REALISATION OF PARALLEL (P,Q) COUNTERS FOR HIGH-SPEED ARRAY MULTIPLIERS

BY

BAKRI MADON

DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

UNIVERSITY COLLEGE LONDON

Submitted in accordance with the requirements

for the degree of

Doctor of Philosophy

September 1989

To my family

ACKNOWLEDGEMENTS

I would like to express my appreciation to the many people who have made this work possible. First, and foremost, my sincere gratitude to my supervisor, Mr. Chris Guy for his patient guidance and continued encouragement throughout the production of this work. Also for his most helpful criticism and comments on the earlier drafts of this thesis.

I am most grateful to Hewlett Packard Santa Clara Technology Center for giving me the opportunity to participate in the development of the HP1X multiplier and the MPC test chip, the use of their SPICE models in the research work and for their kind invitation to allow me to pursue my work in California. In particular, many thanks to Mr. Patrick Byrne for his advice and support during the work on the HP1X multiplier and MPC test chip, and his most helpful guidance during the initial stages of the research work. Also, my sincere gratitude to Mr. William Hillery for his criticism and continued interest in this work.

I am also greatly indebted to members of the IC Design Centre for their constant support, in particular to Mr. Brian Fantini for persevering with my computing demands.

Acknowledgements are due to the government of Malaysia for providing financial support without whom this work might not have been possible.

Last, but not least, I would like to thank my family who has encouraged me to do my best and provided comfort in times of difficulty.

ABSTRACT

With current trend towards single chip digital signal processors and the growing demand for more powerful and real time performance of such processors, further improvements in speed would need to be made on conventional iterative carry-save array (CSA) multipliers. Considerable increases in the speed of array multipliers can be achieved by adding more than one partial product bit at a time by employing higher order parallel (p,q) counters. This approach heavily depends on an efficient realisation of a counter, which ideally should have a delay and complexity comparable to that of a full-adder.

An iterative array multiplier which employs a (5,3) counter was recently reported and based on similar techniques, a novel array multiplier utilising (2,2,3) counter cells was developed in this project. The study shows that both the (5,3) counter and (2,2,3) counter architectures are quite close to conventional array multipliers from a VLSI implementation point of view. Assuming the counters operate at a comparable speed as a CSA full-adder, the (5,3) counter scheme is faster than conventional array multipliers by nearly a factor of two, while the (2,2,3) counter technique offers significant improvements for large operand wordlength. In this work, the (5,3) counter and (2,2,3) counter were studied, principally on the efficiency of operation speed and the viability of the array architectures in the fast bipolar ECL technology. For this purpose, a reconsideration of threshold logic, in view of the better IC processes of today as well as the well-proven cascode ECL technique was investigated. A novel threshold circuit technique based on partial use of negative weighted inputs is proposed to overcome the maximum fan-in weight limitation found in traditional threshold circuits. A method of mapping a logic function onto series gated ECL suitable for software implementation is presented. The work also includes the design of a 16 x 16-bit Booth-encoded multiplier and a test chip composed of ring oscillators, using state-of-the-art bipolar technology.

Simulation results show that the most efficient realisation is the (2,2,3) counter cell implemented in series gated ECL using well-proven gates. Circuit simulations indicate the (2,2,3) counter to be nearly as fast as a CSA full-adder. With such a realisation of the (2,2,3) counter cell, significant improvements in the speed of the (2,2,3) multiplier over that of conventional CSA multiplier can be expected, especially for large operand wordlengths.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Multipliers in digital signal processing

In the past, most digital signal processing has required that digitized signals be recorded and then processed off-line on general-purpose computers. In many cases, however, on-line, real-time and therefore very fast processing is required if digital techniques are to successfully replace analog techniques. In digital signal processing, a sampled point of the analyzed waveform may require one addition and one multiplication. For example, if the analog signal is quantized into eight bits, the required computation is made up of nine additions - one for the real addition and eight simulating a multiplication. It is evident that if the multiplication speed were to match the addition speed, the bandwidth of the signal processor would increase significantly allowing more complex functions to be computed in real-time, and thus the ultimate limitation on performance of digital signal processors (DSP) is multiplication.

Over the years many multiplication algorithms have been proposed and practically used. Major advances in the realizations of multipliers as monolithic integrated circuits over the past 10 years have done much to reduce multiplier delay times, power consumption, size and cost [1.1-1.3]. Recently many researchers have tried to develop high speed multiplier architectures which are suitable for VLSI implementation. As the cost of fabrication per transistor on a chip has continually gone down, parallel algorithms for multiplication become increasingly important. Multiplication algorithms based on different number representations have also been widely investigated.

Depending on the applications there are various approaches to

multiplier design. Almost all DSP functions require a considerable amount of multiplications. Some of these functions require the multiplication to be performed at a faster rate while others concentrate on less hardware and moderate speeds.

The application which aim at hardware simplicity and moderate speeds traditionally use in one form or another [1.4-1.14] the add-shift method for multiplication. For a n x n-bit multiplication, the basic add-shift technique adds sequentially, one row at a time, the intermediate result to the next row of partial products in a large $2n$-bit accumulator, where the carry is allowed to propagate full-length. The principle is similar to the way one multiplies numbers using pen and paper. This carry propagation, along with $(n\text{-}1)$ addition times contribute largely to the slow speed. This type of multiplier was most prominently used in the early digital signal processors [1.7,1.8]. These ranged from discrete serial multipliers to a complete integration with other DSP functions onto a single chip.

Some DSP systems employ serial binary representations of signal sequences in highly multiplexed serial applications such as the processing of telephone voice and data signals [1.15,1.16], where the serial approach was found to be efficient in digital filter organization. The need to rapidly perform multiplications on a stream of numbers packed closely in time led to the concept of pipeline multipliers [1.16-1.23]. A form of the basic shift-and-add algorithm was often implemented in the early pipeline multipliers [1.17-1.20]. Recent pipeline multipliers are based on the parallel approach [1.21-1.23] to take advantage of better and cheaper VLSI technology.

It has widely been recognised that the traditional shift-and-add algorithm of multiplication takes time $O(n^2)$, where n is the maximum wordlengths of the multiplier and multiplicand. This time can be reduced

by combinatorial parallel techniques, but at the cost of increasing the complexity of the circuitry. However, with rapid advancements of VLSI technology which brought about reduced cost of fabrication per transistor on a chip, parallel techniques have become cheaper and more practical. Parallel multipliers were first introduced independently by Wallace [1.24] and Dadda [1.25]. The architectures proposed were based on a tree of full-adder cells to reduce the initial partial product matrix to two operands, which are then summed by fast final adders to obtain the final product. Both Dadda's and Wallace's scheme are optimum in the sense of using a minimum stage of full-adders where the speed is shown as a function of $O(\log n)$. Unfortunately, the irregular structure of Wallace's and Dadda's schemes, especially for large operand wordlengths has prompted research into iterative array multiplier architectures. Coupled with the existence and development of an ever sophisticated suite of IC design tools, iterative array architectures have led a more dominant role. In contrast with the partial product reduction techniques of Dadda and Wallace, iterative array multipliers exhibit speeds as a function of $O(n)$. Although they are inherently slower, the advantages of the array approach in terms of hardware regularity and interconnectivity more than outweigh the speed advantage of Wallace's and Dadda's schemes. Usually, multiplier recoding techniques like the modified Booth algorithm [1.30,1.31] and fast final adders [1.32,1.33] are employed in monolithic multipliers to boost its speed. A type of array architecture that is commonly implemented in practical single-chip multipliers is the carry-save-array (CSA) [1.26-1.28]. In fact, the architecture lends itself well to automatic generation layout tools and has actually been implemented in silicon compilers [1.29].

A more complete review of parallel multipliers is described in Chapter 2 of the thesis. Also reviewed in this chapter are techniques

based on recoding the multiplier bits and recent work on architectures employing different binary number representations.

1.2. Silicon vs. GaAs

A majority of the digital multipliers that have been reported were implemented in the fast bipolar ECL technology to take advantage of its advanced state of maturity. In view of the many recent advances in IC technologies there is a tendency to contemplate employing alternative higher-speed technologies to improve the multiplication process drastically. Indeed, a fair amount of excitement and attention have been paid to the inherently faster GaAs technology. Some authors have reported the impressive speeds at moderate power levels of GaAs multipliers [1.34,1.35].

Significant GaAs - silicon ECL differences were observed in the following areas :

(i) On-chip gate delay

(ii) Transistor count

(iii) gate fan-in and fan-out

Small on-chip gate delays give GaAs an essential edge over silicon for high-performance digital systems. These speed advantages are derived from inherent properties such as higher electron mobility and lower parasitic capacitances.

However, the present practical level of integration of GaAs is at the MSI and LSI level whereas silicon ECL has now reached VLSI densities [1.57,1.58]. Transistor count limitations are primarily due to the poor yield of large GaAs chips. At MSI and LSI densities one must therefore ask what advantages can be accrued solely from the short on-chip gate delays which are already apparent for GaAs and likely to continue in the future, since very high gate densities will not be

available in the near term. One possible answer to this would be that a logic system could be partitioned into small GaAs chips. But, however, it could be argued that little additional performance beyond that of silicon ECL would be gained with on-chip gate delays of as low as 20ps at even LSI device densities, since the chip-to-chip interconnect delays may remain constant and hence limit system performance.

Low fan-in and fan-out of GaAs gates although not believed to be a permanent characteristic, nevertheless currently introduce constraints not found in silicon. Gate fan-out can generally be increased by using larger transistors, as is done in silicon. However, low gate fan-in is a serious problem, particularly for NAND gates. This is because an increase in the number of inputs to a NAND gate reduces the noise margin, and noise margins are very small in GaAs to begin with. In order to achieve working devices with adequate noise margins, very fine control of circuit parameters is required, and this is not yet easily achieved.

Table 1.1, 1.2 and 1.3 compare several recent speed-power figures and delay times of ring oscillators, memory circuits and 8 x 8-bit array multipliers, respectively of GaAs and silicon ECL compiled from [1.36-1.55].

Some conclusions could be drawn from the tables and from some recent papers on bipolar ECL and GaAs [1.56-1.61]. GaAs HEMT, the fastest GaAs circuit may not be significantly faster than silicon ECL at room temperature eventhough it operates at a much higher speed at 77K. In fact, recent results [1.45] show that the performance of HEMT circuits have been shown to be similar to DCFL at 300K. It may require as much power for near equivalent performance to that of silicon ECL. Although many recent papers [1.42,1.53,1.61] have reported impressive speeds of GaAs, these were mainly achieved through high-power consumption and complex circuit techniques such as the Buffered- FET logic (BFL). The

	Typical gate delay(ps)	power per gate(mW)
GaAS DCFL (1μm, 300K)	45	0.5
GaAs HEMT (0.5μm, 77K)	12.8	2.5
Si ECL (1μm, 300K)	90	0.59
Si NTL (0.5μm, 300K)	42	0.5

Table 1.1. Speed-power of Si bipolar and GaAs ring oscillators.

	Access time (ns)	Total power(W)
GaAS DCFL (0.5μm, 300K)	2	0.5
GaAs HEMT (0.5μm, 77K)	0.9	0.4
Si ECL (1μm, 300K)	1	N.A.*

*not available

Table 1.2. Speed-power of Si bipolar and GaAs 1K RAM.

	Multiplication time(ns)	Total power(W)
GaAS DCFL (0.5μm, 300K)	5.2	2.2
GaAs HEMT (0.5μm, 77K)	N.A.	N.A.
Si ECL (1μm, 300K)	5.25	1.0

Table 1.3. Speed-power of Si bipolar and GaAs 8 x 8-bit array multipliers.

less power-hungry Direct-coupled FET logic (DCFL) have gate delays that are 2 to 4 times those of BFL for complex logic circuits.

Major breakthroughs in terms of gate counts and yield of GaAs chips would therefore need to be achieved before it could be considered as a serious alternative in the implementation of high performance systems. This could be a good number of years to come. On the other hand silicon bipolar technology is continually being improved to achieve and challenge GaAs speeds and it is widely considered as the technology of high speed multipliers for many years to come.

1.3. Research objectives

The advent of VLSI technology has undoubtedly permitted the integration of more complex digital functions on to a single chip giving systems that are more compact, faster and reliable. Indeed, these advantages have prompted the current trend towards single chip realizations of digital signal processors.

As was identified earlier, the main constraint to higher performance of DSP is the multiplication speed. Iterative array multipliers such as the CSA multiplier has often been recognized by IC designers to be most suitable for incorporation in single chip DSPs by virtue of its highly regular structure, ease of design and expandability for larger operand wordlengths. Although faster multiplication techniques like Dadda's and Wallace's schemes, and the use of multiplier recoding techniques [1.30,1.31] are feasible, these normally introduce too much complexity and irregularities in the already complicated DSP design, occupying a larger percentage of the silicon area which should otherwise be taken up by other DSP functions. However, with growing demands for more powerful and real-time performance of single chip DSPs, it is obvious that further improvements in speed would need to be made on iterative array

multipliers than that offered by the full-adder CSA architecture.

Dadda first recognized the fact that considerable increases in the speed of parallel multipliers can be achieved by adding more than one partial product bit at a time by employing higher order parallel (p,q) counters [1.25]. This approach heavily depends on a counter which has a delay and complexity comparable to that of a full-adder. The rather complicated and irregular wiring between counters of different sizes in Dadda's scheme has, however hampered the practical use of this approach in large high-speed multipliers. More often, a serious constraint of extending Dadda's technique to higher order parallel counters has been the rather complex and/or slow operation (compared to a full-adder) of such counters. In the last 20 years many researchers have proposed schemes to realise parallel counters with different degrees of success. These include techniques based on two-level gate network [1.62], sequential circuits [1.63,1.64], table look-ups or ROMS [1.65], threshold logic [1.25,1.66-1.68] and networks of full-adders or smaller counters [1.69-1.74].

The aim of this work is to investigate iterative parallel array architectures based on higher order counters as opposed to the partial product reduction techniques proposed by Dadda, suitable for implementation in single chip bipolar DSP applications. The large number of DSP chips that are in the market today are mostly designed in CMOS technology, but with the ever growing demand of real-time performance of DSP applications, especially in image processing, future designs would obviously need to be implemented in the inherently faster bipolar ECL technology. Improved bipolar processes have evolved recently [1.56-1.58] and thus it is appropriate to investigate a more efficient realisation of array multipliers in bipolar technology which would ultimately lead to more powerful, real-time performance single chip digital signal

processors.

In fact, little attention has been paid in the past to iterative array schemes based on larger counters and thus it is necessary to understand how techniques such as the CSA architecture could be extended. Since the viability of the approach is highly dependent on an efficient implementation of the associated parallel counters, a considerable amount of work is needed to examine various possible configurations of the counters. With todays bipolar technology of smaller size and less power-hungry transistors [1.56-1.58], a more combinatorial approach to realize the counters in order to achieve significant improvements in the multiplication speed should be feasible. Bipolar ECL/EFL circuit techniques and threshold logic, which was shown in the past to be efficient in the synthesis of large counters ought to be considered. The work, ultimately includes an examination of the counters inherent speed in contrast with the fastest full-adder design which would then allow a critical comparison of the new array scheme with conventional CSA multipliers.

1.4. Outline of thesis

Chapter 2 reviews the traditional and most commonly used schemes of designing parallel high-speed multipliers. The two main techniques of parallel multipliers - the partial product reduction method of Wallace and Dadda, and the iterative CSA architecture are assessed in terms of speed and their attractiveness for VLSI implementation. Also, multiplier recoding techniques and multiplication based on different binary representations are examined. An introduction to the concept of using parallel (p,q) counters to enhance multiplication speed is presented. The key areas that require further investigations are then identified and considered.

Chapter 3 describes work undertaken to design a high-speed, bipolar 16 x 16-bit multiplier which was designed in collaboration with Hewlett Packard Company, Santa Clara in the IC design Centre, University College London. The architectures commonly employed in a typical high-speed multiplier are demonstrated through discussions of the chip. Also described is a test chip composed of ring oscillators of full-adder cells made up of different transistor sizes and load resistors. The chip was used to characterize the delay of the full-adder cell in order to get an optimum design of the 16 x 16-bit multiplier and also as the basis of comparison with larger (p,q) counters.

In Chapter 4, extension of the CSA approach and novel array multipliers based on higher order parallel (p,q) counters are first studied. An iterative array (5,3) counter multiplier recently reported is highlighted and a novel architecture based on similar concepts of the (5,3) counter multiplier but employing (2,2,3) counters are then presented. These architectures are assessed in terms of speed and their attractiveness for VLSI implementation.

Chapter 5 describes the implementation of the (5,3) counter and (2,2,3) counter cells in cascode ECL and threshold logic. The complexity and efficiency of operation speed of the two counters using these techniques are emphasized. A novel circuit technique which overcomes the earlier problem of needing high fan-in weight threshold gates is presented. A technique of mapping a logic function onto series gated ECL suitable for software implementation is also described.

Using SPICE circuit simulations, a critical examination of the propagation delay of the (5,3) counter and (2,2,3) counter cells implemented in ECL and threshold logic in comparison with a full-adder cell is made in Chapter 6. The novel threshold circuit technique is also analyzed to characterize its maximum fan-in weight. A gate level

simulation is then performed on the optimum novel array multiplier architecture to evaluate its speed in contrast with conventional CSA multipliers.

Chapter 7 gives a discussion of the conclusions made and identifies areas that require further work.

REFERENCES

[1.1] J.R. Mick & J. Springer, " Single chip multiplier expands digital role in signal processing ", Electronics, Vol. 49, pp. 103-108, May 13, 1976.

[1.2] W. Bucklen, J. Eldon, L. Schirm & F. Williams, " Single-chip digital multipliers form basic DSP building blocks ", EDN, PP.153-163, April 1, 1981.

[1.3] S. Waser & A. Peterson, " Real-time processing gains ground with fast digital multiplier ", Electronics, pp. 93-99, Sept. 29, 1977.

[1.4] E.E. Swartzlander Jr., " The quasi-serial multiplier, " IEEE Trans. Computer, Vol. C-22, No. 4, pp. 317-321, April 1973.

[1.5] D. Hampel, K.E. Mcguire & K.J. Prost, " CMOS/SOS serial-parallel multiplier ", IEEE J. Solid-State Circuits, Vol. SC-11, pp. 669-678, Oct. 1975.

[1.6] J.R. Verjans, " A serial-parallel multiplier using the NENDEP technology ", IEEE J. Solid-State Circuits, Vol. SC-12, pp. 323-325, June 1977.

[1.7] N.R. Powell & J.M. Irwin, " A MOS monolithic chip for high-speed flexible FFT processors ", IEEE Int. Solid-State Circuits Conf., pp. 18-19, 1975.

[1.8] N. Ohwada, T. Kimura & M. Doken, " LSI's for digital signal processing ", IEEE Trans. on Electron Devices, Vol. ED-26, No. 4, pp. 292-298, 1979.

[1.9] I. Chen & R. Willoner, " An O(n) parallel multiplier with bit-sequential input and output ", IEEE Trans. Computers, Vol. C-28, No. 10, pp.721-727, Oct. 1979.

[1.10] R. Gnanasekaran, " On a bit-serial input and bit-serial output multiplier ", IEEE Trans. Computers, Vol. C-32, No. 9 pp. 878-880, Sept. 1983.

[1.11] T. Rhyne & N.R. Strader, " A signed bit-sequential multiplier ", IEEE Trans. Computers, Vol. C-35, No. 10, pp. 896-901, Oct. 1986.

[1.12] N.R. Strader & V.T. Rhyne, " A canonical bit sequential multiplier ", IEEE Trans. Computers, Vol. C-31, No. 8, pp. 791-795, August 1982.

[1.13] J. A. Starzyk & V. S. R. Dandu, " Overlapped multi-bit scanning multiplier ", Proc. IEEE ICCD, pp. 363-366, 1985.

[1.14] R. Gnanasekaran, " A fast serial-parallel binary multiplier ", IEEE Trans. Computers, Vol. C-34, No. 8, pp. 741-744, August 1985.

[1.15] L.B. Jackson, J.F. Kaiser & H.S. Mcdonald, " An approach to the implementation of digital filters ", IEEE Trans. Audio Electroacoust., Vol. AU-16, pp. 413-421, Sept. 1968.

[1.16] S.L. Freeny, " Special-purpose hardware for digital filtering ", Proc. IEEE, Vol. 63, pp. 633-648, Apr. 1975.

[1.17] R.F. Lyon, " Two's complement pipeline multipliers ", IEEE Trans. Commun., Vol. COM-12, pp. 418-425, April 1976.

[1.18] J. Kane, " A low-power, bipolar, two's complement serial multiplier chip", IEEE J. Solid-State Circuits, Vol. SC-11, No. 5, pp. 669-678, Oct. 1976.

[1.19] G. L. Baldwin, " A modular, high-speed serial pipeline multiplier for digital signal processing ", IEEE J. Solid-State Circuits, Vol. SC-13, No. 3, pp. 400-408, June 1978.

[1.20] E.K. Cheng & C.A. Mead, " A two's complement pipeline multiplier ", IEEE Int. Conf. Acoustics, Speech and signal processing, pp.647-650, 1976.

[1.21] M. Hatamian & G. L. Cash, " A 70-MHz 8-bit x 8-bit parallel pipelined multiplier in 2.5-um CMOS ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 4, pp. 505-513, August 1986.

[1.22] T. G. Noll et al, " A pipelined 330-MHz multiplier ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 3, pp. 411-416, June 1986.

[1.23] J. Deverell, " Pipeline iterative arithmetic arrays ", IEEE Trans. Computers, pp. 317-322, March 1975.

[1.24] C.S. Wallace, " A suggestion for parallel multipliers, " IEEE Trans. Electron. Comput. Vol. EC-13, pp 14-17, Feb. 1964.

[1.25] L. Dadda, " Some schemes for parallel multipliers, " Alta Frequenza, Vol. 34, pp. 349-356, Mar. 1965.

[1.26] A. Habibi & P.A. Wintz, " Fast Multipliers, " IEEE Trans. Computer, Vol. C-19, pp. 153-157, Feb. 1970.

[1.27] S. Waser, " High speed monolithic multiplier for real time digital signal processing, " Computer, Vol. 11, no. 10, pp. 19-29, Oct. 1978.

[1.28] E.L. Braun, Digital Computer Design, New York, Academic Press, 1963.

[1.29] N.F. Benschop, " Layout compiler for variable array multipliers ",

Proc. Custom Integrated Circuits Conf., pp.47-58, 1983.

[1.30] D. Booth, " A signed binary multiplication technique, " Quart. J. Mech. Appl. Math., Vol. 4, part 2, 1951.

[1.31] L.P. Rubinfield, " A proof of the modified Booth's algorithm for multiplication ", IEEE Trans. Computers, pp. 1014-1015, Oct. 1975.

[1.32] J. Sklansky, " An evaluation of several two-summand binary adders ", IRE Trans. Electronic Computers, pp. 213-226, June 1960.

[1.33] J. Sklansky, " Conditional-sum addition logic ", IRE Trans. Electronic Computers, pp. 226-231, June 1960.

[1.34] Y. Nakayama, K. Suyama, H. Shimuzu, N. Yokoyama, H. Onishi, A. Shibatomi & H. Ishikawa, " A GaAs 16 x 16 bit parallel multiplier, " IEEE J. Solid-State Circuits, Vol. SC-18, No.5, October 1983.

[1.35] F.S. Lee, G.R. Kaelin, B.M. Welch, R. Zucca, E. Shen, P. Asbeck, C.P. Lee, C.G. Kirkpatrick, S.I. Long & R.c. Eden, " A high-speed LSI GaAs 8 x 8 bit parallel multiplier, " IEEE J. Solid-State Circuits, Vol. SC-17, no. 4, pp. 638-647, Aug. 1982.

[1.36] T. Sakai & M. Suzuki, " Super self-aligned bipolar technolgy ", Symp. VLSI technology, Dig. Tech. papers, pp. 16-19, 1983.

[1.37] R. Ranffit & H. Rein, " High-speed bipolar logic circuits with low power consumptionfor LSI- A comparison ", IEEE J. Solid-State Circuits, Vol. SC-17, No. 4, pp. 703-712, August 1982.

[1.38] P.M. Soloman, " A comparison of semiconductor devices for high-speed logic ", Proc. of the IEEE, Vol. 70, No. 5, pp. 489-509, May 1982.

[1.39] T.H. Ning & D.D. Tang, " Bipolar trends ", Proc. of the IEEE, Vol. 74, No. 12, pp. 1669-1677, Dec. 1986.

[1.40] S. Wiedmann, " Advancements in bipolar VLSI circuits and technologies ", IEEE J. Solid-State Circuits, Vol. SC-19, No. 3, pp. 282-290, June 1984.

[1.41] J. Lohstroh, " Devices and circuits for bipolar (V)LSI ", Proc. of the IEEE, Vol. 69, No. 7, pp. 265-279, July 1981.

[1.42] V. Milutinovic & D. Fura, " An introduction to GaAs Microprocessor architecture for VLSI, " IEEE Computer, pp.30-42, March 1986.

[1.43] B.K. Gilbert, " Design and performance trade-offs in the use of Si VLSI and Gallium Arsenide in high clockrate signal processors ", IEEE Publication 1984, pp. 260-266.

[1.44] L. Larson, J. Jensen, P. Greiling & M. Waldner, " GaAs technology for high speed digital computation ", IEEE Publication 1985, pp. 384-389.

[1.45] J.V. DiLorenzo et al, " GaAs - Status and directions ", IEEE Publication 1985, pp. 371-383.

[1.46] S.I. Long et al, " High speed GaAs integrated circuits ", Proc. of the IEEE, Vol. 70, No. 1, pp. 35-45, Jan. 1982.

[1.47] H. Yuan, " The status and prospect of GaAs bipolar LSI ", IEEE Publication 1985, pp. 391-394.

[1.48] R.C. Eden, " Comparison of GaAs device approaches for ultrahigh-speed VLSI ", Proc. of the IEEE, Vol. 70, No. 1, pp. 5-12, Jan. 1982.

[1.49] R.C. Eden, B.M. Welch, R. Zucca & S.I. Long, " The prospects for ultra high-speed VLSI GaAs digital logic ", IEEE J. Solid-Stae Circuits, Vol. SC-14, No.2, pp. 221-239, April 1979.

[1.50] N. Weste, "GaAs design and simulation issues-are they any different from silicon ", IEEE Publication 1984, pp. 268-272.

[1.51] G. Nuzzilat et al, " GaAs MESFET IC's for gigabit logic applications ", IEEE J. Solid-Stae Circuits, Vol. SC-17, No. 3, pp. 569-583, June 1982.

[1.52] P.M. Soloman & H. Morkoc, " Modulation-doped GaAs/AlGaAs heterojunction field-effect transistors (MODFET's), ultrahigh-speed device for supercomputers ", IEEE Trans. Electron devices, Vol. ED-31, No. 8, pp. 1015-1027, August 1984.

[1.53] T.J. Drummond, W.T. Masselink & H. Morkoc, " Modulation-doped GaAs/(Al,Ga) heterojunction field-effect transistors: MODFETs ", Proc. of the IEEE, Vol. 74, No. 6, pp. 773-822, June 1986.

[1.54] B.M. Welch, R.C. Eden & F.S. Lee, " GaAs digital integrated circuit technology ", Chapter 13, Gallium Arsenide, 1985 John Wiley & sons ltd.

[1.55] Morkoc & P.M. Soloman, " The HEMT: a superfast transistor ", IEEE Spectrum, pp. 28-35, Feb. 1984.

[1.56] K. Washio, T. Nakamura, K. Nakazato & T. Hayashida, " A 48ps ECL in a self-aligned bipolar technology ", Digest of Int. Solid-State Circuits Conf., pp. 58-59, Feb. 1987.

[1.57] " Fairchild's radical process for building bipolar VLSI ", Electronics, pp. 55-101, Sept. 4 1986.

[1.58] " Technology to watch, " Electronics, pp. 35-38, April 7, 1986.

[1.59] Y. Oowaki et al, " A sub-10-ns 16x16 multiplier using 0.6-μm CMOS technology ", IEEE J. Solid-State Circuits, Vol. SC-22, N0. 5, pp. 762-765, Oct 1987.

[1.60] Y. Oowaki et al, " A 7.4ns CMOS 16 x 16 multiplier ", Digest of International Solid-State Circuits Conference, pp. 52-53, Feb. 1987.

[1.61] B.K. Gilbert & G. Pau, " The application of GaAs integrated circuit technology to the design and fabrication of future generation digital signal processors : promises and problems ", Proc. of the IEEE, Vol. 76, No. 7, pp. 816-834, July, 1988.

[1.62] E. E. Swartzlander, Jr., " Parallel Counters ", IEEE Trans. Computers, Vol. C-22, No. 11, pp. 1021-1024, Nov. 1973.

[1.63] A. Svoboda, " Adder with distributed control ", IEEE Trans. Computers, Vol. C-19, No. 8, pp. 749-751, August 1970.

[1.64] S. Singh & R. Waxman, " Multiple operand addition and multiplication ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 113-120, Feb. 1973.

[1.65] L. Dadda, " On parallel digital multipliers ", Alta Frequenza, pp. 574-580, Oct. 1976.

[1.66] D. Ferrari & R. Stefanelli, " Some new schemes for parallel multipliers, " Alta Frequenza, Vol. 38, pp. 843-852, Nov. 1969.

[1.67] D.Hampel & R.O. Winder, " Threshold logic ", IEEE Spectrum, pp. 32-39, May 1971.

[1.68] R.O. Winder, " Threshold logic will cut costs, especially with boosts from LSI ", Electronics, pp. 94-103, May 27 1968.

[1.69] C. C. Foster & F. D. Stockton, " Counting responders in an associative memory ", IEEE Trans. Computers, pp. 1580-1583, Dec. 1971.

[1.70] S. Dormido & M. A. Canto, " Synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-30, No. 9, pp. 699-703, September 1981.

[1.71] H. Kobayashi & H. Ohara, " A synthesizing method for large parallel counters with a network of smaller ones ", IEEE Trans. Computers, Vol. C-27, No. 8, pp. 753-757, August 1978.

[1.72] S. Dormido & M. A. Canto, " An upper bound for the synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-31, No. 8, pp. 802-805, August 1982.

[1.73] D. G. Gajski, " Parallel Compressors ", IEEE Trans. Computers, Vol. C-29, No. 5, pp. 393-398, May 1980.

[1.74] L. Dadda, " Composite parallel adders ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 942-946, October 1980.

CHAPTER 2

APPROACHES TO PARALLEL MULTIPLIER DESIGN

2.1. Introduction

Digital multipliers continue to occupy a position of major importance in the design and development of general and special purpose computers. As one of the key elements in digital signal processors, the multiplier has traditionally hampered the consideration of the digital approach to signal processing owing to high cost, poor speed-power performance and inefficient packaging. Major advances in the realization of multipliers as monolithic integrated circuits [2.1-2.13] have done much to reduce delay times, power consumption, size and cost. However, with the trend towards single chip digital signal processors and the increased demand for lower cost, higher speed and higher efficiency realizations of digital signal processors [2.14] continued improvements in the performance of digital multipliers is needed.

This chapter reviews the traditional and well-known schemes of designing monolithic high speed, parallel multipliers commonly employed today. The full-adder carry-save array (CSA) [2.15,2.16], Wallace tree [2.16-2.18], Booth algorithm [2.16,2.19,2.20] and fast final adders [2.21-2.24] are described. Multipliers based on the full-adder CSA and those that employ the Booth algorithm and fast final adders are assessed in terms of the two main criteria - speed and complexity. The key areas that require further investigations are then identified and considered.

Complexity is defined in the context of high-speed multipliers as a measure of how regular the architecture is in terms of layout and how attractive it is for VLSI implementation. The speed of a multiplier is conventionally taken to be the worst case delay, which is the maximum

number of gate delays that a propagating signal could traverse in the architecture, although on average, the computations take less time.

2.2. Traditional approaches to multiplier design

In the development of monolithic high-speed multipliers the classic, simple add-and-shift algorithm [2.16] has been superseded by a more parallel approach to multiplier design. The rapid development and advancement of IC technology over the past twenty years has made the fabrication of parallel multipliers become economically more feasible and encouraged many researchers to look at parallel algorithms [2.1-2.13,2.15-2.18,2.34-2.53] and those based on different binary number representations [2.8,2.53-2.58] which are suitable for VLSI implementation.

Multiplication can be done in either signed or unsigned number representations. From the arithmetic and implementation point of view, two's complement is the most attractive for signed numbers. Addition is straight-forward and significantly faster than say, sign-and-magnitude and multiplication is easier to understand and implement. Two's complement has often been stated in the past to be difficult due to the need for sign correction with negative operands. However, this is a hangover from the days of expensive logic and since then many 'correction' algorithms have been reported [2.42-2.47]. The full-adder [2.33] has always been the basic cell used for the implementation of parallel multipliers. Irrespective of the technology used, the full-adder is easy to design and the associated architectures are simple and regular which makes them ideal for VLSI.

There are three basic methods that have been developed over the years to improve the speed of the simple add-and-shift multiplier. The

first speeds up the addition of partial products by using parallel architectures. Schemes for parallel multiplication are roughly divisible into two classes- iterative array of cells [2.1-2.13,2.15,2.16,2.34-2.47] and generation of a matrix of partial product terms with subsequent reduction of the matrix by means of parallel counters [2.16-2.18,2.25-2.28,2.48-2.51]. The second reduces the number of rows of partial products by employing multiplier recoding techniques [2.29] like the modified Booth algorithm [2.19,2.20]. Lastly fast adders [2.21-2.24] are often used at the last stage to add up the final two rows of sum-carry pairs.

For the purpose of simple illustration, examples of 5 x 5-bit multiplication are given; also dot representation (where each dot is a binary bit) of the partial products is used.

2.2.1. Parallel multiplication schemes

Multiplication is a special case of many consecutive additions. Under these circumstances it is not necessary to let the carry signals propagate until all the partial products have been added. Each full adder bit is a three input device, with two outputs. Thus if these two outputs are fed to two of the three inputs of appropriately weighted bit positions for the next addition, a new partial product can be added via the third input. This type of adder is also known as a Carry Save Adder since the carry is saved and not propagated. A basic carry-save array (CSA) multiplier is shown in Figure 2.1 for a 5 x 5-bit multiplier where an array of these adders are appropriately connected. All the partial products are formed simultaneously, added and shifted appropriately by hardware. Lower significant final product bits up to P_{n-1} i.e. the final product bit of weight 2^{n-1} are generated automatically by the right-most

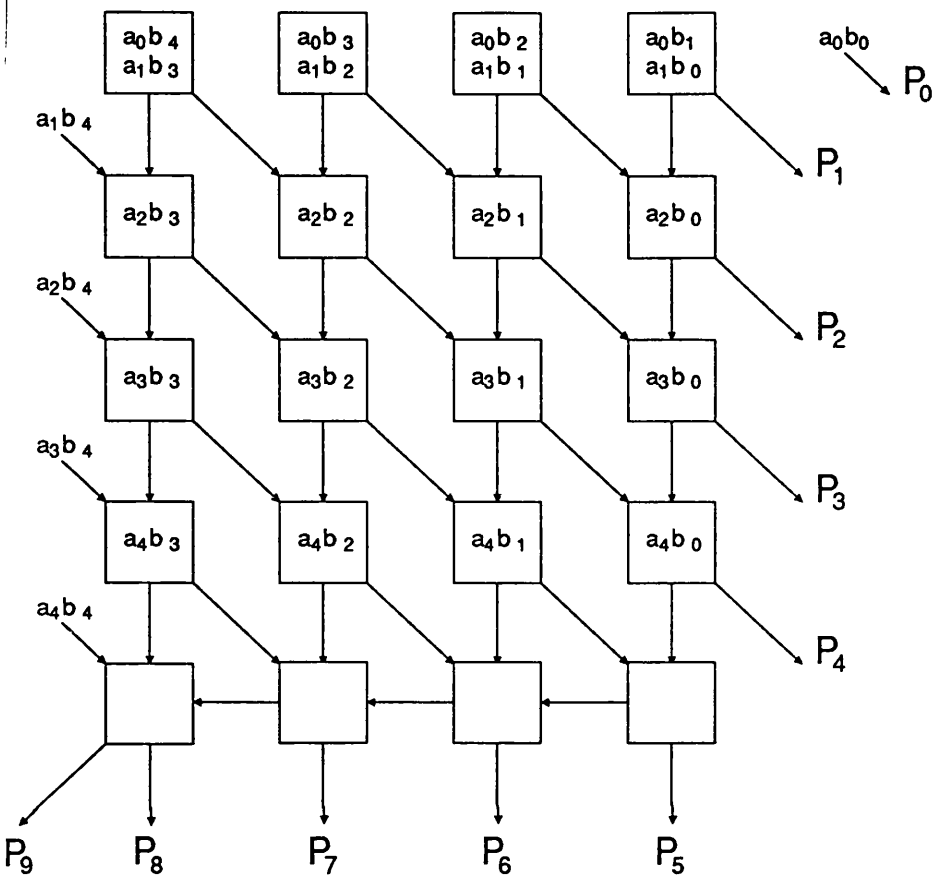


Figure 2.1 Block diagram of a 5 x 5- bit multiplier using carry-save technique [2.15].

column cells whilst the higher significant final product bits are computed by an extra stage of full-adders at the bottom of the array.

For an n x n bit multiplication the array requires 1 stage of half adders and (n-1) stages of full adders. The worst-case delay (including the AND gate delay for the generation of partial product bits) is given by

$$(2n-1) \text{ gate delays} \qquad (2.1)$$

since the longest carry propagate chain in the array is through the right column and bottom edge of the array. Thus this architecture employs one cell for the main multiplication array, and with the regularity, modularity and simple interconnections between adders, is ideal from a

VLSI implementation point of view.

Additional speed can be gained by the use of a Wallace tree [2.15-2.18] which is a form of matrix generation-reduction scheme. In this architecture, a series of full-adders, that operate in parallel, combines groups of three bits of the same binary weight to produce a vector of Sum bits (S) and a vector of Carry bits (C). If more than one C and one S vectors are generated from the first stage, these are combined again in groups of three in the next stage. This procedure continues until one C and one S vector remain. Then a string of full-adders is applied to those two vectors to obtain the final result. In general, for a n x n-bit multiplication the number of gate delays necessary to reduce n partial products to one C and one S vector is given [2.16] by

$$\left[\log_{\frac{3}{2}} n \right] - 1 \qquad (2.2)$$

Figure 2.2 shows a block diagram of a Wallace tree for the multiplication of 5-bit numbers in the ideal case where all bits of the same binary weight are under the same column. However, unless a large chip area is used, this is not the case in an actual design. In a minimum area layout, bits of the same binary weight are in different columns, and additional wiring is required for the interconnection of the appropriate partial products resulting in a much more complex and irregular interconnection pattern.

Dadda [2.25-2.26] also introduced a similar procedure to the Wallace tree for reducing the partial products to two numbers using full-adders. His procedure is optimum in the sense that it uses a minimum number of full-adders. The difference between this technique and Wallace's scheme is in their ways of connecting full adders.

Comparisons between the three architectures on the basis of speed

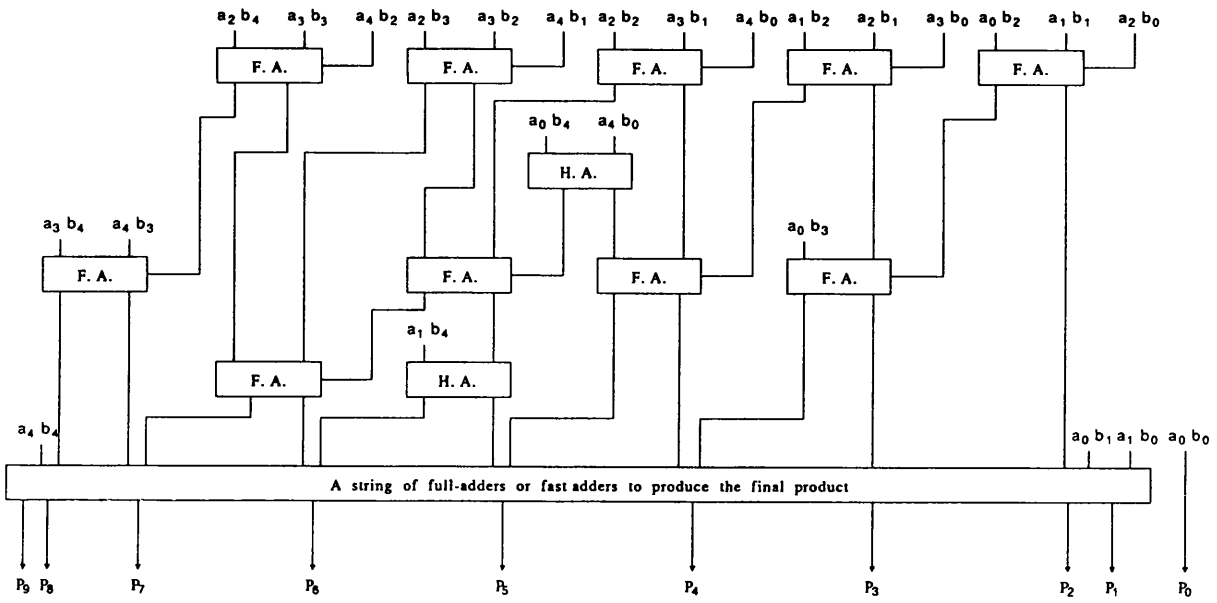


Figure 2.2. Block diagram of a 5 x 5-bit multiplier using Wallace's scheme [2.15].

and complexity are discussed in section 2.2.4

2.2.2. Fewer partial products

Another speed-up technique called Booth's algorithm [2.19], increases speed by reducing the number of partial products by half; this reduces the number of carry-save adder stages, and hence the total multiplication delay. It requires little hardware without also substantially increasing the complexity of the main multiplication array.

Basically, Booth's algorithm allows the multiplication operation to skip over any contiguous strings of all 1s and all 0s, rather than form a partial product for each bit. Skipping a string of 0s is straightforward, but in skipping over a string of 1s, the following property is put to use: a string of 1s can be evaluated by subtracting the weight of the rightmost 1 from the modulus. Thus, the value of the string 11100 computes to 2^5-2^2 = 28. However, in Booth's algorithm consecutive strings

of 3-bits of the multiplier word are recoded with the most significant bit of the string saved for the next higher string to enable us to remember the previous action. All possible permutations are computed from Table 2.1. The algorithm works for two's complement numbers and requires that the multiplier be padded with a 0 to the right of the least significant bit (LSB) to detect the start of a string of 1s. To work with unsigned numbers the multiplier must also be padded with 0s to the left of the MSB so that the multiplier will not be treated as a negative number.

Y_{i+1}	Y_i	Y_{i-1}	Operation
0	0	0	Add zero
0	0	1	Add multiplicand
0	1	0	Add multiplicand
0	1	1	Add 2x multiplicand
1	0	0	Subtract 2x multiplicand
1	0	1	Subtract multiplicand
1	1	0	Subtract multiplicand
1	1	1	Subtract zero

All operations are followed by a 2-bit shift.

Breaking a 8-bit multiplier into five groups of 3 bits each :

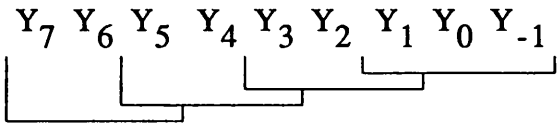


Table 2.1. A modified Booth algorithm.

Although one could take and decode more than 3 bits at a time, the Booth encoder required becomes more complicated and impractical as a vector of three times the multiplicand would need to be generated and this cannot be easily obtained by simple shifting of the partial product.

A full-propagate addition is needed to generate this, limiting the effectiveness of the reduction in the number of carry-save additions.

2.2.3. Fast final adders

If we look at the straight full-adder CSA, Wallace and Dadda tree there is a vector of sum and carry bits (also known as the Intermediate Product [IP] bits) being added by a string of full-adders in order to obtain the final higher significant product bits. However, the IP bits would have settled down to their final values after a gate delay given by the number of stages of full-adders. The string of full-adders is basically a ripple-carry adder [2.21] and though is ideal for VLSI implementation, it is slow due to rippling of the carry bit. Thus, the addition of the higher significant IP bits is one of the main speed-limiting factor of the CSA multiplier and thus calls for a much faster adder architectures than the simple ripple carry adder.

Conventional fast adders can be roughly categorized into 2 classes of algorithms, conditional-sum and carry-look-ahead [2.21]. Conditional-sum was invented by Sklansky [2.21,2.22] and it has been considered by Winograd [2.30] to be faster than the carry-look-ahead. This was evaluated based on an (r,d) circuit, which is a d-valued logical circuit in which each element has a fan-in at most r and can compute any r-argument, d-valued logical function in unit time. Table 2.2. [2.21-2.24,2.77] shows the comparisons of addition speed (in gate delays) of the two architectures with the theoretical lower bound given by Winograd. Other fast adder architectures have been reported over the years like the canonic and Ling [2.59] adders but are limited by implementation problems.

From an implementation point of view, the conditional sum adder is

	Gate delays
Winograds lower bound	$\lceil \log_r 2n \rceil$
Carry-look-ahead	$4\lceil \log_r n \rceil$
Conditional-sum	$2 + \log_r + \lceil \frac{n-r}{r} \rceil$

Table 2.2. Relative speeds of fast adder architectures

more attractive because of its more regular and simpler layout compared to the carry-look ahead. Furthermore, the carry-look ahead is plagued by high gate fan-in and fan-out requirements for large operand wordlength. The conditional-sum adder is thus the most attractive scheme both in speed and layout. For a much faster addition, a combination of these two types of adders could be employed if the area overhead and power requirements are tolerable. In fact, Winograd showed that with an (r,d) circuit, the theoretical lower bound (fastest time) of addition of two operands is nearly achievable with the conditional-sum and carry-look-ahead combination algorithm.

2.2.4. Architectural comparisons

The procedure for the reduction of the partial products is the basic difference between the CSA array, Wallace's and Dadda's schemes. In Wallace's and Dadda's schemes, the number of stages increases as the logarithm of n whereas in the carry save scheme the number of stages is linearly dependent on n as shown in equations (2.2) and (2.1), respectively. Thus Dadda's and Wallace's techniques are much faster than the CSA array for large wordlength n. Both Dadda's and the CSA array

approach are optimum in the sense of using a minimum number of full adders. Wallace's scheme usually needs more full adders.

In general, matrix generation-reduction schemes are much faster than array types since their speed of operation increases with the logarithm of the wordlength [2.16,2.17]. The array schemes, however are more ideal for VLSI implementation because of their regular layout and use of one basic circuit type. The rather complicated and irregular wiring between the adders in a Wallace tree has often hampered their practical use in high-speed, single chip multipliers.

Thus, for large multipliers like a 32 x 32-bit, the advantages of the CSA array in terms of hardware regularity and interconnectivity outweigh the speed advantage of the Wallace tree architecture. Furthermore, the CSA array architecture is easily expandable by adding additional rows and columns of full adders to fit the specific application. However, if the Booth encoder is to be employed in the design this must be accompanied by the use of a fast final adder. Without a fast adder the speed of the multiplier is not significantly better than that of the straight CSA since the critical path for both schemes is along the right column and bottom row of full-adders where the final product bits are computed.

Based on the above considerations, the fastest practical realization of a high-speed multiplier would incorporate the modified Booth encoder, carry-save array and a fast adder as shown in Figure 2.3. This configuration is the basis of many practical, high-speed multiplier designs implemented today and was adopted in the design of the HP1X multiplier (see Chapter 3). The theoretical speed of a n x n-bit multiplier based on this configuration can be derived as follows.

The Booth encoder generates a reduced partial product matrix $\lceil n/2 \rceil$

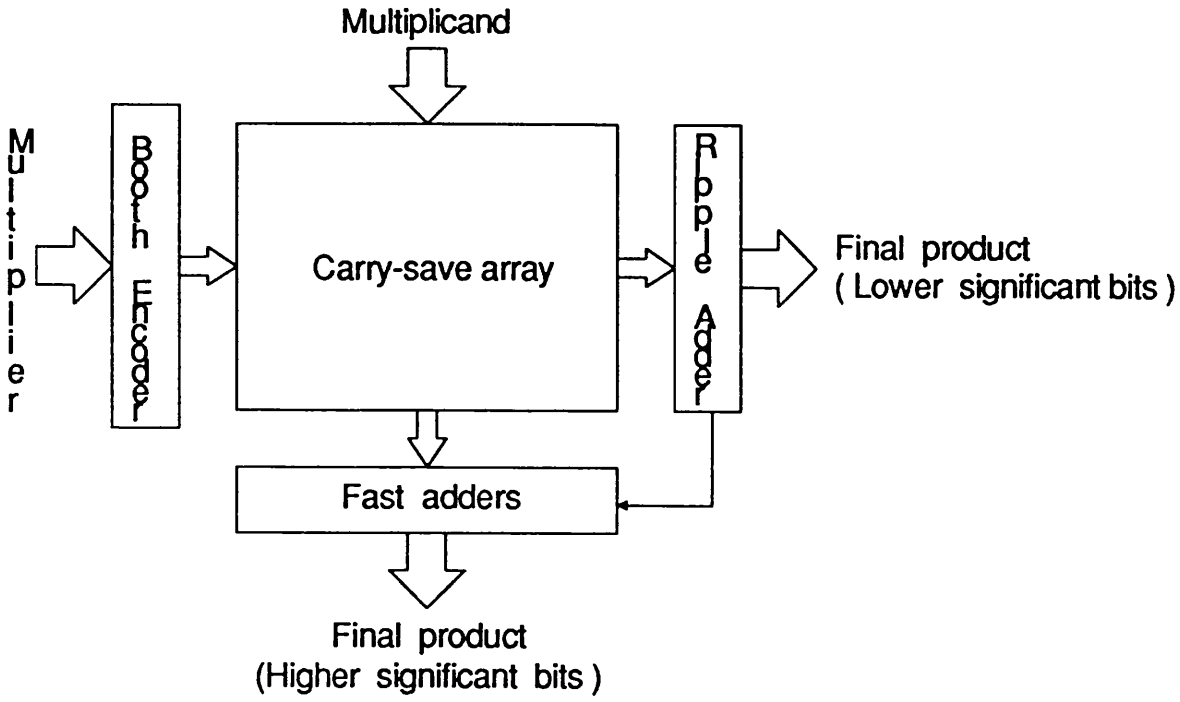


Figure 2.3. Block diagram of a typical high-speed multiplier

rows of n-bits each after a delay D[Booth]. With a carry-save array employed, the lower significant final product bits up to $P_{\lceil n/2 \rceil}$ would have been automatically generated by the array after a delay of $(\lceil n/2 \rceil - 1)$.D[FA]. This leaves two operands, the intermediate product of m-bits to be computed by the fast final adder, where m is given by

$$m = (2n-2) - \lceil n/2 \rceil = \lfloor 3n/2 \rfloor - 2 \qquad (2.3)$$

In the last section we have identified the conditional-sum adder to be the most attractive approach in terms of speed and VLSI implementation. Thus, with this type of adder employed, the maximum delay of a Booth-encoded, CSA of full-adders with conditional-sum (CS) adder is given by

$$D[\text{Booth}] + (\lceil n/2 \rceil - 1).D[\text{FA}] + D[\text{m-bit CS adder}] \qquad (2.4)$$

With the delay of a full-adder equal to 1 gate, the delay of a Booth encoder consists of 2 gate delays, one to decode the truth-table and the

other to select the appropriately shifted or unshifted multiplicand. For a conditional-sum adder (from Table 2.2) with a typical fan-in of 4 which is achievable in most proven technologies the delay of a m-bit fast adder is then given by

$$4 + \lceil (m-4)/4 \rceil \text{ gate delays.}$$

Equation (2.4) thus reduces to

$$5 + \lceil n/2 \rceil + \lceil (m-4)/4 \rceil \qquad (2.5)$$

In cascode ECL technology an estimate of the total gate count, which also represents the relative total power consumption has been evaluated [2.60] for the Booth multiplier and the straight CSA multiplier. The results are shown in Table 2.3. The speed and gate count of the two schemes are also shown graphically in Figure 2.4(a) and (b), respectively.

There is a significant improvement in the multiplication speed with the Booth algorithm and fast adders employed for large n. This is achieved with reasonably low increase in the gate count and hence, power consumption. It was observed that [2.60] the total number of gates in the main carry-save array is actually reduced by employing the Booth encoder. The overall higher gate count of the Booth multiplier in contrast with the straight CSA multiplier is accounted for by the fast final adder architectures. Figure 2.4(b) does not truly reflect the relative silicon

	Multiplication speed (gate delays)	Gate count (ECL)
Straight CSA	$(2n-1)$	$2(n-1)^2 + n^2 + 2(n-1)$
Booth Multiplier	$5 + \lceil n/2 \rceil + \lceil (m-4)/4 \rceil$	$(4n + 9)\lceil n/2 \rceil + 7(\lfloor 3n/2 \rfloor - 2)$

Table 2.3. Speed and ECL gate count of straight CSA multiplier and Booth-encoded CSA multiplier with fast final adders

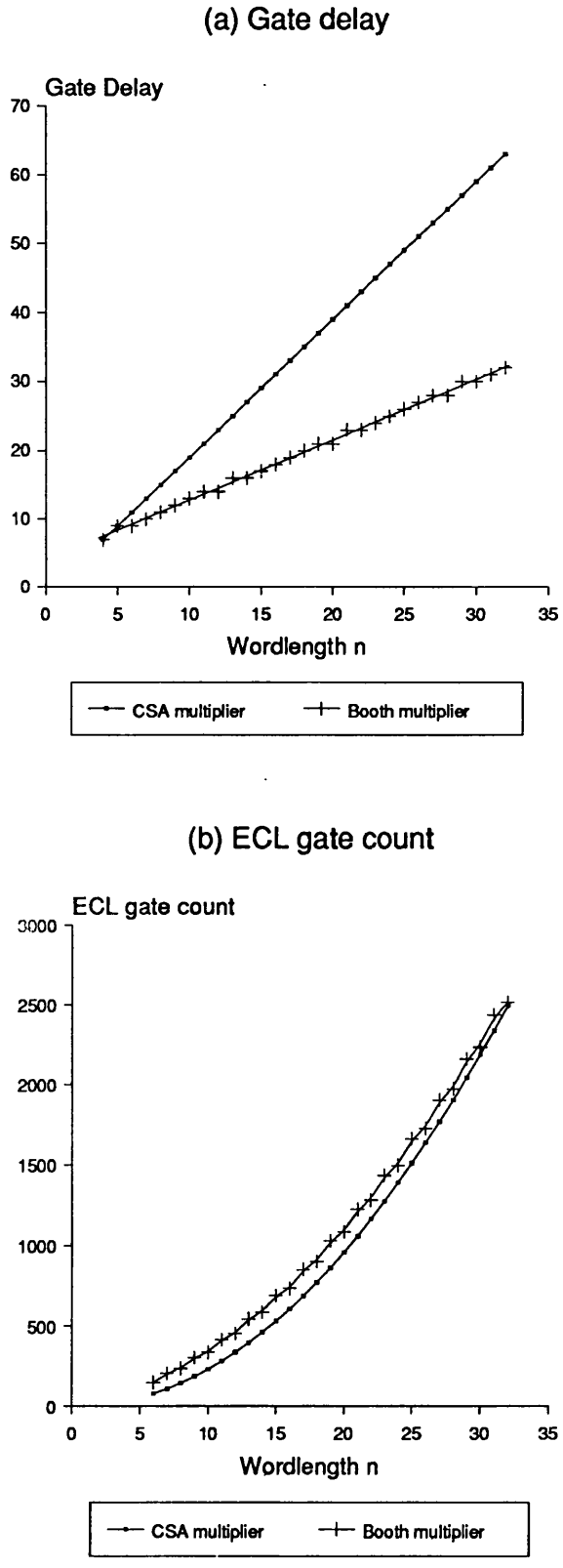


Figure 2.4. **(a)** Relative gate delay and **(b)** ECL gate count of a straight CSA multiplier and Booth-encoded CSA multiplier with fast final adders.

area of the two approaches since this is dependent on the complexity and regularity of wiring between cells. Furthermore, Table 2.3 was obtained assuming the AND gates in the CSA multiplier used to generate the partial product bits are comparable to the gates of the full-adders. Bearing in mind that a more complex and less regular interconnection (in contrast with the carry-save array) is likely to exist in the fast adder architectures, the relative silicon area of the two approaches would be much worse than that shown in Figure 2.4(b). The main advantage of the straight CSA is that it is highly regular which makes it attractive for VLSI implementation with quick design time and thus is ideal for design automation systems like silicon compilers [2.86].

A 16 x 16-bit, bipolar ECL Booth multiplier (called the HP1X) was designed in the Department of Electronic and Electrical Engineering, University College London, in collaboration with Hewlett-Packard Technology Centre, Santa Clara in California. This is described in Chapter 3.

2.3. Bounds on multiplication speed

Winograd [2.61] has derived the theoretical lower bound on multiplication in terms of an (r,d) circuit. It was shown that the theoretical lower bound t on multiplication speed is the same or slightly slower than the theoretical lower bound on addition speed where it is given by

$$t \geq \left\lceil \log_r(n-2) \right\rceil \qquad (2.6)$$

It is interesting to compare the speeds of practical circuits to the theoretical lower bounds. In multiplication, the fastest possible realization [2.16,2.17] uses Booth encoding and a Wallace tree interconnection of full-adders with a conditional-sum/carry-look-ahead

combination for the final adder. This results in a speed of

$$2\lceil \log_{3/2}(n) \rceil + \lceil \log_r n \rceil \qquad (2.7)$$

To illustrate this in terms of numbers, assume r=4 (typical on-chip fan-in) and n=16 bits. The lower bound for addition and multiplication is then three gate delays, whereas the conditional-sum addition is eight gate delays and the Booth encoding/Wallace tree multiplication speed is 18 gate delays.

Why is practical addition much closer to the lower bound than practical multiplication? The answer is that the conventional binary data representation is idealized for addition and not multiplication. Brennan [2.77] states a principle explained by Winograd: any data representation permitting the lower bound of one operation to be approached cannot accomplish the same for the other. This principle is illustrated by the slide rule, which can perform multiplication (using logarithms) but not addition efficiently. This fact has prompted many researchers to look at multiplication implemented in different binary number system.

Two principal areas that have been studied are the redundant binary/signed digit (SD) multiplication [2.5,2.53-2.55] and residue number multiplication [2.8,2.55,2.56]. The residue number system, relying on cells composed of ROMS or PLAs is less promising than the signed digit representation for large wordlength multiplication. In a signed digit multiplier, because of the redundancy property of the binary numbers, parallel addition of two n-bit numbers can be performed in a constant time independent of n without any carry propagation; n-bit multiplication can therefore be carried out in a time proportional to log (n). In a n x n-bit multiplication [2.53], the partial products are added pairwise by means of a tree of redundant binary adders. The main drawback of this approach is the need to convert the conventional binary representation to

the SD representation and vice-versa - this contributes a significant percentage to the total multiplication time. However, because its computation time is proportional to the logarithm of the wordlength this approach has great potential for future practical implementations of large, high-speed multipliers. Compared to the straight CSA multiplier, it was reported that it is about 4 times faster for 32-bit multiplication and about 7 times for 64-bit multiplication.

2.4. Approaches to improve the speed

With the trend towards single chip digital signal processors and the growing demand for higher speed and better efficiency realizations of digital multipliers, further work is needed in this area. While the Booth multiplier is attractive for single chip (especially in CMOS) implementation, the employment of this technique for bipolar, single chip digital signal processors is inhibited to a large extent by high power consumption and silicon area. This is evident from the HP1X 16 x 16-bit multiplier which consumes a power of around 4W with a silicon area of approximately 5 x 5 mm^2. The large number of digital signal processing chips that are in the market today are mostly designed in CMOS technology, but with the ever growing demand of real-time digital signal processing applications especially in image processing, future designs would need to be implemented in the faster bipolar ECL technology. As discussed in Chapter 1, the alternative approach of using GaAs technology to enhance the speed using existing proven architectures is still constrained by the low practical level of integration and poor yield of large GaAs chips, although GaAs parallel multipliers have been reported recently [2.64-2.66]. On the other hand, improved bipolar processes have evolved recently [2.67-2.68] which approached VLSI level of integration

as a consequence of the smaller and less power-hungry transistors. Thus further work is needed to investigate a more efficient realization of digital multipliers in bipolar technology which would ultimately lead to more powerful single chip digital signal processors.

It appears that two of the three methods discussed in section 2.2 i.e. faster addition of partial products and the use of fast final adders could be investigated further to improve the speed of the multiplier. The Booth algorithm cannot be effectively extended to recode more than three bits as it is fundamentally limited by the inability to realise higher multiples of the multiplicand which can be generated by simple shifting. The use of faster final adders to add the sum-carry pairs is expensive in terms of power and silicon area. In fact about 30% of the total power consumption and 25% of the total silicon area in the HP1X multiplier are taken up by the fast adders. Various architectures of fast adders have been exhaustively investigated [2.21-2.24] since the advent of integrated circuits and it seems that little progress will be made in this area. This is added by the fact that Winograd's theoretical lower bound on addition is nearly achievable with the combination of conditional-sum and carry-look-ahead algorithm. Furthermore, the use of fast adders gives irregularities and increases design time. Indeed, the major objective of VLSI designers today is to maximize regularity wherever possible.

Multiplier architectures that have been proposed over the years are either refinements of the carry-save approach or that based on a different number representation [2.1-2.13]. A majority of these techniques still make use of the 3-input full-adder cell. One of the major factors which limits the multiplication speed is that only one row of partial products is added at a time by a full-adder. What is needed, in order to increase the speed significantly is to look at ways of adding

more rows of partial products at one time i.e. to add not just three bits, but four or more bits of the same weight. This subject has been extensively covered by Dadda [2.25-2.27], who investigated the use of logic blocks called parallel (p,q) counters to speed up the addition of partial products.

2.4.1 Generalized counters

Dadda [2.25,2.17] first introduced the notion of a (p,q) counter as a combinatorial network which receives p bits of equal weight as input and produces a q-bit word corresponding to their sum as output. A full-adder, for example would be termed a (3,2) counter.

The value of the output is

$$v = \sum_{i=0}^{p-1} b_i \tag{2.8}$$

where b_i denotes the binary value of bit i of the input column and v denotes the value of the q-bit output. The number of output bits must be sufficient to represent all possible sums of p bits and hence

$$2^q - 1 \geq p \tag{2.9}$$

This class of counters may be extended to include counters which receive several successively weighted input columns and produce their sum, taking the weighting into account. We denote counters of this type as

$(p_{k-1}, p_{k-2}, \ldots\ldots, p_0, q)$ counters

where k is the number of input columns, p_i is the number of input bits in the column of weight 2^i, and q is the number of bits in the output word.

The value of the output is

$$v = \sum_{i=0}^{k-1} \sum_{j=0}^{p_i - 1} b_{ij} 2^i \tag{2.10}$$

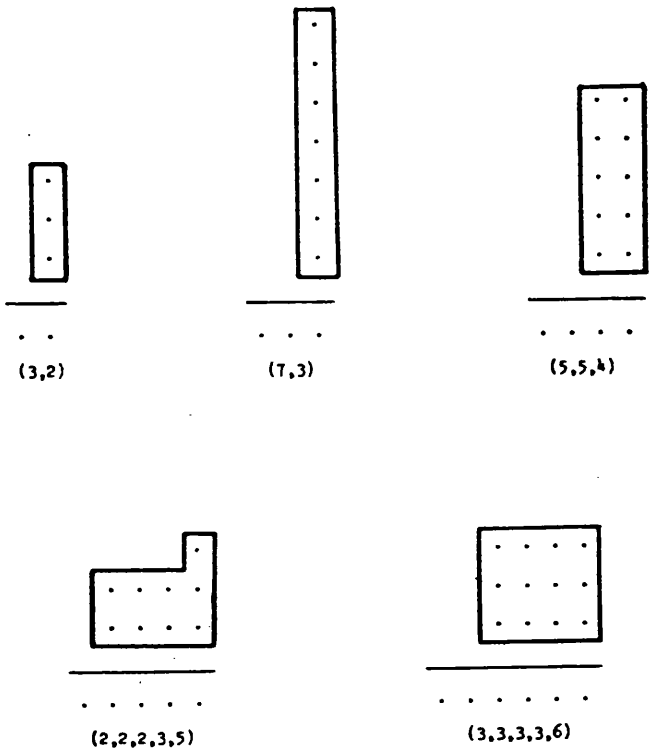


Figure 2.5. Some generalized counters [2.17].

where b_{ij} denotes the value of bit j in column i. Again q is subject to the constraint that

$$2^q - 1 \geq \sum_{i=0}^{k-1} p_i 2^i \tag{2.11}$$

Examples of several counters are shown in Figure 2.5. The effect of a series of counters acting on adjacent sets of input columns is shown in Figure 2.6. The inputs to the counters are shown first, followed by an equivalent representation of the output. The counters shown (with the exception of the (2,2,2,3,5)) all have complete utilization. Complete utilization of counters is not necessary but it is desirable.

Equal column counters are a convenient tool for reducing the regular portion of a matrix and are thus more useful. The regularity of these counters permits us to make the following observation, which are not

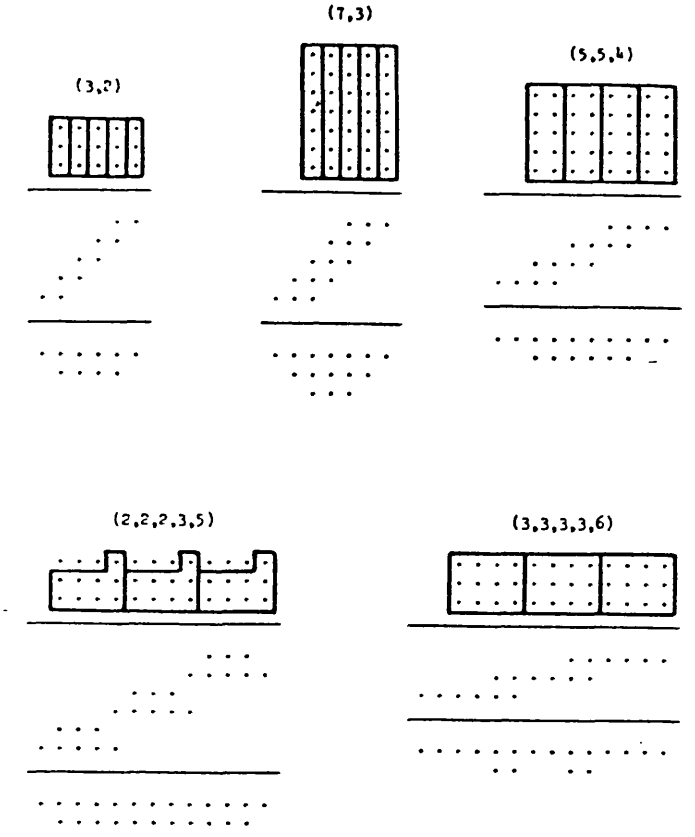


Figure 2.6. Effect of a series of adjacent counters [2.17].

necessarily true of unequal column counters. Equal column counters consume a rectangular matrix of k columns by r rows where

$$r = p_{k-1} = p_{k-2} = \ldots\ldots = p_0$$

The outputs align themselves such that no more than $\lceil q/k \rceil$ outputs contribute to a given column. Hence the number of rows s of the output matrix is simply

$$s = \lceil q/k \rceil \qquad (2.12)$$

The number of resultant output rows has a direct bearing on the number of stages of counters needed to reduce a large matrix. The less the number of stages required the higher the speed, so it is desirable that the number of output rows be small.

If we let

$$v_r = 2^k-1$$

denote the maximum possible value of a single input row and

$$v_0 = 2^q-1$$

denote the maximum representable output value the constraint on the number of input rows may be expressed as

$$r \leq \frac{v_0}{v_r} = \frac{2^q-1}{2^k-1} \qquad (2.13)$$

If q is not divisible by k, the output matrix will be somewhat sparse. It is advantageous to produce a matrix of uniform height since it can be reduced by a single counter type.

These considerations lead us to the notion of a maximally efficient counter as one which produces a uniform output matrix while consuming the largest possible regular portion of the input matrix i.e. it should have equal columns with the largest possible number of rows.

Disregarding cases where strings of counters may be stacked and skewed as in the case previously mentioned, this means that the number of output columns should be a multiple of the number of input columns, or

$q = sk$.

The counter will consume the largest possible portion of the matrix when

$v = rv_r$.

2.4.2 Number of levels for reduction

A matrix r bits high can be reduced to one s bits high through the use of one level of counters. We now consider the number of levels of counters required to reduce a matrix more than r bits in height to one of s bits as this will determine the speed of the multiplication operation. Denote the maximum height of the matrix that may be reduced to s bits

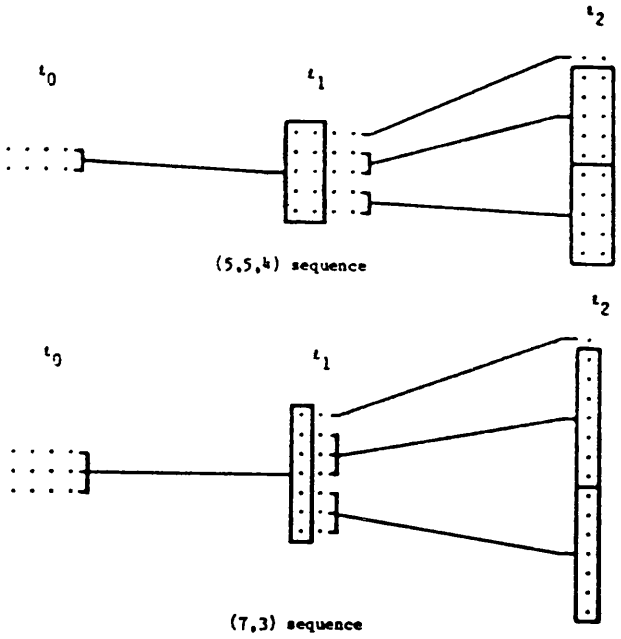


Figure 2.7. Examples of multi-level reduction [2.17].

using j levels of counters as l_j. Note $l_0 = s$; $l_1 = r$. Knowing l_j we may determine l_{j+1} by observing that the l_j bits represent the output of a stack of $\lfloor l_j/s \rfloor$ counters plus a residue of (l_j) mod. s bits which were not reduced by counters (Figure 2.7).

The $\lfloor l_j/s \rfloor$ counters each consume r bits so

$$l_{j+1} = r\lfloor l_j/s \rfloor + (l_j) \bmod s \qquad (2.14)$$

For a (5,5,4) counter the maximum reduction height sequence is

2,5,11,26,65.....

and for a (7,3) counter it is

3,7,15,35,79,.....

A rough approximation of the first few terms of the sequence is

s, $s(r/s)$, $s(r/s)^2$, $s(r/s)^3$,.....

Hence the number of levels of counters required to reduce a matrix h bits high to s rows is roughly

$$\log_{(r/s)}(h) \quad \text{levels} \tag{2.15}$$

It is important to note that this reduction method is similar to that employed in the Wallace tree [2.18]. For an iterative array type of interconnections like the carry-save array, in general more stages of counters would be required which results in a slower speed.

Other sequences may be obtained by combining two or more types of counters to fit the requirements of a particular matrix. If there is a need to economize on the number of large counters, a slight improvement can be obtained by using smaller counters to reduce underutilized large counters wherever possible. The penalty of this technique is a higher complexity in terms of the interconnections resulting from the irregular use of counter sizes.

2.4.3 Implementation and synthesis of parallel counters

The (3,2) counter or full-adder is the commonest form of implementation of the parallel counters concept, and the use of such counters in parallel multipliers has been discussed in the preceding sections. Apart from the synthesis of fast parallel digital multipliers, parallel (p,q) counters has been shown in the past to be useful in multiple operand addition [2.78,2.79], in inner product computation [2.80], in fast digital correlators [2.81] and in merged arithmetic [2.57]. The class of parallel counters were generalised by Meo [2.51] to accept several columns of multiple operands.

Dadda [2.25] has considered and proposed some schemes for high-speed multipliers by employing higher order counters with the reduction technique described in the last section. The use of larger counters in generating the initial partial product matrix has also been thoroughly studied by Ferrari and Stefanelli [2.28]. The architectures proposed by

Dadda and Ferrari lack regularity in terms of the counter size and their interconnections which makes them inattractive for VLSI implementation. However, the practical realization of the higher order parallel counters, has been the main stumbling block in employing this approach for high-speed multipliers. The poor device technology at that time has hampered the use of higher order counters as they were found to be too complex and slow compared to the full-adder.

Direct synthesis of counters in conventional combinatorial logic is obviously feasible when the height of the input column p is small as in the case of the full-adder. As the height increases both the number of gates and propagation delay grows quite disproportionately and there would be a stage where it appears to be of no advantage in trying to sum a lot of bits together.

Many methods have been proposed to implement parallel counters. Three of the more obvious techniques are two-level network of gates [2.71], sequentially [2.48,2.49] and ROM [2.26]. To implement an N-input counter by using a two-level gate network requires (2^N-1) logic AND gates (each with N-inputs) followed by a reduction network that requires M logic OR gates (each with on the order of 2^N-1 inputs), where M is the number of outputs from the counter :

$$M = 1 + \lfloor \text{Log}_2\ N \rfloor \qquad (2.16)$$

The two-level gate network is clearly impractical for most values of N. The ROM approach requires a memory with 2^N words of M bits each to implement an N-input counter. Apart from the slow speed of ROM, the total storage requirement of $M.2^N$ bits is also impractical for most values of N. The technique of employing sequential circuits yields a counter that is too slow for most high-speed operations although it can handle an extremely large number of inputs N with little penalty in hardware cost

compared with other concepts.

Foster and Stockton [2.72] first proposed a method of implementing counters with a network of full-adders. Basically, their design procedure involves grouping the counter inputs into sets of three lines each. Each of these sets of lines are reduced by a network of full-adders until one line of each weight remains. Analysis of this method yields a lower bound on the number of adder delay times δ for a N-input counter which is given by

$$\delta \geq \lfloor \mathrm{Log}_3 (N-1) \rfloor + \lfloor \mathrm{Log}_2 N \rfloor \qquad (2.17)$$

At most N full-adders are required to implement an N-input counter. Although this approach is more practical, the slower speed which results is inadequate for most high-speed arithmetic operations.

A variation of the full-adder network counter that uses full-adders and fast adders was developed by Swartzlander [2.71] which exhibits twice the speed of the former scheme. However, for parallel multiplier applications the technique is found to be too costly both in terms of hardware and power. Swartzlander also proposed an alternative implementation, a quasi-digital (i.e. hybrid) approach that uses analog summing to generate a voltage proportional to the count of 1s and it appears to be potentially faster than the strictly digital approaches. The disadvantage with this method besides the high hardware cost, is that as with all analog or partly analog networks, resistors may have to be trimmed to meet designed tolerances, stray capacitance and inductance must be minimized, and noise suppression may be difficult.

Noting the rapid increase in conventional hardware with p, Dadda suggested the use of threshold logic [2.69,2.70,2.85] for possible economy and speed. The logical definition of threshold gates is shown in Figure 2.8. Counters using threshold logic can be of the non-inverting or

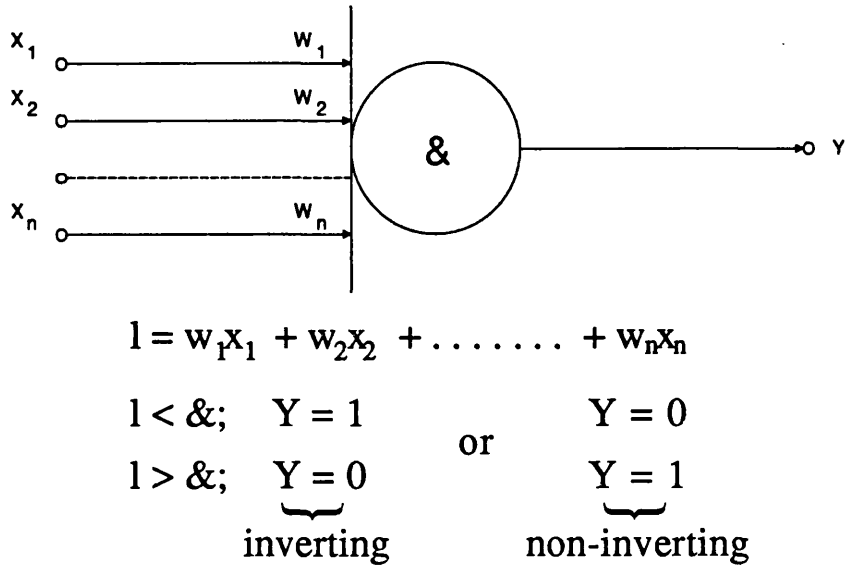


Figure 2.8. The logical definition of threshold gate [2.69].

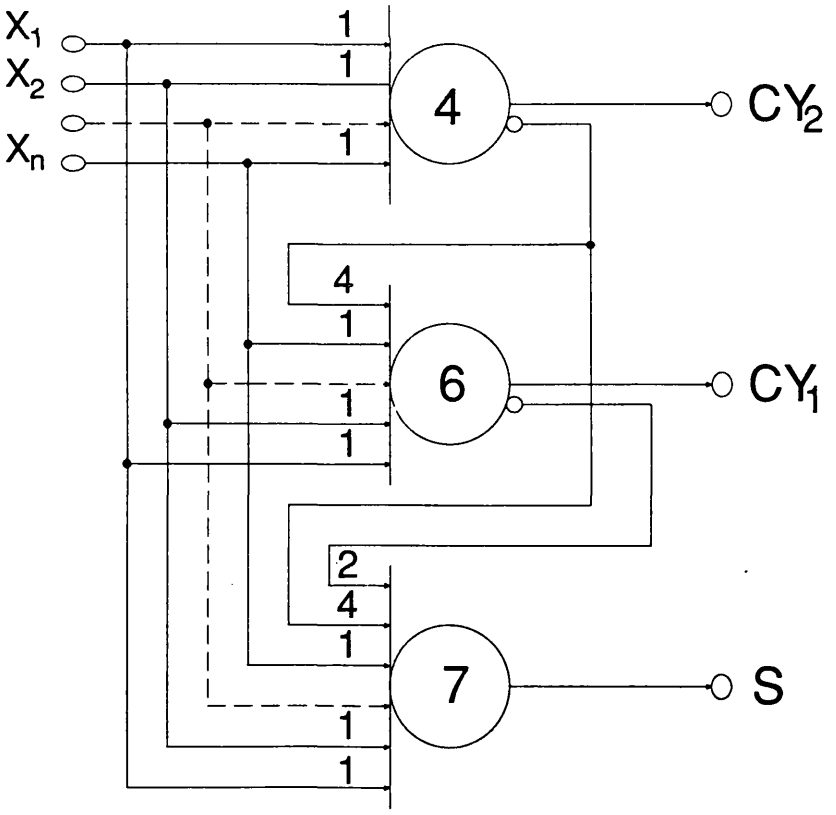


Figure 2.9. A (n,3) counter using inverting threshold logic [2.25].
(where $4 \leq n \leq 7$)

inverting type according to the type of threshold gate used. The advantage of using threshold logic is clearly illustrated in Figure 2.9, where a counter size (n,3), where $4 \leq n \leq 7$ could be realised by just adding extra lines of inputs and without any significant increase in complexity or propagation delay, assuming the threshold gates have a high fan-in input weight. The poor device technology at that time has often hampered this scheme as the circuit depends heavily on a fine control of the process parameters, primarily the resistor absolute values and ratios.

Traditional high-speed threshold gates are primarily of the current mode type [2.82]. However, the employment of this type of circuit in the synthesis of fast (p,q) counters has been constrained in the past by the high fan-in weight requirement of the counter's threshold gates as evident from Figure 2.9. Furthermore, better control on resistor absolute values and ratios was needed. Current [2.83,2.84] developed a four-valued threshold logic approach for implementing parallel counters which offers significant improvements in hardware cost and wiring complexity with a slightly better speed than the full-adder network. The approach, however is only of great advantage for large parallel counters of the order of $N \geq 31$.

2.5. Further work needed

It is not clear as to what the size of the input column p is before the combinatorial approach in realizing the counter becomes too complicated and with a delay that is too slow to be able to gain any significant increase in the overall multiplication speed. How much more complexity is imposed and that which is tolerable depends very much on the logical complexity and its fan-in and fan-out requirements. It is not

expected that large counters such as a (7,3) counter would have a delay comparable to that of a full-adder but for a reasonable size counter like a (4,3) or (5,3) counter the propagation delay would more likely to approach that of the full-adder. The maximum counter delay that can be tolerated before it is no longer useful in terms of the overall multiplication speed depends on its architecture, but ideally the counter delay should be equal to that of the full-adder. An architecture, say which is two times faster than that of the full-adder CSA would still be as fast as the CSA even if its counter delay is slower by a factor of 2. Thus, it is important that a counter that is optimized in speed and complexity must be balanced by an architecture that is faster and a complexity which is not significantly higher than the existing full-adder schemes. Any possible architectures need to address these issues.

The reduction technique on the partial product matrix described by Dadda gives a delay which is a function of the logarithm of the wordlength and is faster than the array technique, whose delay varies linearly with the wordlength. The application of this technique for single chip multipliers is however, already too complicated for counters of the full-adder size due to the irregular wiring. The problem is definitely worsened if higher order counters are employed and this is the main reason why the approach is not practical for VLSI implementation. Further work is therefore needed to investigate the use of the array approach for higher order counters since the advantages of the array in terms of regularity, interconnectivity and expandability would outweigh the speed advantage of Dadda's scheme.

With todays bipolar technology of smaller transistor size and lower power consumption and the evolution of new bipolar circuit techniques it is perhaps more practical now to realize multiplier-dedicated counters in

pure combinatorial logic to achieve the high-speed required. Multi-level current mode logic [2.62] would need to be fully explored again. The inherent advantages of EFL could be made full use of in building up the counters. EFL [2.63] offers an improvement in gate count for some logic functions whilst maintaining the speed-power product of ECL. Features such as high fan-in and fan-out and the ability to wire-OR the outputs could be employed efficiently. Since EFL and ECL are fully compatible in terms of logic levels, a mixture of them could be used where advantageous. Apart from that threshold logic, which depends heavily on resistor ratios or absolute values would benefit from todays improved processing technology which would mean a more efficient realization of the (p,q) counters in terms of speed and complexity. Thus further work is needed in these areas.

This project has considered the above issues. Architectures based on different size counters in an iterative array approach are evaluated in terms of speed, hardware costs and regularity of layout. The realization of counters in conventional combinatorial logic and threshold logic are reconsidered. These are described in subsequent chapters.

2.6. Summary

This chapter has reviewed the traditional and most commonly-used techniques for high-speed multipliers. They are compared on the basis of speed and their attractiveness for VLSI implementation.

The Wallace tree and Dadda's technique are shown to be faster than the carry-save array approach but are unattractive for VLSI implementation. Multiplier recoding techniques like the modified Booth algorithm have proved to be a cost-effective method of enhancing the speed of the multiplication process but are limited by the number of bits

that can be coded effectively. Fast adder architectures that are attractive for incorporation with the Booth encoder multiplier technique are assessed in terms of speed and hardware costs. Work done by Dadda on the concept of parallel (p,q) counters is introduced.

It is found that further investigations is needed in iterative array architectures based on higher order parallel (p,q) counters that are ideal for implementation in the fast silicon bipolar technology which would result in better efficiency realization of high-speed multipliers in single chip, real-time digital signal processors. With today's improved technology of smaller transistors and lower power consumption, a reconsideration of the conventional combinatorial logic and Threshold logic for realization of the counters deserves further attention.

REFERENCES

[2.1] C.P. Lerouge et al, " A fast 16 bit NMOS parallel multiplier ", IEEE Journal Solid-State Circuits, Vol. SC-19, No. 3, pp. 338-342, June 1984.

[2.2] M. Hatamian & G.L. Cash, " A 70-MHz 8-bit x 8-bit parallel pipelined multiplier in 2.5-um CMOS ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 4, pp. 505-513, August 1986.

[2.3] K. Washio et al, " 2.7ns 8 x 8-bit parallel array multiplier using sidewall base contact structure ", IEEE J. Solid-State Circuits, Vol. SC-22, No. 4, pp. 613-614, Aug. 1987.

[2.4] M. Uya, K. Kaneko & J. Yasui, " A CMOS floating point multiplier ", IEEE J. Solid-State Circuits, Vol. SC-19, No. 5, pp. 697-701, Oct. 1984.

[2.5] Y. Harata et al, " High-speed multiplier using a redundant binary adder tree ", Proc. IEEE ICCD, pp. 165-170, Oct. 1984.

[2.6] T.G. Noll et al, " A pipelined 330-MHz multiplier ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 3, pp. 411-416, June 1986.

[2.7] A.R. Hurson, B. Shirazi & S.H. Pakzad, " VLSI layout design of a systolic multiplier unit for 2's complement numbers ", Proc. IEEE ICCD, pp. 354-358, 1985.

[2.8] F.J. Taylor, " A VLSI residue arithmetic multiplier ", IEEE Trans. Computers, Vol. C-31, No. 6, pp. 540-546, June 1982.

[2.9] Y. Oowaki et al, " A sub-10-ns 16x16 multiplier using 0.6-μm CMOS technology ", IEEE J. Solid-State Circuits, Vol. SC-22, N0. 5, pp. 762-765, Oct 1987.

[2.10] S. Nakamura, " A single chip parallel multiplier by MOS technology ", IEEE Trans. Computers, Vol. 37, N0. 3, pp. 274-282, March 1988.

[2.11] N.R. Shanbhag & P. Juneja, " Parallel implementation of a 4x4-bit multiplier using modified Booths algorithm ", IEEE J. Solid-State Circuits, Vol. 23, No. 4, pp. 1010-1013, August 1988.

[2.12] W. Bucklen, J. Eldon, L. Schirm & F. Williams, " Single-chip digital multipliers form basic DSP building blocks ", EDN, PP.153-163, April 1, 1981.

[2.13] S. Waser & A. Peterson, " Real-time processing gains ground with fast digital multiplier ", Electronics, pp. 93-99, Sept. 29, 1977.

[2.14] S.L. Freeny, " Special-purpose hardware for digital filtering ", Proc. IEEE, pp. 633-648, 1975.

[2.15] A. Habibi & P.A. Wintz, " Fast Multipliers, " IEEE Trans. Computer, Vol.C-19, pp. 153-157, Feb. 1970.

[2.16] S. Waser, " High speed monolithic multiplier for real time digital signal processing, " Computer, Vol. 11, no. 10, pp. 19-29, Oct. 1978.

[2.17] W.J. Stenzel, W.J. Kubitz & G.H. Garcia, " A compact high-speed parallel multiplication scheme, " IEEE Trans Comput., Vol. C-26, pp. 948-957, Oct. 1977.

[2.18] C.S. Wallace, " A suggestion for parallel multipliers, " IEEE Trans. Electron. Comput. Vol. EC-13, pp 14-17, Feb. 1964.

[2.19] A. D. Booth, " A signed binary multiplication technique, " Quart. J. Mech. Appl. Math., Vol. 4, part 2, 1951.

[2.20] L.P. Rubinfield, " A proof of the modified Booth's algorithm for multiplication ", IEEE Trans. Computers, pp. 1014-1015, Oct. 1975.

[2.21] J. Sklansky, " An evaluation of several two-summand binary adders ", IRE Trans. Electronic Computers, pp. 213-226, June 1960.

[2.22] J. Sklansky, " Conditional-sum addition logic ", IRE Trans. Electronic Computers, pp. 226-231, June 1960.

[2.23] R. Gosling, Design of arithmetic units for digital computers, Wiley.

[2.24] C.F. Cavanagh, "Digital computer arithmetic, Design and implementation ", Mcgraw Hill 1986.

[2.25] L. Dadda, " Some schemes for parallel multipliers, " Alta Frequenza, Vol. 34, pp. 349-356, Mar. 1965.

[2.26] L. Dadda, " On parallel digital multipliers ", Alta Frequenza, pp. 574-580, Oct. 1976.

[2.27] L. Dadda, " Composite parallel adders ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 942-946, October 1980.

[2.28] D. Ferrari & R. Stefanelli, " Some new schemes for parallel multipliers, " Alta Frequenza, Vol. 38, pp. 843-852, Nov. 1969.

[2.29] O. L. MacSorley, "High-speed arithmetic in binary computers," Proc. IRE, Vol. 49, pp. 67-91, Jan. 1961.

[2.30] S. Winograd, " On the time required to perform addition ", J. Association for Computing machinery, Vol. 12, No. 2, pp. 277-285, April 1965.

[2.31] P.M. Spira, " Computation times of arithmetic and Boolean functions in (d,r) circuits ", IEEE Trans. Computers, Vol. C-22, No. 6, pp. 552-555, June 1973.

[2.32] P.M. Spira, " On the time necessary to compute switching functions ", IEEE Trans. Computers, pp. 104-105, Jan. 1971.

[2.33] T. liu, K.R. Hohulin, L. Shiau & S. Muroga, " Optimal one-bit full adders with different types of gates ", IEEE Trans. Computers, Vol. C-23, No. 1 pp. 63-69, Jan. 1974.

[2.34] S. Nakamura, "Algorithms for iterative array multiplication," IEEE Transactions on computers, Vol. C-35, No.8, August,1986.

[2.35] D.P. Agrawal, " High-speed arithmetic arrays ", IEEE Trans. Computers Vol. C-28, No. 3, pp. 215-224, March 1979.

[2.36] A. Dhurkadas, " Faster parallel multiplier ", Proc. of the IEEE, Vol. 72, No. 1, Jan. 1984.

[2.37] R. De Mori, " Suggestion for an IC fast parallel multiplier ", Electronics Letters, pp. 50-51, Vol. 5, No.3, 6th Feb. 1969.

[2.38] H.H. Guild, " Fully iterative fast array for binary multiplication and addition ", Electronics Letters, pp. 263, Vol. 5, No. 12, 12th June 1969.

[2.39] J.C. Hoffman, B. Lacaze & P. Csillag, " Iterative logical network for parallel multiplication ", Electronics Letters, pp. 178, Vol. 4, No. 9, 3rd May 1968.

[2.40] K.J. Dean, " Logical circuits for use in iterative arrays ", Electronics Letters, pp. 81-82, Vol. 4, No. 5, 8th March 1968.

[2.41] J. Deverell, " Pipeline iterative arithmatic arrays ", IEEE Trans. Computers, pp. 317-322, March 1975.

[2.42] S. D. Pezaris, " A 40ns 17-bit by 17-bit array multiplier ", IEEE

Trans. Computers, pp. 442-447, April 1971.

[2.43] J. C. Majithia & R. Kitai, " An iterative array for multiplication of signed binary numbers ", IEEE Trans. Computers, pp. 214-216, Feb. 1971.

[2.44] R. De Mori, " A parallel structure for signed-number multiplication and addition ", IEEE Trans. Computers, pp. 1453-1454,Dec. 1972.

[2.45] I.D. Deegan, " Cellular multiplier for signed binary multipliers ", Electronics Letters, Vol. 7, No. 151, pp. 436-437, 29th July 1971.

[2.46] C.R. Baugh & B.A. Wooley, " A two's complement parallel array multiplication algorithm, " IEEE Trans. Computer, Vol. C-22, pp.1045-1047,Dec. 1973.

[2.47] J.A. Gibson & R.W. Gibbard, " Synthesis and comparison of two's complement parallel multipliers ", IEEE Trans. Computers, pp. 1020-1027, Oct. 1975.

[2.48] A. Svoboda, " Adder with distributed control ", IEEE Trans. Computers, Vol. C-19, No. 8, pp. 749-751, August 1970.

[2.49] S. Singh & R. Waxman, " Multiple operand addition and multiplication ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 113-120, Feb. 1973.

[2.50] N.G. Kingsbury, " High speed binary Multiplier ", Electronics Letters, Vol. 7, No. 10, pp. 277-278, 20th May 1971.

[2.51] A.R. Meo, " Arithmetic networks and their minimization using a new line of elementary units ", IEEE Trans. Computers, Vol. C-24, No. 3, pp. 258-280, March 1975.

[2.52] T.A. Brubaker & J.C. Becker, " Multiplication using logarithms implemented with read-only memory ", IEEE Trans. Computers, Vol. C-24, No. 8, pp. 761-765, August 1975.

[2.53] N. Takagi, H. Yasuura, S. Yajima, " High-speed VLSI multiplication algorithm with a redundant binary addition, " IEEE Trans. on Computers, Vol. C-34, No. 9, Sept. 1985.

[2.54] A. Avizienis, " Signed-digit number representations for fast parallel arithmetic", IRE Trans. Electron. Comput., Vol. EC-10, pp. 389-400, Sept. 1961.

[2.55] H. L. Garner, " A survey of some recent contributions to computer arithmetic ", IEEE Trans. Computers, Vol. C-25, No. 12, pp. 1277-1282,1976

[2.56] F.J. Taylor, " Residue arithmetic: A tutorial with examples ", IEEE Computer, pp. 50-62, May 1984.

[2.57] E.E. Swartzlander, " Merged arithmetic ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 946-950, October 1980.

[2.58] N. Takagi & S. Yajima, " On-line error-detectable high-speed multiplier using redundant binary representation and three-rail logic ", IEEE Trans. Computers, Vol. C-36, No. 11, pp. 1310-1317, Oct. 1986.

[2.59] H. Ling, "High-speed binary adder ", IBM J. Res. & Develop., Vol. 25, No. 3, pp. 156-166, May 1981.

[2.60] K. Kostantinides, R. Kaneshiro & G. Baldwin, " A 16 x 16 bit GaAs parallel multiplier, Preliminary report: Architectural considerations ", HPlabs, Measurements and Systems Lab., System Tech. Dept., Palo Alto, California.

[2.61] S. Winograd, " On the time required to perform multiplication ", J. of the association for Computing Machinery, Vol. 14, No. 4, pp. 793-802, Oct. 1967.

[2.62] U. Priel & J.W. Hively, " An integrated current mode full-adder ", Digest of Int. Solid-State Circuits Conf., pp. 108-109, Feb. 1967.

[2.63] Z.E. Skokan, " Emitter function logic - Logic family for LSI, " IEEE Journal of Solid-State Circuits, Vol. SC-8, No. 5, October, 1973.

[2.64] Y. Nakayama, K. Suyama, H. Shimuzu, N. Yokoyama, H. Onishi, A. Shibatomi & H. Ishikawa, " A GaAs 16 x 16 bit parallel multiplier, " IEEE J.Solid-State Circuits, Vol. SC-18, No.5, October 1983.

[2.65] F.S. Lee, G.R. Kaelin, B.M. Welch, R. Zucca, E. Shen, P. Asbeck, C.P. Lee, C.G. Kirkpatrick, S.I. Long & R.c. Eden, " A high-speed LSI GaAs 8 x 8-bit parallel multiplier, " IEEE J. Solid-State Circuits, Vol. SC-17, no. 4, pp. 638-647, Aug. 1982.

[2.66] K. Gonoi et al, " A GaAs 8 x 8-bit multiplier/accumulator using JFET DCFL", IEEE Journal Solid-State Circuits, Vol. SC-21, No. 4, August 1986.

[2.67] " Technology to watch, " Electronics, pp. 35-38, April 7, 1986.

[2.68] " Inside technology, " Electronics, pp. 67-77, June 25, 1987.

[2.69] D.Hampel & R.O. Winder, " Threshold logic ", IEEE Spectrum, pp. 32-39,May 1971.

[2.70] R.O. Winder, " Threshold logic will cut costs, especially with boosts from LSI ", Electronics, pp. 94-103, May 27 1968.

[2.71] E.E. Swartzlander, Jr., " Parallel Counters ", IEEE Trans. Computers, Vol. C-22, No. 11, pp. 1021-1024, Nov. 1973.

[2.72] C.C. Foster & F.D. Stockton, " Counting responders in an associative memory ", IEEE Trans. Computers, pp. 1580-1583, Dec. 1971.

[2.73] S. Dormido & M.A. Canto, " Synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-30, No. 9, pp. 699-703, September 1981.

[2.74] H. Kobayashi & H. Ohara, " A synthesizing method for large parallel counters with a network of smaller ones ", IEEE Trans.

Computers, Vol. C-27, No. 8, pp. 753-757, August 1978.

[2.75] S. Dormido & M.A. Canto, " An upper bound for the synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-31, No. 8, pp. 802-805, August 1982.

[2.76] D.G. Gajski, " Parallel Compressors ", IEEE Trans. Computers, Vol. C-29, No. 5, pp. 393-398, May 1980.

[2.77] J.F. Brennan, " The fastest time of addition and multiplication ", IBM research reports, Vol. 4, No. 1, 1968.

[2.78] I. T. HO & T. C. Chen, " Multiplie addition by residue threshold functions and their representattion by array logic " IEEE Trans. Computers, Vol. C-22, N0. 8 pp. 762-767, Aug. 1973.

[2.79] L. Dadda, " Multiple addition of binary serial numbers ", Proc. 4th Symposium on Computer Arithmetic, pp. 140-148, Oct. 1978.

[2.80] E.E. Swartzlander, B.K. Gilbert & I.S. Reed, " Inner product computers ", IEEE Trans. Computers, Vol. C-27, pp. 21-31, 1978.

[2.81] R.P. Cheung, K.W. Current & E.E. Swartzlander, " A very high-speed digital correlation technique ", in 1976 Nat. Telecommun. Conf. Rec., Nov. 1976, paper 52.4.

[2.82] J.J. Amodei & R.O. Winder, " An integrated threshold gate ", Digest of Int. Solid-State Circuits Conference, pp. 114-115, February 1967.

[2.83] K.W. Current & D.A. Mow, " Implementing parallel counters with four-valued threshold logic ", IEEE Trans. Computers, Vol. C-28, No. 3, pp. 200-204, March 1979.

[2.84] K.W. Current, " Pipelined binary parallel counters employing latched quarternary logic full adders ", IEEE Trans. Computers, Vol. C-29, No. 5, pp. 400-403, May 1980.

[2.85] L. Dadda, " Synthesis of threshold logic combinatorial networks, " Alta Frequenza, no. 3, pp. 224-28E-231, 1961.

[2.86] N.F. Benschop, " Layout compiler for variable array multipliers, Proc. Custom Integrated Circuits Conference, pp. 47-58, 1983.

CHAPTER 3

THE HP1X MULTIPLIER AND MPC TEST CHIP

3.1. Introduction

The HP1X 16 x 16-bit multiplier was designed in collaboration with Hewlett-Packard company at the IC design centre, Department of Electronic and Electrical Engineering, University College London, and fabricated in Santa Clara using their latest bipolar HP1X process. Also done in parallel was an MPC test chip composed of 6 ring oscillators which has been fabricated and tested successfully. This chip provides a fast and convenient route to obtain accurate propagation delay of full-adder cells, verifying the validity of transistor model parameters and allows us to make final 'tweaking' of the full-adder cells employed in the HP1X multiplier.

The appropriate SPICE [3.5] and logic (HILO) [3.6] simulations were first performed on the various logic functions of the multiplier to characterize their speed-power behaviour. The mask layout was done using a technology-independent interactive layout editor called CAESAR. It was first necessary to adapt CAESAR for the HP1X process by setting up the necessary technology files concerned. A design rule-checker called LYRA was also modified for the process.

3.2 The HP1X's architecture

The multiplier utilizes a carry-save full-adder array to generate 32 sum-carry pairs which are subsequently summed by blocks of high-speed final adders to form a 32-bit product in two's complement representation. Figure 3.1 shows a block diagram of the multiplier. This configuration is similar to the typical high-speed multiplier shown in

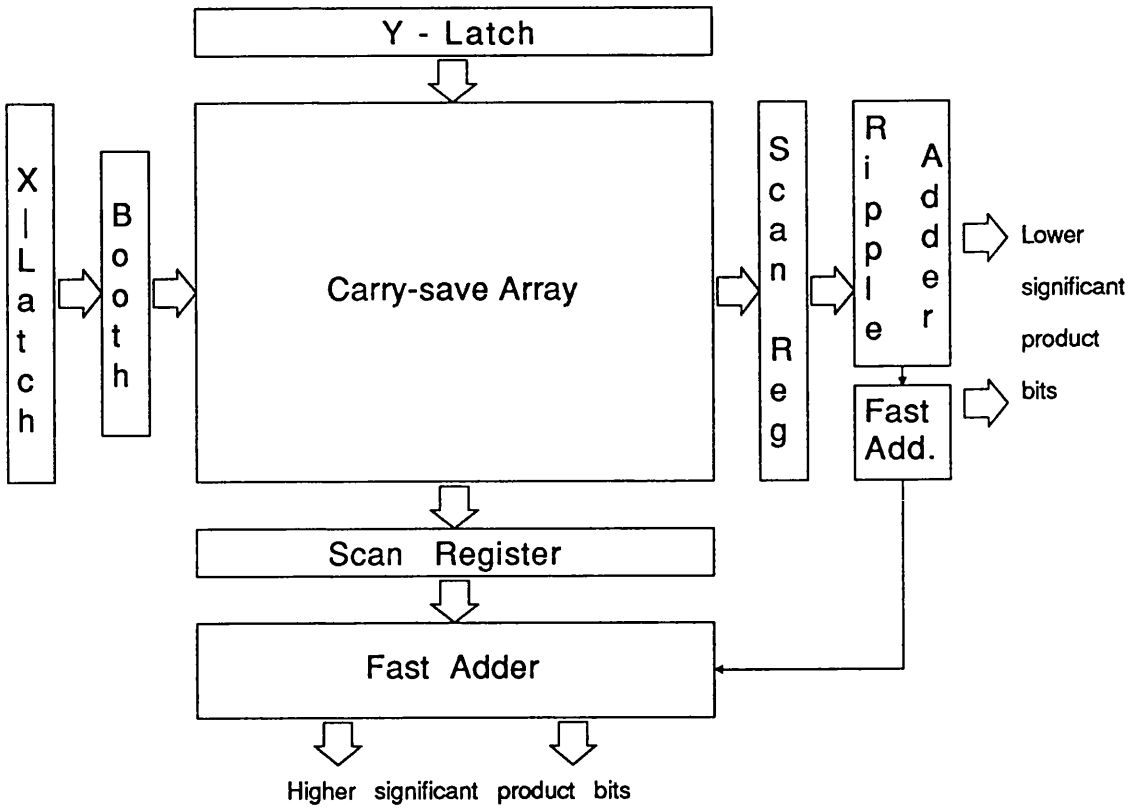


Figure 3.1. Block diagram of the HP1X multiplier.

Figure 2.3. The multiplier architecture enables simultaneous loading of the multiplier and multiplicand word. The X-latch and Y-latch are 16 and 17-bit input data latches for the multiplier and the multiplicand, respectively. The output drivers of the Y-latch are able to drive a total fan-out of 16 gates. The Booth encoder examines three consecutive bits of the multiplier to generate a shift, complement and zero signals. In essence the three bits are represented by the truth table of 2.1. These three signals are fed into the partial product generator in the carry-save array which is basically composed of a 17 x 8 matrix of alternating rows and columns of partial product generators and full-adders. The partial product generator takes the three outputs from Booth encoder and drives the full-adder which is located directly below

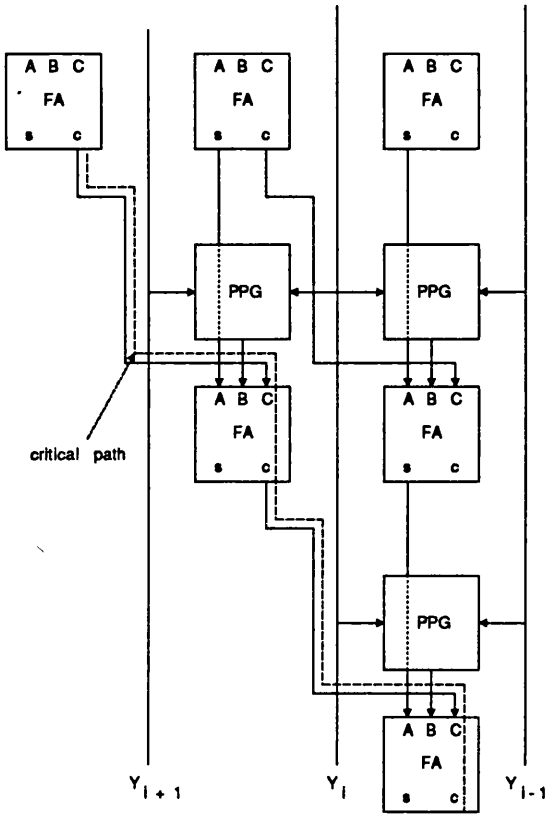


Figure 3.2. Part of the carry-save array.

it (Figure 3.2). The full-adder takes the output of the partial product generator, and the sum and carry output of the appropriate previous full-adders. It in turn generates a sum and a carry. The critical path of this array is the diagonal path which traverses through eight full-adders. The final adders are mainly composed of ripple adders to produce the lower significant product bits, and blocks of conditional-sum adders of different sizes for the higher significant product bits. In order to ease initial testing of the chip, a scan path register is placed between the carry-save array and the final adders. This feature enables independent testing of the two blocks.

The expected high operation speed of the multiplier (simulations show a total multiplication time of under 5ns for a total power

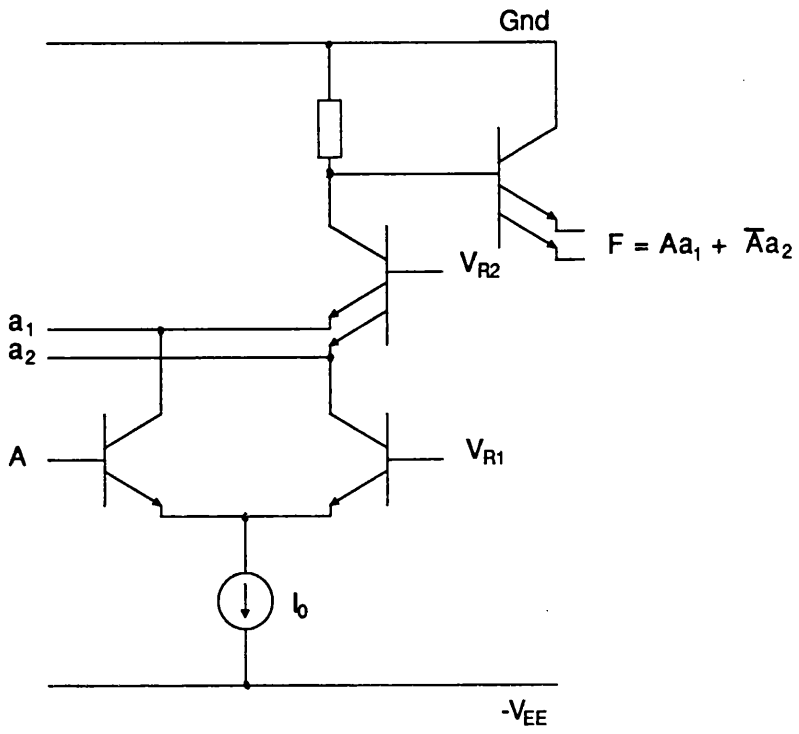


Figure 3.3. (a) A simple EFL structure [3.3].

dissipation of around 4W) owes very much to the bipolar technology and the circuit technique used. A 3-level ECL circuit approach [3.1] was employed to realize most of the logic functions of the multiplier with the exception of the final adders, which are built up by blocks of the simple emitter-function logic (EFL) [3.2-3.4] structure of Figure 3.3. Here the EFL cell was found to be attractive in compromising the speed, power and complexity of conditional-sum adders of different sizes.

Because one of the factors that determines the speed of the multiplier is the propagation delay of the full-adder, it becomes necessary to characterize its delay with respect to power and different load conditions. Also it is not easy to accurately extract (from the layout) the parasitics of bipolar circuits due to the complex geometry of bipolar transistors and its interconnections. Furthermore, since the HP1X

process was still at the development stage it was necessary to verify the validity of the SPICE model parameters evaluated so far. Thus a more accurate and convenient method of characterizing the on-chip delay than that estimated from simulation was required. A ring oscillator provides a fast and easy route to obtain accurate propagation delay of logic circuits on a chip as well as verifying the accuracy of the transistor model parameters. Such a chip was designed and tested; this is discussed next.

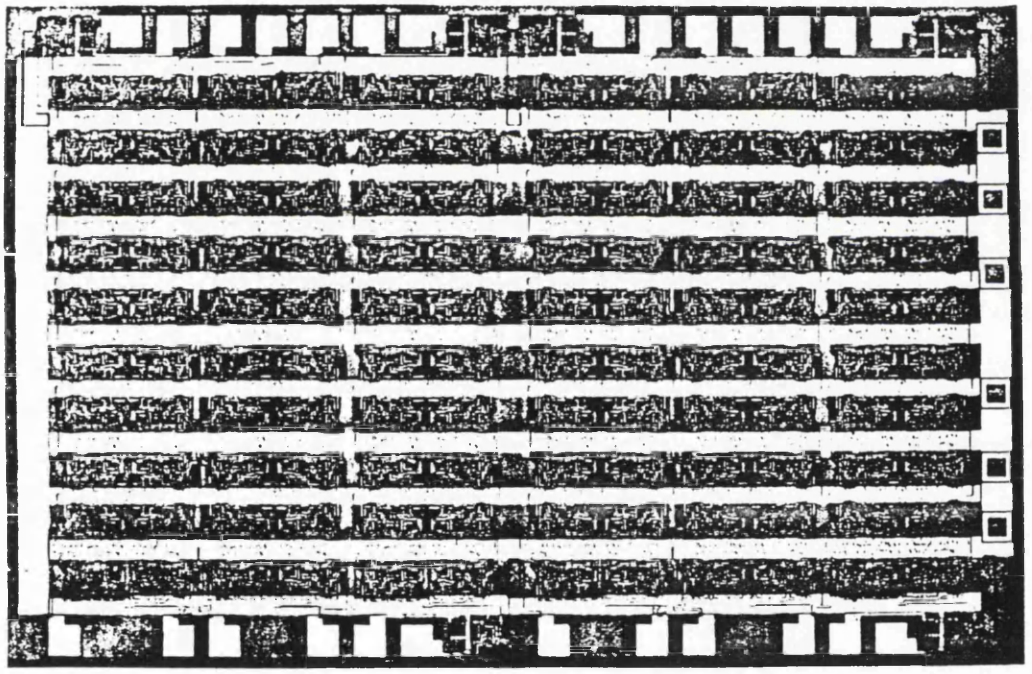


Figure 3.4. Microphotograph of MPC test chip.

3.3. The MPC test chip

The MPC test chip (Figure 3.4) comprises six 20-stage ring oscillators each of which is made up of 19 stages of non-inverting full-adders (S_0 = '0') and one full-adder (Figure 3.5) which can be switched between the non-inverting and inverting condition. The circuit

which sets the 19 stages in the non-inverting configuration is called SET1 (the schematic is shown in Figure 3.6(b)). The circuit that allows switching between the non-inverting and inverting conditions is called SET2 and is shown schematically in Figure 3.6(c).

If " IN " is left open (unconnected) then $\overline{B}$ = '0' and the full-adder will be inverting (states 3 and 7 of Table 3.1) causing the ring oscillator to be on. In order to stop the oscillation, B is set to '0' (states 1 and 5 of Table 3.1) which means " IN " will need to be above

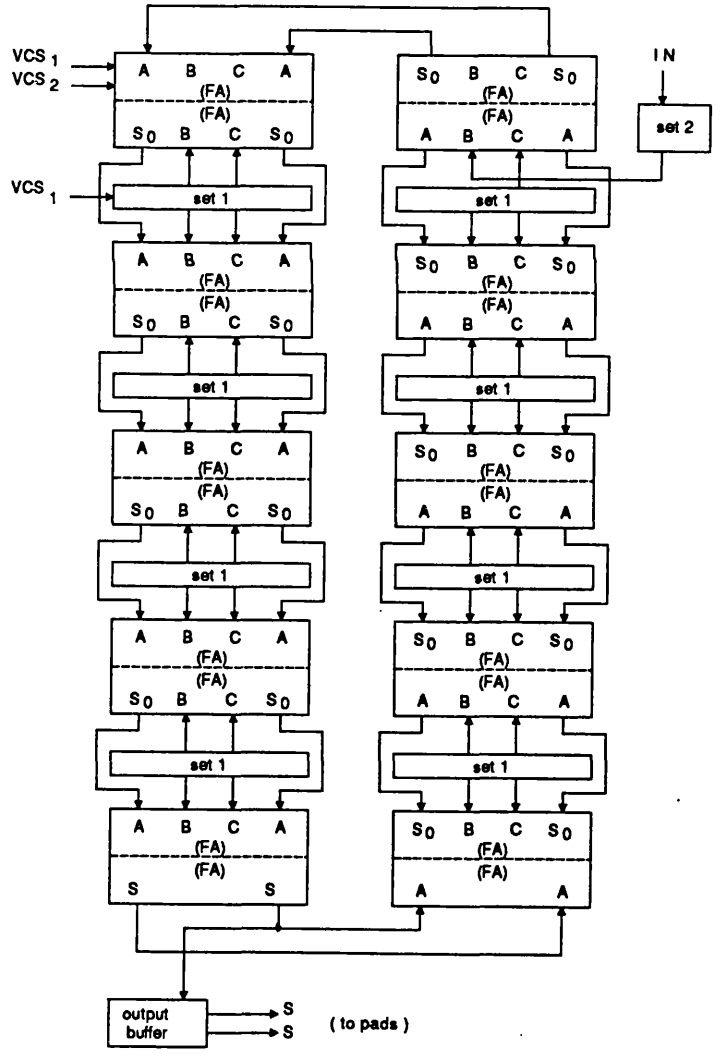


Figure 3.5. A 20-stage ring oscillator.

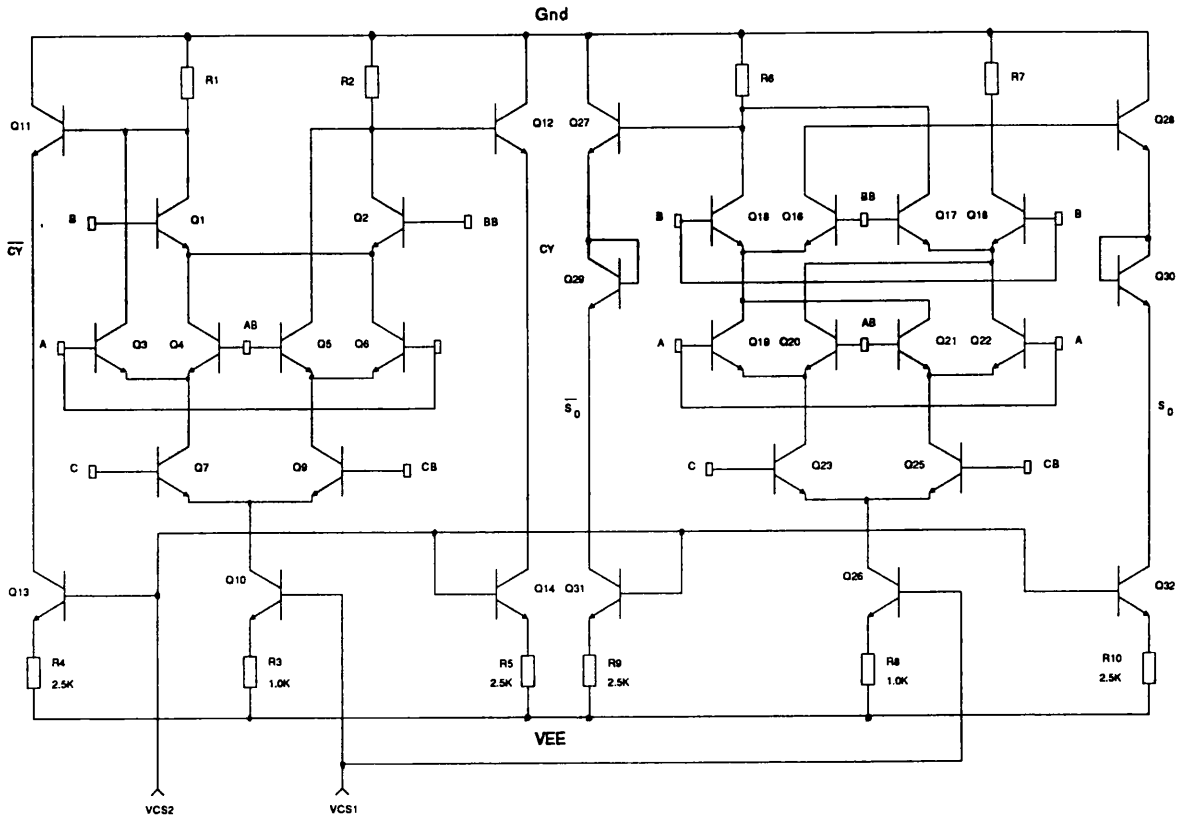


Figure 3.6(a). Full-adder schematic.

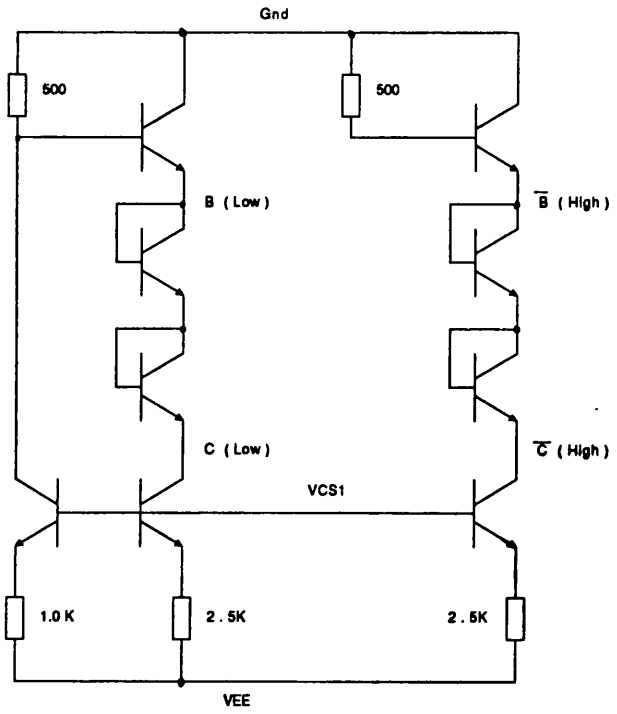


Figure 3.6(b). SET1 schematic.

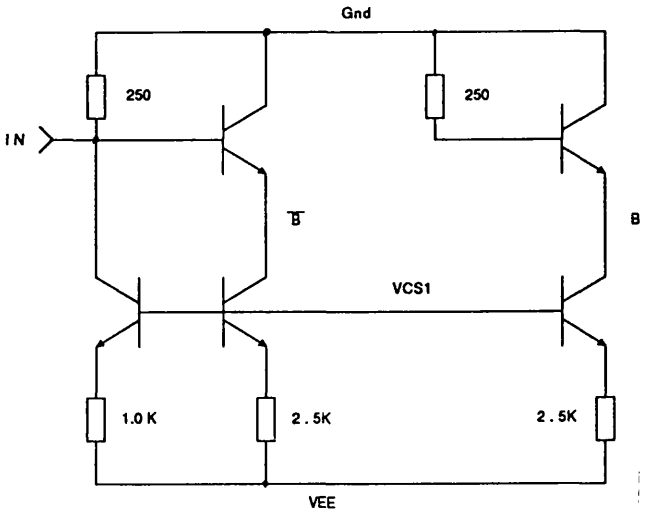


Figure 3.6(c). SET2 schematic.

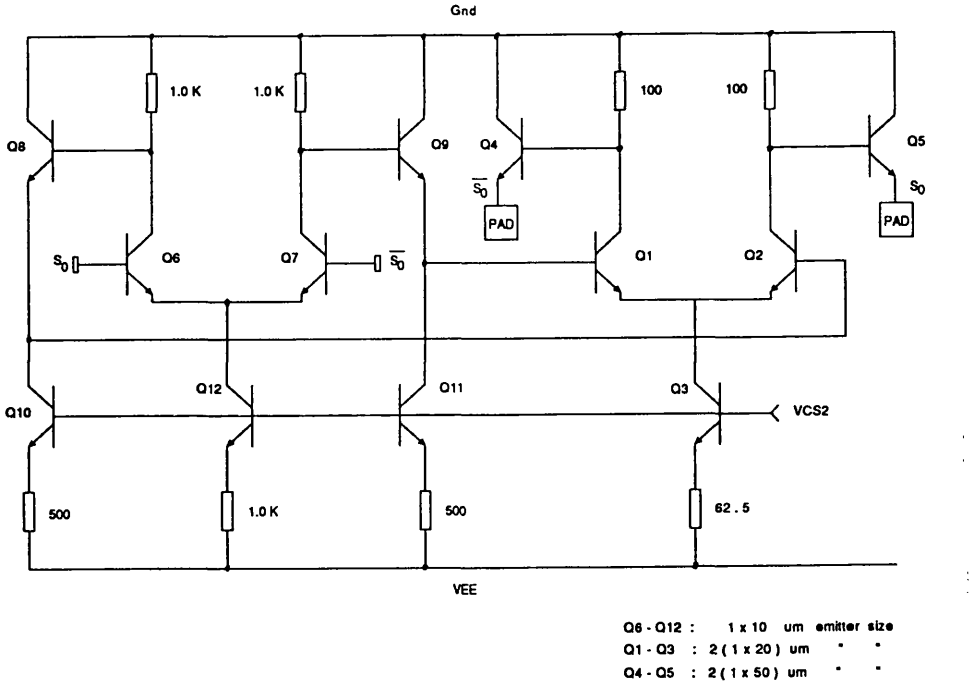


Figure 3.6(d). Output buffer schematic.

A	B	C	S_0	State
0	0	0	0	1
0	0	1	1	2
0	1	0	1	3
0	1	1	0	4
1	0	0	1	5
1	0	1	0	6
1	1	0	0	7
1	1	1	1	8

Table 3.1. Truth-table of the sum function of a full-adder.

ground by about 200mv.

Thus,

" IN " = -200mv osc. on.

" IN " = +200mv osc. off.

Vcs1 controls the current I_{OT} through the tail current sources and the input drivers (SET1 and SET2). Vcs2 controls the current I_{OE} through the emitter follower current sources and the output drivers.

3.3.1. Results

A packaged chip was first mounted and soldered on a copper-clad board to provide a good ground plane. Whilst keeping the supply voltage of -5.2v constant, each ring oscillator's frequency was measured as Vcs1 was varied for a fixed value of Vcs2. This procedure was repeated for different values of Vcs2. The results were as shown in Figures 3.7 to 3.12, where they are compared with those obtained from simulations.

As can be seen, the variation of full-adder delay against tail current is as expected. The usual V-shaped curve of the gate delay can be attributed to the variation of logic swing, transistor current gain β and

junction capacitances variations with collector current as Vcs1 and Vcs2 is varied. The leftmost points in the measured data graphs show the minimum value of tail current that would give adequate voltage swing to switch the gates reliably. The simulated results were obtained using four stages of full-adder cells with the appropriate SET1 and SET2 circuits. As can be seen, the simulated results exhibit the trend of the measured ones quite well although their absolute values do not agree reasonably close. As the HP1X technology is a relatively new process, better accuracy of the SPICE results could be expected with the availability of an improved SPICE model parameter values in the future.

The results obtained from the test chip were used for the final 'tweaking' of the full-adder cells in the HP1X multiplier. By changing the load resistor and/or transistor size and also by selecting a certain value of current source resistance, the full-adder cell could be tailored to meet a specific speed-power requirement.

Also, the simulated results were used as the basis for comparison of parallel counters described in subsequent chapters.

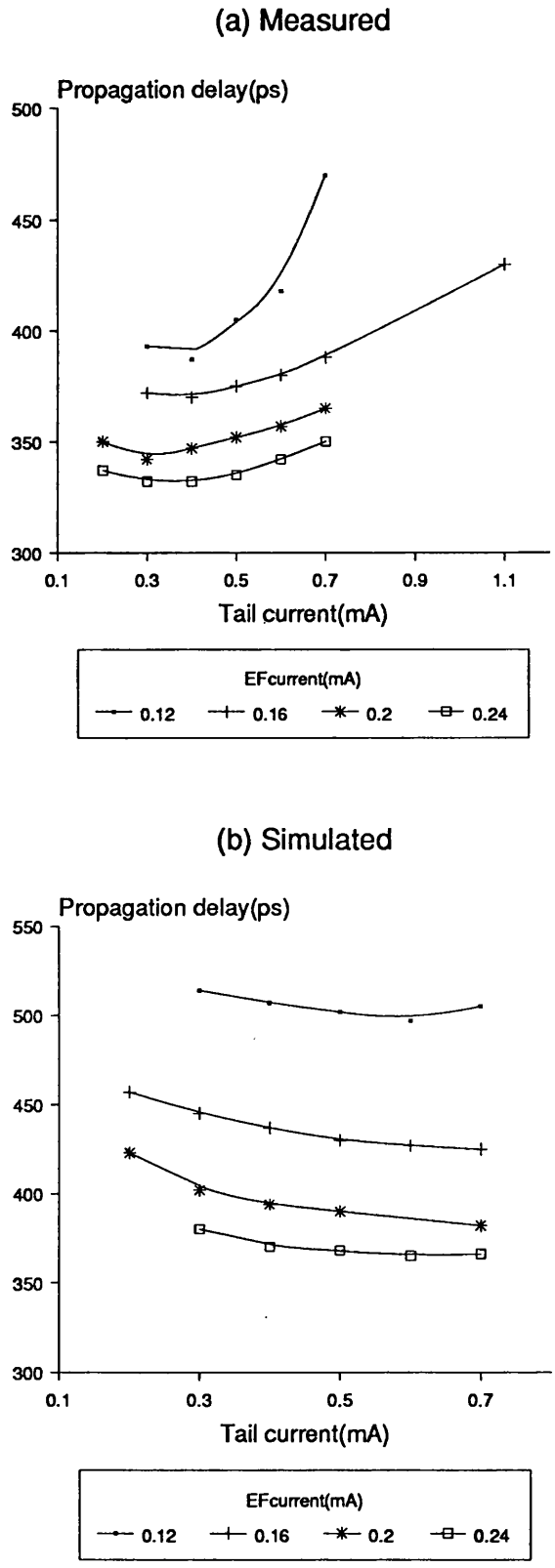


Figure 3.7. Propagation delay of ring oscillator 1;
Load resistance = 500Ω; Transistor emitter size = 1 x 10μm.
(a) Measured, and **(b)** Simulated.

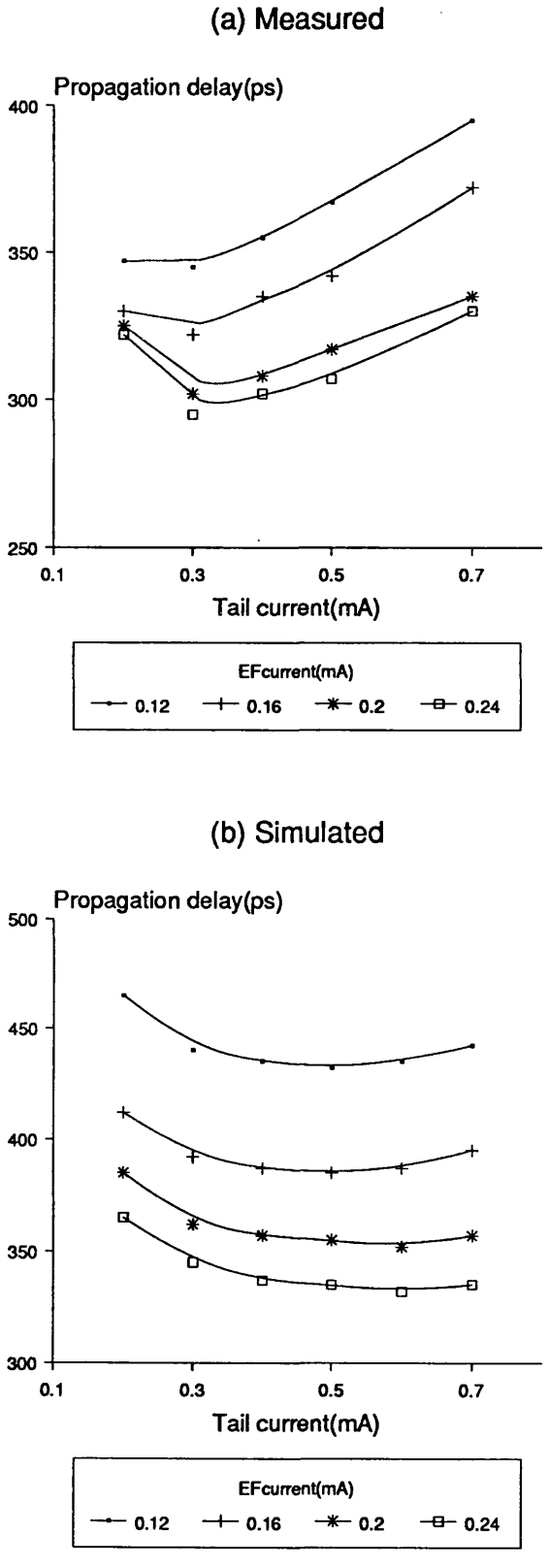


Figure 3.8. Propagation delay of ring oscillator 2;
Load resistance = 500Ω; Transistor emitter size = 1 x 5μm.
(a) Measured, and **(b)** Simulated.

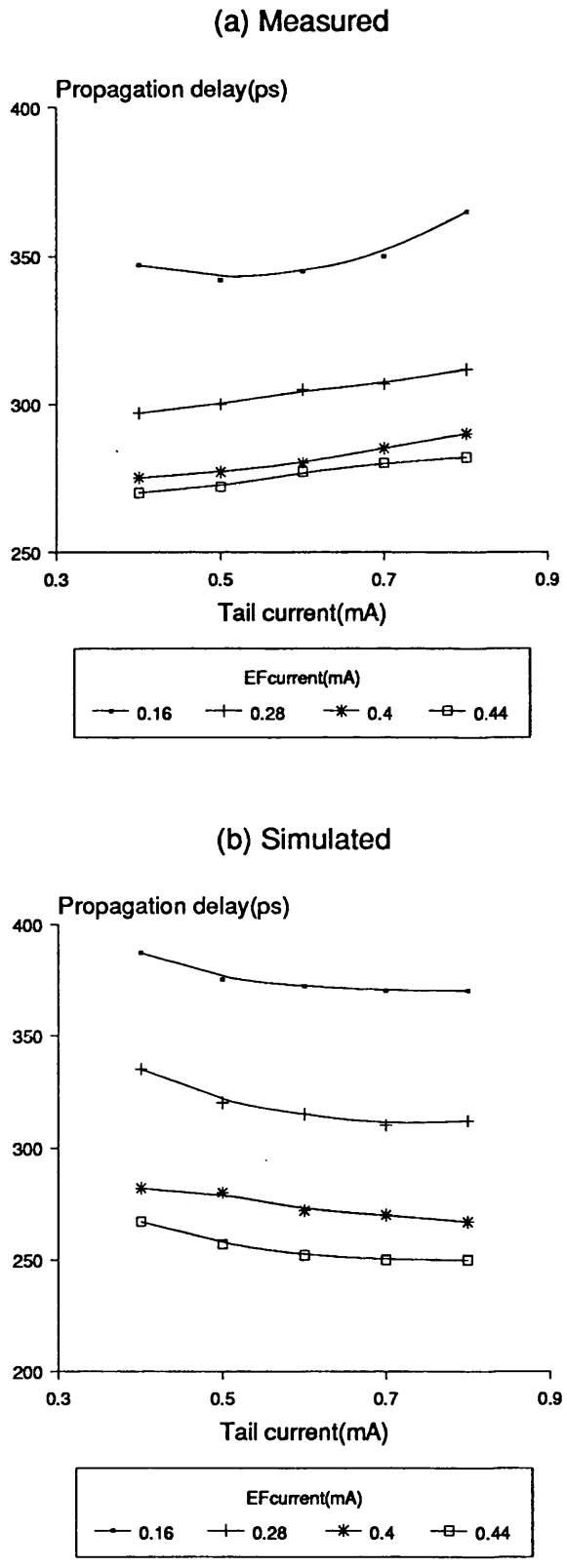


Figure 3.9. Propagation delay of ring oscillator 3;
Load resistance = 250Ω; Transistor emitter size = 1 x 10μm.
(a) Measured, and (b) Simulated.

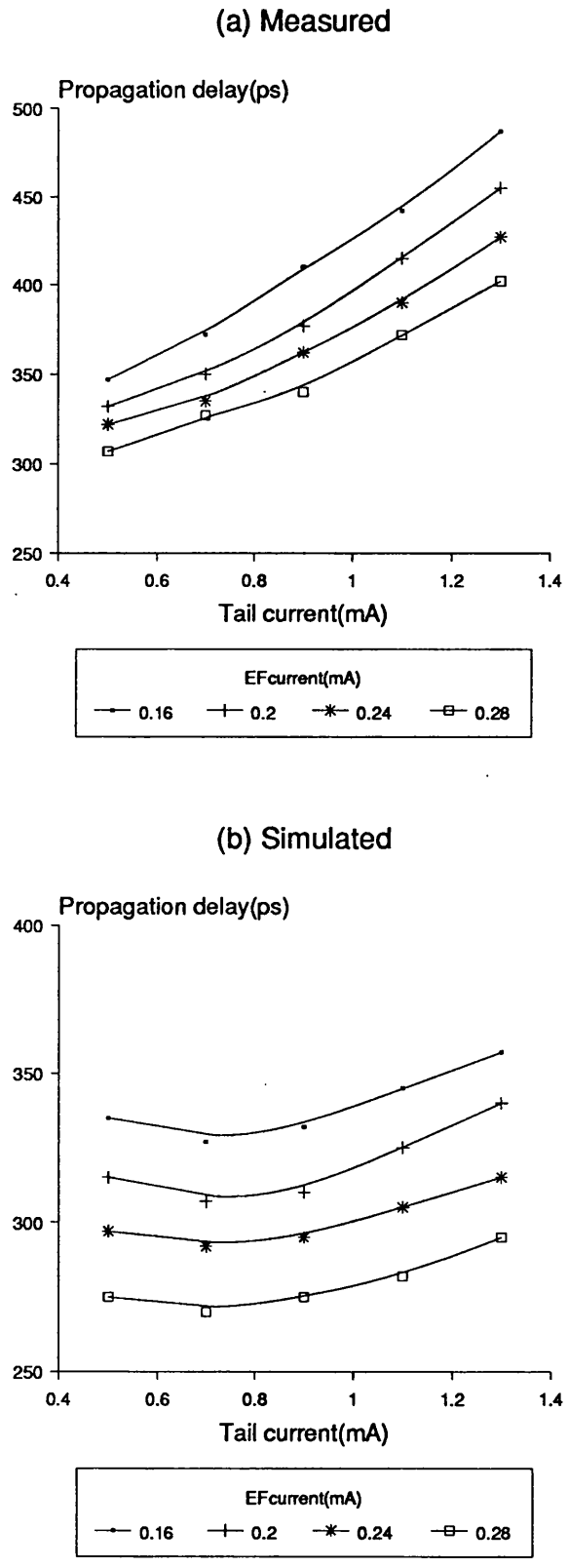


Figure 3.10. Propagation delay of ring oscillator 4;
Load resistance = 250Ω; Transistor emitter size = 1 x 5μm.
(a) Measured, and **(b)** Simulated.

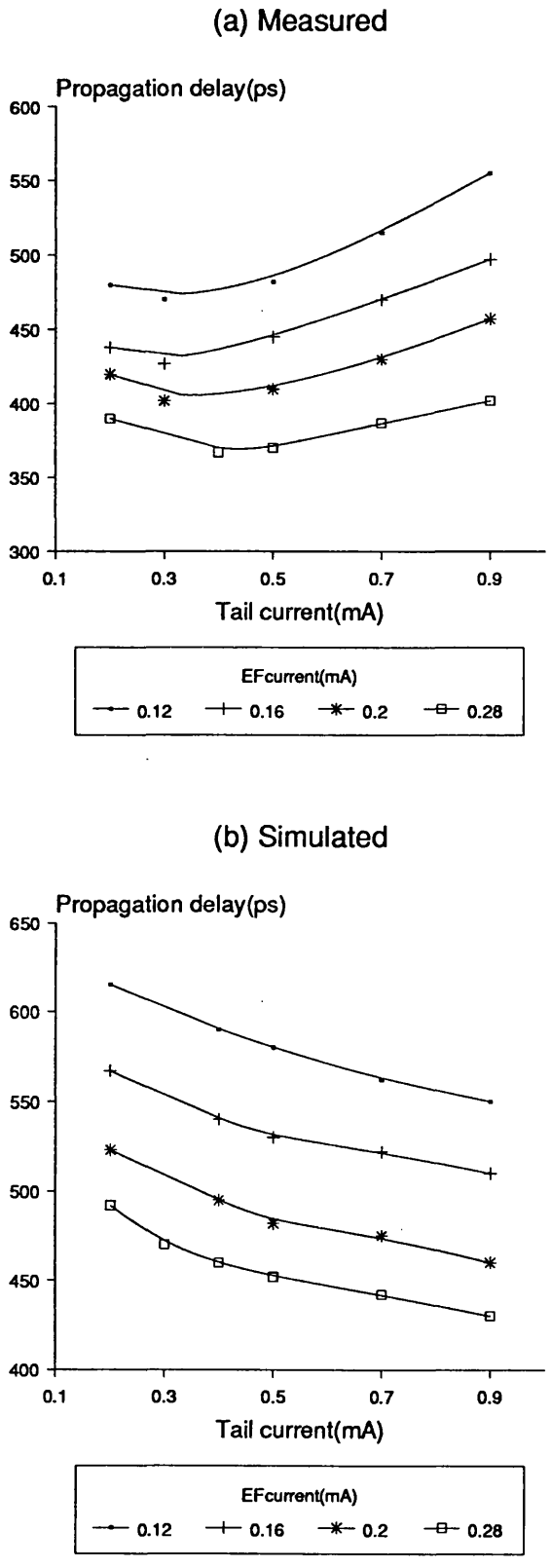


Figure 3.11. Propagation delay of ring oscillator 5;
Load resistance = 750Ω; Transistor emitter size = 1 x 10μm.
(a) Measured, and **(b)** Simulated.

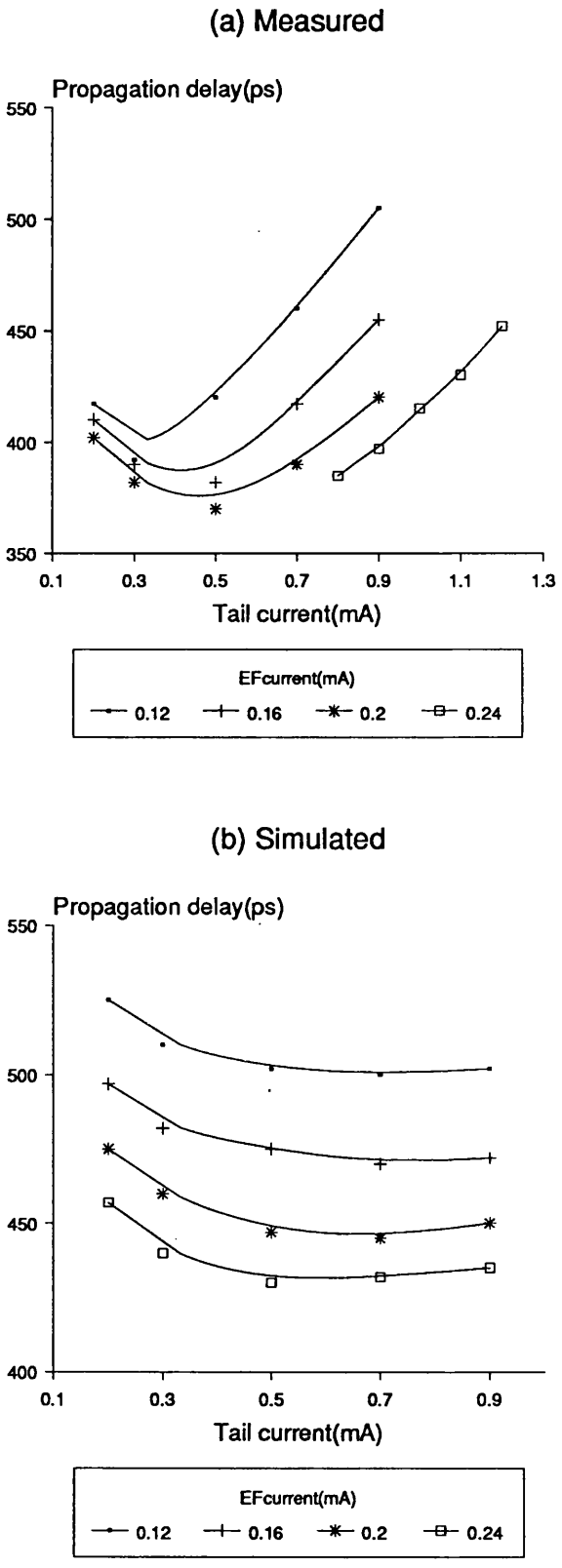


Figure 3.12. Propagation delay of ring oscillator 6;
Load resistance = 750Ω; Transistor emitter size = 1 x 5μm.
(a) Measured, and **(b)** Simulated.

REFERENCES

[3.1] U. Priel & J.W. Hively, " An integrated current mode full-adder ", Digest of Int. Solid-State Circuits Conf., pp. 108-109, Feb. 1967.

[3.2] Z. E. Skokan, " Emitter function logic - Logic family for LSI, " IEEE Journal of Solid-State Circuits, Vol. SC-8, No. 5, October, 1973.

[3.3] M. I. Elmasry, " Logic Design using EFL structures, " IEEE Transactions on Computers, pp. 952-956, Sept. 1976.

[3.4] M. I. Elmasry, " Digital bipolar integrated circuits, " John Wiley & sons, ISBN 0 471-05571-9.

[3.5] L.W. Nagel, " SPICE2 : A computer program to simulate semiconductor circuits ", Memo ERL-M520, University of California, Berkeley, CA, May 9, 1975.

[3.6] HILO-3 user manual, GenRad Inc., Revision 2, 15 June, 1985.

CHAPTER 4

ARCHITECTURES FOR ITERATIVE ARRAY MULTIPLICATION

4.1. Introduction

The multiplication operation is one of the most vital operations in many digital signal processing and computer applications. Because of the inherent complexity of the operation, the execution speed of multiplication tends to be the dominant factor in the entire processing time [4.1-4.4]. With the advent of VLSI technology, which brought about the reduced cost of fabrication per transistor on a chip, parallel algorithms for multiplication become increasingly important.

Schemes for parallel multiplication are roughly divisible into two classes - iterative array of full-adder cells [4.1,4.18-4.41] and generation of a matrix of partial product terms with subsequent reduction of the matrix [4.1,4.5-4.8,4.42-4.45]. Compared to the linear delay of the iterative array multiplier, matrix reduction is faster, especially for large wordlength n as it is a function of $O(\log n)$. For VLSI implementations of a single chip multiplier, however, the reduction technique, requiring a large number of irregular interconnections between different types of cells, is at a major disadvantage over the iterative array algorithm, which has a more regular layout. Since the current trend is towards single chip digital signal processor, the importance of a regular architecture becomes more urgent as a strategy to cope with the increased complexity. With the growing demand in real-time digital signal processing, better array multiplication speed than can achieved by the full-adder carry-save array (CSA) architecture is needed.

As discussed in Chapter 2, considerable increases in the speed of digital multipliers can be achieved by adding more than one partial

product at a time [4.5-4.17] by employing higher order parallel (p,q) counters. This approach largely depends on a counter which has a delay and complexity comparable to that of a full-adder. Chapter 2 also describes previous techniques employed to synthesize such counters.

In this chapter, extension of the CSA approach and novel array multipliers based on higher order parallel counters are first studied. An iterative array (5,3) counter multiplier architecture recently reported is highlighted and a novel architecture based on (2,2,3) counters that was developed in this project is then presented. These architectures are assessed on the basis of speed and their attractiveness for VLSI implementation, and are critically compared with their full-adder counterparts. Emphasis is put on the complexity of wiring between modules, the regularity of distribution of partials products in the array and the expandability of the architecture for longer wordlength. Also, for comparison purposes the implementation of the counters in the matrix reduction technique employed by Dadda is evaluated to determine how well it fares against the array technique in terms of speed. Examples of 8 x 8-bit multiplication are given for simple illustrations.

The study shows that both the (5,3) counter and (2,2,3) counter architectures are quite close to conventional array multipliers from a VLSI implementation point of view. In terms of speed, assuming the said counters operate at a comparable speed to a full-adder the (5,3) counter architecture offers a speed enhancement of nearly a factor of 2 whilst the (2,2,3) counter architecture is found to give a significant improvement for large operand wordlength.

4.2. Extension of the CSA technique

The advantages of the full-adder CSA multiplier architecture are well-understood and it is the most commonly-used technique in monolithic multipliers and digital signal processors due to its inherent regular structure and ease of design. A study of the concept of parallel (p,q) counters, first introduced by Dadda [4.5] leads one to the question of whether the CSA technique could be extended to higher order counters in order to achieve better speeds.

4.2.1. (4,3) counter multiplier architectures

A full-adder or (3,2) counter is basically a combinatorial network which takes in three bits of the same weight and produces a 2-bit output word, whose individual bits are commonly called the sum and carry bits. The next higher order counter is a (4,3) counter, the extra output bit is required as a consequence of equation 2.9. The employment of this cell (Figure 4.1(a)) in a CSA scheme results in the architecture shown in Figure 4.1(b) for a 8 x 8-bit multiplication. It can be observed that the (4,3) counter does not offer any significant improvement in speed over the full-adder CSA. In fact the number of stages and the total number of cells is the same as the full-adder CSA. The reason behind this is not difficult to see - except for the first two stages, only one row of partial products could be added at each stage as a consequence of the three output lines needed to be computed from the previous stage.

If we use the reduction algorithm employed by Dadda (see section 2.4.2), the maximum reduction height sequence is

3,4,5,6,8,10,13,17,.....

and compare this to that of the full-adder case whose maximum height sequence is

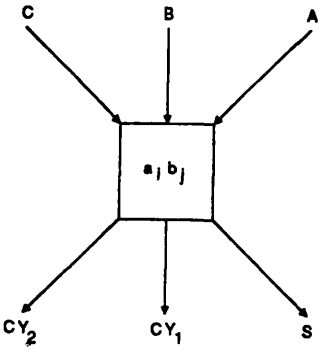


Figure 4.1(a). A (4,3) counter cell.

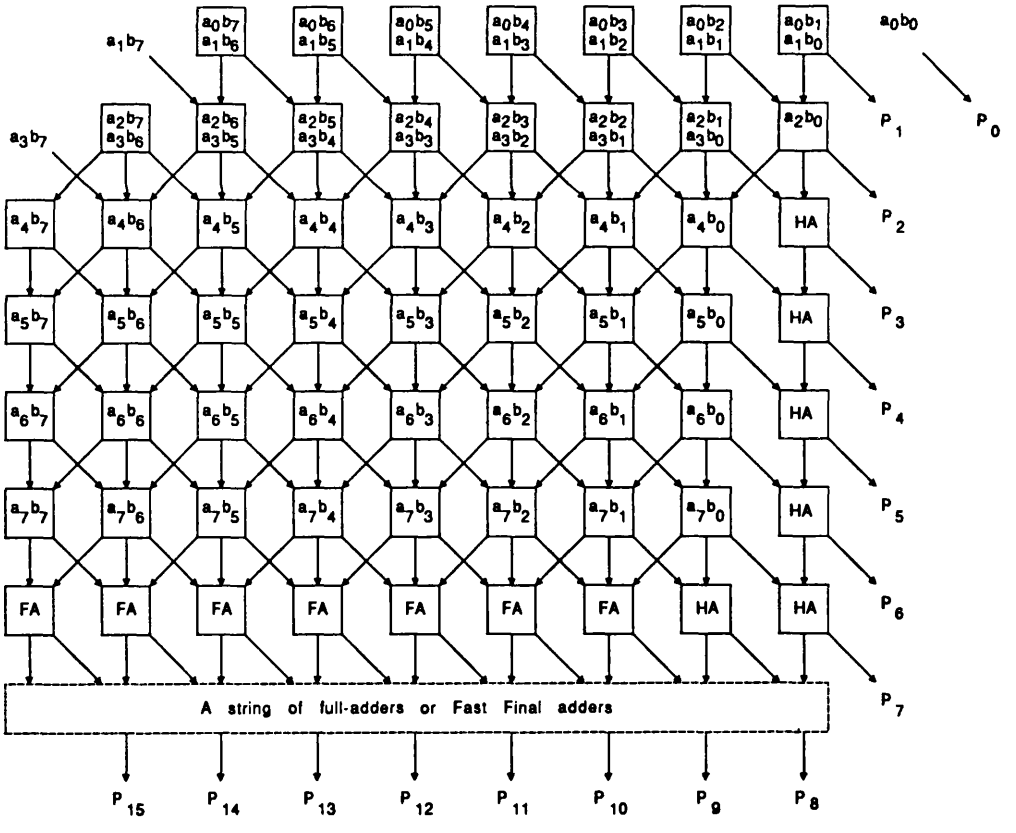


Figure 4.1(b). A 8 x 8-bit (4,3) counter CSA multiplier.

A	B	C	D	S	CY_1	CY_2
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	1	1	0
1	0	0	0	1	0	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	0	0	1

$$S = A \oplus B \oplus C \oplus D$$

$$CY_1 = \bar{A}BC + \bar{B}CD + AC\bar{D} + AB\bar{D} + B\bar{C}D + A\bar{C}D$$

$$CY_2 = ABCD$$

Table 4.1. Truth-table of a (4,3) counter.

2,3,4,6,9,13,19,28,.....

In this configuration, the (4,3) counter approach would probably offer a better speed (bearing in mind the high-speed required of the counter) especially for large wordlength. Unfortunately, the irregular layout of Dadda's scheme prohibits its practical use.

The main conclusion that can be drawn from observation of the (4,3) counter array architecture is that in order to attain any possible speed improvements in the array approach, at least two rows of partial products should be accommodated at each stage taking into account the number of output lines produced and which have to be computed at the next level. It is interesting to note that a 3-bit output word, as shown by equation 2.9

could code up to a maximum of 7-bits of the same weight. Obviously, with a (4,3) counter the full range of the output word is not efficiently utilised. These considerations leads one to the case of the (5,3) counter.

4.2.2. (5,3) counter multiplier architectures

Figure 4.2(b) demonstrates a multiplier architecture based on an array of (5,3) counters. Here, the counters, being able to sum two partial product bits at each stage are more efficient in reducing the number of stages in the array. For a 8 x 8-bit multiplication, there are 3 fewer stages compared to a full-adder CSA. In general, for a n x n-bit multiplication, the number of counter stages is given by $\left(\lfloor n/2 \rfloor + 1\right)$ for $n \geq 4$. For even n, this would give almost the same speed as the Booth multiplier (if the same fast final adder is employed for both) providing the (5,3) counter has a delay comparable to that of a full-adder.

Using Dadda's algorithm, the maximum reduction height sequence is

3,5,7,11,17,27,.....

Thus, for 8 x 8-bit multiplication the number of stages required is just one less than the array approach, but the scheme is clearly more effective for large wordlength.

However, like all good things, the (5,3) counter introduces more complexity (as Table 4.2 suggests) and it might be more difficult, compared to the (4,3) counter to achieve a counter speed that would approach the full-adder's. It is seen that the distribution of partial product terms in the array is less regular than the full-adder CSA and also there would be unused counter inputs if the homogeneous nature of the array in terms of counter size is preserved.

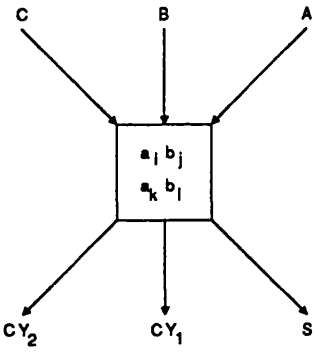


Figure 4.2(a). A (5,3) counter cell.

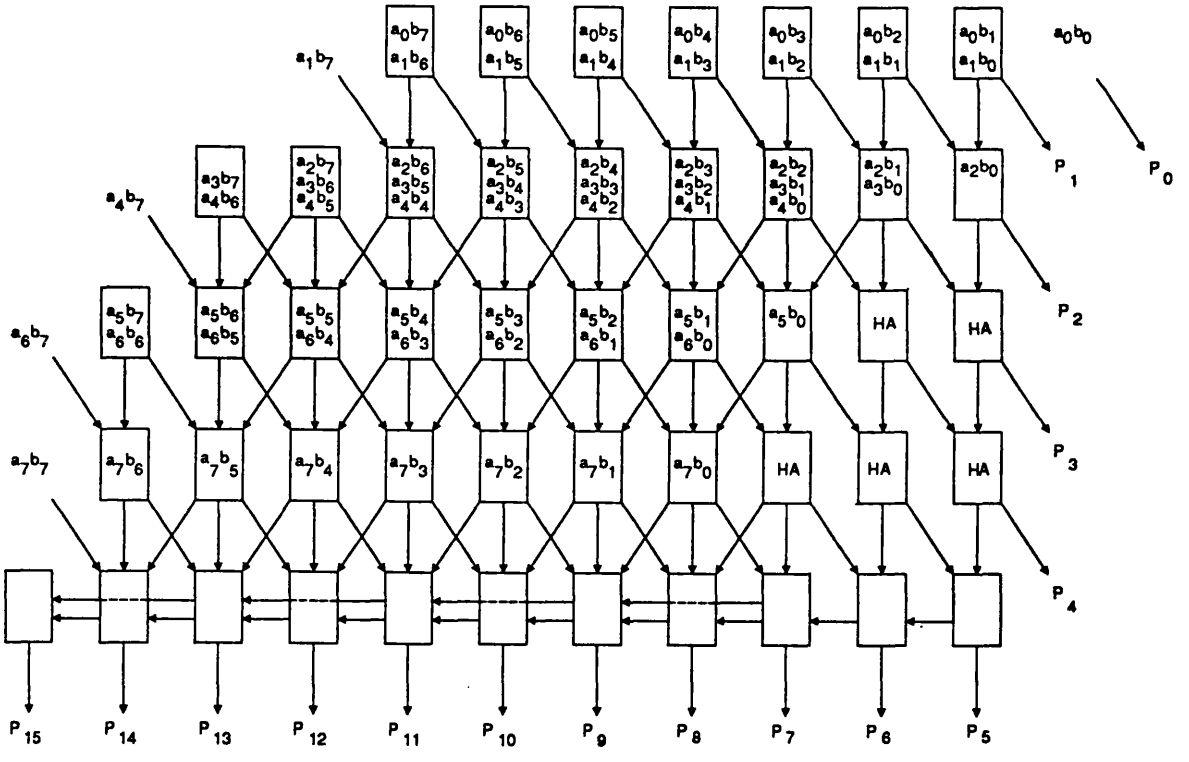


Figure 4.2(b). A 8 x 8-bit (5,3) counter CSA multiplier.

A	B	C	D	E	S	CY_1	CY_2
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0
0	0	1	0	0	1	0	0
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	1	1	0
0	1	0	0	0	1	0	0
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	1	1	0
0	1	1	0	0	0	1	0
0	1	1	0	1	1	1	0
0	1	1	1	0	1	1	0
0	1	1	1	1	0	0	1
1	0	0	0	0	1	0	0
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	0	0	1	0
1	0	1	0	1	1	1	0
1	0	1	1	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	0	0	0	1	0
1	1	0	0	1	1	1	0
1	1	0	1	0	1	1	0
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0
1	1	1	0	1	0	0	1
1	1	1	1	0	0	0	1
1	1	1	1	1	1	0	1

$$S = A \oplus B \oplus C \oplus D \oplus E$$

$$CY_1 = A(BC\bar{D} + C\bar{D}\bar{E} + \bar{B}\bar{D}\bar{E} + \bar{B}\bar{C}D + \bar{B}D\bar{E}) + \bar{A}(B\bar{C}E + BC\bar{D} + \bar{B}CE + C\bar{D}E + \bar{C}DE) + B\bar{C}D\bar{E}$$

$$CY_2 = BCDE + ABDE + ABCE + ABCD + ACDE$$

Table 4.2. Truth-table of a (5,3) counter.

It is important to note that assessing the speed of the array architecture on the basis of number of counter stages required is misleading. It can be observed that the critical path of the (5,3) counter array architecture (as well as the (4,3)) is along the right-most column and bottom row of the array which, for a n x n-bit multiplication is given by (2n-2) counter delays. This figure is the same as the critical delay of the full-adder CSA (or worse since the (5,3) counter is likely to be slower than the full-adder) as shown by equation 2.1 (excluding the AND gate delay for the generation of partial product bits). Although the number of stages are reduced, the generation of the final product bits at the edge of the array becomes the speed-limiting factor. An obvious answer to this would be to use fast adders as is commonly considered, but this adds complexities and irregularities in the array. Thus, in order to obtain better speeds in an array multiplier, the employment of a higher order counter, which reduces the number of stages required must be accompanied by a corresponding improvement in the delay due to the generation of the final product bits at the edges of the array. This should be achieved, as far as possible through preservation of the homogeneous nature of the array in terms of the size of the counter cell, their interconnectivity and the distribution of partial product terms, in order to render the architecture ideal for VLSI implementation. These factors presents an enormous challenge to VLSI designers.

However, a recent (5,3) counter architecture reported by Nakamura [4.30] appears to have all the features required of an array multiplier and is found to be quite close to that of the conventional CSA multipliers; this is discussed next.

4.3. An optimum (5,3) counter array multiplier

Nakamura [4.27,4.30] has reported an optimum (5,3) counter iterative array multiplier architecture that has a high degree of regularity without substantially increasing the total hardware complexity. It was claimed that the operation speed is nearly twice as fast as the conventional array multiplier providing the (5,3) counter has a comparable delay to that of the full-adder's.

In this scheme, the array of partial products can be collated in a square form, as shown in Figure 4.3 for a 8 x 8-bit multiplication, where i is the ith row from the bottom and j is for the jth column from the left. The coordinates are arranged so that at the location (i,j) the weight for the partial product $a_i b_j$ is 2^{i+j}. It is clear that all partial product bits of the same weight in the square array are placed along in a diagonal direction from upper left to lower right. By the same coordinates used in Figure 4.3, we denote the cell at the (i,j) position as C_{ij}. If the square matrix is folded into a triangular shape at the main diagonal line of $i = j$, as shown in Figure 4.4, each C_{ij} $(n > i > j \geq 0)$, contains two partial products of $a_i b_j$ and $a_j b_i$, and on the main diagonal C_{ii} $(n > i \geq 0)$, contains only one partial product of $a_i b_i$. The multiplier architecture is based on this triangular array. In Figure 4.4, the connections for the sum and carrys are obvious; by assuming the flow of additions from left to right, the sum and carrys should be chained along on the equally weighted diagonal (incrementally weighted horizontal) direction.

The entire interconnection of the input and output lines of the counter cells in a 8 x 8-bit multiplier results in the architecture shown in Figure 4.5(b). In general for a n x n-bit multiplication, the scheme has interconnections defined as follows :

i \ j	0	1	2	3	4	5	6	7
7	a_7b_0	a_7b_1	a_7b_2	a_7b_3	a_7b_4	a_7b_5	a_7b_6	a_7b_7
6	a_6b_0	a_6b_1	a_6b_2	a_6b_3	a_6b_4	a_6b_5	a_6b_6	a_6b_7
5	a_5b_0	a_5b_1	a_5b_2	a_5b_3	a_5b_4	a_5b_5	a_5b_6	a_5b_7
4	a_4b_0	a_4b_1	a_4b_2	a_4b_3	a_4b_4	a_4b_5	a_4b_6	a_4b_7
3	a_3b_0	a_3b_1	a_3b_2	a_3b_3	a_3b_4	a_3b_5	a_3b_6	a_3b_7
2	a_2b_0	a_2b_1	a_2b_2	a_2b_3	a_2b_4	a_2b_5	a_2b_6	a_2b_7
1	a_1b_0	a_1b_1	a_1b_2	a_1b_3	a_1b_4	a_1b_5	a_1b_6	a_1b_7
0	a_0b_0	a_0b_1	a_0b_2	a_0b_3	a_0b_4	a_0b_5	a_0b_6	a_0b_7

Figure 4.3. A 8 x 8-bit collated square array of partial product bits [4.27].

i \ j	0	1	2	3	4	5	6	7
7	a_0b_7 a_7b_0	a_1b_7 a_7b_1	a_2b_7 a_7b_2	a_3b_7 a_7b_3	a_4b_7 a_7b_4	a_5b_7 a_7b_5	a_6b_7 a_7b_6	a_7b_7
6	a_0b_6 a_6b_0	a_1b_6 a_6b_1	a_2b_6 a_6b_2	a_3b_6 a_6b_3	a_4b_6 a_6b_4	a_5b_6 a_6b_5	a_6b_6	
5	a_0b_5 a_5b_0	a_1b_5 a_5b_1	a_2b_5 a_5b_2	a_3b_5 a_5b_3	a_4b_5 a_5b_4	a_5b_5		
4	a_0b_4 a_4b_0	a_1b_4 a_4b_1	a_2b_4 a_4b_2	a_3b_4 a_4b_3	a_4b_4			
3	a_0b_3 a_3b_0	a_1b_3 a_3b_1	a_2b_3 a_3b_2	a_3b_3				
2	a_0b_2 a_2b_0	a_1b_2 a_2b_1	a_2b_2					
1	a_0b_1 a_1b_0	a_1b_1						
0	a_0b_0							

Figure 4.4. A 8 x 8-bit folded triangle array of . partial product bits [4.27].

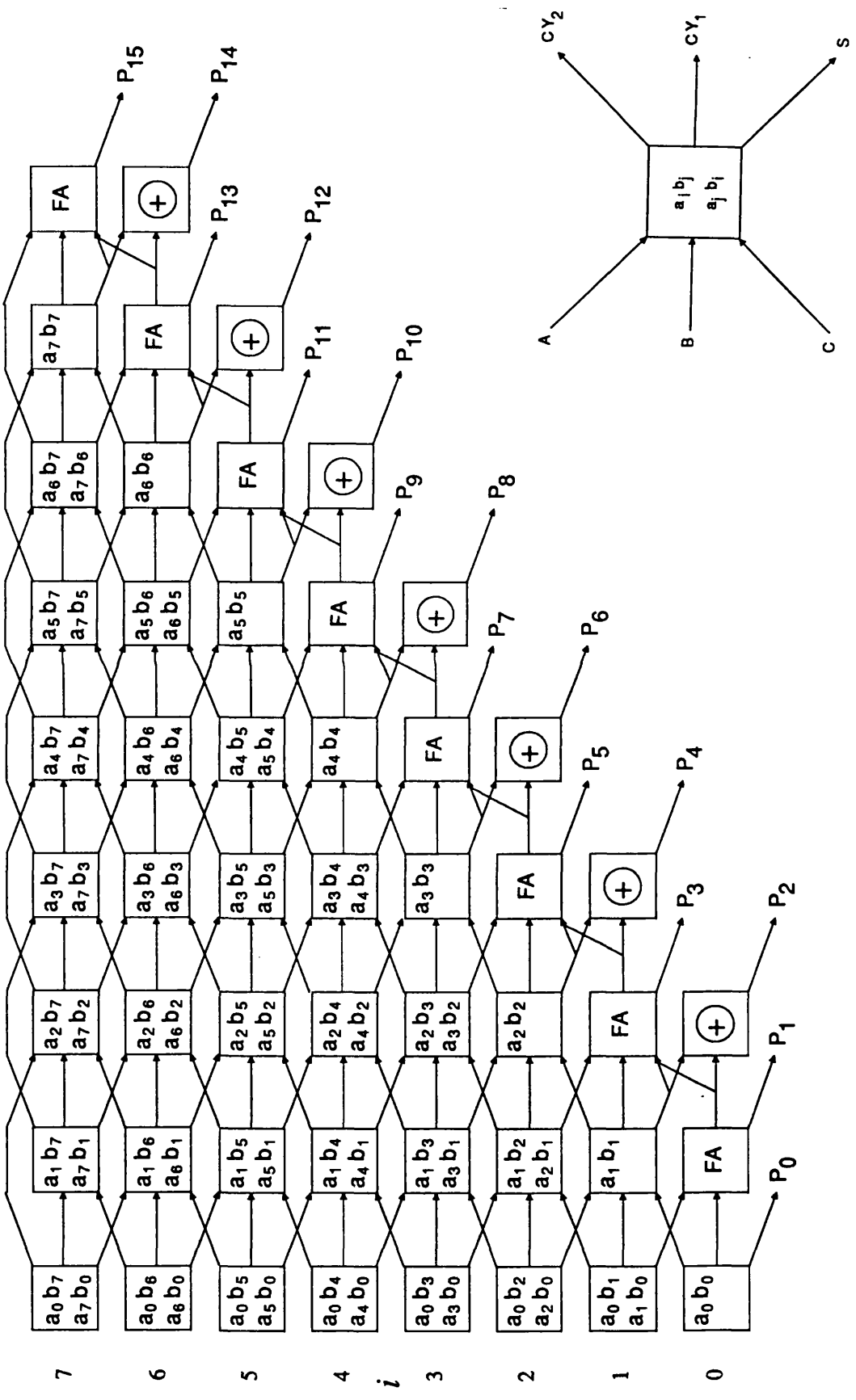


Figure 4.5(a). A (5,3) counter cell.

Figure 4.5(b). An optimum 8 x 8-bit array multiplier based on a (5,3) counter [4.30].

(i) For $i,j > 0$ and $i + j < n$, C_{ij} has inputs from product terms a_ib_j and a_jb_i. The A, B, and C input lines are connected to $C_{i+1,j-1}$, $C_{i,j-1}$, $C_{i-1,j-1}$ cells through their S, CY_1 and CY_2 output lines, respectively. The S, CY_1 and CY_2 lines of C_{ij} are connected to the $C_{i-1,j+1}$, $C_{i,j+1}$ and $C_{i+1,j+1}$ through their input lines A, B and C, respectively.

(ii) The input lines A, B and C of the leftmost column cells $C_{i,0}$ are connected to 0.

(iii) The diagonal cells C_{ij} where $i = j$, have only one partial product term a_ib_i. Here the cells are designed as (3,2) counters and in addition, another stage of n modified full-adders. These diagonal full-adders are slightly different from the normal full-adder in that there are four inputs instead of three. The reason is that two of the four inputs have a positional weight of 1/2 since both bits ANDed together gives a carry bit. Thus, the sum and carry output of the diagonal cells are defined as

$$S = \text{sum}(w,x,y,z) = w \oplus x \oplus (yz)$$

$$C = \text{carry}(w,x,y,z) = wx + wyz + xyz$$

where w = S of $C_{i+1,j-1}$, x = CY_1 of $C_{i,j-1}$, y = S of $C_{i,j-1}$, and z = CY_1 of $C_{i-1,j-1}$. By employing these cells, the computation of the final products is speeded up since such a modified full-adder with four inputs is simpler and faster than a (5,3) counter. As seen later, a combination of the EX-OR gate and the modified full-adder is basically a (2,2,3) counter and is the basis of a novel array architecture developed in this project.

The total number of (5,3) counters, half-adders, modified full-adders and EX-OR gates required for a n x n-bit multiplication is summarized in Table 4.3 (excluding AND gates for partial product generation).

As shown in Figure 4.5(b), a carry propagates from left to right and the longest delay for a n x n-bit multiplication is (n + 1) counter

	Number of cells
(5,3) counters	$\sum_{x=1}^{n-1} x$
Half-adders (HA)	n
Modified FA	n
EX-OR gate	(n-1)

Table 4.3. No. of counters for a n x n-bit (5,3) counter multiplier.

delays. Some minor improvements in speed and hardware savings can be made by replacing the 0th column cells $C_{i,0}$ by half-adders. If the delay of a single (5,3) counter is equal to that of a full-adder's, the new algorithm is nearly twice as fast as the old one. Apart from that, the multiplier is very close to the full-adder CSA since the advantages of this array architecture are obvious; there is a regular interconnection of counters. Although there are more wires running in between the cells and more than one type of cell employed compared to the full-adder CSA, the wiring between the different types of cells are still regular, and all the interconnections between cells are made between either nearest neighbours or next nearest neighbours. This property is vital for VLSI implementation because wiring in integrated circuits occupies a lot of silicon area. The architecture is also easily expandable by adding extra rows of counters to fit the specific application. Furthermore, there is a regular distribution of the partial product terms in the array as illustrated by Figure 4.6, and no counter inputs are unutilised. These

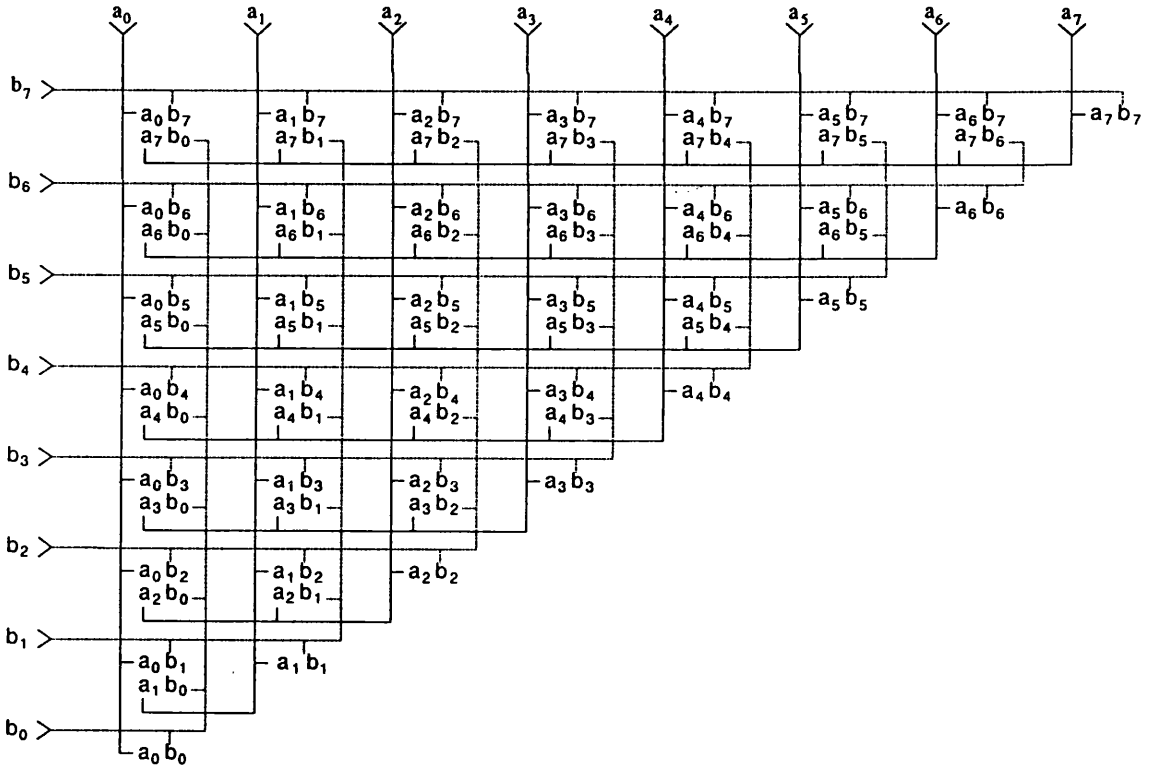


Figure 4.6. Distribution and wiring of partial product terms.

factors make the architecture ideal from a VLSI implementation point of view.

The major issue in this array multiplication algorithm is the delay and complexity of a single (5,3) counter cell. The attractiveness of the architecture in terms of speed and VLSI implementation of single chip multipliers and digital signal processors makes it worthwhile to investigate possible schemes to realise a (5,3) counter which would give a delay comparable to that of the full-adder without incurring too much complexity. According to the complexity measure estimated by Nakamura which is based on the table look-up operation on ROM [4.8], the (5,3) counter is three times more complex than the full-adder. This comparison, however, is based on a particular implementation method which may not

give practical values for certain logic families with different implementation methods.

In practical applications of hardware algorithms, one of the problems we often face is the fact that the efficiency of the implemented hardware depends not only on the algorithm but also on the technology that implements the design. Indeed, in many cases the implementing technology has a dominant effect on the efficiency of design. Therefore, in this project the (5,3) counter multiplier is discussed in a different perspective, namely the efficiency of operation speed of the (5,3) counter and the viability of the architecture based on a particular technology - bipolar, since this is the fastest silicon technology that has reached an advanced stage of maturity. This subject is thoroughly described in subsequent chapters.

4.4. Novel iterative (2,2,3) counter multiplier architecture

The folded triangle array of partial products proved to be one of the key factors which gives the optimum (5,3) counter architecture its regular structure in terms of the interconnections between cells, the even distribution of partial products and the complete utilisation of the counters. In addition, the employment of the modified full-adder for the diagonal cells enhances the speed of the critical path in the array by a possible factor of nearly 2. These features are made use of in the development of the (2,2,3) counter multiplier architecture described below.

Attempting to extend the optimum (5,3) counter architecture based on single input column, higher order counters i.e. (6,3), (7,3) and so on, using the folded partial product array would not result in any improvement in the critical delay. The employment of these counters would

merely reduce the sum delay path while the carrys delay paths remained unaffected, which means that there would still be the same number of cells in the *j*-direction (and hence the same speed) as the (5,3) counter multiplier. Furthermore, the distribution of the partial product terms would no longer be regular since the higher order counters would depend on a partial product array that has more than two terms paired together regularly and this is difficult, if not impossible, to obtain. Besides, as Table 4.1 and Table 4.2 suggest, the logical functions of higher order counters increases quite disproportionately that there is no advantage in trying to extend the architecture larger than the (5,3).

It can be observed that, ideally, any attempt to improve the critical delay of the (5,3) counter multiplier should concentrate on ways of reducing the number of cell delays in both the *i* and *j* direction . This entails summing four neighbouring pairs of partial product terms in a delay time of at worse, two full-adder delays to be able to maintain the speed advantage. Such a counter required to sum the four pairs of partial product terms (a total of eight input variables) would be far too complicated and impractical. A compromise is, however possible and this stems from the fact that the delay due to the generation of two consecutive final product bits can be achieved in a unit counter delay with the help of the diagonal cell combination of an EX-OR gate and the modified full-adder. What is more important is reducing the delay across the array i.e. in the j-direction and this could be implemented with the same EX-OR gate and modified full-adder combination.

Using the concept of counters which successively receive several input columns introduced in Chapter 2, the EX-OR gate and modified full-adder composed together is really a (2,2,3) counter as illustrated in Figure 4.7. The logic function of the three outputs are as defined in

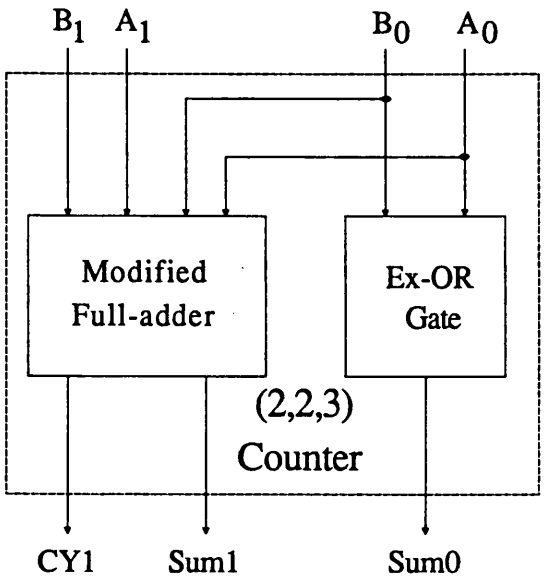


Figure 4.7. A (2,2,3) counter cell.

B_1	A_1	B_0	A_0	CY1	Sum1	Sum0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	0

$$Sum0 = A_0 \oplus B_0$$

$$Sum1 = A_1 \oplus B_1 \oplus (A_0B_0)$$

$$CY1 = A_1B_1 + A_1A_0B_0 + B_1A_0B_0$$

Table 4.4. Truth-table of (2,2,3) counter.

Table 4.4. Notice that equation 2.11 is satisfied with a 3-bit output word. The realisation of the modified full-adder, with only four inputs, in principle, could be achieved with a delay between that of a full-adder's and a (5,3) counter's. It is seen that the Sum0 output propagation delay, being dependent on only two input variables would be faster than both the Sum1 and CY1 outputs. These considerations lead us to the (2,2,3) counter array multiplier architecture shown in Figure 4.8 for a 8 x 8-bit multiplication.

4.4.1. Architectural description of (2,2,3) counter multiplier

The (2,2,3) counter multiplier differs from the (5,3) counter multiplier in that two consecutive partial product pairs of adjacent weights in the folded square matrix are summed together simultaneously. Since this operation produces 3 output lines of different weights, a network of (2,2,3) counters and full-adders are interconnected to form megacells to add up the partial product bits with the output lines from previous stages. In Figure 4.8, each box with partial product bits is a megacell composed of (2,2,3) counters and full-adders whereas the diagonal cells are solely half-adders or (2,2,3) counters. The interconnection of the megacells and diagonal cells are similar to the (5,3) counter architecture except that there is now a fourth output line (the highest carry) possible from each megacell, depending on the number of partial product bits present as well as the number of output lines from previous stage cells (taking their relative weighting into account). Using the coordinates shown in Figure 4.8, this occurs for cells $C_{i,0}$ and $C_{i,j}$ (where $i \neq j$; $i \neq$ n-1 and $j \neq 0$). This highest carry line of such megacell, say C_{ij} has to go to the cell to the right of it i.e. cell $C_{i,j+1}$ in order to maintain the same number of output lines from each

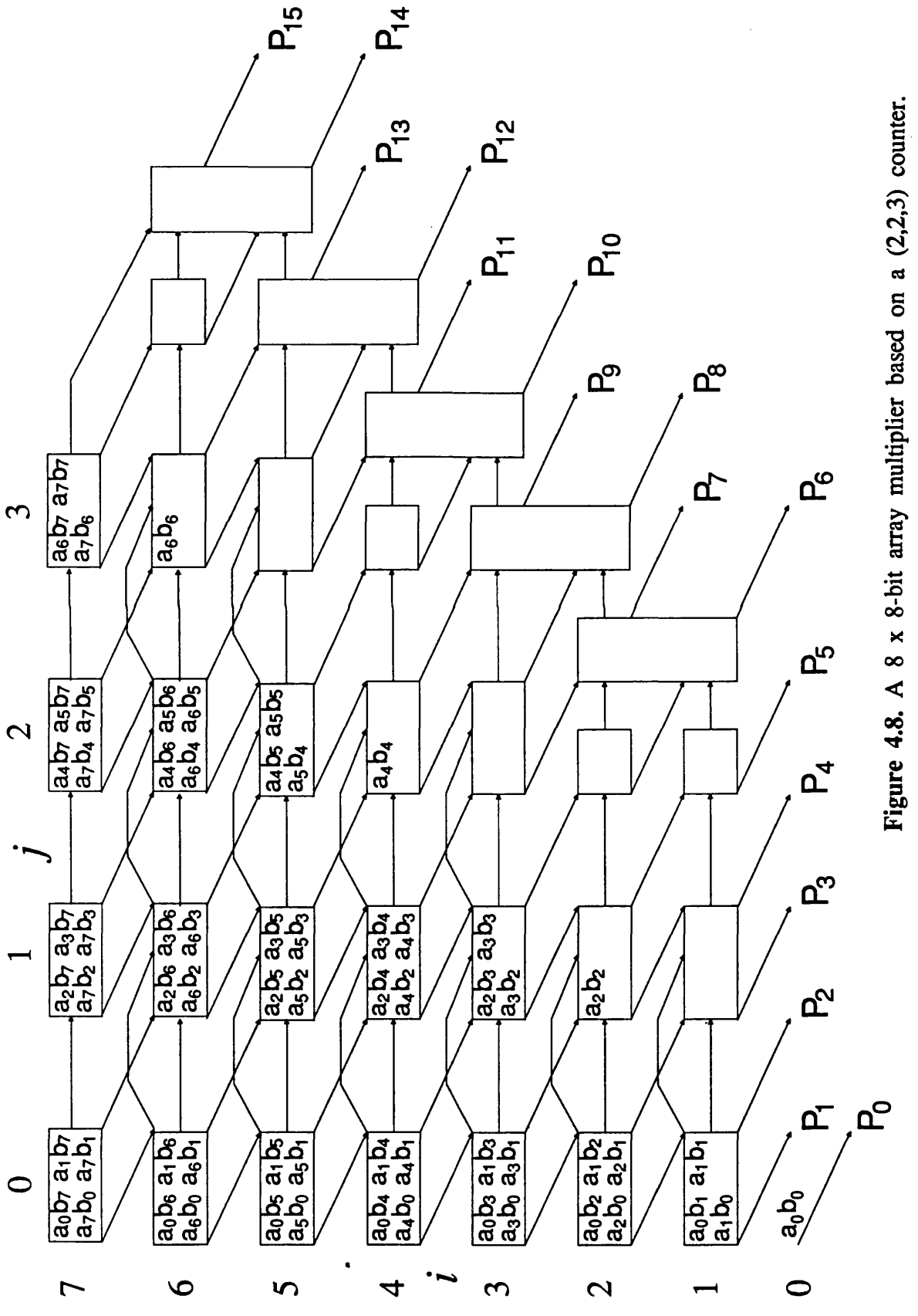


Figure 4.8. A 8 x 8-bit array multiplier based on a (2,2,3) counter.

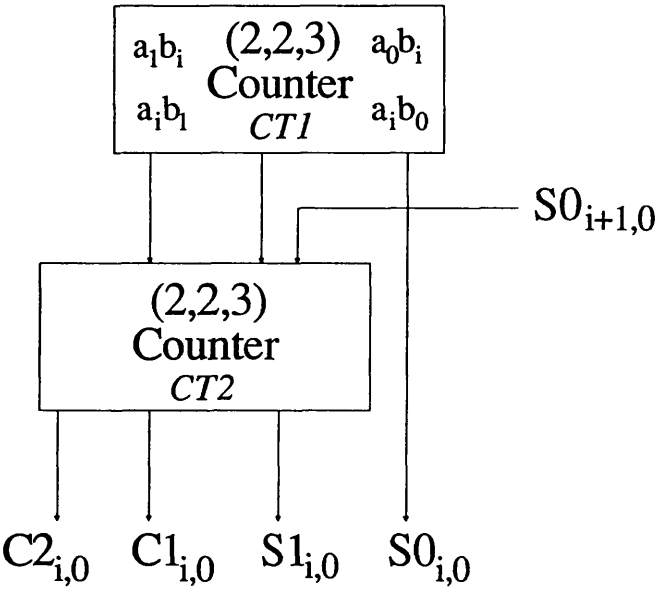


Figure 4.9(a) Composition of megacell $C_{i,0}$.

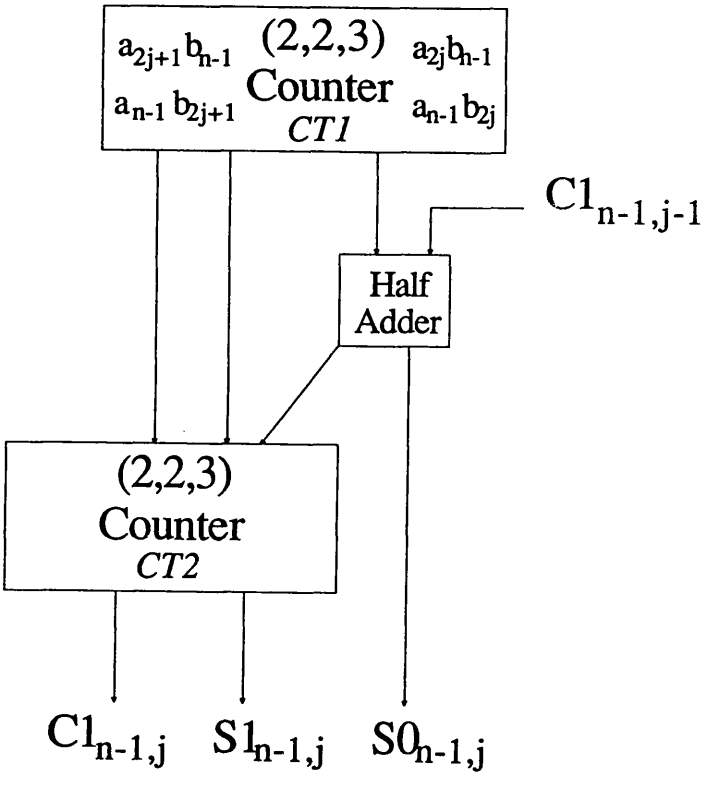


Figure 4.9(b). Composition of megacell $C_{n-1,j}$.

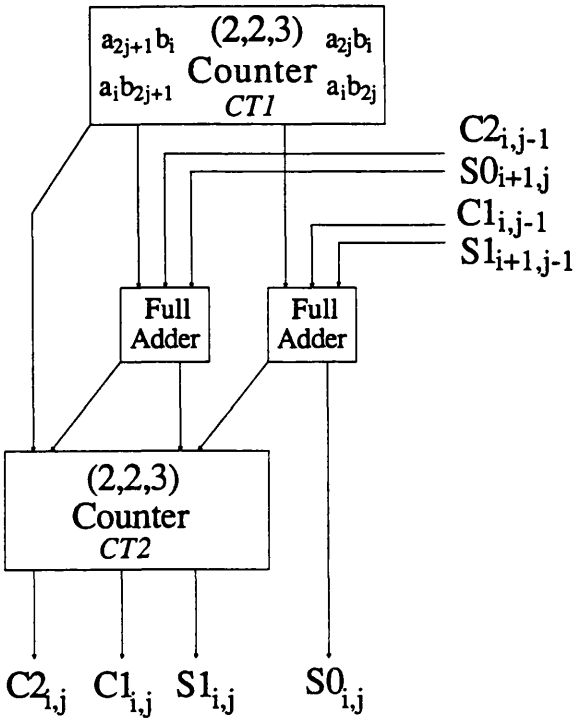


Figure 4.9(c). Composition of megacell $C_{i,j}$.

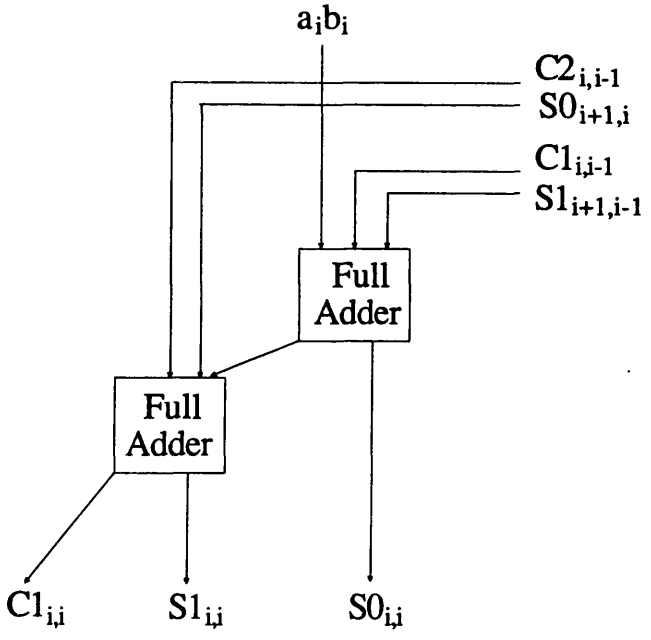


Figure 4.9(d). Composition of megacell $C_{i,i}$.

megacell at subsequent stages. Furthermore, if the highest carry is instead connected to cell $C_{i+1,j+1}$ as is similarly done in the (5,3) counter multiplier, a more complicated and slower megacell would result. The main disadvantage of connecting the highest carry this way is that, it rules out the possibility of using the same network of EX-OR gate and modified full-adder for the diagonal cells as implemented in the (5,3) counter architecture, in the computation of the final product bits. A slightly more complicated but, still regular network of half-adders and (2,2,3) counters for the diagonal cells is needed in this architecture.

The composition of the megacells are shown in Figures 4.9(a)-(d). In general for a n x n-bit multiplication, there are (n-1) rows and n/2 columns of megacells, which are divided into four classes as described below :

(i) For cells in the leftmost column i.e. $C_{i,0}$ (where $1 \leq i \leq (n-1)$), each cell is made up of a (2,2,3) counter *CT1*, which initially reduces the two adjacent pairs of partial product terms and another (2,2,3) counter *CT2* as depicted in Figure 4.9(a). One of the lower significant inputs to *CT2* is a propagating input, which comes from the S0 output of cell $C_{i+1,0}$. The propagation delay of this megacell is one (2,2,3) counter delay for the S1, C1 and C2 outputs whilst the S0 line do not suffer any further delay.

(ii) The composition of the topmost row megacells $C_{n-1,j}$ (where $j > 0$) is similarly illustrated in Figure 4.9(b). A fourth output line C2 is not possible in this case since the full-range of the 3-bit output word is adequate to code the two pairs of partial product terms plus the bit due to propagating input C1 from cell $C_{n-1,j-1}$.

Some minor improvements in speed and hardware savings could be achieved by replacing *CT2* in cell $C_{i,0}$ and *CT2* in cell $C_{n-1,j}$ with a

(1,2,3) counter.

(iii) For megacells $C_{i,j}$ (where $i \neq j$, $i \neq$ n-1 and $j \neq 0$), the presence of two pairs of input lines of adjacent weights from the previous stage necessitates the use of two blocks of full-adders to add them with the summed partial product bits. This is shown in Figure 4.9(c).

(iv) Figure 4.9(d) shows the composition of megacells $C_{i,i}$, where i is even and there is only one partial product term present. Notice again that the three output lines are sufficient to code the addition of the two pairs of input lines of adjacent weights from the previous stage, with the single partial product bit.

The propagation delays of the appropriate outputs of the four types of megacells are summarized in Table 4.5 by making the assumption that all the logic blocks could be realised with a unit gate delay. It must be stressed that in all the megacells except $C_{i,i}$, in principle the S1 output's propagation delay would be faster than the higher carry output(s) since logically, S1 is merely an EX-OR function of two variables as illustrated in Figure 4.7.

The diagonal cells differ from those of the (5,3) counter architecture as a result of the presence of the fourth output line C2. A slightly more involved, but regular network is needed to compute the final product bits. This, however does not actually worsen the critical delay resulting from generation of the final product bits. Figure 4.10 illustrates this where the delay paths near the neighbourhood of the diagonal cells of part of a n x n-bit array are shown after taking into account the different propagation delays of the outputs of the megacells as summarized in Table 4.5. In fact, the sum and carry delay paths through the megacells is effectively equalised by the delay path resulting from generation of the final product bits (dotted arrow) by

megacell	Propagation delay (unit gate delay)			
	C2	C1	S1	S0
$C_{i,0}$	1	1	1	0
$C_{n-1,j}$	-	2	2	1
$C_{i,j}$	2	2	2	1
$C_{i,i/2}$	-	2	2	1

Table 4.5. Propagation delays of megacells

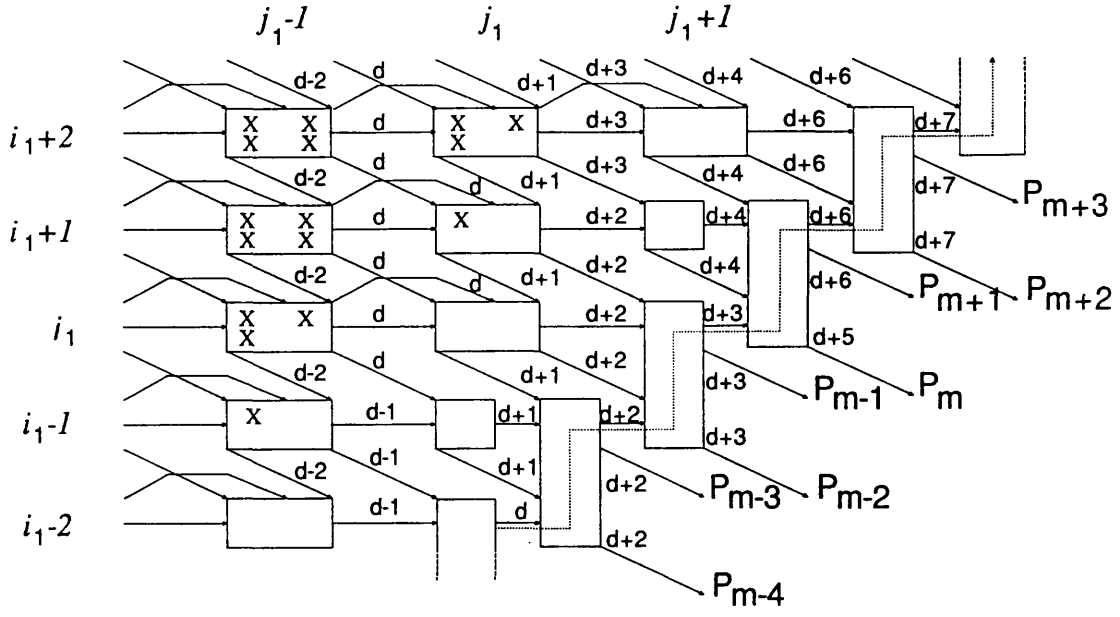


Figure 4.10. A network of half-adders and (2,2,3) counters for the diagonal cells.

	No. of megacells/cells	
	even n	odd n
$C_{i,0}$	n-1	n-1
$C_{n-1,j}$	$\frac{n}{2}-1$	$\lfloor \frac{n}{2} \rfloor -1$
$C_{i,j}$	$\sum_{x=1}^{n/2-2} 2x$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (2x+1)$
$C_{i,i}$	$\frac{n}{2}-1$	$\lfloor \frac{n}{2} \rfloor$
Half-adders	$\frac{n}{2}$	$\lfloor \frac{n}{2} \rfloor$
(2,2,3) counters	$\frac{3}{2}n-4$	$3\lfloor \frac{n}{2} \rfloor -5$

Table 4.6(a). No. of megacells/cells of a (2,2,3) counter multiplier.

	Number of counters	
	Even n	Odd n
2-input EXOR	$\sum_{x=1}^{n/2-2} 4x + (\frac{11}{2}n-9)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (4x+2) + (\frac{11n-21}{2})$
2-input AND	$\sum_{x=1}^{n/2-2} 8x + (7n-8)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (8x+4) + (7n-10)$
Full-adders (FA)	$\sum_{x=1}^{n/2-2} 4x + (n-2)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (4x+2) + (n-1)$
Modified FA	$\sum_{x=1}^{n/2-2} 4x + (\frac{9}{2}n-8)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (4x+2) + (\frac{9n-17}{2})$

Table 4.6(b). Total no. of parallel counters of a (2,2,3) counter multiplier

employing a network of half-adders and (2,2,3) counters for the diagonal cells.

Table 4.6(a) summarizes the total number of different types of megacells including the diagonal ones for a n x n-bit multiplication. In Table 4.6(b), the total number of counters of different sizes after flattening the hierarchy of the megacells are given.

4.4.2. Critical delay path

The last section shows how the delay due to the generation of the final product bits could be equalised to the sum and carry paths through the megacells by using a slightly more complicated network of half-adders and (2,2,3) counters for the diagonal cells. In order to identify the critical path of the multiplier, it is thus necessary to look for the sum and/or carry path through the megacells which gives the largest delay. In this architecture, the sum delay path of any weight w, where $2^4 < 2^w \leq 2^{2n-1}$ comprises alternate gate delays of 1 and 2 (except for the first two column of megacells where the sum delay is a unit gate delay) corresponding to the S0 and S1 output delay, respectively for each addition of a pair of partial product terms. One would suspect then that the longest delay path through the the megacells is in the vicinity of the path which has the largest number of partial product bits and this occurs for the path whose partial product weight is 2^n.

Since, in principle, the modified full-adder part of a (2,2,3) counter is slower than the 2-input EXOR gate, the actual critical path through the megacells starts from the top left cell $C_{n-1,0}$ and follows a " staircase " path traversing through alternate S0 and C1 paths as illustrated in Figure 4.11 (shaded area) where the diagonal cells have been deleted for the sake of clarity. Using the notation IP_w, for the

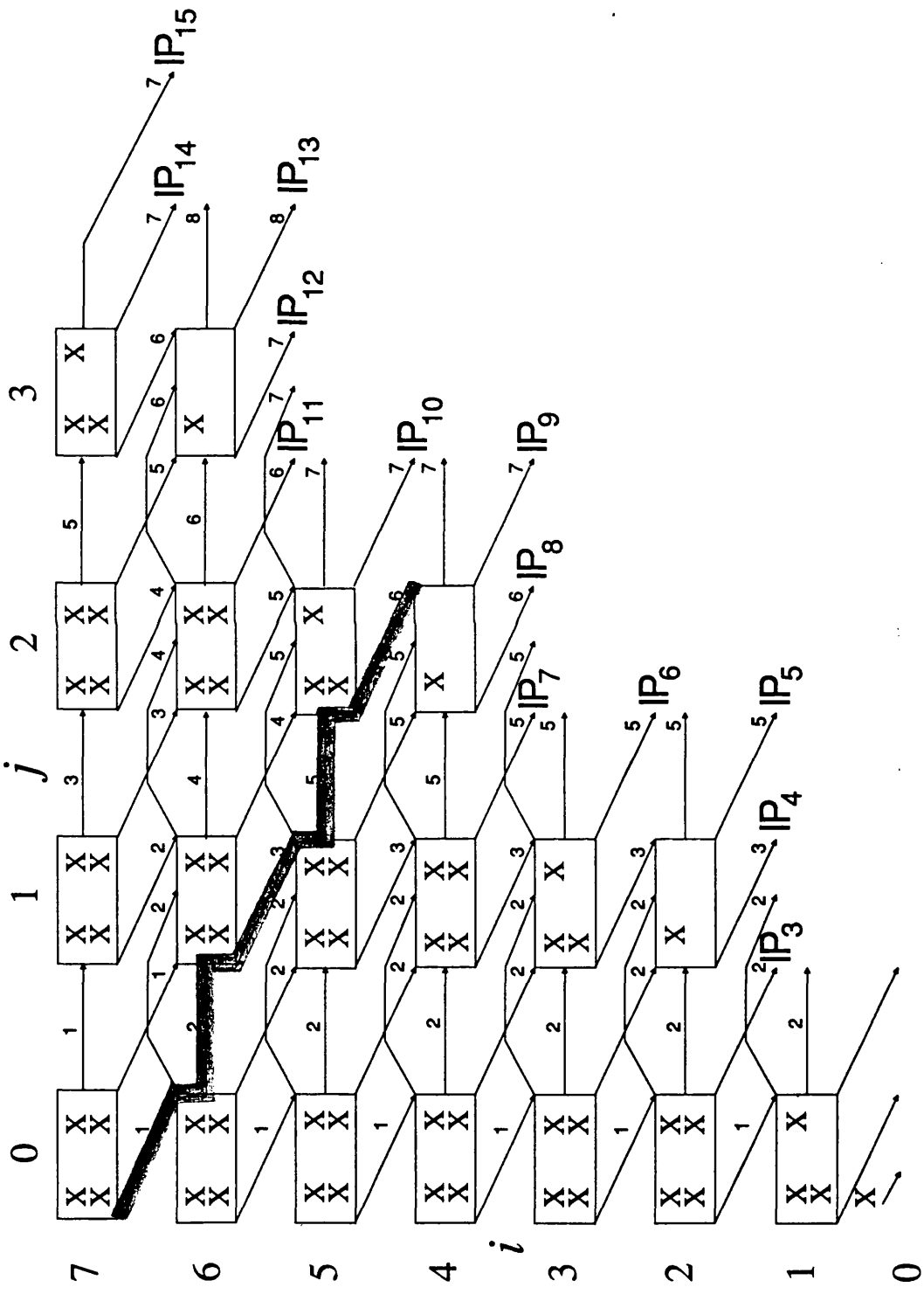


Figure 4.11. The critical path through the megacells of a 8 x 8-bit array.

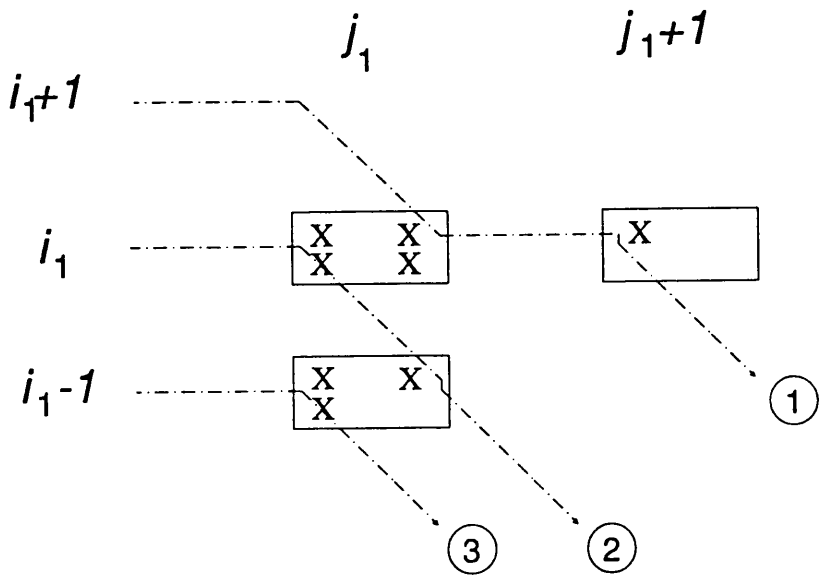


Figure 4.12. Diagram demonstrating possible paths where IP_c ends.

intermediate product bit (IP) of weight w, such a delay path results in the intermediate product IP_c, whose weight is 2^c having the largest delay through the array of megacells (c = 9 for the 8 x 8-bit multiplication). For $w > c$, the delay of the intermediate product bits is equal to or less than the delay of IP_c (this is proved in section 4.4.2.2) which verifies that the " staircase " path that results in IP_c does represent the critical path through the megacells. For a n x n-bit multiplication, the value of c is derived in the next section.

4.4.2.1. Derivation of the weight of the critical intermediate product bit

Because of the slight irregularities of the megacells neighbouring the diagonal cells over two rows, determining the value of c and characterising the delay behaviour of IP_c directly would be rather awkward. Instead, observations were made on IP_c over a range of n and these are summarized in Figure 4.12, which shows the possible paths where

the critical path of IP_c could end, depending on the nature of n. Path 1 would result for IP_c where n has the property that (n-1) is a multiple of 3. Path 2 and path 3 applies when n and (n-2) respectively is a multiple of 3. Table 4.7 tabulates c and the total gate delay of IP_c over a certain range of n. From Table 4.7, it can be seen that for every multiple of 3, c increases by 1. Hence,

$$c = n + \left\lfloor \frac{n-4}{3} \right\rfloor \quad (4.1)$$

The delay of IP_c, d can be generalised as follows:

(a) If (n-2) is a multiple of 3 then

$$d = n-2 \quad (4.2)$$

(b) or else

$$d = n-1. \quad (4.3)$$

Wordlength n	c	Total gate delay d of IP_c
5	5	3
6	6	5
7	8	6
8	9	6
9	10	8
10	12	9
11	13	9
12	14	11
13	16	12
14	17	12
15	18	14
16	20	15

Table 4.7. The weight and delay of the critical intermediate product bit over a range of wordlength.

4.4.2.2. A proof that the higher significant intermediate product bits have a delay not more than the delay of IP_c

The delay of the outputs of the topmost row cells $C_{n-1,j}$, unlike the megacells in the bulk of the array is not affected by the presence of a slower propagating input bit and this results in a faster delay (by one gate) for the outputs concerned. This, in turn sets the delay of the megacells of the following stages resulting in a delay pattern which increases linearly down the array by a unit gate delay for a given column of cells in the part of the array above the critical path of IP_c. Thus, above the array where IP_c ends, Figure 4.13 results for the case where (n-2) is not a multiple of 3 and where the delay of IP_c is given by (d-1). It can be seen that there is a linear increase in the delay of the same outputs of the same column of megacells as we move down the array. The same delay pattern can also be observed at the part of the array above the critical delay path of IP_c, for any other appropriate multiple of n corresponding to the paths shown in Figure 4.12.

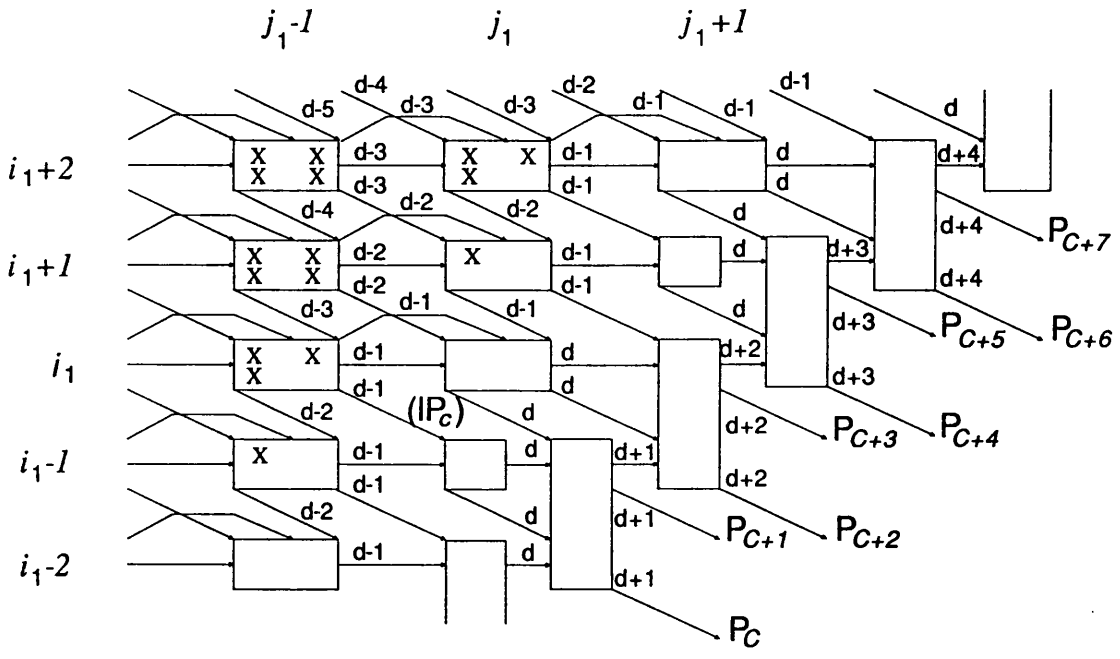


Figure 4.13. Part of an array above the critical delay path IP_c.

From Figure 4.13, for each column of megacells added for the computation of the next higher significant intermediate product bit, the extra gate delay required per intermediate product bit is actually compensated by the lower delay (by one gate for each row) of the carry signals C1 and C2 from the previous column.

Thus, the total gate delay of the intermediate product bits higher than IP_c cannot be greater than the delay of IP_c.

In order to determine the total worst case delay for a n x n-bit multiplication it is necessary to note the following two observations :

(i) It can be seen from Figure 4.13 where (n-2) is not a multiple of 3, a further 2 gate delays is needed to obtain P_c and P_{c+1} in addition to the delay d of IP_c by using the network of half-adders and (2,2,3) counters. Thus, the number of remaining final product bits greater than P_{c+1} is given by

$$(2n-1) - (c+1) \qquad \textbf{(4.4)}$$

For the case where (n-2) is a multiple of 3, the same observation can be made except that P_{c+1} cannot be obtained simultaneously with P_c by using the same diagonal (2,2,3) counter. This gives the number of remaining higher significant final product bits greater than P_c as

$$(2n-1) - c \qquad \textbf{(4.5)}$$

(ii) The total delay d of IP_c is dependent on whether (n-2) is a multiple of 3 or not, as derived in equations 4.2 and 4.3, respectively.

The total worst-case delay of the architecture can now be evaluated, first, for the case where (n-2) is not a multiple of 3. An assumption is made that the (2,2,3) counter has a unit gate delay. In order to compute the higher significant final product bits P_w, where $(2n-1) \geq w > (c+1)$, it can be observed that every two consecutive final product bits can be obtained in one gate delay by using the same network of diagonal cells

throughout the array. These higher significant final product bits can be obtained in a further delay totalling

$$\frac{(2n-1) - (c+1)}{2} \tag{4.6}$$

The worst case delay D_n is then equal to the total delay of P_c (and P_{c+1}) and the delay due to the computation of the higher significant product bits, P_w; this is given by

$$D_n = (n-1) + 2 + \frac{(2n-1) - (c+1)}{2} = \frac{3}{2}n - \frac{1}{2}\left\lfloor\frac{n-4}{3}\right\rfloor \tag{4.7}$$

The same argument as above can be applied for the case when (n-2) is a multiple of 3 where it is found that

$$D_n = (n-2) + 2 + \frac{(2n-1) - (c+1)}{2} = \frac{3}{2}n - \frac{1}{2}\left(1+\left\lfloor\frac{n-4}{3}\right\rfloor\right) \tag{4.8}$$

Equations 4.7 and 4.8 can be generalised to give

$$D_n = (n+1) + \left\lfloor\frac{n}{3}\right\rfloor \tag{4.9}$$

4.5. Architectural Comparisons of (5,3) and (2,2,3) counter multipliers

A comparison of the (2,2,3) counter and (5,3) counter array multipliers are shown in Figure 4.14(a) with the conventional CSA multiplier in terms of speed against the operand wordlength where it has been assumed that the (5,3) counter and (2,2,3) counter has a propagation delay equal to that of the full-adder. The relative hardware costs of the three schemes could be roughly estimated by assuming a single cascode ECL gate that could implement a logic function of up to five variables. The total ECL gate count of the straight CSA multiplier has been calculated in Chapter 2 (section 2.2.4) and an estimation of the hardware cost of the (5,3) counter and (2,2,3) counter multipliers are summarized in Table 4.8 including the AND gates for generation of partial product terms. Figure 4.14(b) illustrates the relative hardware costs of the three

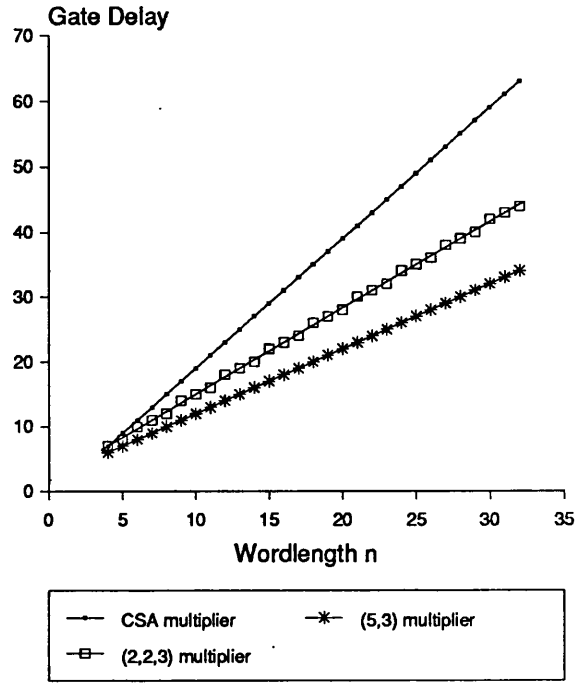
	Total ECL gate count	
(2,2,3) Counter multiplier	even n	odd n
	$\sum_{x=1}^{n/2-2}(28x) + (\frac{47}{2}n-37)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2}(28x+14) + (\frac{47n-79}{2})$
(5,3) counter multiplier	$\sum_{x=1}^{n-1}(3x) + n^2 + (5n-4)$	

Table 4.8. ECL gate count estimate of (2,2,3) counter and (5,3) counter multiplier.

different array techniques. Figure 4.14(a) demonstrates clearly the superior speeds of array multipliers through employment of higher order parallel counters. The (5,3) counter multiplier is faster than the conventional CSA multiplier by nearly a factor of two whilst the (2,2,3) counter array scheme has a speed between that of the CSA and (5,3) counter technique but is obviously more effective for large operand wordlength. It is interesting to note that the (5,3) counter architecture is about as fast as the Booth-encoded CSA multiplier with fast final adders (see Figure 2.4(a)).

In terms of the relative hardware costs, the (5,3) counter scheme is better than the CSA technique while the (2,2,3) counter architecture has a total gate count about 20%-35% more than the CSA multiplier. It must be emphasised that these figures give a rough estimate of the speed and hardware cost trade-offs of the three schemes implemented in ECL technology based on the assumptions that functions of up to five variables can be implemented with a single ECL gate with equal

(a) Gate delay



(b) ECL gate count

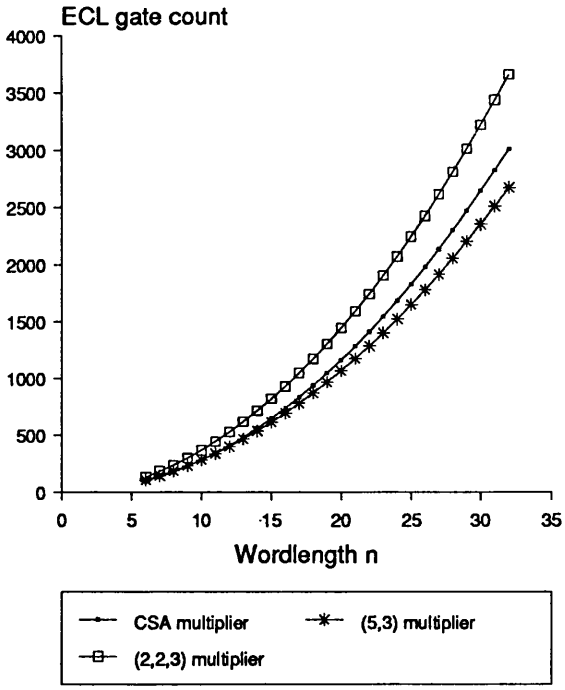


Figure 4.14. A comparison of different array multiplier schemes. **(a)** Relative gate delay, and **(b)** ECL gate count.

propagation delay. This may not give actual figures for practical ECL circuits and for certain logic families with different implementation methods since it depends among many things, on the maximum fan-in and fan-out, and tolerable noise margins under standard operating conditions. A fan-in of 4 is typical in most proven technologies which suggests that the curve of the (2,2,3) counter (which has four input variables) in Figure 4.14(b) would more likely to approach that for practical circuits.

From a VLSI implementation point of view both the (5,3) counter and (2,2,3) counter architectures are quite close to the conventional CSA multiplier although there are more than one type of cell used and slightly more wiring between them. The employment of the folded array of partial product bits which gives the (5,3) counter architecture a regular structure in terms of counter size and distribution of partial product terms, is retained in the (2,2,3) counter technique. Both the architectures are easily expandable to larger wordlengths by adding extra rows of cells to fit the specific application.

It is interesting to note that in all the megacells except $C_{i,i}$ of the (2,2,3) counter architecture, the S1 output's propagation delay, in principle would be faster than the higher carry outputs (C1 and C2) since logically, S1 is merely an EX-OR function of two variables as illustrated in Figure 4.7. The implication of this is that, because the delay path through the megacells traverses a majority of alternate logic blocks of full-adders and (2,2,3) counters, the likely slower delay of the modified full-adder component of a (2,2,3) counter would be compensated by the faster EX-OR gate of the (2,2,3) counter of the next stage. This makes the (2,2,3) counter multiplier a more attractive proposition than the (5,3) counter architecture since the speed requirement on the (2,2,3) counter is less stringent than the (5,3)

counter. Besides, the (2,2,3) counter, having four input variables as opposed to five for the (5,3) counter could be designed in practice with a faster propagation delay.

For those multiplications which are based on signed binary numbers, the conversion from unsigned multiplication to signed two's complement is simple for both architectures. For instance, by applying the usual technique of two's complement addition [4.46], a two's complement (5,3) counter multiplier can be realized, without disturbing the regularity of the array simply by inverting the partial product terms in the topmost row cells [4.30]. A similar type of conversion can be made for the (2,2,3) counter multiplier since the distribution pattern of the partial product bits is retained.

In this project the (5,3) counter and (2,2,3) counter architectures are studied in a different perspective, principally the efficiency of operation speed of the counters and the viability of the architecture in the fast bipolar ECL technology. Several methods for realising the counters both at the logic and circuit level are investigated for speed, noise immunity and ease of implementation for VLSI. This subject is discussed in subsequent chapters.

4.6. Summary

Extension of the conventional full-adder CSA approach to higher order parallel (p,q) counters has been studied in this chapter. A novel iterative (5,3) counter multiplier recently reported is highlighted and from this a (2,2,3) counter developed in this project are assessed in terms of speed and their attractiveness for VLSI implementation.

The study shows that both the (5,3) counter and the (2,2,3) counter architectures are quite close to conventional array multipliers from a

VLSI implementation point of view. Although more than one type of cell is involved in the architectures, there is a regular interconnection between cells and distribution of partial product bits, and both the architectures are easily expandable. In terms of speed, assuming the counters operate at a comparable speed as a full-adder the (5,3) counter architecture offers a speed enhancement of nearly a factor of 2 whilst the (2,2,3) counter architecture is found to give a significant improvement over that of conventional CSA multiplier for large operand wordlength.

REFERENCES

[4.1] S. Waser, " High speed monolithic multiplier for real time digital signal processing, " Computer, Vol. 11, no. 10, pp. 19-29, Oct. 1978.

[4.2] S. Waser & A. Peterson, " Real-time processing gains ground with fast digital multiplier ", Electronics, pp. 93-99, Sept. 29, 1977.

[4.3] W. Bucklen, J. Eldon, L. Schirm & F. Williams, " Single-chip digital multipliers form basic DSP building blocks ", EDN, PP.153-163, April 1, 1981.

[4.4] S.L. Freeny, " Special purpose hardware for digital filtering ", Proc. IEEE, pp. 633-648, 1975.

[4.5] L. Dadda, " Some schemes for parallel multipliers, " Alta Frequenza, Vol. 34, pp. 349-356, Mar. 1965.

[4.6] W.J. Stenzel, W.J. Kubitz & G.H. Garcia, " A compact high-speed parallel multiplication scheme, " IEEE Trans Comput., Vol. C-26, pp. 948-957, Oct. 1977.

[4.7] D. Ferrari & R. Stefanelli, " Some new schemes for parallel multipliers, " Alta Frequenza, Vol. 38, pp. 843-852, Nov. 1969.

[4.8] L. Dadda, " On parallel digital multipliers ", Alta Frequenza, pp. 574-580, Oct. 1976.

[4.9] S. Singh & R. Waxman, " Multiple operand addition and multiplication ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 113-120, Feb. 1973.

[4.10] A. Svoboda, " Adder with distributed control ", IEEE Trans. Computers, Vol. C-19, No. 8, pp. 749-751, August 1970.

[4.11] E.E. Swartzlander, Jr., " Parallel Counters ", IEEE Trans. Computers, Vol. C-22, No. 11, pp. 1021-1024, Nov. 1973.

[4.12] C.C. Foster & F.D. Stockton, " Counting responders in an associative memory ", IEEE Trans. Computers, pp. 1580-1583, Dec. 1971.

[4.13] L. Dadda, " Composite parallel adders ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 942-946, October 1980.

[4.14] S. Dormido & M. A. Canto, " Synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-30, No. 9, pp. 699-703, September 1981.

[4.15] H. Kobayashi & H. Ohara, " A synthesizing method for large parallel counters with a network of smaller ones ", IEEE Trans. Computers, Vol. C-27, No. 8, pp. 753-757, August 1978.

[4.16] S. Dormido & M. A. Canto, " An upper bound for the synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-31, No. 8, pp. 802-805, August 1982.

[4.17] D. G. Gajski, " Parallel Compressors ", IEEE Trans. Computers, Vol. C-29, No. 5, pp. 393-398, May 1980.

[4.18] C.P. Lerouge et al, " A fast 16 bit NMOS parallel multiplier ", IEEE Journal Solid-State Circuits, Vol. SC-19, No. 3, pp. 338-342, June 1984.

[4.19] M. Hatamian & G.L. Cash, " A 70-MHz 8-bit x 8-bit parallel pipelined multiplier in 2.5-um CMOS ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 4, pp. 505-513, August 1986.

[4.20] K. Washio et al, " 2.7ns 8 x 8-bit parallel array multiplier using sidewall base contact structure ", IEEE J. Solid-State Circuits, Vol. SC-22, No. 4, pp. 613-614, Aug. 1987.

[4.21] M. Uya, K. Kaneko & J. Yasui, " A CMOS floating point multiplier ", IEEE J. Solid-State Circuits, Vol. SC-19, No. 5, pp. 697-701, Oct. 1984.

[4.22] Y. Harata et al, " High-speed multiplier using a redundant binary adder tree ", Proc. IEEE ICCD, pp. 165-170, Oct. 1984.

[4.23] T.G. Noll et al, " A pipelined 330-MHz multiplier ", IEEE J. Solid-State Circuits, Vol. SC-21, No. 3, pp. 411-416, June 1986.

[4.24] A.R. Hurson, B. Shirazi & S.H. Pakzad, " VLSI layout design of a systolic multiplier unit for 2's complement numbers ", Proc. IEEE ICCD, pp. 354-358, 1985.

[4.25] F.J. Taylor, " A VLSI residue arithmetic multiplier ", IEEE Trans. Computers, Vol. C-31, No. 6, pp. 540-546, June 1982.

[4.26] Y. Oowaki et al, " A sub-10-ns 16x16 multiplier using 0.6-μm CMOS technology ", IEEE J. Solid-State Circuits, Vol. SC-22, No. 5, pp. 762-765, Oct 1987.

[4.27] S. Nakamura, " A single chip parallel multiplier by MOS technology ", IEEE Trans. Computers, Vol. 37, No. 3, pp. 274-282, March 1988.

[4.28] N.R. Shanbhag & P. Juneja, " Parallel implementation of a 4x4-bit multiplier using modified Booths algorithm ", IEEE J. Solid-State Circuits, Vol. 23, No. 4, pp. 1010-1013, August 1988.

[4.29] A. Habibi & P.A. Wintz, " Fast Multipliers, " IEEE Trans. Computer, Vol.C-19, pp. 153-157, Feb. 1970.

[4.30] S. Nakamura, "Algorithms for iterative array multiplication," IEEE Transactions on computers, Vol. C-35, No.8, August,1986.

[4.31] D.P. Agrawal, " High-speed arithmetic arrays ", IEEE Trans. Computers Vol. C-28, No. 3, pp. 215-224, March 1979.

[4.32] A. Dhurkadas, " Faster parallel multiplier ", Proc. of the IEEE, Vol. 72, No. 1, Jan. 1984.

[4.33] R. De Mori, " Suggestion for an IC fast parallel multiplier ", Electronics Letters, pp. 50-51, Vol. 5, No.3, 6th Feb. 1969.

[4.34] H.H. Guild, " Fully iterative fast array for binary multiplication and addition ", Electronics Letters, pp. 263, Vol. 5, No. 12, 12th June 1969.

[4.35] J.C. Hoffman, B. Lacaze & P. Csillag, " Iterative logical network for parallel multiplication ", Electronics Letters, pp. 178, Vol. 4, No. 9, 3rd May 1968.

[4.36] K.J. Dean, " Logical circuits for use in iterative arrays ", Electronics Letters, pp. 81-82, Vol. 4, No. 5, 8th March 1968.

[4.37] J. Deverell, " Pipeline iterative arithmatic arrays ", IEEE Trans. Computers, pp. 317-322, March 1975.

[4.38] S. D. Pezaris, " A 40ns 17-bit by 17-bit array multiplier ", IEEE Trans. Computers, pp. 442-447, April 1971.

[4.39] J. C. Majithia & R. Kitai, " An iterative array for multiplication of signed binary numbers ", IEEE Trans. Computers, pp. 214-216, Feb. 1971.

[4.40] R. De Mori, " A parallel structure for signed-number multiplication and addition ", IEEE Trans. Computers, pp. 1453-1454,Dec. 1972.

[4.41] I.D. Deegan, " Cellular multiplier for signed binary multipliers ", Electronics Letters, Vol. 7, No. 151, pp. 436-437, 29th July 1971.

[4.42] J.A. Gibson & R.W. Gibbard, " Synthesis and comparison of two's complement parallel multipliers ", IEEE Trans. Computers, pp. 1020-1027, Oct. 1975.

[4.43] N.G. Kingsbury, " High speed binary Multiplier ", Electronics Letters, Vol. 7, No. 10, pp. 277-278, 20th May 1971.

[4.44] A.R. Meo, " Arithmetic networks and their minimization using a new line of elementary units ", IEEE Trans. Computers, Vol. C-24, No. 3, pp. 258-280, March 1975.

[4.45] C.S. Wallace, " A suggestion for parallel multipliers, " IEEE Trans. Electron. Comput. Vol. EC-13, pp 14-17, Feb. 1964.

[4.46] C.R. Baugh & B.A. Wooley, " A two's complement parallel array multiplication algorithm, " IEEE Trans. Computer, Vol. C-22, pp.1045-1047, Dec. 1973.

CHAPTER 5

REALISATION OF PARALLEL COUNTERS

5.1. Introduction

Large higher order parallel (p,q) counters, having a multitude of inputs are very useful in realising various kinds of parallel arithmetic operations [5.18-5.23]. The concept of employing higher order parallel (p,q) counters [5.1-5.5] to enhance the speed of iterative array multipliers has been studied and demonstrated in the last chapter. Assuming the counter cells operate at a comparable speed to that of a full-adder, the (5,3) and (2,2,3) counter multiplier schemes are shown to offer significant speed improvements over conventional carry-save array (CSA) multipliers. Thus the main issue now is how such a counter could be designed in the fast bipolar ECL technology.

A full-adder [5.6] can be considered as a (3,2) counter (it is the 'smallest' one). Irrespective of the technology used, the full-adder is easy to design by purely combinatorial logic techniques and the associated multiplier architectures are simple and regular which makes them ideal for VLSI implementation. However, the complexity involved in implementing higher order counters is well-known [5.7]. Considerable attention has been paid to methods of synthesizing large parallel counters in the past. Previous work has concentrated on using look-up tables or ROMs [5.2], sequential circuits [5.8,5.9], networks of full-adders or smaller counters [5.7,5.10-5.15] and threshold logic [5.1,5.4,5.16,5.17,5.24-5.26,5.49,5.51]. These methods however, result in counters that are too costly and too slow to gain any significant improvement in the speed of digital multipliers. A more detailed review of previous methods of synthesizing parallel (p,q) counters has been

discussed in Chapter 2 (see section 2.4.3).

In this chapter two forms of parallel counters, the (5,3) and (2,2,3) counter are examined with emphasis on a more combinatorial approach of implementation. Two different techniques based on series-gated ECL and threshold logic were investigated. The complexity and efficiency of operation speed of the two counters using these techniques are stressed. A novel circuit technique which overcomes the earlier problem of needing high fan-in weight threshold gates is presented. Also described is a technique of mapping a logic function on to series-gated ECL suitable for software implementation.

5.2. Design considerations of the (5,3) and (2,2,3) counters

The viability of the (5,3) counter and (2,2,3) counter multiplier architectures described in Chapter 4 depends heavily on the efficiency of operation speed and the complexity of a counter cell. Ideally, the counter cell should have a delay and complexity comparable to that of a full-adder cell in the fast bipolar ECL technology [5.27].

In order to achieve a fast implementation of the (5,3) and (2,2,3) counter cell a more combinatorial approach rather than the inherently slower methods reviewed in Chapter 2 is considered. At the logic design level considerations, the speed of a combinational circuit can be described by the number of gate levels, or the height of the circuit, and the complexity can be represented by the number of gates and inputs. In this sense, a circuit realised by a sum-of-products form or its equivalent forms, such as NOR/NAND form, should have the fastest speed. A full-adder, realised in the sum-of-products form requires a total of 12 Boolean gates with three NOT, three two-input NOR, five three-input NOR, and one four-input NOR. The speed of such a cell can be three gate

delays, assuming no complemented inputs are available. A similar straight forward realization of the (5,3) counter and (2,2,3) counter, however, results in a cell of enormous complexity. For the (5,3) counter, a total of 44 gates with one 16-input NOR, two 10-input NOR, 16 five-input NOR, ten four-input NOR, and ten two-input NOR are needed. The speed, on the other hand, is still the same three gate delays. Therefore, at the logic design level, (5,3) counter cell design with the comparable speed of a full-adder is not difficult. This suggests that for those logic families with low impedance switching elements in which the increased fan-in/fan-out does not penalize speed, a fast (5,3) counter multiplier is possible with certain drawbacks in the hardware complexity. In many practical logic families, however, their performance in speed depends not only on the height of the circuit but also on the maximum fan-in and fan-out numbers.

In other words, the number of connections to a (p,q) counter, the complexity of the interconnection pattern, and the fan-in and fan-out capability present some limitations to the circuit and logic designer. The reduction in hardware complexity is crucial to realise a fast parallel counter. Thus, it is important to assess all the high speed 'tricks' available, how complex functions could be realised with little increase in complexity and power consumption, and how they can be utilised to design a parallel counter of reasonable complexity.

Two different techniques of implementing fast (5,3) and (2,2,3) counters are considered. The first looks at their realisation in the well-proven high-speed ECL technology to challenge the speed of a similar realisation of the fastest full-adder cell. Threshold logic, which is known to be efficient in synthesizing higher order parallel counters with minimum increase in speed and complexity is reconsidered in view of the

better bipolar processes of today.

5.2.1. Emitter-coupled logic

Emitter-Coupled logic (ECL) is one of the fastest silicon technology and is commonly employed in high performance integrated circuits. Although it has the disadvantage of high power consumption, ECL offers good logic functional complexity compared to other logic families. The fastest practical implementation of a full-adder in bipolar technology, or in silicon technology for that matter, is with a series gated ECL (Figure 3.6(a)). Any design of a (5,3) and (2,2,3) counter need to be critically compared with this.

In ECL, there are five different ways of getting multiple levels of gating, with essentially just one level of propagation delay. Three of these are series gating, emitter dotting (wire ORing) and collector dotting (wire ANDing). The fourth is the property that because of the differential nature of the ECL structure, a signal and its complement are both available as the gate's outputs. Other technologies require at least one other level of delay (an inverter) to obtain the complement. The fifth is the ability to parallel input transistors to create the OR/NOR function with little additional delay.

Series gating (cascode or stacked logic) [5.28] is a cost effective and convenient way to obtain NAND/NOR functions in ECL. In series gating one current source is used to drive two or more levels (corresponding to the number of variables of the function) of differential pairs forming a transistor tree. This permits a better speed-power product since the same current may be used several times per logic decision. An emitter follower is normally employed to give a high fan-out capability.

It is interesting to note that the delay behaviour of a series gated

ECL gate is dependent on the complexity of the transistor tree and at any given time is set by the number of steady input signals and propagating input signals which then determines the number of levels of differential pairs that are forced to switch. An analysis of the parameters that contributes to the differential pair delay has actually been studied thoroughly by Barna [5.36,5.37] where factors such as the the effects of collector-base capacitances are characterized, with about 30-40% accuracy. A more convenient and fast route to characterize differential delays accurately is to use computer circuit simulations such as SPICE [5.53]. However, as a rule-of-thumb the more complicated the transistor tree and the more levels of differential pairs are forced to switch, the higher the propagation delay. Thus to get as high a speed as possible, the transistor tree is normally minimized to eliminate redundant differential switches with the steady signals fed to the lowest levels in order to reduce the number of levels of differential delays that can be forced to switch.

In the next section, a simple technique of mapping a logic function on to series gated ECL with minimization of the transistor tree suitable for software implementation is given.

A repartitioned form of ECL called emitter-function logic (EFL) [5.27,5.30-5.34] gives an increase in logic function complexity at minimum increment in the device and circuit complexity. This is achieved without additional power consumption in comparison to the single function current switch. The technique was initially considered in the synthesis of parallel counters. Although it gives the advantage of minimizing the complexity of parallel counters through modularity of the design, it was found that because the differential pairs are driven in a single-ended fashion the actual delay of the counters, which would require at least

two levels of EFL gates is actually worse than the cascode ECL full-adder. In addition, the need to have a reference voltage for the single-ended differential switch would further complicates the design.

5.2.1.1. A technique for mapping a logic function on to cascode ECL

A top-down approach is employed in this technique. Rather than using the minimized Boolean equation as the mapping specification the technique looks at the truth-table of the function and works its way down the transistor tree in order to arrive at its minimized form.

Each differential switch can be visualized as a branch with three nodes (Figure 5.1). The coupled emitters can be thought of as an input node through which current can flow in, while the two collectors are considered as output nodes where the current can flow out. Which collector node the current will flow through is determined by which logic input (A or $\bar{A}$) is higher. If A is higher than $\bar{A}$ then current is steered through the collector node of A and vice-versa. Thus a tree of differential switches can be built up as shown in Figure 5.2(a) for a 3-variable function.

Current can be thought of as flowing from a reference node (corresponding to a current source node), through a path determined by

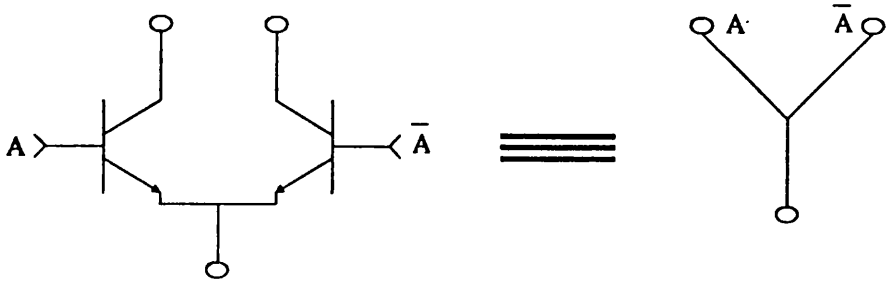


Figure 5.1. A differential switch equivalent symbol.

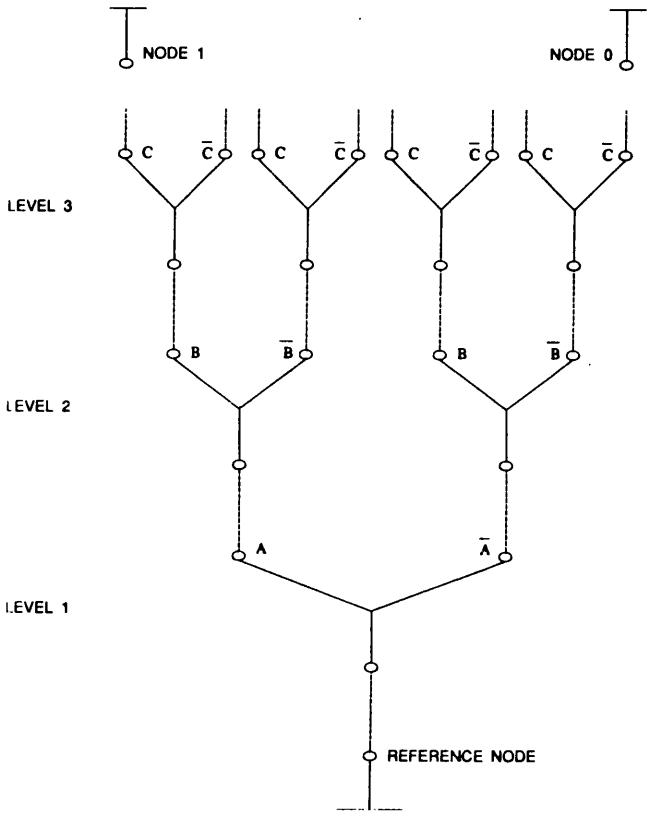


Figure 5.2(a). The possible current paths for a 3-variable function.

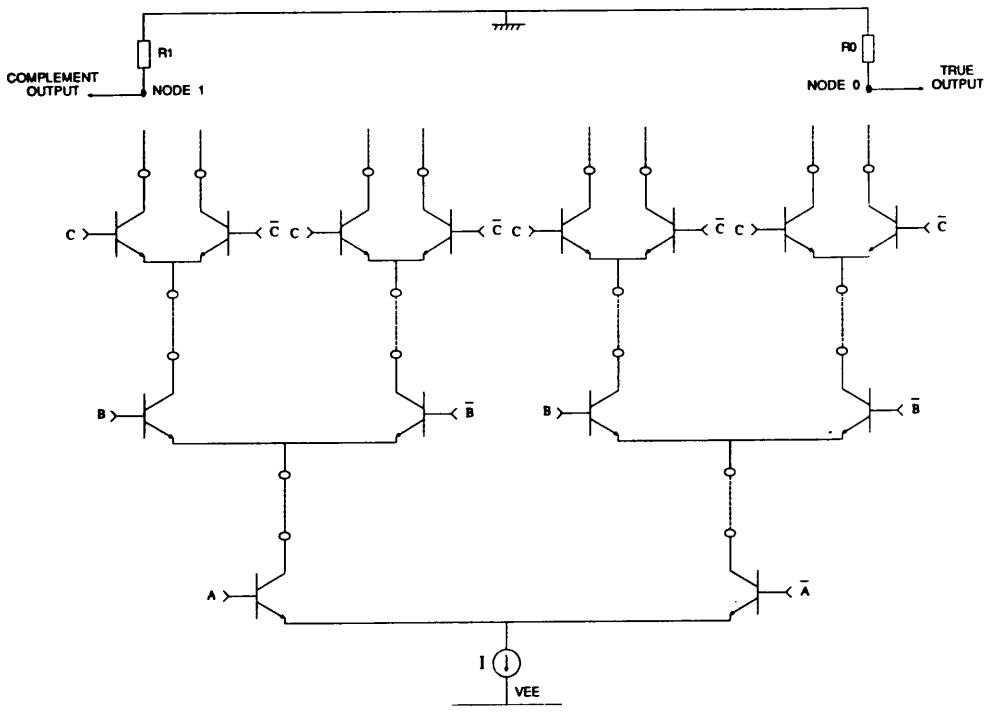


Figure 5.2(b). Equivalent electrical schematic.

the states of the inputs, up to either node 0 or node 1. The broken lines in Figure 5.2(a) shows the possible paths where the current can be steered. Each path represents the possible states of the logic inputs e.g. the path $\overline{A}$-$\overline{B}$-$\overline{C}$ corresponds to state 0 of a 3-variable function. For an n-variable function there are 2^n possible states, thus there are 8 possible paths for a 3-variable function as shown in Figure 5.2(a). As we show later, the broken lines are links that can be broken in the process of minimizing the tree.

At the top level, each output node can be connected either to node 1 or node 0 depending on what the output F corresponding to that state is. If F='1' then connect the appropriate output node to node 1 else if F='0' then connect it to node 0. Nodes 0 and 1 are actually equivalent to the electrical nodes of the two resistors R0 and R1, respectively (Figure 5.2(b)). This is true since if the current is steered to node 1, the true logic output of the gate is high and if it flows to node 0 the true logic output is low.

Thus there are two main steps involved in the mapping and synthesis of the tree. First, a full-blown realisation of the tree is done by connecting the highest level differential switches output current nodes to node 1 or 0 as defined by the output column of the truth-table of the function. Next, redundant differential switches are eliminated starting from the highest level down to the lower levels in order to obtain a minimized tree of the same function. An example is shown in Figure 5.3(a) for the Sum1 output of a (2,2,3) counter whose truth-table is illustrated in Table 4.4. Figures 5.3(b) and 5.3(c) illustrate the minimization process of the tree of four variables by eliminating redundant differential switches.

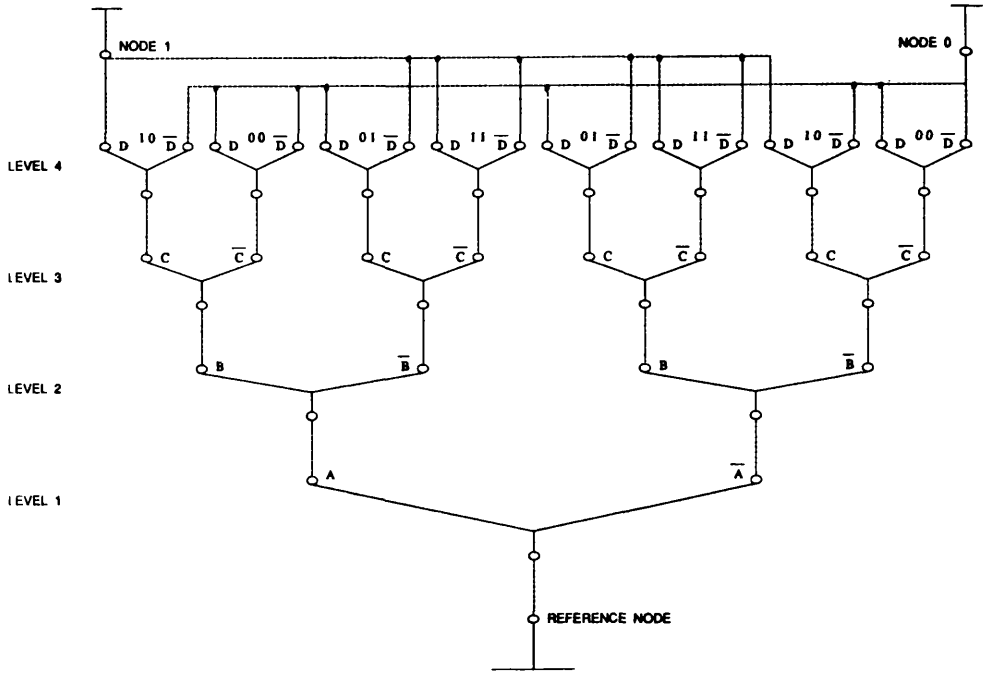


Figure 5.3(a). A full-blown tree of the Sum1 function of a (2,2,3) counter.

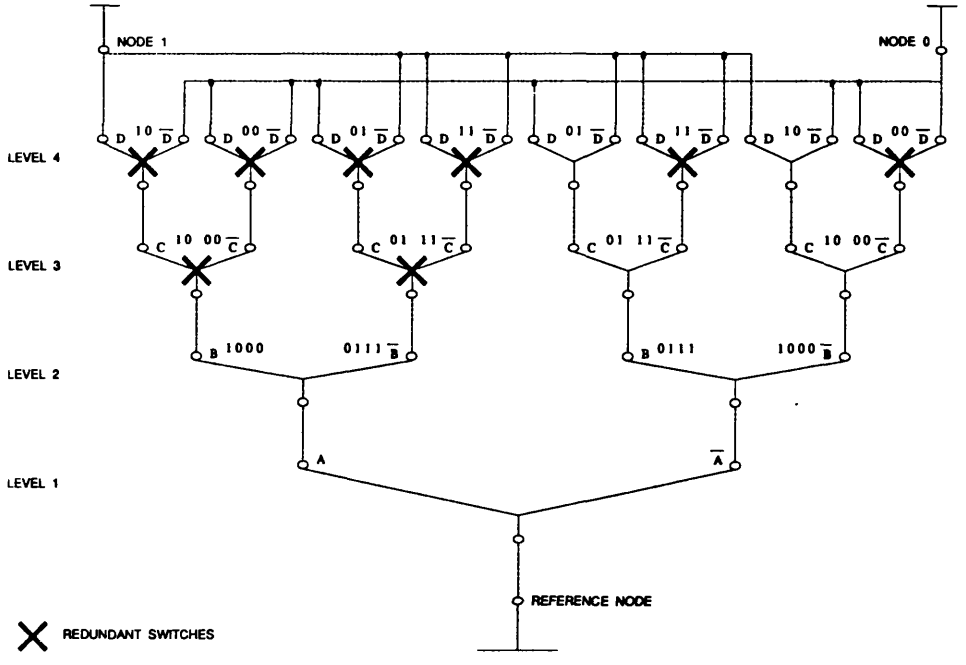


Figure 5.3(b). Identification and elimination of redundant switches.

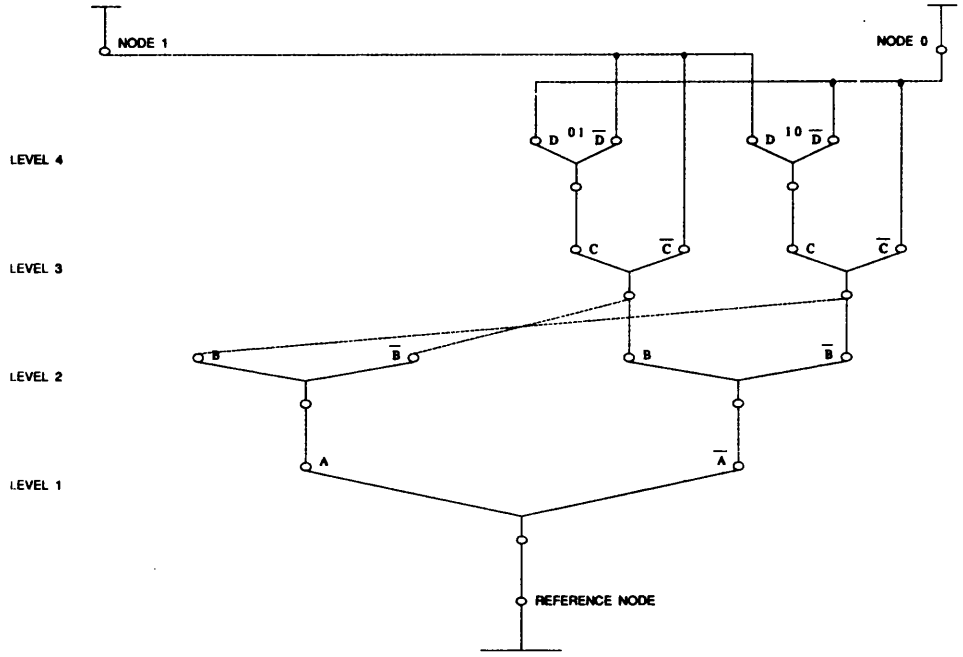


Figure 5.3(c). A minimized tree of the same function.

There are 2 types of redundant differential switches :

(i) Differential switches at any level whose two output nodes are connected both to node 1 or both to node 0. What this means logically is that the output of the differential switch is independent of the logic input concerned. Thus all such switches can be eliminated altogether and the lower level output nodes can be connected directly to node 1 or 0.

(ii) Beginning with the highest level, differential switches which have their two output nodes connected to node 1 and 0 in the same configuration i.e. the true output node of each switch is connected to node 1 and the complement output node connected to node 0, or vice-versa. In this case all but one of these differential switches at the level concerned can be eliminated. The appropriate lower level output nodes are then connected to the input current node of this remaining differential

switch. This process can be repeated at the lower levels by noting the equivalence of the differential switches at these levels as far as the path to node 1 or node 0 is concerned.

Verification of the minimized tree can be done by looking at each state of the inputs and its corresponding output bit from the truth-table of the function and then looking at the tree to see the existence of a path for the current from the reference node to node 1 or 0 accordingly. Notice that in order to get a simple tree, a long series of 0's and/or 1's or frequent occurrences of 01 or 10 in the output column of the truth-table of the function are necessary. For certain functions such as the (2,2,3) counter with input bits of different weights, depending on which variables are mapped to the various levels, differing transistor trees can be obtained since the output column of the truth-table of the function would change. The higher level differential switches are inherently faster than the lower levels in switching the gate's logical output and thus a computer program implementation of the technique should prove useful in order to fully explore the trade-offs between speed and complexity for the gate of such functions. As we have shown above the algorithm to do this is fairly simple with the steps clearly well-defined and thus the technique can be efficiently implemented in software.

5.2.2. Threshold logic

Threshold logic [5.16-5.18,5.24-5.26,5.35,5.38-5.52] is of great interest since they are known to have the edge over conventional Boolean realisations in terms of reduced numbers of components, improved speed for higher logic complexity and are versatile. Furthermore, these gates are compatible with normal digital LSI technology.

A threshold gate has binary inputs and outputs just like any other

logic gate. The difference however, is that in the threshold gate the inputs are weighted, and eventually, a binary decision made as to whether the total weight is more or less than some reference (see Figure 2.8). This principle of weighting and summing the inputs rather than simply noting the presence of all inputs as high (as in an AND gate) or one input high (as in an OR gate) is the reason that a threshold gate can tell more about the state of the inputs, thus providing greater "logic power".

The attractiveness of using threshold logic is clearly illustrated in Figure 2.9. The technique is of great interest since a counter size of up to a (7,3) could be realised by just adding extra rows of inputs and without any significant increase in gate count or propagation delay, assuming the gates have a high fan-in weight. Apart from that it has none of the irregular and complex wiring of equivalent function Boolean gates, and as such is ideal for VLSI implementation because wiring occupies a lot of silicon area. However, the implementation of fast parallel counters in threshold logic is often hampered by the need to have a high fan-in weight threshold gate. This is evident from Figure 2.9 where a gate with a total fan-in weight of 9 and 11 is required for the CY_1 and S functions, respectively of a (5,3) counter. The maximum practical fan-in weight of traditional threshold gates is limited to about 7 [5.38,5.39] as a consequence of the need to have adequate noise margin and to avoid the transistors from saturating which would degrade their propagation delay. In section 5.3.2 a novel circuit technique is proposed which overcomes this maximum fan-in problem through the partial use of negative weighted inputs.

5.3. (5,3) counter cell

In the design of the (5,3) counter cell it is important to note that out of the five input signals to the counter, two of the signals are partial product terms $a_i b_j$ and $a_j b_i$, which are steady while the rest are propagating signals from the outputs of previous stage cells. This property of the signals has great implications on the efficient design of the counter cell.

5.3.1. ECL implementation of a (5,3) counter

Nakamura [5.29] has come up with an optimised design of the (5,3) counter at the logic level by separating the signals of the two partial product terms $a_i b_j$ and $a_j b_i$ associated with each cell from the input signals A, B and C from the outputs of the previous stage (see Figure 4.5(a)). Inputs A, B and C are referred to as propagating signals since they are ready only after the completion of operations in the previous stage cells. The other two inputs $a_i b_j$ and $a_j b_i$ are considered as steady signals because once the primary input operands **a** and **b** to the multiplier stabilise, $a_i b_j$ and $a_j b_i$ are ready right after one AND gate delay, and there are no propagation delays associated with them. This property of the signals is fully utilised to give an optimized design for the (5,3) counter cell. The circuits for the steady signals do not have to be extremely fast because this part only adds a constant delay for the entire multiplication speed. The other part, which is most sensitive to speed, has to be designed by fast, flat circuits; but with only propagating signals, this part can be much simpler and faster than the straightforward, brute force realisation of the (5,3) counter.

Consider all possible combinations of values on the steady signals $a_i b_j$ and $a_j b_i$ and the effect of each of the states on the outcome of the

counter's outputs S, CY_1 and CY_2 (see Table 4.2, noting that here a_ib_j = D and a_jb_i = E). Examination of Table 4.2 as similarly considered by Nakamura yields Table 5.1, which represents a simple, minimized form of the (5,3) counter where :

S^0 = sum function of a full-adder

C^0 = carry function of a full-adder

$C^1 = A\bar{C} + B\bar{C} + A\bar{B}$

$C^2 = ABC$

The equivalence of row 2 and 3 of Table 5.1 is expected since the outputs depend on the weight of the steady signals i.e. the number of 1's.

An optimized logic design of the (5,3) counter is illustrated in Figure 5.4(a). This is similar to the logic design of the (5,3) counter developed by Nakamura for implementation in CMOS technology except that, in ECL, as we shall see later the a_ib_j and a_jb_i need not be reduced to one select signal for the multiplexers (Figure 5.4(b)). The problem thus reduces to the efficient design of logic functions C^1, C^2 and the multiplexer circuits.

Logic functions C^1 and C^2 could be realised in series-gated ECL as

a_ib_j	a_jb_i	S	CY_1	CY_2
0	0	S^0	C^0	0
0	1	$\overline{S^0}$	C^1	C^2
1	0	$\overline{S^0}$	C^1	C^2
1	1	S^0	$\overline{C^0}$	C^0

Table 5.1 Minimized logic function of a (5,3) counter

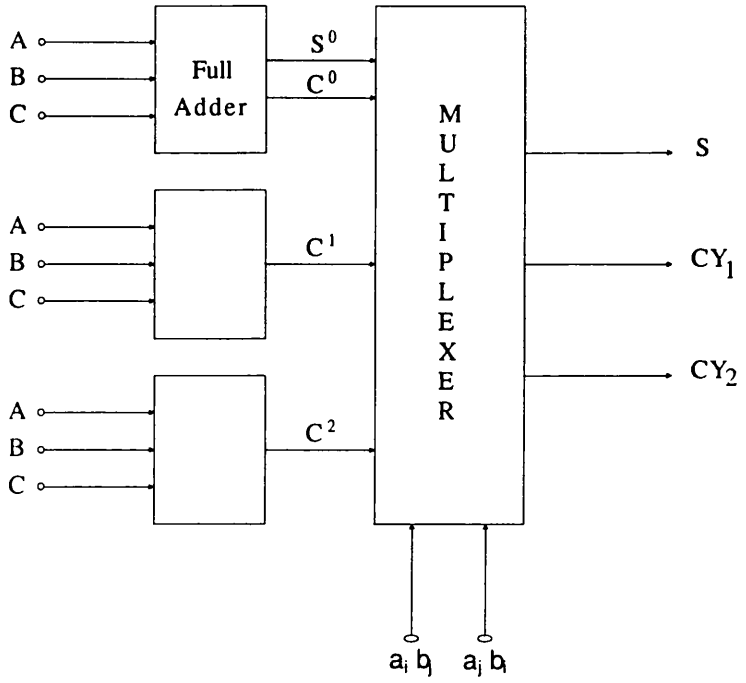


Figure 5.4(a). An optimized logic design of a (5,3) counter.

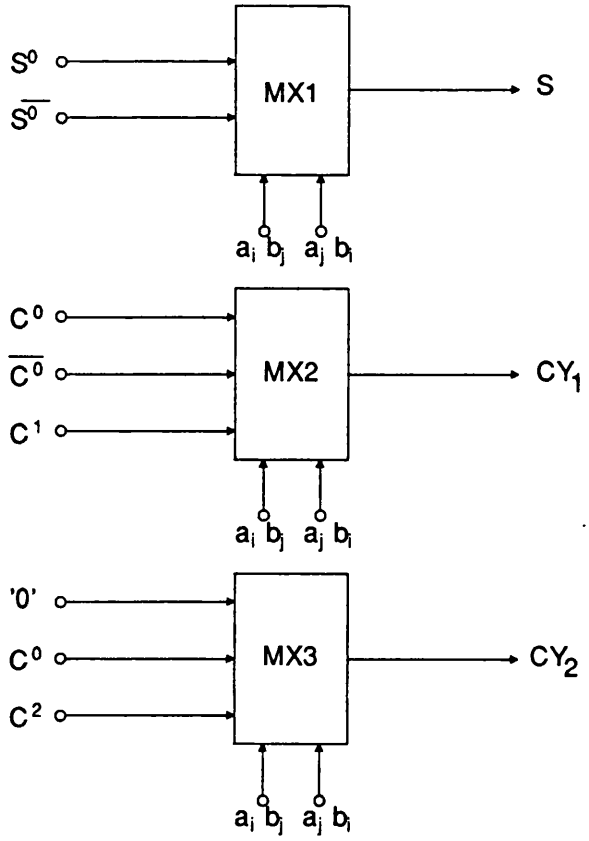
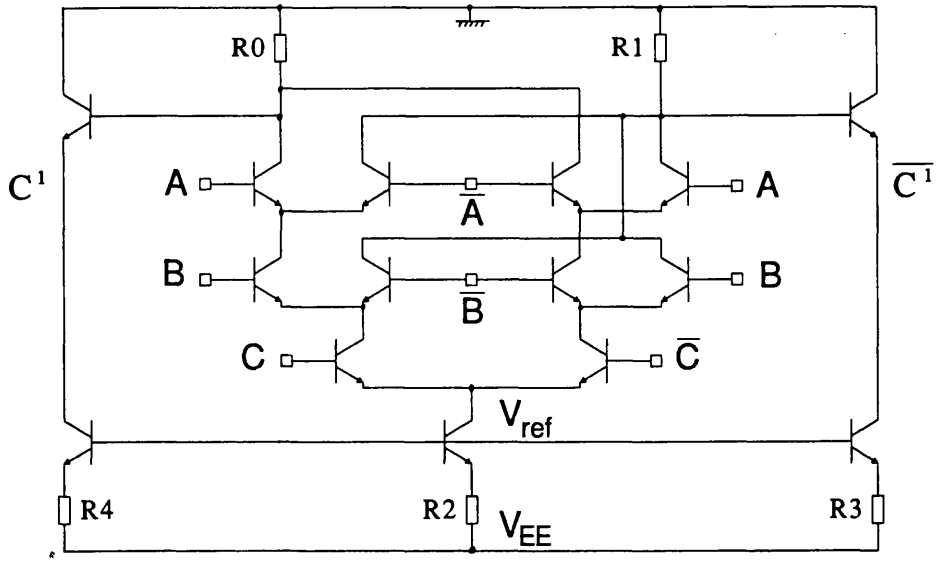


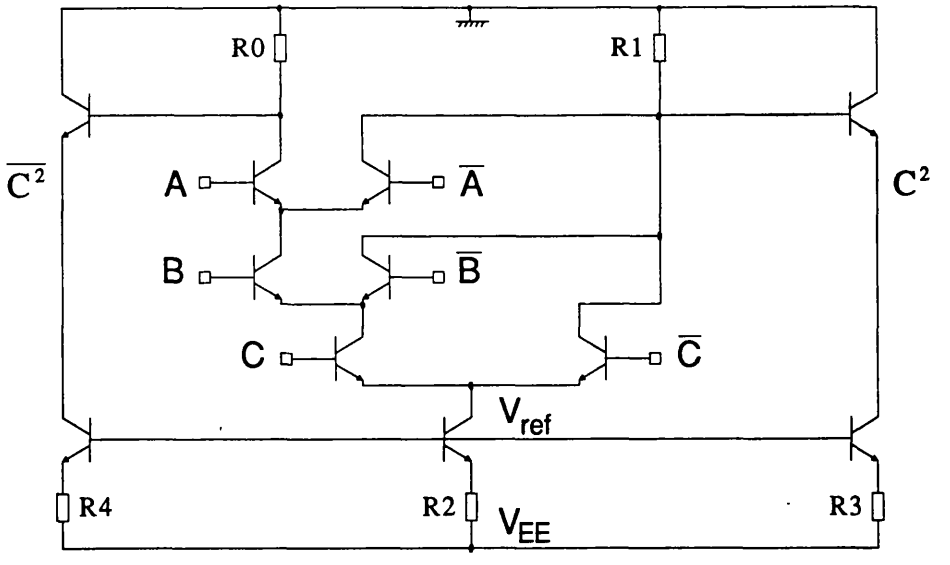
Figure 5.4(b). Multiplexers for the (5,3) counter.

commonly implemented for the full-adder cell with the same propagation delay and complexity. The circuits to realise these are shown in Figure 5.5(a) and 5.5(b). A high-speed design of the multiplexer circuits MX1, MX2 and MX3 are similarly implemented in series-gated ECL and are schematically shown in Figure 5.5(c), 5.5(d) and 5.5(e), respectively.

Bearing in mind that each of the logic blocks has the delay of a full-adder, this suggests that the (5,3) counter has twice the delay of a full-adder. However, it must be stressed that the propagation delay behaviour of a cascode ECL gate depends to a large extent on the number of levels of differential switches in the transistor tree that are liable to switch. In other words, the delay is dependent on the number of steady and propagating signals. The multiplexer circuits, having two steady signals $a_i b_j$ and $a_j b_i$ at the two lowest levels of the transistor tree should be faster than a full-adder having all three inputs changing. Because the two steady signals of the partial product bits set the current path of the two lowest level of the transistor tree at the very early stage of each computation, the input signals to each multiplexer just pass through with only one level of differential delay. This implies that the (5,3) counter may have a delay between one and two times the delay of a full-adder. Thus, a closer examination of the number of steady signals and propagating signals of the (5,3) counter multiplier needs to be done and critically compared to the signals of a full-adder CSA multiplier before a true, accurate measure of the cell delay could be obtained; this would then reflect any improvements in the speed of the (5,3) counter multiplier over the conventional CSA scheme. Besides, the critical delay path of the (5,3) counter multiplier is also determined by the delay of the modified full-adder diagonal cells which is basically a critical component of a (2,2,3) counter. Thus, a characterization of not

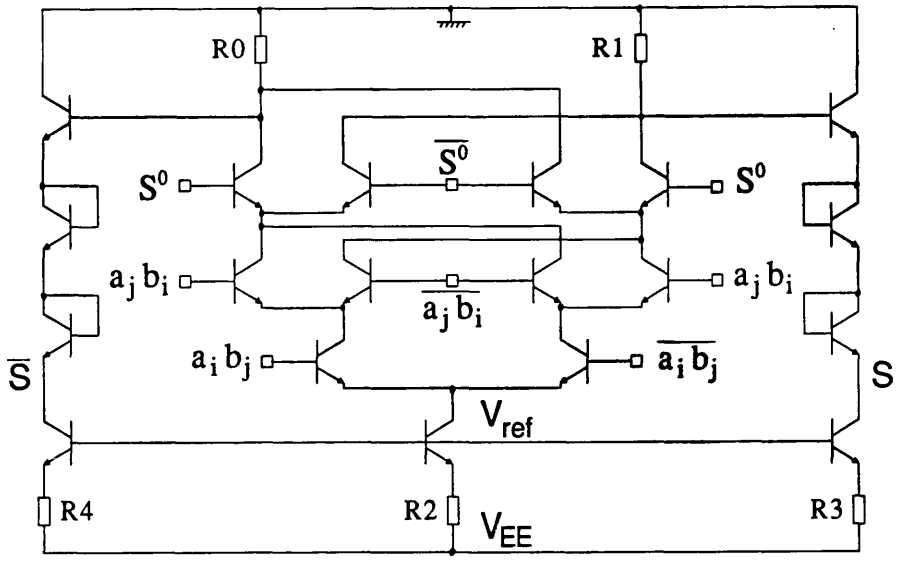


(a)

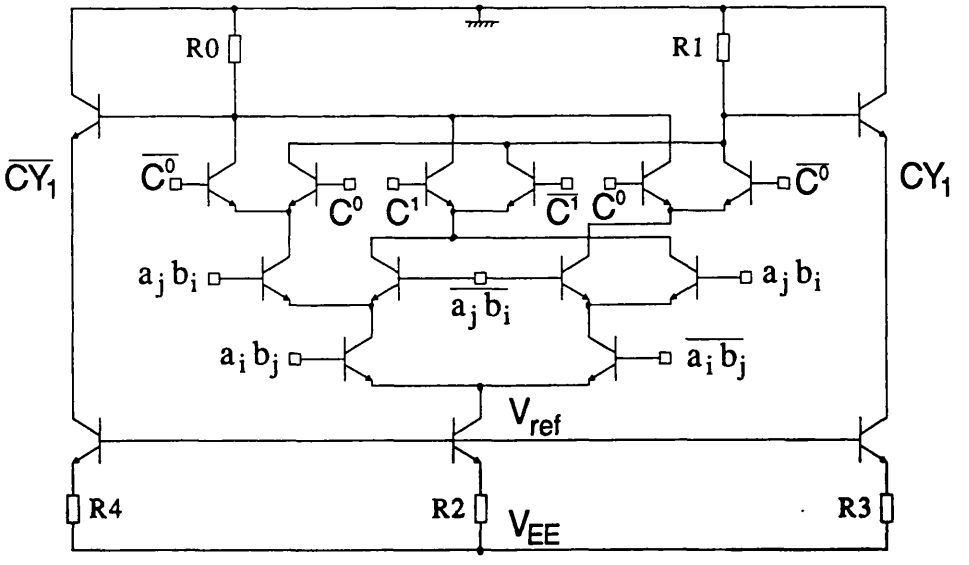


(b)

Figure 5.5. ECL implementation of the various logic blocks of a (5,3) counter: (a) C^1 gate, (b) C^2 gate (c) MX1 gate, (d) MX2 gate, and (e) MX3 gate.

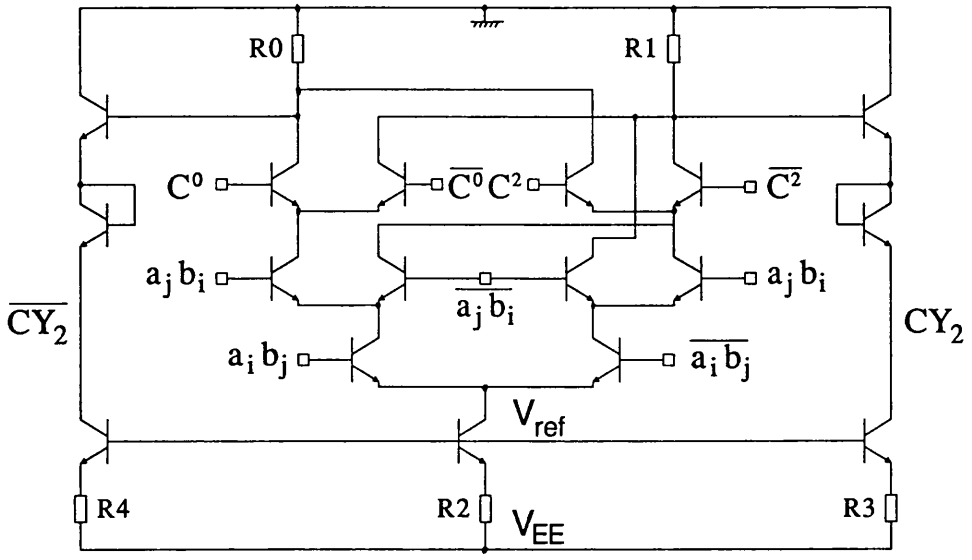


(c)



(d)

Figure 5.5. (Continued)



(e)

Figure 5.5. (continued)

just the (5,3) counter delay but also the delay of the modified full-adder component of a (2,2,3) counter is required before the efficiency of operation speed of the (5,3) counter multiplier could be reliably evaluated. The results are discussed in the next chapter. In terms of complexity, the (5,3) counter is about three times more complex than a full-adder.

In the proposed design in ECL, the process of computing the steady signals before the propagating signals reach the cell yields a cell of low complexity and delay since the critical part of the circuit sensitive to the performance in speed can be made simpler and faster. Therefore, the larger the number of steady signals and the less the number of propagating signals of a cell, the faster the speed of the cell. In this sense, the (5,3) counter cell design is advantageous against the

C_1^p	S^p	S	CY_1	CY_2
0	0	S^0	C^0	0
0	1	$\overline{S^0}$	C^1	C^2
1	0	S^0	$\overline{C^0}$	C^0
1	1	$\overline{S^0}$	$\overline{C^1}$	C^3

Table 5.2 Logic table of a (6,3) counter.

C_2^p	C_1^p	S^p	S	CY_1	CY_2
0	0	0	S^0	C^0	0
0	0	1	$\overline{S^0}$	C^1	C^2
0	1	0	S^0	$\overline{C^0}$	C^0
0	1	1	$\overline{S^0}$	$\overline{C^1}$	C^3
1	0	0	S^0	C^0	1

Table 5.3. Logic table of a (7,3) counter.

full-adder design because the ratio of the number of steady signals and propagating signals for the (5,3) counter is 2/3, while the ratio for the full-adder is 1/2. The same three-input and three-output propagation signals per cell can accommodate not only two steady signals but also three or at most four steady signals, corresponding to a (6,3) counter

and (7,3) counter, respectively. A similar study of the dependence of the outputs of these counters on the weight of the steady signals of the partial product bits yields Table 5.2 and Table 5.3, where :

$C^3 = A + B + C$,

and S^p, C_1^p and C_2^p, which represent the select signals of the multiplexers are the sum and carries obtained from the addition of the steady partial product bits (only C_2^p is possible in the (7,3) counter which has four partial product bits). The computation of the steady partial product signals to give S^p, C_1^p and C_2^p is necessary to reduce the complexity and delay of the multiplexers to give the same counter speed as the (5,3). However, in terms of cost-effectiveness the (5,3) is the most attractive since the (6,3) and (7,3) counters require more gates for the computation of S^p, C_1^p and C_2^p, although they have the same propagation delay.

5.3.2. Threshold logic implementation of a (5,3) counter

A circuit technique which overcomes the maximum fan-in problem through partial use of negative-weighted inputs is proposed. In threshold logic the same logic function could be accomplished through the inversion of an input variable and the sign of its weight along with an offset in the threshold value [5.35]. Therefore, if input x_i is inverted, the sign of its weight w_i must be changed, and the threshold t must also be offset to $(t\text{-}w_i)$ in order that these changes may still express the same logic function. Using this formula, an alternative logic for the (n,3) counters could be formed as shown in Figure 5.6.

Assume a general threshold gate of threshold T which has as positive weighted inputs $X_1,....,X_n$ of arbitrary weights $W_{x1},....,W_{xn}$, respectively and negative weighted inputs $Y_1,....,Y_p$ of arbitrary weights $-W_{y1},....,-W_{yp}$, respectively (Figure 5.7). The novel circuit to

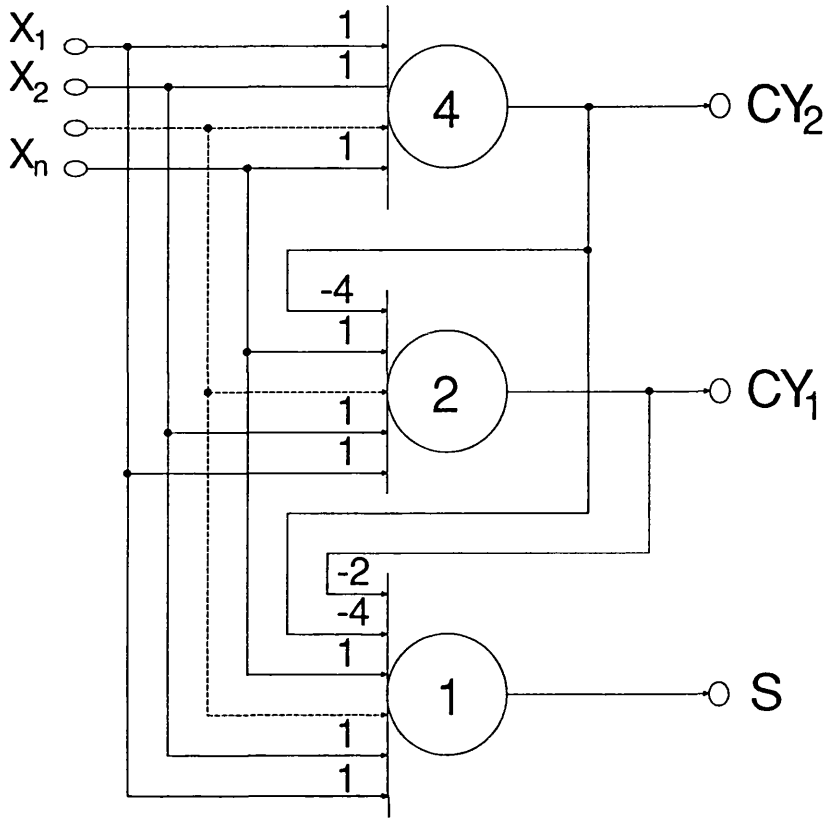


Figure 5.6. Alternative threshold logic realization of a (n,3) counter (where $4 \leq n \leq 7$).

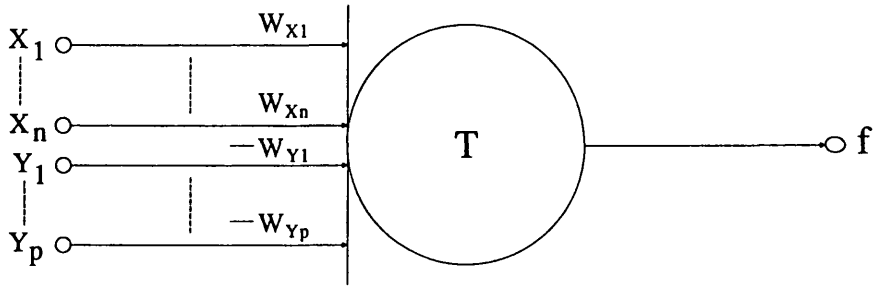


Figure 5.7. Threshold gate symbol with positive and negative weighted inputs.

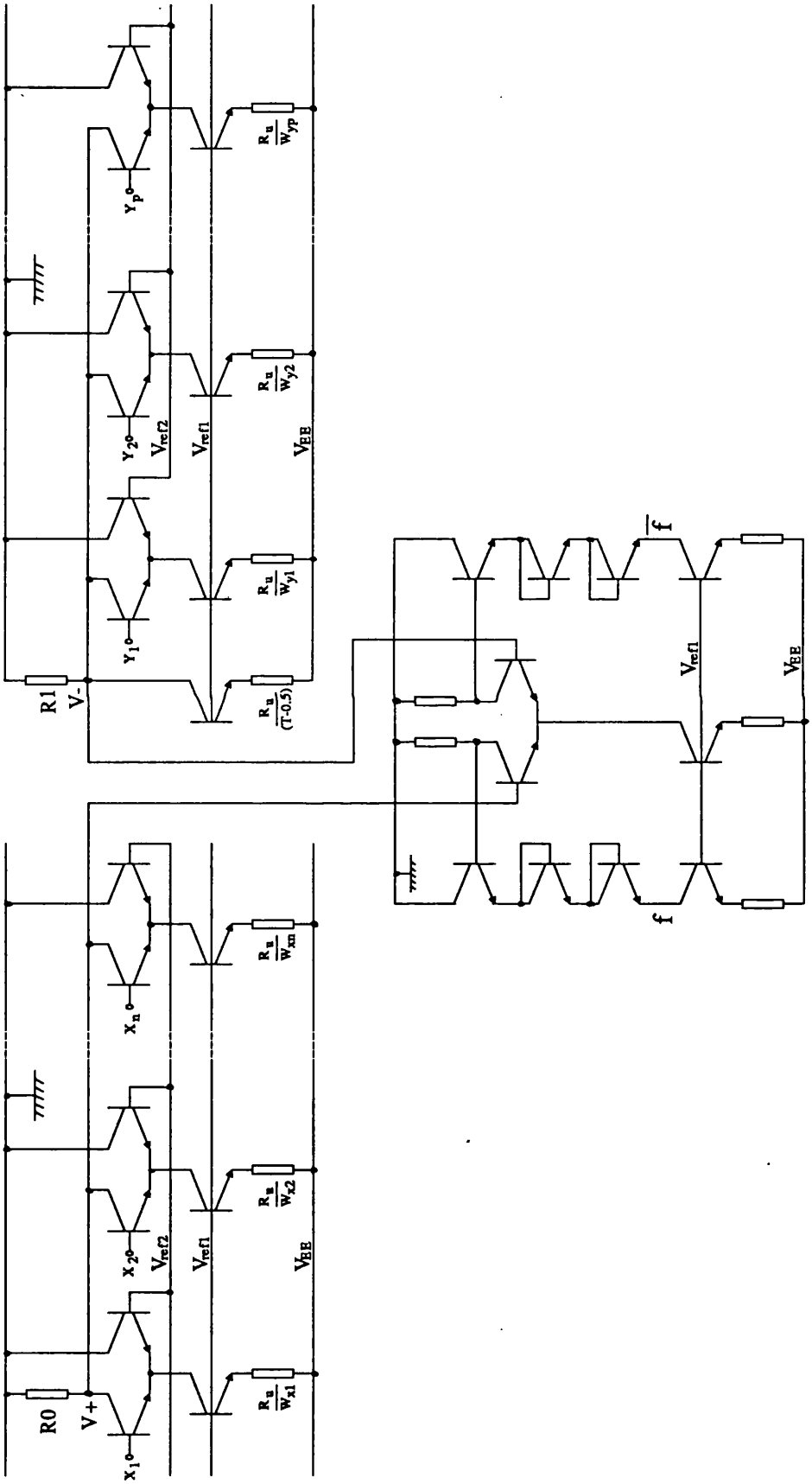


Figure 5.8. A novel threshold logic circuit technique to realise positive and negative weighted inputs.

functionally realise threshold gates with negative weights is illustrated in Figure 5.8. The circuit technique is similar to traditional threshold circuits except that the negative weighted inputs are separated from the positive weighted ones by two distinct ECL circuits.

Each input feeds a differential switch that serves the dual purpose of :

(1) deriving a weighted current (by virtue of a emitter degenerated current source)

(2) making a binary decision as to the state of the input.

The positive weighted inputs whose differential switches establish the node voltage V+ effectively codes the total weight of its inputs. The node voltage V- sets the threshold level of the circuit depending on the weight of the negative weighted inputs. V+ and V- are called the summing nodes of the positive weighted inputs and negative weighted inputs, respectively. Thus, unlike traditional threshold gates whose threshold level is fixed, the threshold in this case is varied according to the total weight of the negative-weighted inputs. This variation of the threshold level, in essence, realises the negative nature of the weight of these inputs. The logic decision of the output is carried out by another differential circuit which compares V+ and V- to give the required output. If V+ is greater than V- (in magnitude) the output goes high, otherwise it goes low. The summing nodes can be fed to emitter followers to provide high drive capability for the comparator circuit.

Each positive weighted input (or negative weighted input) that is in a high state i.e. above the reference V_{ref2}, draws a number of units of current, corresponding to the weight of the input through a summing resistor R0 (or R1). For an input X_i of unit weight ($W_i = 1$), the unit weight current I_u is generated through a unit weight resistor R_u, where

$$I_u = \frac{V_{EE} - V_{ref1} - V_{be}}{R_u} \tag{5.1}$$

This pulls down the node voltage V+ (or V-) by V_u volts, where $V_u = I_u R0$ (note that R1 = R0). For an arbitrary weight W_k i.e. $I_k = W_k I_u$, the current source resistor required is given by R_{u/W_k}. The current source with resistor value $R_{u/(T-0.5)}$ which is not under the influence of any negative weighted input, sets the threshold level to its lowest value when all the negative weighted inputs are logically low. The term V_u, known as the unit weight voltage is an important parameter and as will be discussed next, determines the maximum fan-in of the circuit for high-speed applications.

Let m and l be the total weight of the positive weighted inputs and negative weighted inputs that are high, respectively. Then

$$V+ = - \left\{ mV_u + V_{be1} \right\} \tag{5.2}$$

$$V- = - \left\{ \left[(T-0.5) + l \right] V_u + V_{be2} \right\} \tag{5.3}$$

The maximum value of m and l, which in fact represents the maximum fan-in weight of the positive weighted and negative weighted inputs, respectively depends on how far V+ or V- can be reduced. For high-speed operation, the lowest value that V+ and V- can be pulled down to occurs when all the differential input transistors just enters into saturation. This, in turn depends on the lowest value of V_{ref1} that is sufficient to give a well-controlled current source for the weighted currents. With the lowest value of V+ and V- determined under normal operating conditions, in order to get a high fan-in, V_u should be as small as possible without degrading the noise margin of the differential pair which compares V+ and V-.

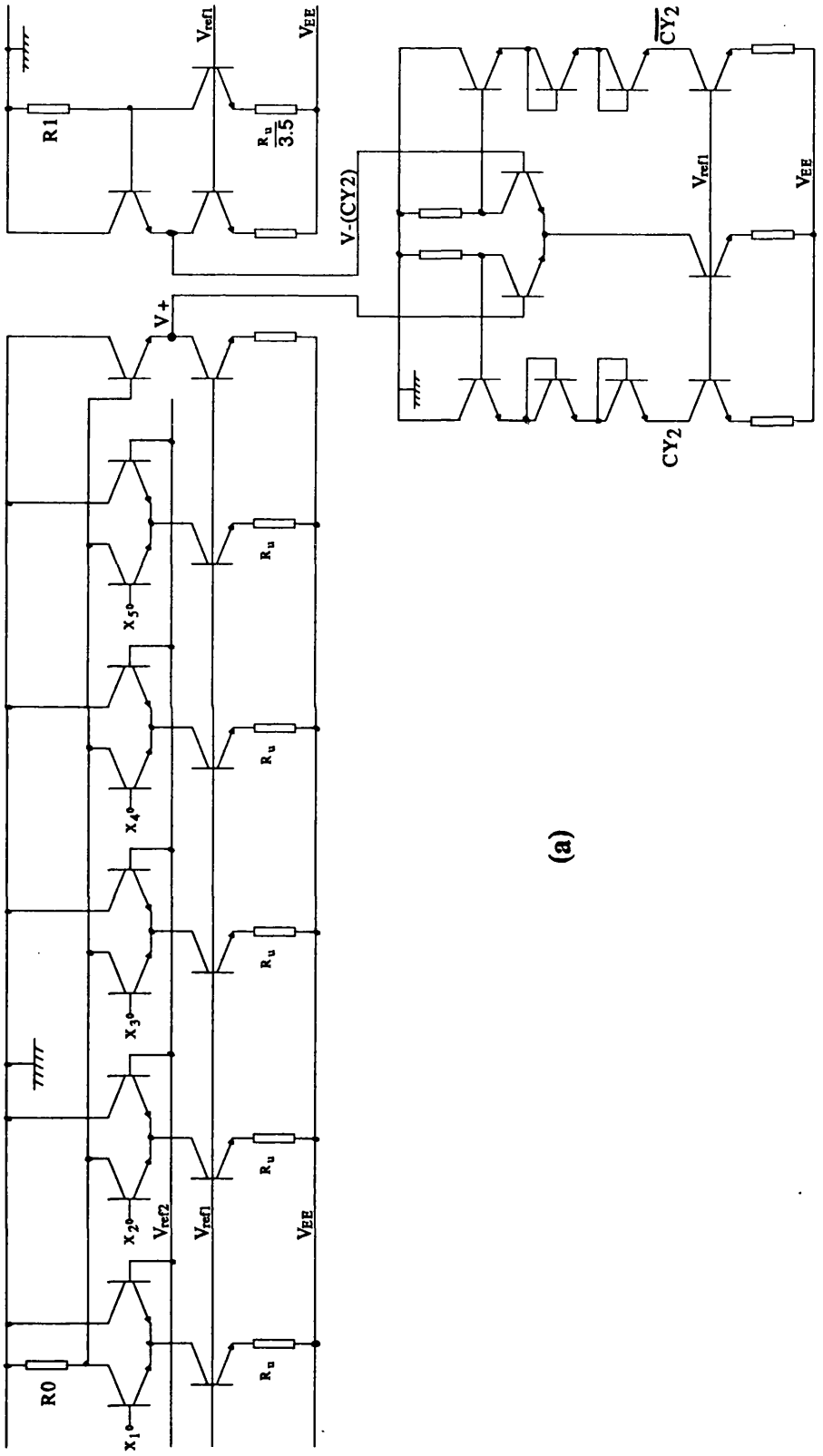


Figure 5.9. Implementation of a (5,3) counter using the new threshold circuit. **(a)** CY_2 function, **(b)** CY_1 function, and **(c)** S function.

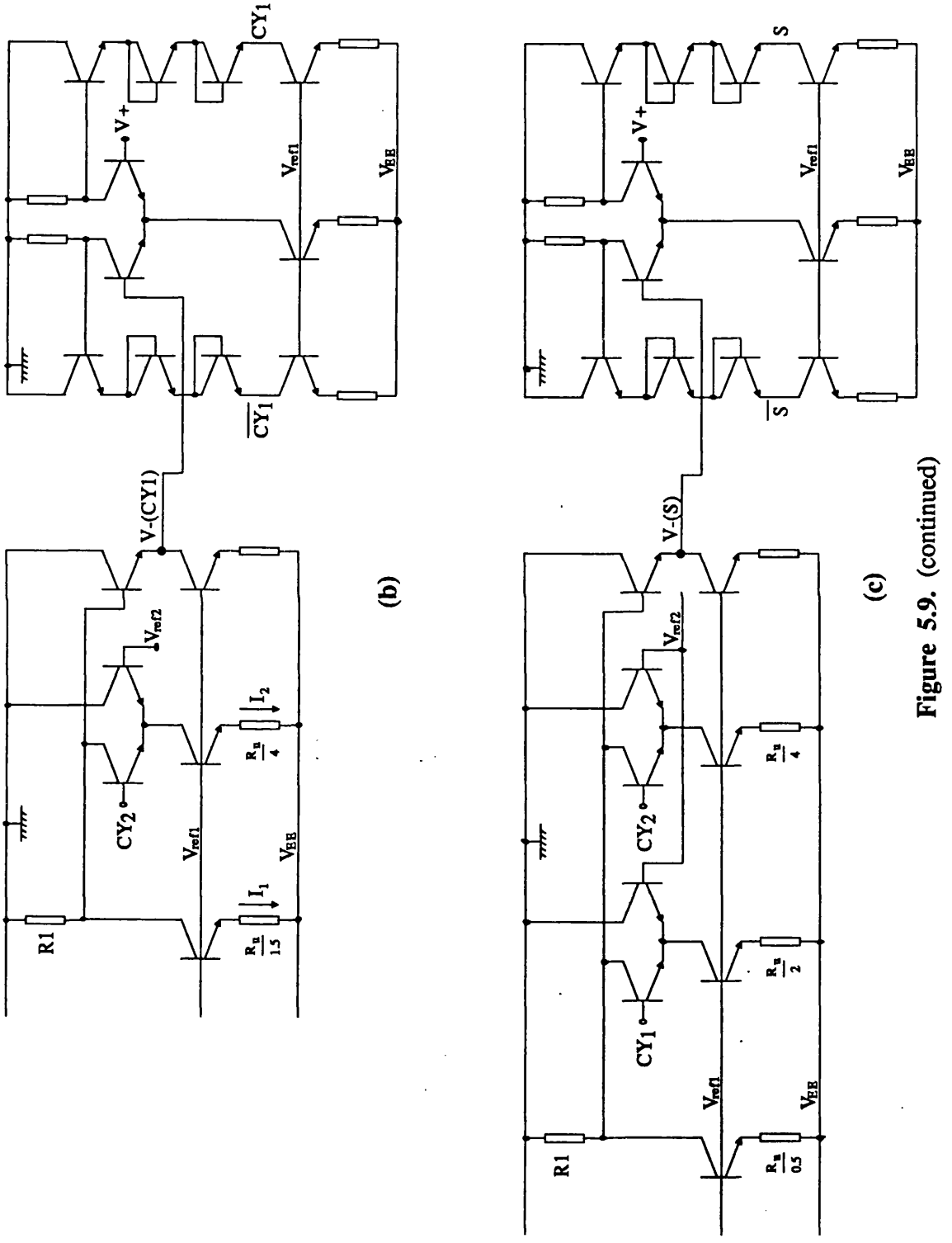


Figure 5.9. (continued)

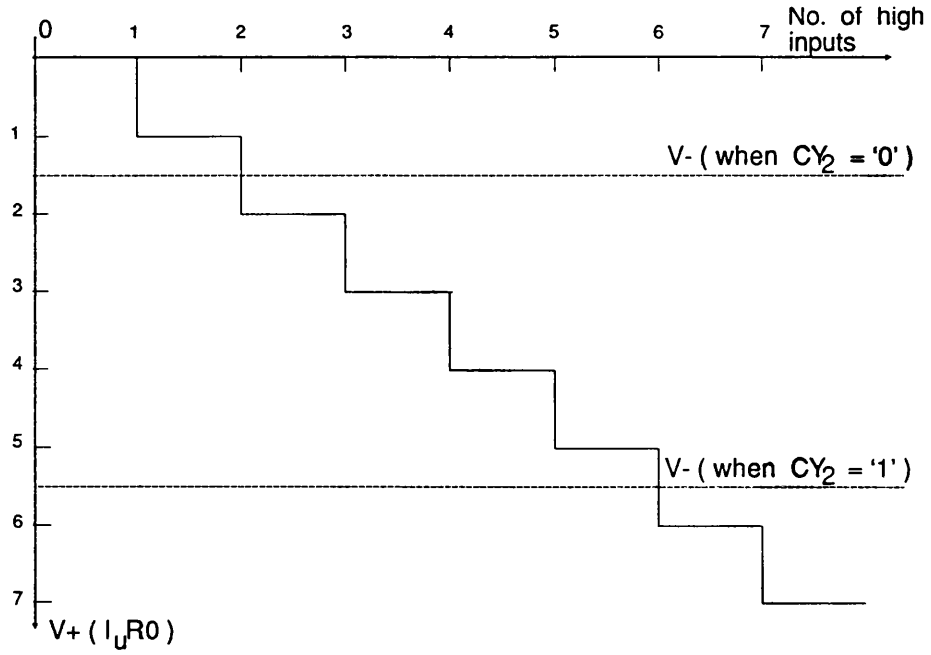


Figure 5.10. Graph showing the ideal characteristics of V+ and V- vs. no. of high inputs, of the the CY_1 gate.

The attractiveness of using this technique is illustrated in Figures 5.9(a), (b) and (c), which realises the three functions of the (5,3) counter. The fan-in problem has been eliminated by separating the inputs into positive weighted and negative weighted inputs through the use of two distinct ECL gates. Here, the summing nodes are first fed to emitter followers to provide a high drive capability. There is an added advantage of using this technique; the positive weighted inputs need only be summed once to obtain V+ at the first stage and this is fed to the next two stages to compute CY_1 and S. If traditional threshold gates are employed to implement the entirely positive weighted inputs, these inputs have to be computed for each of the functions of the counter thus increasing the fan-out and wiring complexity of the counter.

The operation of the circuit is best understood by looking at the CY_1 function of the (n,3) counter. The value of the parameters I_u, I_1,

I_2, R0, and R1 are chosen such that the ideal characteristics of the circuit as shown in Figure 5.10 is obtained in order to accomplish the correct functionality of the gate. Here, ideally

$I_1R_1 = 1.5I_uR0$

$(I_1 + I_2)R_1 = 5.5I_uR0.$

Notice that when V+ is at least $2I_uR0$ with CY_2 = '0' and at least $6I_uR0$ when CY_2 = '1' the output goes high as required since the threshold of the gate is 2.

The technique should thus prove useful in applications where high fan-in weight threshold gates are needed. One good advantage is that since the threshold gate is basically a current mode type of circuit, they can be mixed where advantageous with normal Boolean ECL gates under the same bipolar process.

In terms of speed, the worst-case delay of the counter is three threshold gate delay suggesting a speed worse than that of a CSA full-adder. However, the scheme is of great importance since a higher order counter up to a (7,3) could be realised without any significant increase in delay and complexity by just adding extra lines of inputs. Such higher order counters could be useful in the inherently faster matrix generation-reduction architectures of Wallace and Dadda.

An analysis of the circuit technique based on SPICE simulation results is discussed in the next chapter to characterize the maximum fan-in weight, minimum unit weight voltage and the propagation delay of the circuit under standard operating conditions.

5.4. (2,2,3) counter cell

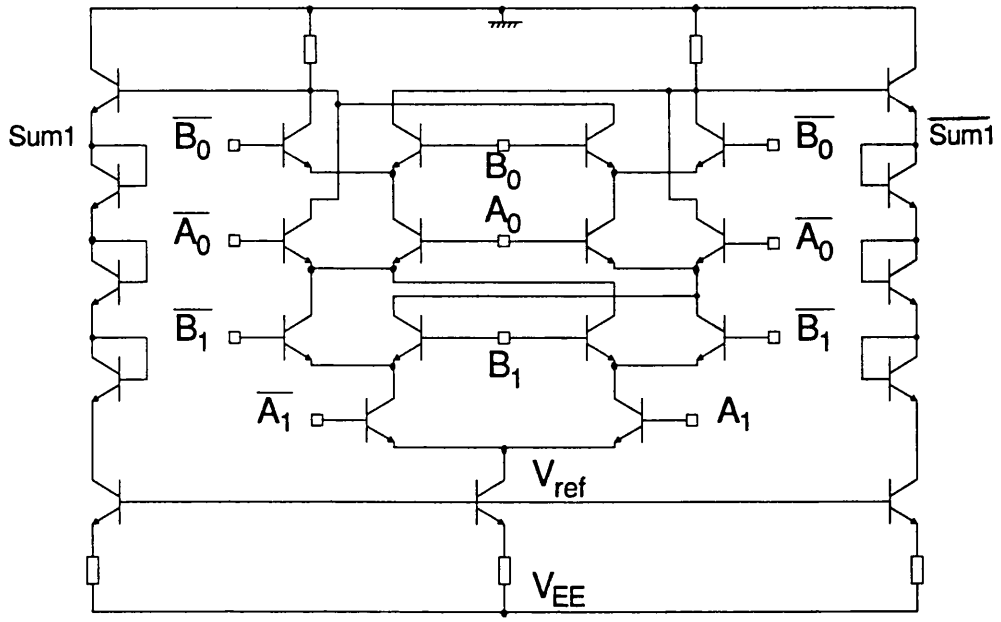
The (2,2,3) counter is basically composed of a 2-input EX-OR and a modified full-adder which are used for the diagonal cells of the (5,3)

counter multiplier (Figure 4.5(b)). The main issue of this counter is the delay and complexity of the modified full-adder since the 2-input EX-OR gate could be designed, in principle with a delay and complexity lower than a full-adder. An efficient fast design of the modified full-adder is not only important for the viability of the (2,2,3) counter multiplier but also for the (5,3) counter scheme since it is one of the diagonal cells which lie in the critical path of the multiplier.

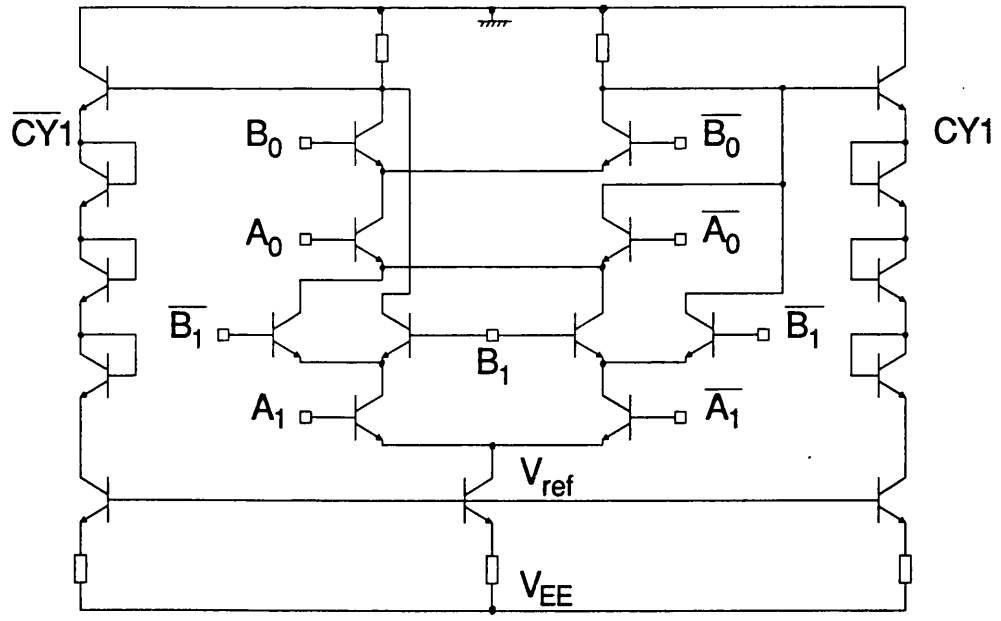
5.4.1. ECL implementation of a modified full-adder

In three of the megacells of the (2,2,3) counter multiplier (see Figures 4.9(a)-4.9(c)), the second stage (2,2,3) counter *(CT2)* is the most critical part of the cell. The first stage (2,2,3) counter *(CT1)* which reduces the two pairs of partial product bits to three output lines need not be very fast since they settle to their final value after one counter delay. Thus a simpler and slow (2,2,3) counter could be implemented for this like a network of full-adders as proposed by Foster and Stockton [5.10]. To realise a high-speed design of the second stage (2,2,3) counter (or rather the modified full-adder) it is necessary to note that the input(s) which come directly from the first stage (2,2,3) counters are steady signals while the rest are propagating signals. The property of these signals are utilized in the implementation of the modified full-adder in cascode ECL.

Two different circuit techniques are proposed for the implementation of the modified full-adder in ECL as illustrated in Figures 5.11 and 5.12. The first scheme uses a single cascode ECL gate of four levels of differential pairs. The tree of differential switches in the first technique is obtained by mapping the truth table of the (2,2,3) counter using the mapping technique described in section 5.2.1.1. In the second

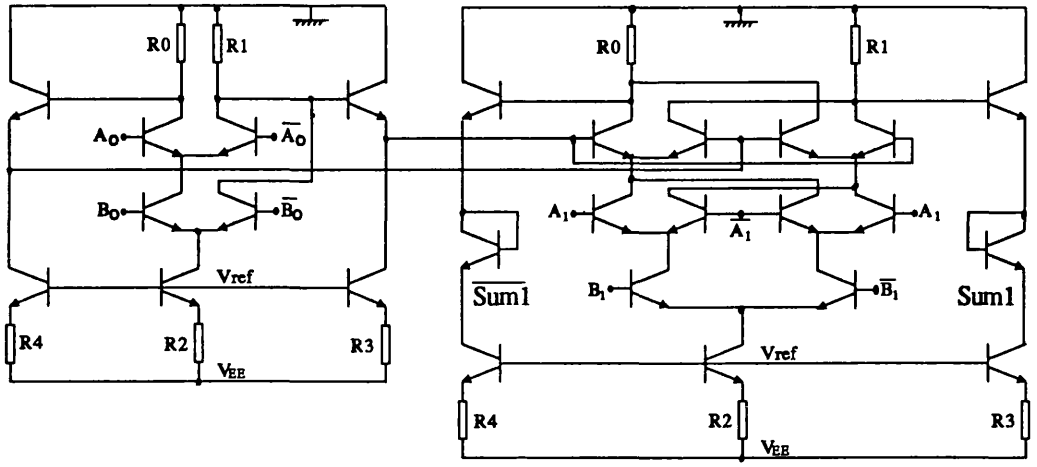


(a)

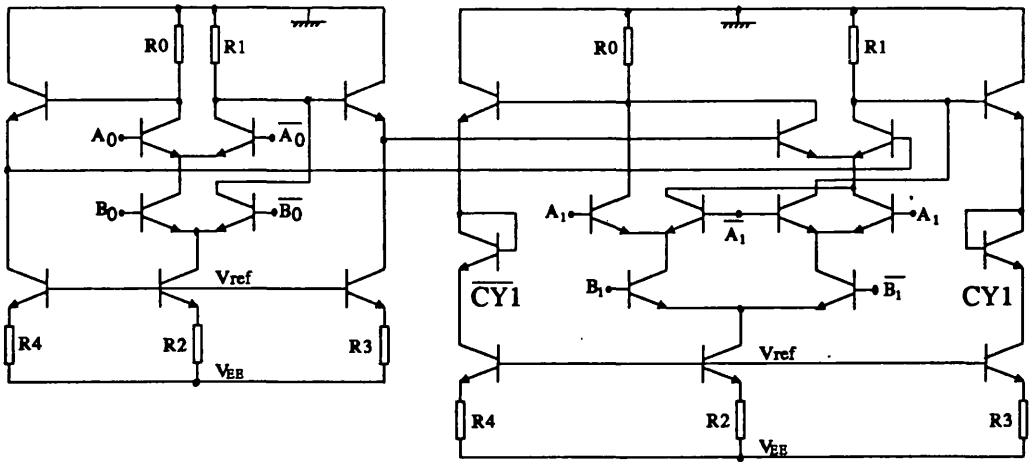


(b)

Figure 5.11. Implementation of the modified full-adder of a (2,2,3) counter by 4-level series gated ECL. **(a)** Sum1 gate, and **(b)** CY1 gate.



(a)



(b)

Figure 5.12. Implementation of the modified full-adder of a (2,2,3) counter by using AND gate and normal full-adder. **(a)** Sum1 function, and **(b)** CY1 function.

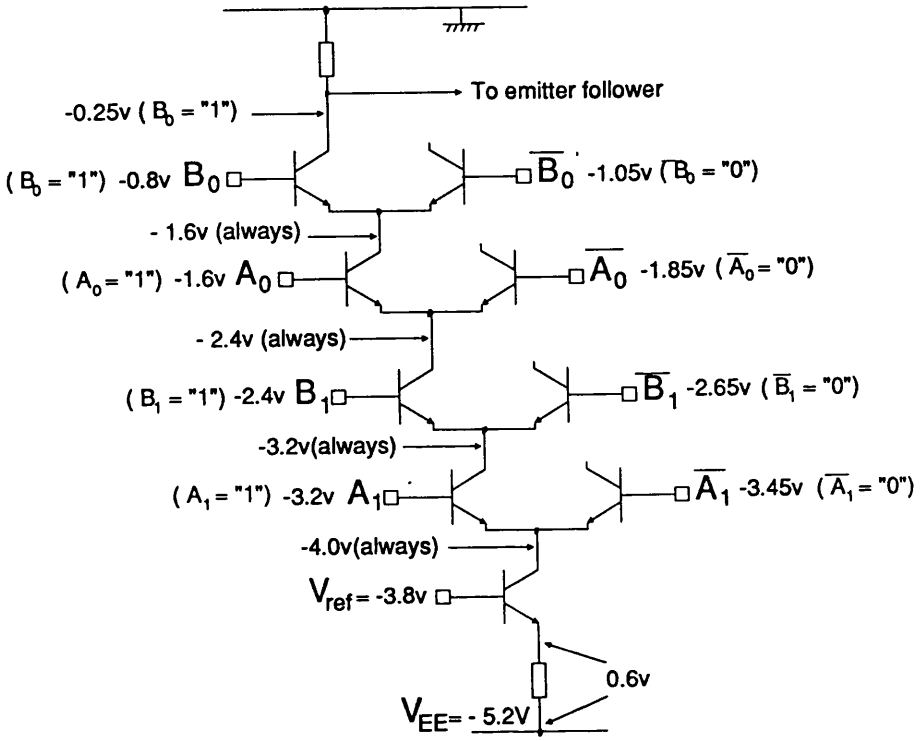


Figure 5.13. A 4-level transistor tree showing node voltages.

method, a two-input AND gate with its output fed to a normal full-adder are employed which, in essence realises directly the Boolean equations defined in Table 4.4.

For a given power supply there is a limit to the number of levels that could be accommodated in a series-gated ECL gate in order to maintain high-speed operation of the circuit. To prevent saturation of the transistors, a voltage of V_{be} is normally required between the bases of input transistors of adjacent levels. The use of differential inputs allows a logic swing of 250mv (this swing was shown to be workable for the MPC test chip) and sets the emitter voltage of a differential switch that is turned on, to a fixed value. With V_{be} = 0.8v, the total voltage required for the four-level switching tree is -4.0v, making possible the

use of a standard supply of -5.2v. Figure 5.13 demonstrates this point clearly. Notice that V_{ref} can be set to around -3.8v to give adequate control of the current source without degrading the speed of the circuit since the base-collector junction, being forward-biased by 0.2v, would not be sufficient to cause heavy saturation of the transistor concerned. To achieve compatibility between the input and output levels, collector-base connected transistor diodes in the emitter followers are used to level-shift the outputs to the appropriate level.

It is seen that by feeding the output from the first stage (2,2,3) counter of the megacells to the lowest input level, three levels of differential switches in the transistor tree are under the influence of the propagating signals and this represents the maximum propagation delay of the counter.

In the second method (Figures 5.12(a) and (b)), two distinct ECL gates are employed. The first computes the AND function of the lower significant bits of the counter whose output is then fed to the highest level of a normal full-adder. The method suggests that the modified full-adder could be slower than a normal full-adder; but looking at the input signals in the context of steady and propagating signals implies otherwise. With the output from the first stage (2,2,3) counter fed to the lowest input level of the normal full-adder, the modified full-adder's propagation delay can be nearly as fast as a normal full-adder. The reason is that although there are four propagating signals, the signal at the second input level of the full-adder would have switched the current to its probable path by the time the output of the AND gate starts to switch the current at the highest level to its final path. It is interesting to note that even if the modified full-adder is not as fast as a normal full-adder, as discussed in the

last chapter, the delay path through the megacells of the (2,2,3) counter array architecture traverses a majority of alternate logic blocks of full-adders and (2,2,3) counters. This means that the slower delay of the modified full-adder component of a (2,2,3) counter would be compensated by the faster EX-OR gate component of the next stage (2,2,3) counter giving an average propagation delay of one full-adder per (2,2,3) counter cell. Compared to the first method, the second method is slightly more expensive in terms of power since two separate ECL gates are employed.

Better multiplication speed and hardware savings could be achieved by replacing the second stage (2,2,3) counters in megacells $C_{i,0}$ and $C_{n-1,j}$ with a (1,2,3) counter. Such a counter with three input variables allows the modified full-adder to be designed with a single cascode ECL gate with the same delay, power and complexity as a normal full-adder.

A series of SPICE simulations runs and circuit analysis are carried out on the two forms of modified full-adder ECL implementation in order to characterize the tradeoffs in speed, power and noise immunity under standard operating conditions. The results are presented in the next chapter.

5.4.2. Threshold logic implementation of a modified full-adder

A threshold logic realization of the modified full-adder is shown in Figure 5.14. Here, traditional threshold gates with a maximum fan-in weight of 7 could be reliably employed. However, for better savings in hardware and a reduction in fan-out and wiring complexity the novel circuit technique of using negative weighted inputs can be applied. In terms of speed, the worst case propagation delay is two threshold gate delay or worse since a AND operation , A_0B_0 is necessary for one of the inputs. Thus, this method is not as fast as the ECL implementation.

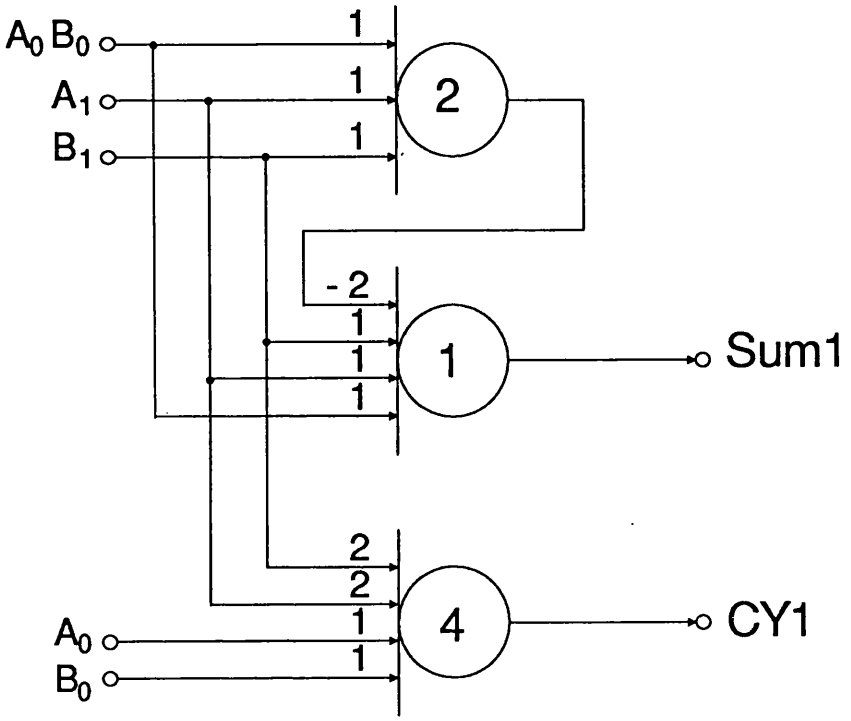


Figure 5.14. Threshold logic realization of a modified full-adder.

Unlike the (n,3) counters where the threshold implementation could be extended to a (7,3) counter with little increase in delay and hardware, the (2,2,3) counter threshold implementation cannot be similarly adapted to larger number of inputs without significantly increasing the hardware complexity and delay, due to the fact that the inputs are of different weights.

5.5. Summary

The efficiency of operation speed of the (5,3) counter and (2,2,3) counter have been investigated in bipolar technology using series gated ECL and threshold logic. A novel threshold logic circuit technique using negative weighted inputs to overcome the maximum fan-in weight problem of parallel counters is proposed. Also given is an attractive approach of

mapping a logic function on to series gated ECL for software implementation.

It is shown that the series gated ECL (5,3) counter has a delay between one and two full-adder delays with an increase in complexity of a factor of 3 compared to the full-adder. Although the threshold implementation of the (5,3) counter gives a delay that is about three full-adder delays, the scheme is of great interest since a counter of size (7,3) could be realised with little increase in hardware and delay. The modified full-adder component of a (2,2,3) counter designed in series gated ECL gives a cell that is potentially as fast as a full-adder with about the same degree of complexity.

REFERENCES

[5.1] L. Dadda, " Some schemes for parallel multipliers, " Alta Frequenza, Vol. 34, pp. 349-356, Mar. 1965.

[5.2] L. Dadda, " On parallel digital multipliers ", Alta Frequenza, pp. 574-580, Oct. 1976.

[5.3] L. Dadda & R. Stefanelli, "Digital multipliers : A unified approach ",Alta Frequenza, Vol. 37, pp. 1079-1086, Nov. 1968.

[5.4] D. Ferrari & R. Stefanelli, " Some new schemes for parallel multipliers, " Alta Frequenza, Vol. 38, pp. 843-852, Nov. 1969.

[5.5] W.J. Stenzel, W.J. Kubitz & G.H. Garcia, " A compact high-speed parallel multiplication scheme, " IEEE Trans Comput., Vol. C-26, pp. 948-957, Oct. 1977.

[5.6] T. Liu, K.R. Hohulin, L. Shiau & S. Muroga, " Optimal one-bit full adders with different types of gates ", IEEE Trans. Computers, Vol. C-23, No. 1 pp. 63-69, Jan. 1974.

[5.7] E. E. Swartzlander, Jr., " Parallel Counters ", IEEE Trans. Computers, Vol. C-22, No. 11, pp. 1021-1024, Nov. 1973.

[5.8] A. Svoboda, " Adder with distributed control ", IEEE Trans. Computers, Vol. C-19, No. 8, pp. 749-751, August 1970.

[5.9] S. Singh & R. Waxman, " Multiple operand addition and

multiplication ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 113-120, Feb. 1973.

[5.10] C. C. Foster & F. D. Stockton, " Counting responders in an associative memory ", IEEE Trans. Computers, pp. 1580-1583, Dec. 1971.

[5.11] S. Dormido & M. A. Canto, " Synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-30, No. 9, pp. 699-703, September 1981.

[5.12] H. Kobayashi & H. Ohara, " A synthesizing method for large parallel counters with a network of smaller ones ", IEEE Trans. Computers, Vol. C-27, No. 8, pp. 753-757, August 1978.

[5.13] S. Dormido & M. A. Canto, " An upper bound for the synthesis of generalized parallel counters ", IEEE Trans. Computers, Vol. C-31, No. 8, pp. 802-805, August 1982.

[5.14] D. G. Gajski, " Parallel Compressors ", IEEE Trans. Computers, Vol. C-29, No. 5, pp. 393-398, May 1980.

[5.15] L. Dadda, " Composite parallel adders ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 942-946, October 1980.

[5.16] D.Hampel & R.O. Winder, " Threshold logic ", IEEE Spectrum, pp. 32-39, May 1971.

[5.17] R.O. Winder, " Threshold logic will cut costs, especially with boosts from LSI ", Electronics, pp. 94-103, May 27 1968.

[5.18] I. T. HO & T. C. Chen, " Multiplie addition by residue threshold functions and their representattion by array logic " IEEE Trans. Computers, Vol. C-22, N0. 8 pp. 762-767, Aug. 1973.

[5.19] L. Dadda, " Multiple addition of binary serial numbers ", Proc. 4th Symposium Computer Arithmetic, Oct. 1978, pp. 140-148.

[5.20] E.E. Swartzlander, B.K. Gilbert & I.S. Reed, " Inner product computers ", IEEE Trans. Computers, Vol. C-27, pp. 21-31, 1978.

[5.21] R.P. Cheung, K.W. Current & E.E. Swartzlander, " A very high speed digital correlation technique ", in 1976 Nat. Telecommun. Conf. Rec., Nov. 1976, paper 52.4.

[5.22] E. E. Swartzlander, " Merged arithmetic ", IEEE Trans. Computers, Vol. C-29, No. 10, pp. 946-950, October 1980.

[5.23] A.R. Meo, " Arithmetic networks and their minimization using a new line of elementary units ", IEEE Trans. Computers, Vol. C-24, No. 3, pp. 258-280, March 1975.

[5.24] K.W. Current & D.A. Mow, " Implementing parallel counters with four-valued threshold logic ", IEEE Trans. Computers, Vol. C-28, No. 3, pp. 200-204, March 1979.

[5.25] K.W. Current, " Pipelined binary parallel counters employing latched quarternary logic full adders ", IEEE Trans. Computers, Vol.

C-29, No. 5, pp. 400-403, May 1980.

[5.26] J.J. Amodei & R.O. Winder, " An integrated Threshold gate ", Digest of Int. Solid-State Circuits Conference, pp. 114-115, February 1967.

[5.27] M. I. Elmasry, " Digital bipolar integrated circuits, " John Wiley & sons, ISBN 0 471-05571-9.

[5.28] U. Priel & J.W. Hively, " An integrated current mode full-adder ", Digest of Int. Solid-State Circuits Conf., pp. 108-109, Feb. 1967.

[5.29] S. Nakamura, " A single chip parallel multiplier by MOS technology ", IEEE Trans. Computers, Vol. 37, N0. 3, pp. 274-282, March 1988.

[5.30] Z.E. Skokan, " Emitter function logic - Logic family for LSI, " IEEE Journal of Solid-State Circuits, Vol. SC-8, No. 5, October, 1973.

[5.31] M.I. Elmasry, " Logic Design using EFL structures, " IEEE Transactions on Computers, pp. 952-956, Sept. 1976.

[5.32] R.J. Baumert, L.E. Cameron & R.A. Wilson, " A mixed EFL I^2L digital telecommunication integrated circuit ", IEEE J. Solid-Stae Circuits, Vol. SC-19, No. 1, pp. 26-31, Feb. 1984.

[5.33] M.I. Elmasry & P.M. Thompson, " Two-level emitter-function logic structures for logic-in-memroy computers ", IEEE Trans. Computers, Vol. C-24, No. 3, pp. 250-257, March 1975.

[5.34] M.I. Elmasry & P.M. Thompson, " Logic partition for multiemitter two-level structures ", IEEE Trans. Circuits and Systems, Vol. CAS-21, No. 3, pp. 354-359, May 1974.

[5.35] S. Muroga, " Threshold logic and its applications ", pp. 14-16, Wiley-Interscience, 1971.

[5.36] A. Barna, " Propagation delay in current mode switching circuits ", IEEE J. Solid-State Circuits, pp. 123-124, April 1975.

[5.37] A. Barna, " Analytic approximations for propagation delays in current mode switching circuits including collector-base capacitances ", pp. 597-599, Oct. 1981.

[5.38] C.R. Baugh & B.A. Wooley, " Stastistical analysis of a differential threshold logic circuit configuration ", IEEE Trans. Computers, Vol. C-25, No. 7, pp. 745-754, July 1976.

[5.39] B.A. Wooley & C.R. Baugh, " An integrated m-out-of-n detection circuit using threshold logic ", IEEE J. Solid-State Circuits, Vol. SC-9, No. 5, pp. 297-306, October 1974.

[5.40] D. Hampel, " Multifunction threshold gates ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 197-203, February 1973.

[5.41] W.C. Seelbach et al, " Variable threshold logic family ", Digest of Int. Solid-State Circuits Conference, pp. 40-41, February 1965.

[5.42] A.L. Larson, " A TTL compatible threshold gate ", IEEE J. Solid-State Circuits, pp. 30-31, February 1974.

[5.43] T.T. Dao, " Threshold I^2L and its applications to binary symmetric functions and multivalued logic ", IEEE J. Solid-State Circuits, Vol. SC-12, No. 5, pp. 463-472, October 1977.

[5.44] D.R. Haring, " A technique for improvimg the reliability of certain classes of threshold elements ", IEEE Trans. Computers, pp. 997-998, October 1968.

[5.45] C.H. Han, C.K. Kim & G.H. Yoo, " Feasibility of substrate fed threshold logic ", IEEE J. Solid-State Circuits, Vol. SC-18, No. 2, pp. 160-164, April 1983.

[5.46] A. Pal, " A four variable programmable universal logic module using digital summation threshold logic gates ", Proc. of the IEEE, Vol. 72, No. 12, pp. 1813-1814, December 1984.

[5.47] C.R. Edwards, " The application of the Rademacher-Walsh transform to Boolean function classification and threshold logic synthesis ", IEEE Trans. Computers, Vol. C-24, No. 1, pp. 48-62, January 1975.

[5.48] R.O. Winder, " The status of threshold logic ", RCA review, pp. 62-84, March 1969.

[5.49] L. Dadda, " Synthesis of threshold logic combinatorial networks, " Alta Frequenza, n. 3, Vol. xx, pp. 224-28E-231, 1961.

[5.50] M.A. Fischler and E.A. Poe, " Threshold realization of arithmetic circuits, " IEEE Trans. Electron. Comput., Vol. EC-11, pp. 287-288, April 1962.

[5.51] C.H. Gustafson et al, " Synthesis of counters with threshold elements, " Switching theory and logical design, pp. 25-35, Oct. 1965.

[5.52] I.J. Gabelman, " The synthesis of Boolean functions using a single threshold element, " IRE Trans. Electron. Comput., Vol. EC-11, pp. 629-642, Oct. 1962.

[5.53] L.W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits ", Memo ERL-M520, University of California, Berkeley, CA, May 9, 1975.

CHAPTER 6

SIMULATION RESULTS

6.1. Introduction

Various schemes for realising the (5,3) counter and (2,2,3) counter have been discussed in the last chapter. The efficiency of operation speed of the counters have been investigated for the two techniques in bipolar technology, that is using series gated ECL and threshold logic which are shown to offer promising tradeoffs in speed and complexity.

In this chapter, the propagation delays of the (5,3) and (2,2,3) counter when connected in their multiplier architectures are critically compared with that of a full-adder. For this purpose, a series of SPICE [6.1] simulations were carried out on the various logic cells of the (5,3) and (2,2,3) counter. The propagation delay of the full-adder cell under different load resistors, bias currents and transistor sizes have been simulated and compared with the actual figures obtained from measurements of the MPC test chip (see Chapter 3). The HP1X transistor model used in the test chip simulations, although shown to have a certain degree of inaccuracy, is, however, representative of state-of-the-art bipolar technology and is employed in the simulations of the (5,3) and (2,2,3) counter for comparison purposes.

SPICE simulations were performed on the counters under the same operating conditions with all gates having the same load resistors of 250Ω, transistor emitter size of 1 x 10μm, and emitter follower stages (for both the true and complement signal) to give differential outputs. Also all inputs are driven differentially. Simulation results of the (5,3) counter synthesized in threshold logic are first presented. The worst-case delays of series gated ECL realization of the (5,3) and

(2,2,3) counter are then given. A general evaluation and characterisation on the delay behaviour of series gated ECL gates are then determined which allows us to make assumptions and develop delay models (for HILO) of the various logic blocks. Using this ECL delay behaviour a logic simulation using HILO [6.2] is performed on the (2,2,3) counter architecture for a 8 x 8-bit multiplication and compared with an equivalent conventional CSA multiplier.

The results show that the (2,2,3) counter offers a significant speed improvement over the CSA multiplier whilst the (5,3) counter multiplier gives no increase in speed.

6.2. (5,3) counter cell

Two techniques were proposed in the last chapter for synthesis of the (5,3) counter. The first, which is better in terms of speed is to use blocks of series-gated ECL multiplexers. The second method, implemented in threshold logic, although do not have the potential to offer any significant speed improvement in the (5,3) counter multiplier, is of great interest since a counter of size up to a (7,3) could possibly be designed with the same delay and with only a marginal increase in complexity. Such a counter should be useful for the Wallace tree type multipliers and in applications where a large parallel (p,q) counter with low complexity is of prime importance.

6.2.1. (5,3) counter realised in threshold logic

A novel threshold logic circuit technique to overcome the maximum fan-in limitation was proposed in the last chapter. The technique has the potential to improve the maximum fan-in by a factor of two by the partial use of negative-weighted inputs.

This section presents the results obtained from SPICE simulation runs on the threshold gates and from this, their delay behaviour are characterised. An analysis of the circuit to estimate the desired minimum logic swing, unit weight voltage and current, and maximum fan-in attainable under worst-case conditions are first discussed. Although the analysis is done on a specific type of circuit function, it is representative of the novel circuit technique and should be applicable to any other circuits of different functions.

6.2.1.1. Analysis of novel threshold circuit

Resistor values for the emitter degenerated current sources are first determined to give the required current weighting for each of the inputs. The resistor value for a given input weight current is given by equation 5.1. Ideally, to achieve high-speed switching of the circuit, unit weight current should corresponds to the region of maximum cut-off frequency, f_t. For the HP1X process, a collector current of around 0.5mA flowing in a npn transistor of emitter size 1 x 10 μm lies in the region of maximum f_t, this value is taken as the unit weight current in our simulations.

In order to obtain a high fan-in weight, the reference voltage Vref1 should be as low as possible so that saturation of the transistors are avoided, but without degrading the output impedance of the current sources. With I_u = 0.5mA, a Vref1 of -3.8v is found to give adequate control of the current sources. A 50mv change in Vref1 causes only about 40μA change in I_u, from SPICE simulations.

Hence, with I_u = 0.5mA and Vref1 = -3.8v, a unit weight resistance, R_u = 1200Ω

is required for the unit weight current sources. Other inputs with

different weightings are given resistances which are scaled accordingly as illustrated in Figures 5.9(a)-(c). For the comparator circuits, the required tail current and load resistors which sets the differential voltage swing is determined in the next section.

In the design of the circuits, it is clearly seen that the correct operation of the circuit depends heavily on controlling and keeping the voltage levels of V+ and V- to their ideal values under circuit parameter variations and other sources of noise. The design should thus allow for power supply, reference voltage, transistor and resistor parameter variations when the circuits are committed on to silicon. This can be achieved by having an adequate separation between V+ and V- to ensure that there is enough voltage swing for each comparator's differential pair to completely switch after taking into account such variations. This value, Vdiff where

$$Vdiff = |V+ - V-| \tag{6.1}$$

is essentially directly related to the unit weight voltage V_u as shown in Figure 5.10, where it is apparent that

$$Vdiff = \frac{1}{2} V_u \tag{6.2}$$

Thus, the maximum fan-in weight achievable in a given design situation is a function of both the minimum value of V_u that can be used reliably, and the total voltage swing available at the comparators inputs. The desired minimum value of V_u is crucial since a small V_u would render the comparator circuits ineffective whilst a large V_u would seriously limit the maximum fan-in weight of the circuits. Thus a compromise needs to be determined.

The desired minimum value of V_u can be evaluated by doing a statistical analysis of the circuits as a function of component tolerances and various design constraints, as similarly done by Baugh and

Wooley [6.2] on their threshold circuits. However, such an analysis, although reliable, comparatively accurate and represents an optimistic view of the circuit, is quite an involved and complicated process. Nevertheless, an estimate on V_u could be made to a first order for the case when V+ and V- approach closest to each other under worst-case conditions due to extreme power supply, reference voltage, transistor and resistor parameter variations.

6.2.1.2. Estimation of unit weight voltage and maximum fan-in weight

To determine the worst-case variation of Vdiff i.e. its minimum value, assumptions are made on various circuit parameters to simplify the analysis. The transistors in the proposed threshold circuit are assumed to be nominally identical where Vbe is assumed to be the only parameter to vary significantly. All resistors in the circuit of Figures 5.9(a)-(c) are implemented as series combinations of a basic resistor with the same variation. This results in the best possible matching and, together with the differential approach, provides for minimum circuit sensitivity to processing and operating environment. Where an input with a weighting w (where $w > 1$) is required in the circuit, this is implemented with w unit weight current sources which are all switched by this common input. Within a circuit chip, the power supply and reference voltage of the current sources are assumed to be evenly distributed with the same variation throughout. Since the power supply variations considered here correspond only to variations within a single chip, it is expected that they are kept quite small. The current sources reference voltage is assumed to be driven from a single reference circuit where the effects of component variation in the reference circuit is reflected entirely in the variation of Vref1. A very low resistance in the interconnection

metallization is assumed when the circuit is laid out onto silicon.

A convenient criterion for establishing the minimum usable value of V_u is

$$\min\{Vdiff\} = V_m + (\Delta V- + \Delta V+) \qquad (6.3)$$

which provides a margin of ($\Delta V-$ + $\Delta V+$) deviation against the voltage V_m, where V_m is chosen to ensure nearly complete switching of each comparator's differential pair.

Assume V+ to be higher than V-, where V+ and V- are given by equations 5.2 and 5.3, respectively. Due to variations of VEE, Vref1, Vbe and resistor ratios R0/Ru, V+ is offset by $\Delta V+$, where it is given by

$$\Delta V+ = n\{\Delta VEE - \Delta Vref1 - \Delta Vbes\}\Delta(R0/Ru) + \Delta Vbef \qquad (6.4)$$

where n is the total input weight, $\Delta Vbes$ and $\Delta Vbef$ are the variations in Vbe of the current source transistors and emitter follower transistor, respectively.

The maximum value of $\Delta V+$ occurs when both $\Delta Vref1$ and $\Delta Vbes$ are offset extremely in the negative sense along with an equivalent positive offset on ΔVEE, $\Delta Vbef$ and $\Delta(R0/Ru)$. It is apparent from equation 6.4 that the influence of the various parameters can be minimised by increasing the emitter degeneration resistance of the current sources. However, for a given unit weight current, this would imply a higher Vref1 is needed which then further limits the maximum fan-in weight. Typically (for the HP1X process),

ΔVEE = +1%

$\Delta Vbef$ = +5%

$\Delta R0/Ru$ = +2%

$\Delta Vref1$ = -4%

$\Delta Vbes$ = -5% .

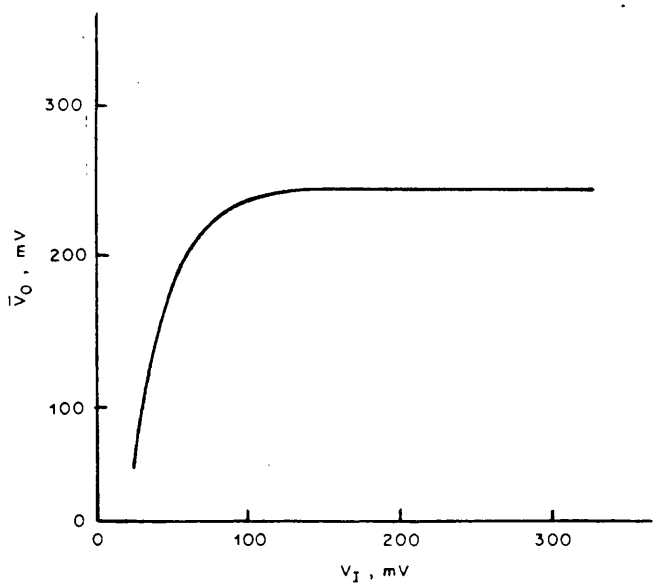


Figure 6.1. Differential output as a function of differential input driving the comparators switch.

With VEE = -5.2v, Vref1 = -3.8v, Vbe = 0.8v, R0 = 1K and Ru = 1.2K, $\Delta V+ \simeq$ 60mv for a fan-in weight of 7. By a similar argument, V- would be offset, at worse, by 60mv in the opposite direction. This gives a total deviation of 120mv for Vdiff. Figure 6.1 provides a basis for choosing V_m in equation 6.3. With V_m = 120mV, at least 99% of the current in the differential switches is ensured to flow through the ON side of the switch. Thus, from equations 6.2 and 6.3, a unit weight voltage of 0.5v would prove sufficient to accommodate the possible variations of power supply, reference voltage, transistor and resistor parameters, and other possible environmental sources of noise. In terms of its variation with temperature, both V+ and V- should track fairly well because of their equal dependence on most circuit parameters. As a result, Vdiff should be well maintained under any temperature variations. In fact SPICE simulation shows that the circuit could operate reliably up to fairly high temperatures.

For a unit weight voltage of 0.5v, a tail current of 0.5mA is employed in the comparator circuits with a load of 1KΩ. For the purpose of simulations collector currents of 1mA are used for the comparators emitter follower output stages.

With V_u = 0.5v, the lowest that the summing nodes of the resistors corresponding to V+ and V- can be pulled down to without saturation of the input transistors is -2.5v and -3.0v (of the S gate), respectively for a (5,3) counter. In the comparator differential pairs, load resistors of 500Ω are used to give a differential output swing of 250mv. Three diode collector-base connected transistors in the emitter followers are actually needed to level shift the differential outputs to -3.2v for a logic high and -3.45v for a logic low, thus avoiding saturation of the input transistors. Although this means that the base-collector junction of the current sources transistors are forward biased slightly by 0.2v, the effect of this is minimal, since a bias of around 0.6v is normally required to bring the transistors into heavy saturation. It follows that for a fan-in weight of 7, the base-collector junction of the differential pair transistors would be forward-biased by only 0.3v which thus shows that a maximum fan-in weight of 7 should be practical to allow a reliable design of higher order counters up to a (7,3).

6.2.1.3. SPICE Results

The (5,3) counter was simulated to verify its functional correctness (Table 6.1) and also to determine its worst-case propagation delay. The gates were switched exhaustively under all possible input transitions taking into account that two of the five inputs are steady signals of the two partial product bits; delays and voltages were measured. A Bandgap reference source [6.6] was also included in all the simulations

No. of high inputs	CY_2	CY_1	S
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1

Table 6.1. Truth-table of (5,3) counter in terms of no. of high inputs.

to model as precisely as possible the actual conditions under which the circuits are expected to operate.

Worst-case propagation delay, at first thought, occurs when V+ and V- are furthest from each other before they switch over which corresponds to a maximum input voltage swing to each comparator. However, it was found that the worst case delay happens when V+ and V- of the S gate switches level relative to each other, causing multiple logic transitions at its output before the eventual output state is attained. Since the threshold level of the S function is determined by both the CY_2 output and CY_1 output, which in turn is also controlled by the CY_2 output, changes in these due to transitions of the propagating inputs A, B and C from previous stage counters would cause the threshold level V- of S to switch over momentarily relative to V+ (because the CY_2 output changes earlier than the CY_1 output) until such a time when the CY_1 output eventually switched to its final state. This is further aggravated by changes in V+ itself. Thus, during this time the logic output of S would have changed unnecessarily several times due to relative changes in V+ and V- which worsens its propagation delay. This is best illustrated in Figure 6.2. Table 6.2 summarizes some of the input transitions which cause this type of delay to occur.

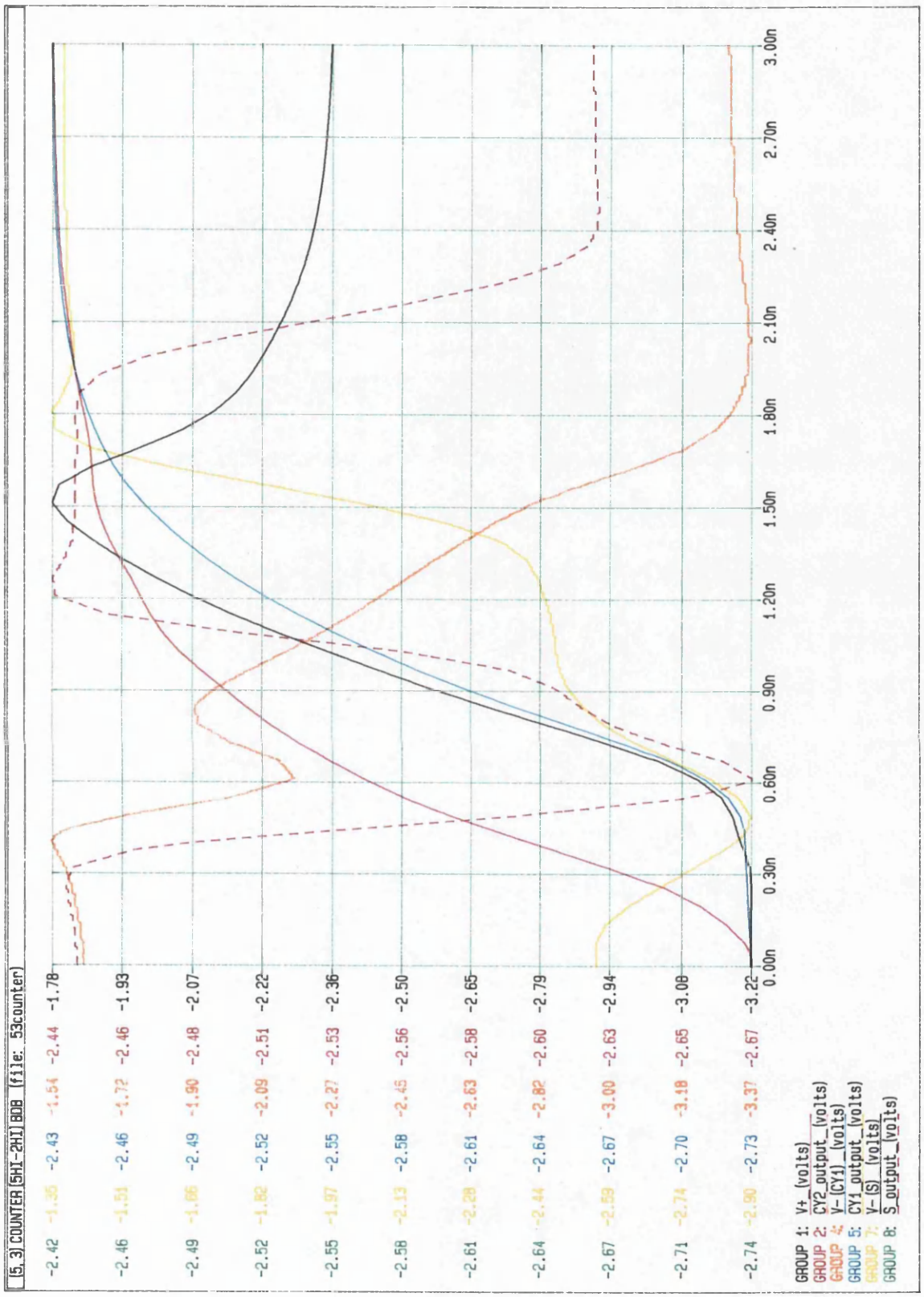


Figure 6.2. Worst-case delay characteristics of threshold logic (5,3) counter

Input change	Propagation delay(ps)
0 HI ⟶ 5 HI	1975
1 HI ⟶ 5 HI	1945
4 HI ⟶ 2 HI	1970
5 HI ⟶ 2 HI	1990

Table 6.2. Worst-case delays of a threshold (5,3) counter.

One way to alleviate this problem is to interchange the two inputs to the S gate which comes from the partial product bits with the CY_1 output as shown in Figure 6.3. Notice that the outputs and their weightings concerned have been appropriately changed to obtain the same function. Since the partial product bits would have settled to their final values after one AND gate delay, the threshold of S is only affected by changes to the CY_2 output. Thus, any transitions in A, B or C from previous stage counters would only cause the threshold of S to change once to its eventual level. This reduces the critical delay to the case when only V+ momentarily changes its level once due to transitions in A, B or C until such a time when the CY_1 output starts to switch to its final state. Using the logic configuration of Figure 6.3, simulations show an improvement of about 300ps to 400ps. The penalty for designing the counter this way is that more devices and power are needed since the V+ of the S gate is no longer identical to the other two functions as Figure 6.3 suggests.

Compared to the fastest full-adder cell, the (5,3) counter implemented in this technique has a worst-case delay which is several times slower although the high fan-in weight requirement has been solved. Clearly, such counters would be of little use in the (5,3) counter array

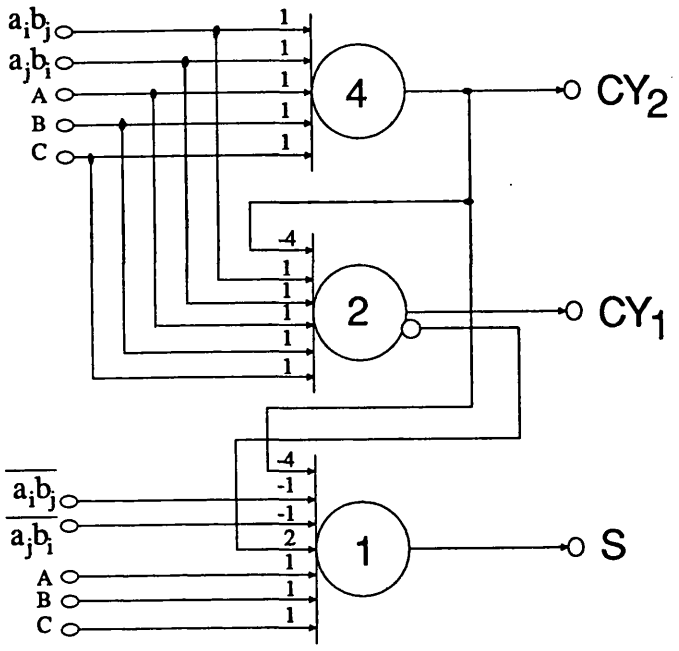


Figure 6.3. Alternative Threshold logic configuration of a (5,3) counter to improve propagation delay.

multiplier. However, in the inherently faster architectures of Wallace and Dadda, and in applications where a large counter needs to be designed with little increase in complexity, this technique should prove favourable.

It is apparent that extreme excursions of V+ and V- determine the worst-case delay of the (5,3) counter since this corresponds to a relatively high voltage swing at the comparator's inputs and cause unnecessary changes at the logic outputs. Unless an efficient method of limiting V+ and V- like clamping them to a certain value is employed, the circuit is not expected to rival equivalent Boolean ECL gates in terms of speed. Some attempts were made to clamp both V+ and V- but without much success.

6.2.2. (5,3) counter implemented with ECL multiplexers

The separation of the inputs to the (5,3) counter in terms of steady and propagating signals have led to a potentially efficient and fast realization of the counter where it was estimated that the propagation delay lies between one and two full-adder delay (see Figure 5.4). The steady signals i.e. the two partial products bits serve as the control inputs to three multiplexers which then selects the correct output under a given input state.

The worst case delay clearly occurs when the C input to the counter changes state; this corresponds to all three differential levels of the full-adder, cells C^1 and C^2 to switch. Since the two lower level inputs to the multiplexers are steady signals of the partial product bits, these cells only introduce one further level of differential delay which then gives roughly a total worst-case delay of four differential delays. However, the true delay, as was discovered, was much worse than two CSA full-adder delays.

Two stages of the counter were simulated under all possible input states for each of the outputs to verify its functionality and determine its worst case delay. Figure 6.4(a) shows the relative worst-case delay of the full-adder, gates C^1 and C^2, whilst Figure 6.4(b) depicts the propagation delay introduced by the multiplexers MX1, MX2 and MX3, as a function of tail current for a fixed value of emitter follower current. These results were obtained under the same operating conditions where load resistors of 250Ω and transistors of emitter sizes 1 x 10μm were employed, and under actual loading conditions. The multiplexers delays were obtained with the outputs of MX3, MX2 and MX1 driving the lowest, middle and highest input level of the logic gates of the following (5,3) counter, respectively. Such connection between the outputs and inputs of

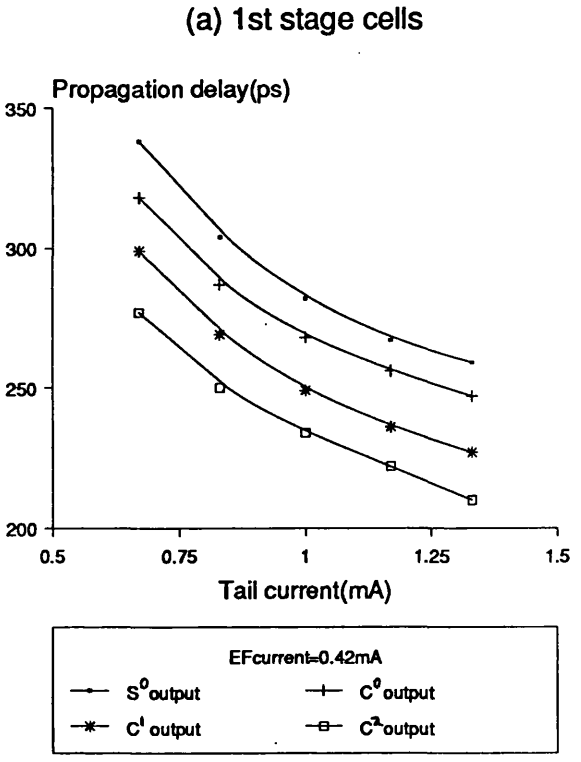


Figure 6.4. Critical delay of various gates of a ECL (5,3) counter. **(a)** 1st stage cells, and **(b)** Multiplexers MX1, MX2 and MX3.

adjacent (5,3) counters was made based on observation that with a fanout of 1, the multiplexer MX3 is inherently faster than MX2, which in turn is slightly better than MX1. To minimise the delay, it is thus appropriate to feed the lowest input level of the following gates with the fastest propagating output signal from previous stage.

The inherently better performance of gates C^1 and C^2 compared with the full-adder can be attributed to their simpler transistor tree. The eventual significantly slower speed of multiplexer MX2 and MX3 compared to MX1 is due to the need to have level shifting diodes at their emitter follower stages where each of them is required to drive three gates of the following stage.

It is thus apparent from examination of Figures 6.4(a) and (b) that the critical delay path of the (5,3) counter is through the S^0 gate and multiplexer MX2. If a comparison is made with a full-adder cell connected in a CSA multiplier, under the same operating conditions (see Figure 3.9(b)), the critical delay of the (5,3) counter is more than two times the full-adder's, mainly due to two factors. Each output of a (5,3) counter has to drive three gates of the following stage as opposed to two for a CSA full-adder. Also all three differential levels of the first stage cells i.e. S^0, C^0, C^1 and C^2 are liable to change compared to two for a CSA full-adder cell (see Figure 2.1) and this is further worsened by delays imposed by 2 level shifting diodes in the emitter followers of MX3. Note that only one level-shifting diode at most, is needed to connect successive stages of the main array full-adder cells in a CSA multiplier. Typically, each level-shifting diode introduces 20-50ps depending on the fanout.

It is important to note that the results in Figure 3.9(b) were obtained for a fan-out of 1. When connected in a CSA multiplier, each

gate of a full-adder would be driving two gates of the following stage. Simulations show that there is an increase of around 20-40ps for each fanout increment. However, even if this figure is added to each curve of Figure 3.9(b), the (5,3) counter speed is nowhere significantly better to justify the increase in complexity put into designing the (5,3) counter multiplier. It is also interesting to note that if the diagonal cells i.e. the modified full-adder and 2-input EX-OR gates are as fast as a CSA full-adder, it is not expected that there will be any considerable improvement in the overall multiplication speed without the (5,3) counter being quite as fast, since a majority of the possible critical paths in the multiplier architecture traverses a number of (5,3) counter cells. Thus, in these respects, the (5,3) counter multiplier scheme is worse than a full-adder CSA.

6.3. (2,2,3) counter cell

The critical component of the (2,2,3) counter is the modified full-adder cell in which two techniques in series gated ECL were proposed in the last chapter. The two input EX-OR gate should, in principle be faster than a CSA's full-adder cell. Figure 6.5 clearly illustrates the superior performance of this gate (for fanout = 1) compared to a full-adder. The first technique uses four-level ECL gates to directly implement the functions of the modified full-adder whilst the second method employs two gates, a two-input AND gate followed by a normal full-adder.

A discussion of the 4-level ECL gates will first be given followed by the AND gate-normal full-adder combination and a comparison of the two implementions is then presented.

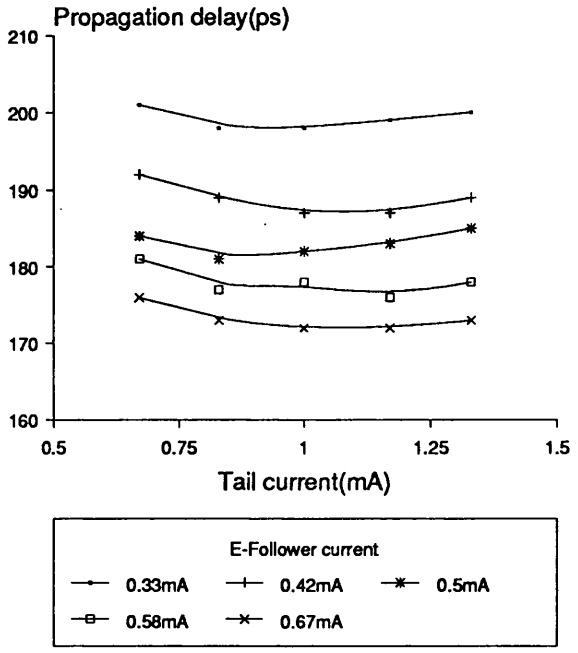


Figure 6.5. Critical delay of a 2-input EXOR series-gated ECL gate (Fanout=1).

6.3.1. Modified full-adder implemented with 4-level ECL gates

It was shown in the last chapter that it is theoretically possible to implement the two functions of the modified full-adder component of the (2,2,3) counter in a 4-level series gated ECL gate with a standard power supply of -5.2v. Figure 5.13 clearly demonstrates this point.

The worst-case delay obviously occurs when all four levels of the gates are forced to switch. The results are shown in Figure 6.6(a)-(b) as a function of tail current and emitter follower current. Although all four levels of the transistor tree are liable to switch it should be noted that this happens only in megacells $C_{i,0}$ for each operation of the multiplier. However, in megacells $C_{n-1,j}$ and $C_{i,j}$ the first stage (2,2,3) counter *CT1* which adds the partial product bits does not lie in the critical path and thus only the second stage (2,2,3) counter *CT2* needs to be addressed. Since the lowest level input to the second stage counter is

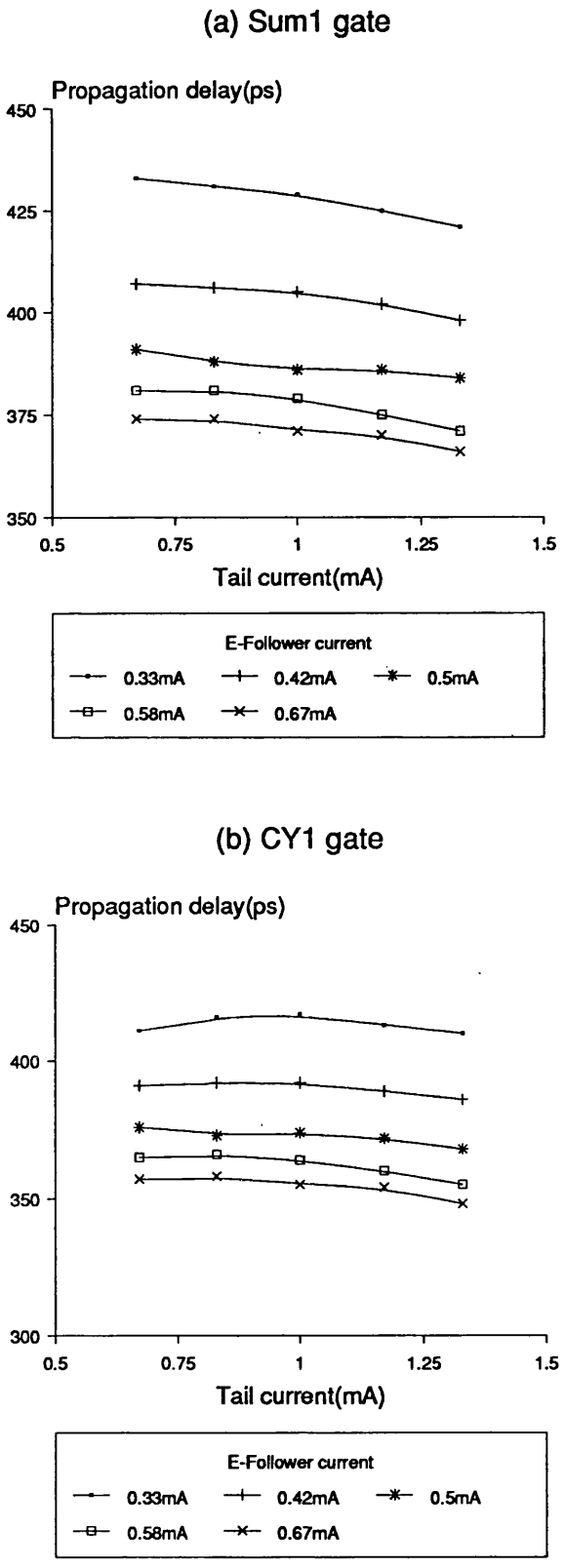
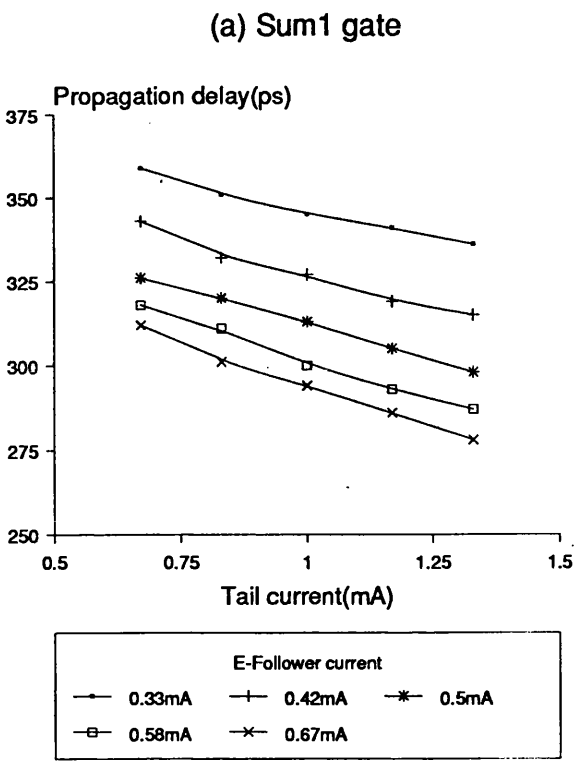


Figure 6.6. Worst-case delay of a modified full-adder implemented with 4-level series gated ECL when all four differential levels switch (Fanout=1). **(a)** Sum1 gate, and **(b)** CY1 gate.



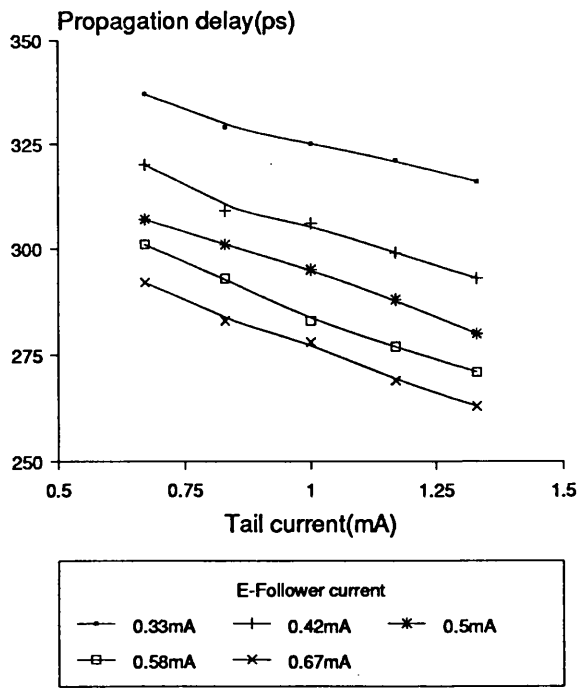


Figure 6.7. Worst-case delay of a modified full-adder implemented with 4-level series gated ECL when three differential levels switch (Fanout=1). **(a)** Sum1 gate, and **(b)** CY1 gate.

a steady signal produced from counter *CT1*, at worse, the propagation delay roughly corresponds to three differential delays. The speed of the modified full-adder of counter *CT2* in megacells $C_{n-1,j}$ and $C_{i,j}$ for this type of delay are presented in Figure 6.7(a)-(b). Both the results shown in Figures 6.6(a)-(b) and Figures 6.7(a)-(b) were obtained for a fanout of 1 and with their outputs appropriately level-shifted by three and two diodes, respectively.

A comparison in speed and the practicality of this technique is made in section 6.3.3. with the AND gate-normal full-adder implementation.

6.3.2. Modified full-adder implemented with AND gate-normal full-adder

The AND gate-normal full-adder combination of the modified full-adder component of the (2,2,3) multiplier was similarly simulated as described in the last section.

It is apparent from Figures 5.12(a)-(b) that based on the number of differential delays, the possible critical delay path occurs when either input B_0 to the AND gate or input B_1 to the normal full-adder changes state. The results of simulations are shown in Figure 6.8(a)-(b) and Figure 6.9(a)-(b) for changes at input B_1 and B_0, respectively. These results were obtained for a fanout of 1.

It can be seen that the actual worst-case delay is due to changes in input B_1 i.e. when all three differential levels of the normal full-adder are switching. Although changes at input B_0 gives the same number of differential delays, the actual smaller delay resulting from changes at input B_0 can be attributed to the inherently better performance of the 2-input AND gate, which by design has a relatively simple transistor tree and only needs to drive one gate without being level-shifted any further. Figure 6.10 demonstrates the superior speed of the 2-input AND gate. In

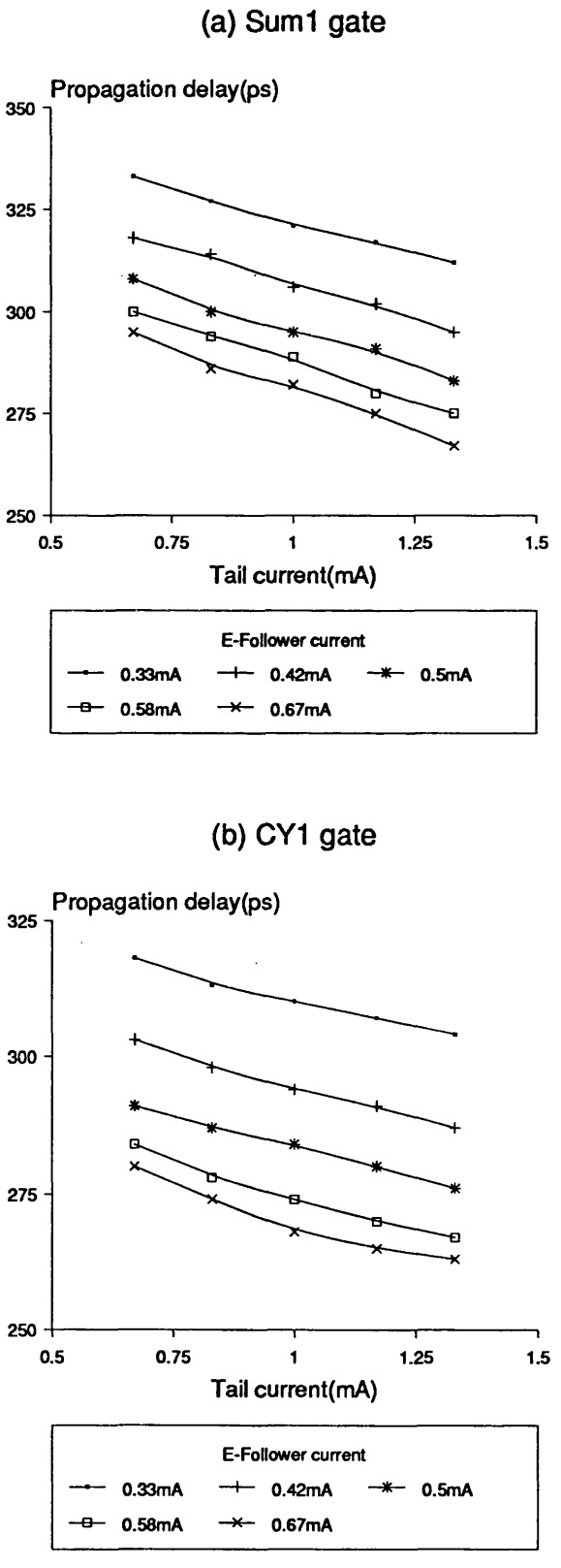


Figure 6.8. Worst-case delay of a modified full-adder implemented with AND gate-normal full-adder when input B_1 to full-adder changes (Fanout=1). **(a)** Sum1 gate, and **(b)** CY1 gate.

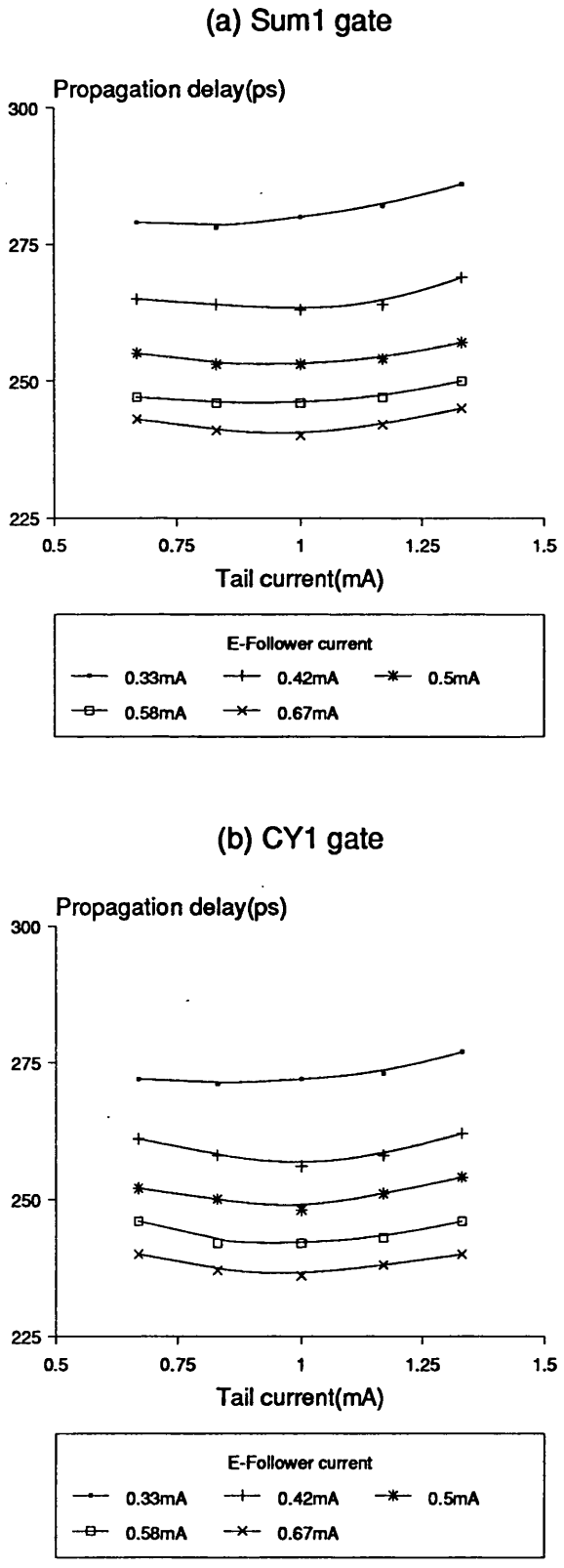


Figure 6.9. Worst-case delay of a modified full-adder implemented with AND gate-normal full-adder when input B_0 to AND gate changes (Fanout=1). **(a)** Sum1 gate, and **(b)** CY1 gate.

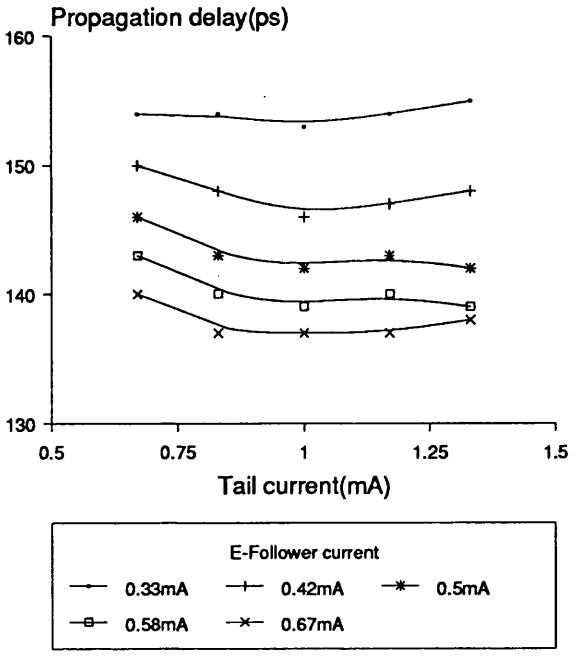


Figure 6.10. Worst-case delay of a 2-input AND series gated ECL gate (Fanout=1).

fact, it is sufficient to use one AND gate to drive both the sum and carry functions of a normal full-adder without any considerable loss of speed. Compared to a full-adder, under the same operating conditions, the AND gate's worst-case delay is about 100ps better than a CSA full-adder. Furthermore, if a comparison is made, say for an emitter follower current of 0.4mA, the propagation delay of the modified full-adder resulting from changes at the AND gate is nearly as fast as a CSA full-adder (compare the curves of Figure 6.9(a) with those of Figure 3.9(b)). Note that the Sum function of a normal full-adder is inherently slower that the Carry function and so it is only necessary to compare the Sum gate's propagation delay.

If we examine its implementation in the megacells of the (2,2,3) counter multiplier, during each operation of the multiplier, the worst-case delay of a modified full-adder resulting from changes at the

B_1 input would only occur once in megacell $C_{i,0}$. However, in megacells $C_{n-1,j}$ and $C_{i,j}$ where the lowest level input to the modified full-adder component of counter *CT2* is a steady signal from counter *CT1*, the critical delay of the modified full-adder in *CT2* would thus corresponds to changes at the B_0 input of the AND gate. As evident from Figure 6.9(a) this is nearly as fast as a CSA full-adder cell. Although each modified full-adder has to drive three gates (this can be reduced to two by using a single AND gate for both functions of a modified full-adder) of the next stage as opposed to two for a CSA full-adder, any slight increase in the delay of the modified full-adder due to the extra fanout would be more than compensated by the faster EXOR gate of the following stage. Thus, in this respect the modified full-adder can be assumed, on average to be as fast as a CSA full-adder.

6.3.3. Comparisons of the two implementations of a modified full-adder

The results show that there is better performance in the propagation delay of the AND gate-normal full-adder configuration in megacells $C_{n-1,j}$ and $C_{i,j}$ compared to its implementation in 4-level ECL gates. The reason is not difficult to see; with the lowest level input being a steady signal, the arrival of three propagating inputs gives a maximum number of three differential delays for the 4-level ECL implementation. However, the AND gate-normal full-adder configuration is far better than the 4-level ECL implementation, since at most only three differential delays are liable to switch, whether caused by changes at input B_0 or input B_1. This also means that in megacells $C_{i,0}$ of the (2,2,3) counter multiplier, the implementation of the modified full-adder in the AND gate-normal full-adder configuration would give a far superior speed than the 4-level ECL configuration.

In fact, when used in the diagonal cells of the multiplier architecture, the AND gate-normal full-adder combination, in general, would prove to be significantly better in terms of speed since at most two diodes drops are required in the emitter followers to level shift the outputs compared to three for the 4-level ECL gates. Each level shifting diode in the emitter follower, as found from simulations, typically adds about 45ps (for a fan-out of three) to the propagation delay. This becomes crucial especially in the diagonal cells of the (2,2,3) counter multiplier where a series of (2,2,3) counter cells are connected.

Furthermore, with the AND gate-normal full-adder configuration of the modified full-adder, the total number of different types of logic gates required is the same as a CSA multiplier scheme. These are the sum and carry functions of a full-adder and half-adder, which is basically composed of a 2-input EXOR gate and a 2-input AND gate.

Although SPICE shows that the 4-level ECL gate implementation appears workable for a power supply of -5.2v, the practical implementation of 4-level gates is questionable since no known designs have been fabricated with such gates. There are some major factors that still need to be addressed and investigated. For one thing, a logic swing of 0.25v although proven to work for the 3-level gates of the MPC test chip, might not be adequate for the 4-level gates. It was observed from SPICE that there was some degree of attenuation of the bias current through the transistor tree as we go from the lower level input to the highest level, and also of the output voltage through the level shift network used to get from the collector output down to the base input. With variations expected in the on-chip circuit parameters, the degree of attenuation might be worse which thus calls for a much higher logic swing. A higher logic swing unfortunately increases the possibility of

the transistors to saturate heavily and increases the propagation delay. Also, the reference voltage Vref has to be tightly controlled to -3.8v since any deviations would probably caused the onset of heavy saturation or limit the noise margin across the resistors of the current sources. In fact some ringing in the output was observed as Vref was reduced to -3.85. This is probably due to coupling of the signal across the collector-base of the current source transistor. Since the reference generator has some finite impedance (it is not unusual for this impedance to appear inductive depending on the design of the reference), the coupling could caused the reference voltage to be modulated and therefore give rise to ringing of the current sources connected to it which might then degrade the speed. Thus, for a power supply of -5.2v, the 4-level transistor tree might prove to be a bit of a squeeze when actually designed on silicon. The seriousness of the above problems could not be determined unless the 4-level gates are fabricated and as yet, no data is available from any known designs.

The three-level ECL gates for a power supply of -5.2v, on the other hand are well-proven in previous designs such as the MPC test chip. Thus, based on available data from known designs and from SPICE simulation results, in terms of speed, reliability and practicality of design under standard operating conditions, the AND gate-normal full-adder would prove superior than the 4-level gate implementation.

In Chapter 4, it was evaluated that if the (2,2,3) counter is as fast as a CSA full-adder, a considerable improvement in speed can be achieved with the (2,2,3) counter multiplier compared to a CSA multiplier, for large operand wordlength. The degree of improvement over a conventional CSA multiplier will first be determined for a 8 x 8-bit multiplication under true operating conditions taking into account

effects such as fanout on the delays of the series gated ECL gates. This will allow us to determine the validity of equation 4.9, which describes the speed of a n x n-bit multiplication based on the assumptions made and thus an evaluation of the speed improvement of a n x n-bit (2,2,3) counter multiplier over the CSA multiplier can then be done. This is discussed in the next sections.

6.4. Speed comparison of the (2,2,3) counter and conventional CSA multiplier

Because of the slightly irregular structure of the (2,2,3) counter multiplier architecture in terms of counter size and interconnection pattern, a straight-forward estimate of their critical delay cannot be made based on one type of cell, unlike the highly regular full-adder CSA multiplier. This is further complicated by the fact that in the (2,2,3) multiplier there are four types of megacells with different number of steady signals (partial product terms) and propagating signals.

Ideally, to get as accurate a comparison as possible, a SPICE simulation has to be performed on the whole multiplier architecture. This is obviously not practical with the available VAX11/750 minicomputer. However, a fairly good estimate could be determined by doing a gate-level simulation using HILO where the gates concerned are given delays as obtained from SPICE simulation results. Although the exact behaviour of ECL gates is not built in HILO, their delay characteristics could be modelled using the avaiable HILO gate primitives.

6.4.1. Delay characteristics of ECL gates

The delay of ECL gates is basically divided, to a first order, into three parts. The bulk of the propagation delay is accounted for by the number of differential levels that are forced to switch. This is followed

by the emitter follower delay which also includes the delay due to any diode drops needed to level shift the outputs. The third factor which determines the delay is the fanout.

From the simulation results obtained on the full-adder cell and the 4-level ECL gates, roughly the delay for a given gate is proportionately dependent on the number of differential levels in the transistor tree that are forced to switch. The less than ideal linear behaviour can be attributed, among many factors, to the different wiring capacitances at the collector nodes of each differential pair since the number of wires that connects adjacent levels varies according to the function of the gate. An analysis of the parameters that determines a differential pair delay has actually been widely studied by Barna [6.4-6.5].

The emitter follower, which serves to give a high drive capability and as a level shifting network normally adds a small percentage to the total delay depending on the fanout and switching current. Each level shifting collector-base connected diode in the emitter follower typically adds around 25-45ps (for the HP1X process) depending on the bias current and fanout.

A comparison in speed of the (2,2,3) counter multiplier and CSA multiplier is made for a given value of tail and emitter follower current. Within a circuit chip the reference voltage is assumed to be generated by a single reference circuit and distributed throughout the chip so it is appropriate to consider the tail and emitter follower current sources to be controlled by the same reference. A single AND gate is employed to drive both the sum and carry function of a modified full-adder since this is found to be adequate without any significant degradation in its speed, and also to make some savings in the number of gates (and, hence power) required. By doing this, it can be noticed that

the fanout of any gate at any level of the hierarchy in both the (2,2,3) counter multiplier and the conventional CSA multiplier is two which would greatly simplify the design process of the multipliers and make comparisons in their relative speeds more precise. A reference voltage of -3.8v is employed by virtue of the fact that adequate control of the current sources could be attained with fairly high value resistors (meaning high output impedance of the current sources) for a given current. All the gates are given load resistors of 250Ω, transistors of emitter size 1 x 10μm, and resistances of 600Ω and 1.2KΩ for the tail current sources and emitter follower current sources, respectively.

Gate	No. of levels switching (ps)		Further delays due to level-shift diodes(ps)	
	1	2	1	2
2-input AND	103	144	34	62
2-input EXOR	122	181	26	53

Table 6.3. Delays of a 2-input AND and 2-input EXOR gate for fanout = 2 (Tail current = 1mA; Emitter Follower current = 0.5mA).

Gate	No. of levels switching (ps)			Further delays due to level-shift diodes(ps)	
	1	2	3	1	2
Sum	122	200	266	41	60
Carry	117	179	242	47	79

Table 6.4. Delays of sum and carry function of a full-adder for fanout=2 (Tail current = 1mA; Emitter Follower current = 0.5mA).

The results of propagation delay of all the gates employed in both multiplier schemes are summarized in Table 6.3 and Table 6.4, as a function of number of differential levels switching and level-shifting diodes used for a fanout of two. The tables show only the actual fanout, number of differential levels that can switch and level-shifting diodes needed by each gate in both multipliers.

6.4.2. Modelling the delay behaviour of ECL gates in HILO

The delay due to a change at any input level is modelled into HILO by using the buffer primitive elements which are given the appropriate delays as obtained from SPICE, before the actual definition of the gate's function. The delays embedded in these buffers include the total differential delays plus the delay due to an emitter follower stage without any level shifting diodes, driving the number of gates encountered by the function in either the CSA multiplier or (2,2,3) counter multiplier. As identified in the last section all the gates throughout the hierarchy of the (2,2,3) counter multiplier have a fanout of two by employing just one AND gate in the modified full-adder which then gives the same fanout as the gates employed in the CSA multiplier. Where level shifting diodes are required, their contribution towards the total delay is also modelled with buffer primitive elements, but connected at the output of the function. HILO primitive gates which define the function of the cells are set with zero delays in order to preserve the actual delay of the function as obtained from SPICE. The HILO input description and schematics of the various cells used in the logic simulation of a 8 x 8-bit (2,2,3) counter and conventional CSA multipliers are illustrated in Appendix B. The large plot of the (2,2,3) counter multiplier demonstrates clearly the regularity of the

architecture. The CSA multiplier has the same interconnections as that shown in Figure 2.1 for a 5 x 5-bit multiplication with extra rows and columns of cells added to give a 8 x 8-bit multiplier.

The procedure to model the gates in HILO was done mostly using Computervision's schematic capture design package. This allows quick data entry and gives us the capability to explore and exercise many special features of HILO easily and interactively.

6.4.3. HILO simulation results

The results of simulation runs on a 8 x 8-bit CSA multiplier and an equivalent (2,2,3) counter multiplier are shown in Table 6.5 and Table 6.6, respectively for some random input vectors. In Table 6.5, simulation (b) exercises one of the critical path of the CSA multiplier which shows a worst-case delay of 2743ps. This path corresponds to the diagonal sum path from the top left cell to the bottom right cell and through the carry path of the bottom row cells.

In Table 6.6, simulation (a) and (c) demonstrates the variation of the sum delay path through the megacells of the (2,2,3) counter multiplier. Simulation (d) and (f) illustrates some of the longest delay encountered when all of the final product bits change states; these, however do not actually exercise the critical path of the multiplier. From discussions in section 4.4.2, the critical path is the "staircase" path which traverses alternately along the C1 output and S0 output of megacells $C_{i,0}$, $C_{i,j}$ and $C_{i,i}$, and through the half-adders and (2,2,3) counters (the carry path) of the diagonal cells. Attempts were made to determine the input vectors which truly exercises the critical path of the multiplier but this was found to be difficult. Instead, calculations were made based on observations of HILO and SPICE simulation results

```
Parameters=printfile,displayfile,initfile,scratch,coresize=1000 Kbytes
Computervision Corp.
CDS3000 version - Fault Free Simulator.

HILO MARK 3.1I.5  -  23-JUN-1989 13:07

(C) GenRad,  Inc.  1987
*rsf multrun
END OF SUBFILE
8 OF PPGEN1
8 OF PPGEN
48 OF PPGEN2
7 OF HADDER
42 OF FADDER
7 OF FADDR1
7 OF EXOR2
Delayscale = NS
Time to load 3.38 cpu secs.
MULTCSA LOADED OK
Initialisation complete   cpu secs = 0.66    (total = 0.66)
                                        FFFFFF
                              FFFFFFFFFFPPPPPP
          AAAAAAAA BBBBBBBB PPPPPPPPPP[[[[[[
          [[[[[[[[ [[[[[[[[ [[[[[[[[[[111111
          01234567 01234567 0123456789012345
          ]]]]]]]] ]]]]]]]] ]]]]]]]]]]]]]]]]

----TIME---

        0    11111111 00000001 0000000000000000                    (a)

      513    11111111 00000001 0000000000000010
      532    11111111 00000001 0000000100000010
      839    11111111 00000001 0000000111111110

     5000    11111111 10000001 0000000111111110                    (b)
     5178    11111111 10000001 1000000111111110
     5385    11111111 10000001 1100000111111110
     5626    11111111 10000001 1110000111111110
     5867    11111111 10000001 1111000111111110
     6108    11111111 10000001 1111100111111110
     6349    11111111 10000001 1111110111111110
     6590    11111111 10000001 1111111111111110
     6850    11111111 10000001 1111111011111110
     7057    11111111 10000001 1111111001111110
     7158    11111111 10000001 1111111000111110
     7275    11111111 10000001 1111111000011110
     7392    11111111 10000001 1111111000001110
     7509    11111111 10000001 1111111000000110
     7626    11111111 10000001 1111111000000010
     7697    11111111 10000001 1111111000000011
     7743    11111111 10000001 1111111000000001
```

Table 6.5. HILO simulation results of a 8 x 8-bit CSA multiplier.

10000	11111111	00100000	1111111000000001	(c)
10178	11111111	00100000	0111111000000001	
10385	11111111	00100000	0011111000000001	
10448	11111111	00100000	0011111000000000	
10513	11111111	00100000	0001111000000010	
10532	11111111	00100000	0001111100000010	
10626	11111111	00100000	0011111100000010	
10736	11111111	00100000	0011111110000010	
10839	11111111	00100000	0011111100111110	
10937	11111111	00100000	0011111100000000	
11737	11111111	00100000	0011111101000000	
12044	11111111	00100000	0011111111000000	
15000	11111111	01000000	0011111111000000	(d)
15292	11111111	01000000	0111111111000000	
16901	11111111	01000000	0111111110000000	
20000	10101010	01000000	0111111110000000	(e)
20492	10101010	01000000	0101111110000000	
20974	10101010	01000000	0101011110000000	
21456	10101010	01000000	0101010110000000	
21875	10101010	01000000	0101010100000000	
25000	00111100	01000000	0101010100000000	(f)
25251	00111100	01000000	0001010100000000	
25974	00111100	01000000	0001110100000000	
26456	00111100	01000000	0001111100000000	
26716	00111100	01000000	0001111000000000	
30000	01010101	10001000	0001111000000000	(g)
30385	01010101	10001000	0101111000000000	
30754	01010101	10001000	0101101000000000	
30852	01010101	10001000	0101100000000000	
30988	01010101	10001000	0101100100000000	
31015	01010101	10001000	0101000100000000	
31255	01010101	10001000	0101000000010000	
31422	01010101	10001000	0101000010010000	
31497	01010101	10001000	0101001010010000	
31562	01010101	10001000	0101001011010000	
31617	01010101	10001000	0101001111010000	
31850	01010101	10001000	0101001011010000	
35000	00001111	00011000	0101001011010000	(h)
35344	00001111	00011000	0001001011010000	
35713	00001111	00011000	0001011011010000	
35811	00001111	00011000	0001010011010000	
35826	00001111	00011000	0000010011010000	
36214	00001111	00011000	0000010111010000	
36254	00001111	00011000	0000010111011000	
36349	00001111	00011000	0000000111011000	
36378	00001111	00011000	0000000101001000	
36412	00001111	00011000	0000000101101000	
36496	00001111	00011000	0000000001101000	
36536	00001111	00011000	0000000000101000	

Table 6.5. (continued).

```
    36803    00001111 00011000 0000000001101000
    36850    00001111 00011000 0000000101101000

    40000    10111101 00011011 0000000101101000                    (i)
    40472    10111101 00011011 0001100101101000
    40513    10111101 00011011 0001101101101010
    40532    10111101 00011011 0001101001101010
    40736    10111101 00011011 0001101011011010
    40825    10111101 00011011 0001101011011011
    40837    10111101 00011011 0001101011001011
    40871    10111101 00011011 0001101011001001
    40954    10111101 00011011 0001110011000001
    41071    10111101 00011011 0001110011000101
    41074    10111101 00011011 0001110111000101
    41080    10111101 00011011 0001110101110101
    41178    10111101 00011011 0001110100100101
    41195    10111101 00011011 0001111100100101
    41238    10111101 00011011 0001111101100101
    41315    10111101 00011011 0001111001100101
    41378    10111101 00011011 0001111011110101
    41476    10111101 00011011 0001111010111101
    41593    10111101 00011011 0001111010111001
    41762    10111101 00011011 0001111011111001

finish at 50000
finish simulation   cpu secs = 1.56    (total = 2.22)
*quit

HILO END OF RUN 23-JUN-1989 13:07
```

Table 6.5. (continued).

```
Parameters=printfile,displayfile,initfile,scratch,coresize=1000 Kbytes
Computervision Corp.
CDS3000 version - Fault Free Simulator.

HILO MARK 3.1I.5  -  23-JUN-1989 17:45

(C) GenRad,  Inc.  1987
*rsf multrun
END OF SUBFILE
68 OF PPGEN
40 OF COUNTER223
7 OF MCELLI0
7 OF HADDER
3 OF MCELLII
6 OF MCELLIJ
3 OF MCELLN1J
47 OF EXOR2
40 OF MODFADDER
58 OF FADDER
Delayscale = NS
Time to load 5.33 cpu secs.
MULT223 LOADED OK
Initialisation complete   cpu secs = 0.95   (total = 0.95)
                                        FFFFFF
                                FFFFFFFFFFPPPPPP
          AAAAAAAA  BBBBBBBB  PPPPPPPPPP111111
          01234567  01234567  0123456789012345

----TIME---

        0    11111111  10000000  0000000000000000              (a)

      144    11111111  10000000  1000000000000000
      385    11111111  10000000  1100000000000000
      566    11111111  10000000  1110000000000000
      747    11111111  10000000  1111000000000000
     1007    11111111  10000000  1111100000000000
     1088    11111111  10000000  1111110000000000
     1339    11111111  10000000  1111111000000000
     1568    11111111  10000000  1111111100000000

     5000    00000000  00000000  1111111100000000              (b)
     5103    00000000  00000000  0111111100000000
     5344    00000000  00000000  0011111100000000
     5525    00000000  00000000  0001111100000000
     5706    00000000  00000000  0000111100000000
     5966    00000000  00000000  0000011100000000
     6047    00000000  00000000  0000001100000000
     6298    00000000  00000000  0000000100000000
     6527    00000000  00000000  0000000000000000
```

Table 6.6. HILO simulation results of a 8 x 8-bit (2,2,3) counter multiplier.

10000	11111111	00000001	0000000000000000	(c)
10940	11111111	00000001	0000000000000100	
10955	11111111	00000001	0000000000000110	
11289	11111111	00000001	0000000000001110	
11328	11111111	00000001	0000000000011110	
11399	11111111	00000001	0000000000111110	
11469	11111111	00000001	0000000001111110	
11475	11111111	00000001	0000000101111110	
11592	11111111	00000001	0000000111111110	
15000	11111111	10000001	0000000111111110	(d)
15144	11111111	10000001	1000000111111110	
15385	11111111	10000001	1100000111111110	
15566	11111111	10000001	1110000111111110	
15747	11111111	10000001	1111000111111110	
16007	11111111	10000001	1111100111111110	
16088	11111111	10000001	1111110111111110	
16339	11111111	10000001	1111111111111110	
16568	11111111	10000001	1111111011111110	
16658	11111111	10000001	1111111001111110	
16710	11111111	10000001	1111111001011110	
16804	11111111	10000001	1111111000011110	
16922	11111111	10000001	1111111000011100	
16928	11111111	10000001	1111111000010100	
16931	11111111	10000001	1111111000010000	
16991	11111111	10000001	1111111000000000	
16996	11111111	10000001	1111111000000001	
20000	00000000	00000000	1111111000000001	(e)
20103	00000000	00000000	0111111000000001	
20344	00000000	00000000	0011111000000001	
20525	00000000	00000000	0001111000000001	
20706	00000000	00000000	0000111000000001	
20899	00000000	00000000	0000111000000101	
20946	00000000	00000000	0000111000000100	
20966	00000000	00000000	0000011000000100	
21047	00000000	00000000	0000001000000100	
21209	00000000	00000000	0000001000000000	
21287	00000000	00000000	0000001000010000	
21298	00000000	00000000	0000000000010000	
21428	00000000	00000000	0000000001010000	
21597	00000000	00000000	0000000001000000	
21681	00000000	00000000	0000000000000000	
25000	01000000	00000010	0000000000000000	(f)
26502	01000000	00000010	0000000100000000	
30000	01000101	10010010	0000000100000000	(g)
30385	01000101	10010010	0100000100000000	
30941	01000101	10010010	0100100100000000	
30992	01000101	10010010	0100100100000100	
31088	01000101	10010010	0100110100000100	
31370	01000101	10010010	0100110100010100	
31493	01000101	10010010	0100110100110100	

Table 6.6. (continued).

```
    31568    01000101  10010010  0100110000110100
    31700    01000101  10010010  0100110001110100

    35000    00000000  00000000  0100110001110100          (h)
    35344    00000000  00000000  0000110001110100
    35900    00000000  00000000  0000010001110100
    35992    00000000  00000000  0000010001110000
    36047    00000000  00000000  0000000001110000
    36370    00000000  00000000  0000000001100000
    36452    00000000  00000000  0000000001000000
    36543    00000000  00000000  0000000000000000

    40000    01111111  00000010  0000000000000000          (i)
    40935    01111111  00000010  0000000000001000
    41033    01111111  00000010  0000000000001100
    41158    01111111  00000010  0000000000101100
    41411    01111111  00000010  0000000000111100
    41477    01111111  00000010  0000000010111100
    41502    01111111  00000010  0000000110111100
    41552    01111111  00000010  0000000111111100

    45000    01111111  10000010  0000000111111100          (j)
    45385    01111111  10000010  0100000111111100
    45566    01111111  10000010  0110000111111100
    45747    01111111  10000010  0111000111111100
    46007    01111111  10000010  0111100111111100
    46088    01111111  10000010  0111110111111100
    46339    01111111  10000010  0111111111111100
    46568    01111111  10000010  0111111011111100
    46604    01111111  10000010  0111111010111100
    46665    01111111  10000010  0111111000111100
    46735    01111111  10000010  0111111000011100
    46852    01111111  10000010  0111111000001100
    46909    01111111  10000010  0111111000001000
    47014    01111111  10000010  0111111000000000
    47098    01111111  10000010  0111111000000010

finish at 50000
finish simulation   cpu secs = 1.36    (total = 2.31)
*quit

HILO END OF RUN 23-JUN-1989 17:45
```

Table 6.6. (continued).

in which the worst-case delay path of each of the megacells and the diagonal cells were determined in HILO and then verified by equivalent SPICE simulations. Logic simulation shows that the worst-case delay of megacell $C_{i,0}$ occurs when there is a change in partial product $a_i b_1$ and not due to propagating input $S0_{i+1,0}$ (refer to Figure 4.9(a) and Appendix B-3) as assumed in section 4.4.2. The total delay of output $C1_{i,0}$ when this happens is 695ps. For megacell $C_{i,j}$, changes at input $C1_{i,j-1}$ introduce a delay of 122ps for the $S0_{i,j}$ output whilst input $S0_{i+1,j}$ causes output $C1_{i,j}$ to change after a delay of 388ps. In megacell $C_{i,i}$ a change at the $S0_{i+1,i}$ input gives a delay of 179ps at its $C1_{i,i}$ output. For the diagonal cells, transitions at the A0 input of a (2,2,3) counter introduce a delay of 220ps at the CY1 output while the half-adder contributes a delay of 103ps to its carry output. Adding up all the delays introduced by the cells in the critical path, a worst-case delay of 2249ps was obtained for the (2,2,3) counter multiplier. SPICE simulations done on each of the megacells and diagonal cells actually verify their worst-case delay paths to within ± 70ps. Although this means the HILO descriptions do not accurately model the ECL gates, the logic simulation results should give a fairly good estimate of the relative speeds of the (2,2,3) counter and CSA multiplier since the same type of gates, which have identical fanout are used in both (ignoring effects of different wiring parasitics).

It can thus be observed that for a 8 x 8-bit multiplication, the (2,2,3) counter multiplier is better than the CSA multiplier by 494ps which represents about 18% improvement over the CSA multiplier. In contrast, if we use equations 2.1 and 4.9 to estimate their relative speeds, the (2,2,3) counter multiplier is faster by four unit gate delays which is about 26.7% of the CSA multiplier's speed. The difference can be

attributed to two related factors. In equation 4.9 we have assumed that the (2,2,3) counter has a unit gate delay comparable to a CSA full-adder cell. While the results show this to be true when a comparison is made with all but the bottom row full-adder cells of the carry-save array, it is not exactly valid when compared with the bottom row full-adder cells. Essentially, the bottom row full-adder cells are ripple adders with the carry propagating across the cells. Each pass through these cells introduce only one differential delay as opposed to two in the main array cells and from SPICE simulation this is about 65% of the delay of the main array full-adder cell. On the other hand, each diagonal (2,2,3) counter cell which serve the same purpose as the bottom row cells of the CSA, is slightly faster than two passes through the bottom row CSA. To get a much better assessment of the relative speeds of the CSA multiplier and (2,2,3) counter multiplier, a more involved analysis that would take into account the above factors is obviously needed. However, equations 2.1 and 4.9 can be used to give a first order estimate of both multipliers. As evident from Figure 4.14, greater improvements in multiplication speed can be expected for large operand wordlength. For a 16 x 16-bit multiplication, equation 4.9 shows a faster speed over the CSA multiplier by about 29%, which in numbers is about 800ps. It is interesting to note that the speed of a (2,2,3) counter multiplier lies between that of a straight CSA multiplier and a Booth-encoded CSA multiplier incorporated with fast final adders. Some minor improvements in speed and hardware savings can be achieved for the (2,2,3) counter multiplier by replacing the second stage (2,2,3) counter *(CT2)* in megacells $C_{i,0}$ and $C_{n-1,j}$ with a (1,2,3) counter although this would mean adding a different type of gate in the implementation of the multiplier.

Table 4.6(b) and Table 4.8 illustrates the relative gate count and

power consumption of both schemes based on the assumption that any single logic function of up to five input variables can be implemented with a single cascode ECL gate. However in the eventual implementation, each modified full-adder is composed of three gates, a 2-input AND and the two gates of a normal full-adder. Thus, the factor contributed by the AND gate for each modified full-adder in Table 4.6(b), is given by

$$\sum_{x=1}^{n/2-2} 4x + (\frac{9}{2}n-8) \qquad \text{for even n.}$$

$$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (4x+2) + (\frac{9n-17}{2}) \qquad \text{for odd n.}$$

Using Table 4.8 and the AND gate contribution given above, for a n x n-bit multiplication, the total ECL gate count can be summarized in Table 6.7 where it is compared with that for a CSA multiplier.

For a 8 x 8-bit multiplication, the total gate count required for the (2,2,3) counter multiplier is about 56% more than the CSA multiplier,

	Total ECL gate count	
(2,2,3) Counter multiplier	even n	odd n
	$\sum_{x=1}^{n/2-2} (32x) + (28n-45)$	$\sum_{x=0}^{\lfloor n/2 \rfloor -2} (32x+16) + (28n-48)$
CSA multiplier	$2(n-1)^2 + n^2 + 2(n-1)$	

Table 6.7. Total ECL gate count of (2,2,3) counter multiplier (where modified full-adder is implemented with AND gate-normal full-adder) and conventional CSA multiplier.

whilst for 16 x 16-bit multiplication, the figure is 46% more. These figures would give the relative power consumption of both schemes if all gates are driven with the same tail current and emitter follower current. However, a much lower power consumption can be achieved for the (2,2,3) counter multiplier without any appreciable loss of speed by driving the relatively large number of gates such as the first stage (2,2,3) counters in the megacells which do not lie in the critical path of the multiplier, at lower power dissipation.

6.5. Summary

The results show that the (2,2,3) counter multiplier has a significant improvement in speed over the CSA multiplier by implementing the modified full-adder with a AND gate-normal full-adder configuration, which has a comparable delay to that of a CSA full-adder. It was determined that the 4-level ECL implementation of the modified full-adder is worse in terms of speed, reliability and practicality of design. For a 8 x 8-bit multiplication, simulations show that the (2,2,3) counter is faster than the CSA multiplier by about 18% and is expected to give higher improvements for larger operand wordlengths.

The (5,3) counter cell implemented by the multiplexer approach, shows a speed larger than twice the delay of a CSA full-adder which means a multiplication speed worse than the CSA multiplier. Using the novel threshold logic circuit technique, it is feasible to design higher order counters as large as a (7,3); however the propagation delay was much worse than expected due to unnecessary and extreme transitions of the threshold levels of the gates.

REFERENCES

[6.1] L.W. Nagel, " SPICE2 : A computer program to simulate semiconductor circuits ", Memo ERL-M520, University of California, Berkeley, CA, May 9, 1975.

[6.2] HILO-3 user manual, GenRad Inc., Revision 2, 15 June, 1985.

[6.3] C.R. Baugh & B.A. Wooley, " Stastistical analysis of a differential threshold logic circuit configuration ", IEEE Trans. Computers, Vol. C-25, No. 7, pp. 745-754, July 1976.

[6.4] Barna, " Propagation delay in current mode switching circuits ", IEEE J. Solid-Stae Circuits, pp. 123-124, April 1975.

[6.5] A. Barna, " Analytic approximations for propagation delays in current mode switching circuits including collector-base capacitances ", pp. 597-599, Oct. 1981.

[6.6]. P.R. Gray & R.G. Meyer, Analysis and design of Analog Integrated Circuits, Wiley, 1984.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

7.1. Background

As one of the basic components of digital signal processors (DSP), general and special-purpose computer applications, the digital multiplier is often one of the most vital elements, and because of the inherent complexity of the operation, the execution speed tends to be the dominant factor in the entire processing time. With the advent of VLSI technology which brought about reduced cost of fabrication per transistor on a chip, parallel algorithms for multiplication has become increasingly important.

Schemes for parallel multiplication are roughly divisible into two classes - iterative array of full-adder cells such as the full-adder carry-save array (CSA) approach [7.1], and generation of a matrix of partial product terms with subsequent reduction of the matrix e.g. Dadda and Wallace tree [7.1-7.4]. Compared to the linear delay of the iterative array multiplier, matrix reduction method is faster, especially for large operand wordlength n as it is logarithmically dependent on n. For VLSI implementations of a single chip multiplier, however, the reduction technique, requiring a large number of irregular connections between different types of cells is at a major disadvantage over the iterative array approach, which has a highly regular structure. Because of this, the iterative array technique such as the CSA multiplier lends itself well to automatic generation design tools like a silicon compiler [7.5] and is usually employed in single chip DSP. Often multiplier recoding techniques like the modified Booth algorithm [7.6] and fast final adders [7.7,7.8] are incorporated with the iterative array technique to boost its speed. This method, usually employed in most single chip multiplier applications is, however, more complicated and normally would occupy a

relatively large area of silicon and power consumption apart from the fact that it takes considerably longer to design, and thus, is unsuitable for implementation in single chip DSP applications.

With current trends towards single chip digital signal processors and the growing demand for more powerful and real-time performance of such processors, it is obvious that further improvements in speed would need to be made on the full-adder based CSA multiplier architecture. Considerable increases in the speed of array multipliers can be achieved by adding more than one partial product bit at a time by employing higher order parallel (p,q) counters [7.4]. This approach largely depends on a counter which has a delay and complexity comparable to that of a full-adder.

Attempting to extend the CSA approach to higher order parallel (p,q) counters results in architectures that is actually worse than the full-adder CSA scheme in terms of speed and its attractiveness for VLSI implementation. No significant reduction in the number of counter stages needed in the array could be attained and also it was observed that there is an uneven distribution of partial product bits and incomplete utilisation of some counter cells if the homogeneous nature of the array were to be preserved. Considerable attention has been paid to methods of synthesizing large parallel counters in the past. This includes techniques based on look-up tables or ROMs [7.9], sequential circuits [7.10,7.11], networks of full-adders or smaller counters [7.12,7.13] and threshold logic [7.3,7.14]. These methods, however, result in counters that are too slow, costly and impractical to gain any significant improvement in the speed of digital multipliers.

7.2. Research summary

An iterative array multiplier based on a (5,3) counter cell was recently reported by Nakamura [7.15,7.16] and it was claimed that, assuming the counters operate at a comparable speed as a CSA full-adder, the architecture is faster than the conventional CSA multiplier by nearly a factor of two. A novel array architecture based on a (2,2,3) counter was developed in this project as an extension of the (5,3) counter scheme and it was shown to offer significant enhancement in speed for large operand wordlength. The study shows that both the (5,3) counter and (2,2,3) counter architectures are quite close to conventional array multipliers from a VLSI implementation point of view. The folded collated square matrix of partial product terms introduced by Nakamura proved to be one of the key factors which gives the optimum (5,3) counter and (2,2,3) counter multiplier architectures its regular structure in terms of interconnections between cells, the even distribution of partial product terms and the complete utilisation of counter inputs.

In this work, the (5,3) counter and (2,2,3) counter were studied, principally on the efficiency of operation speed and the viability of the array architectures in the fast bipolar ECL technology. For this purpose a reconsideration of threshold logic, in view of the better processes of today as well as the well-proven series gated ECL technique was investigated.

Two techniques were employed to realise the (5,3) counter . In the proposed design in series gated ECL, the process of computing the steady signals before the propagating signal reach the cell yields a counter which is basically composed of logic blocks of full-adder and some other functions corresponding to the possible output state of the counter that were then selected by multiplexers. It is also possible to extend this technique to a (7,3) counter but in terms of cost-effectiveness the (5,3)

counter is better since the higher order counters require more gates and without any possible enhancement in speed. In the second method, a novel threshold circuit technique based on partial use of negative weighted inputs was proposed to overcome the maximum fan-in weight limitation found in traditional threshold circuits which would make it more feasible to synthesize a (5,3) counter of low complexity. Although, it was observed that the technique might not be as fast as an equivalent series gated ECL full-adder cell, it is of great interest since higher order counters up to a (7,3) could be designed with little increase in complexity and propagation delay. Such large counters should be useful in the inherently faster architectures of Wallace's and Dadda's [7.2,7.3] and in applications where a large counter needs to be designed with little increase in complexity.

The (2,2,3) counter is composed of a 2-input EXOR gate and a modified full-adder which are basically the diagonal cells of the (5,3) counter multiplier. The critical component of this counter is the 4-input modified full-adder since a 2-input EXOR gate can be designed, in principle with a lower propagation delay than a normal full-adder. Two techniques were proposed to synthesize the modified full-adder. The first method uses four levels of series gated ECL to realise each of the functions of the modified full-adder. In theory, it was observed that it is feasible to accommodate four levels of differential switches for a standard power supply of -5.2v without causing the transistors to saturate which could seriously degrade the speed. An efficient method of mapping a logic function onto series gated ECL suitable for software implementation was introduced. In the second method, a 2-input AND gate and a normal full-adder are employed to realise the modified full-adder. Both methods suggest a modified full-adder that is potentially as fast as a normal full-adder without a considerable increase in complexity.

SPICE simulations were carried out to characterize the propagation delay behaviour of the various cells of different techniques and to make a critical comparison of their relative speeds with a CSA full-adder cell. These were performed for a given load resistance, transistor size and as a function of tail current and emitter follower current.

7.3. Conclusions and discussions

The results show that for the (5,3) counter implemented with the novel threshold logic circuit technique the maximum fan-in weight is effectively double that of traditional threshold circuits by separating the inputs to negatively weighted and positively weighted inputs. This allows a reliable design of the (5,3) counter and can be extended to higher order counters up to a (7,3). However, the critical propagation delay of the circuits was found to be much worse than expected. Extreme excursions of V+ and V- of each threshold gate determine the worst-case delay since this corresponds to a relatively high voltage swing at the comparators differential pair, and since the threshold of one gate is controlled by the output of a previous gate, momentary changes at the counters outputs can occur which seriously degrades its propagation delay. Although some improvements can be made by changing the threshold logic configuration slightly to reduce this effect, the (5,3) counter is still several times slower than a CSA full-adder. Unless an efficient method of limiting V+ and V- under extreme input states can be made like clamping them to a certain maximum value, the circuit is not expected to rival equivalent Boolean ECL gates in terms of speed.

The (5,3) counter implemented with multiplexers show a critical delay worse than two times the delay of a CSA full-adder. The need to drive three gates of the following stage (5,3) counter along with the requirement to level shifts the outputs by two diode drops has actually

considerably degraded the speed of the multiplexers although they can be observed to introduce only one further differential delay. This is further aggravated by the fact that all three differential levels of the first stage logic gates of each (5,3) counter are liable to switch. With such counters which has a delay twice that of a CSA full-adder, the (5,3) counter multiplier architecture is not expected to be faster than the conventional CSA multiplier.

Simulations show that the 4-level gates of a modified full-adder has a worst-case delay between one and two times the delay of a CSA full-adder. Although SPICE simulations suggest that the 4-level ECL implementation of a modified full-adder appears workable for a standard power supply of -5.2v, it was determined that problems could arise when the circuits are actually fabricated. A much higher logic swing than that employed for 3-level ECL gates might be required for the gates which would accordingly reduce the number of levels that can be accommodated if saturation of the transistors were to be avoided. This implies that a higher supply voltage of more than -5.2v might be required which would then seriously complicates the design of the circuits. No known designs of a 4-level series gated ECL gate has been reported in the past. In contrast, the AND gate-normal full-adder configuration of the modified full-adder which are well-proven in previous known designs such as the MPC test chip, proved to be superior than the 4-level ECL implementation not only from the point of view of reliability and practicality of design under standard operating conditions, but also in terms of speed. The speed of the modified full-adder when implemented in all but megacell $C_{i,0}$ of the (2,2,3) counter multiplier is nearly as fast as a CSA full-adder. Any marginal increase in speed is also more than compensated by the faster EXOR gate of the following stage megacell, since the delay path through the megacells traverses a majority of alternate blocks of

full-adders and modified full-adder. Thus based on available data from previous designs and from SPICE simulation results, in terms of speed, reliability and practicality of design under standard operating conditions, the AND gate-normal full-adder configuration of the modified full-adder proves to be superior than the 4-level ECL implementation.

A gate-level logic simulation (using the HILO package) was performed on both the (2,2,3) counter multiplier and the conventional CSA multiplier to assess their relative speeds. This was done based on SPICE results obtained with all gates having the same load resistors, transistor size and for a given value of tail current and emitter follower current. The results show that for a 8 x 8-bit multiplication, the (2,2,3) counter multiplier is faster than the conventional CSA multiplier by a factor of more than 18% with an increase in gate count of about 56% (and hence, power consumption if all gates are assumed to be biased with the same tail current and emitter follower current).

In general, for a n x n-bit multiplication, the worst-case delay D_n of a (2,2,3) counter multiplier (including the delay due to generation of partial product terms) can be approximated in terms of unit gate delays given by

$$D_n = (n+1) + \left\lfloor \frac{n}{3} \right\rfloor$$

whereas the critical delay for a conventional CSA multiplier is given by

$D_n = (2n - 1)$.

It is interesting to note that the speed and gate count of a (2,2,3) counter multiplier lies between that of a straight CSA multiplier and a Booth-encoded CSA multiplier incorporated with fast final adders. For a 16 x 16-bit multiplication, it is estimated that the (2,2,3) counter is faster than the CSA multiplier by about 29% with a hardware increase of 46%. Thus, the (2,2,3) counter is more cost-effective for large operand

wordlength, as also evident from Figures 4.14(a)-(b). Reduction in the power consumption of the (2,2,3) counter multiplier can be achieved without any loss of speed by operating a large number of gates which do not lie in the critical path, at lower tail current and emitter follower current.

In terms of VLSI implementation, the (2,2,3) counter multiplier is as attractive as the CSA multiplier. Although, observation of the (2,2,3) counter architecture at the highest level of the hierarchy suggests that there are more types of cells involved, the same type of gates as used in the CSA multiplier i.e. the two functions of a half-adder and full-adder are actually needed in the (2,2,3) counter scheme. Eventhough the (2,2,3) counter architecture is not as regular as a CSA multiplier, this should not be a serious constraint, since, once a layout has been made for a given size multiplication, the scheme can be easily extended by adding extra rows and columns of cells to fit the specific application.

The (2,2,3) counter array multiplier should thus be attractive for future implementations of high-speed multipliers, particularly, in single chip real-time digital signal processors, where a much faster but yet regular array multiplier is often needed to cope with the increased complexity of such chips, and where a multiplier is required which can be designed quickly.

7.4. Recommendations for future work

Future work will concentrate on fabricating a prototype (2,2,3) counter multiplier and a conventional CSA multiplier for a small size multiplication, say, for a 8 x 8-bit, using state-of-the-art bipolar technology such as the HP1X process. Since the same type of gates are employed in both architectures, this will enable us to address the relative wiring complexity, ease of design, a more accurate speed-power

product, and also this will give some figures of the total silicon area needed by the (2,2,3) counter multiplier in comparison with a CSA multiplier. In terms of gate count, the results show that for a 8 x 8-bit multiplication, the (2,2,3) counter scheme, in contrast with the CSA multiplier, is larger by a factor of about 56%, whilst for a 16 x 16-bit multiplication, it is 46% more. These figures, however, do not reflect accurately the relative total silicon area occupied since it depends to a large extent on the complexity and regularity of wiring between cells. Furthermore, these figures were obtained based on the assumption that all gates i.e. the 2-input EXOR and AND gate, and the full-adder gates are identical in all respects. It is also interesting to find out how far the (2,2,3) architecture can be expanded for larger wordlength before the power consumption and silicon area needed becomes a constraint. It was identified that some of the gates in the megacells of the (2,2,3) counter multiplier which do not lie in the critical paths of the multiplier can be operated at lower power consumption without any loss of speed incurred. Some work should therefore be done to investigate on how much power can be saved by this treatment. These work will then allow a more complete assessment on the attractiveness of the (2,2,3) counter multiplier architecture for incorporation in single chip digital signal processors. In the long term, it is hoped that a program will be written which will generate automatically the layout of a (2,2,3) counter multiplier for a given size wordlength. This is quite feasible, since, once the layout of the various cells/megacells have been designed for a small size multiplication the architecture can be easily expanded like the CSA multiplier, by adding extra rows and columns of cells to fit the specific application. Such program should prove useful in automatic generation design tools such as silicon compilers [7.5].

The viability of the (2,2,3) counter multiplier architecture in

other technologies such as MOS technology will need to be investigated. Indeed, a majority of the digital signal processors that are available in the market today are MOS chips. This is not surprising since MOS technology offers higher packing density and lower power consumption which allows more complex DSP functions to be integrated onto a single chip. A higher performance array multiplier architecture would undoubtedly enhance the computation speed to compensate the inherently higher propagation delay of MOS circuits. The efficiency of operation speed and the complexity of a modified full-adder would need to be investigated. In fact, Nakamura [7.16] did not address the implementation of a modified full-adder in MOS technology, and thus it is appropriate that future work will need to look at the viability of the (2,2,3) counter array scheme in this technology.

Several other important areas can also be identified where further work would prove salutary. These are summarized below.

(i) It was shown that it is feasible to overcome the fan-in weight constraint found in traditional threshold circuits by using the proposed novel threshold circuit technique. Indeed, the analysis suggests that the circuit should be able to accommodate counters as large as a (7,3) counter, which might be useful in the inherently faster architectures of Wallace's and Dadda's. The feasibility of using such higher counters in these multiplication schemes would need to be fully investigated. The speed of the threshold circuits was found to be limited by extreme excursions of the summing node voltage V+ and threshold level V-. Unless an efficient method of limiting V+ and V- to a certain value is employed it is unlikely that such circuits will rival equivalent Boolean ECL gates. Thus some work is needed to investigate at ways of clamping V+ and V- efficiently. Also a more accurate analysis of the minimum unit weight voltage is needed than the worst-case approximation treatment presented

in this work. Ideally, a statistical analysis of the threshold circuits would need to be done along with an equivalent Monte Carlo simulation to verify the analysis. This then gives a more optimistic view of the circuits operation and thus better evaluation of the maximum fan-in weight possible can be made.

(ii) A method of mapping a logic function onto series gated ECL was proposed where it was shown to be suitable for software implementation. Such a program would prove beneficial for IC designers to speed up their design process.

(iii) It is apparent that the key to the fast operation of the (2,2,3) counter multiplier is the inherent high-speed of the modified full-adder cell. Some work should therefore be undertaken to investigate the attractiveness of employing such cells in other arithmetic operations such as in high-speed binary additions and divisions.

(iv) Multiplier array architectures based on different binary number representation have received relatively little attention in the past. One potential area is the signed digit (SD) number system [7.17,7.18], which was shown to be attractive for future VLSI implementations of large multipliers. It was estimated that it can be much faster than equivalent conventional array multipliers since their critical delay behaviour depends logarithmically on the wordlength n (similar to the matrix reduction techniques of Dadda and Wallace), and yet can be as regular as conventional array multipliers. Array architectures should therefore be investigated based on such number representation.

REFERENCES

[7.1] S. Waser, " High speed monolithic multiplier for real time digital signal processing, " Computer, Vol. 11, no. 10, pp. 19-29, Oct. 1978.

[7.2] C.S. Wallace, " A suggestion for parallel multipliers, " IEEE Trans. Electron. Comput. Vol. EC-13, pp 14-17, Feb. 1964.

[7.3] L. Dadda, " Some schemes for parallel multipliers, " Alta Frequenza, Vol. 34, pp. 349-356, Mar. 1965.

[7.4] W.J. Stenzel, W.J. Kubitz & G.H. Garcia, " A compact high-speed parallel multiplication scheme, " IEEE Trans Comput., Vol. C-26, pp. 948-957, Oct. 1977.

[7.5] N.F. Benschop, " Layout compiler for variable array multipliers ", Proc. Custom Integrated Circuits Conference, 1983.

[7.6] L.P. Rubinfield, " A proof of the modified Booth's algorithm for multiplication ", IEEE Trans. Computers, pp. 1014-1015, Oct. 1975.

[7.7] J. Sklansky, " An evaluation of several two-summand binary adders ", IRE Trans. Electronic Computers, pp. 213-226, June 1960.

[7.8] J. Sklansky, " Conditional-sum addition logic ", IRE Trans. Electronic Computers, pp. 226-231, June 1960.

[7.9] L. Dadda, " On parallel digital multipliers ", Alta Frequenza, pp. 574-580, Oct. 1976.

[7.10] A. Svoboda, " Adder with distributed control ", IEEE Trans. Computers, Vol. C-19, No. 8, pp. 749-751, August 1970.

[7.11] S. Singh & R. Waxman, " Multiple operand addition and multiplication ", IEEE Trans. Computers, Vol. C-22, No. 2, pp. 113-120, Feb. 1973.

[7.12] C. C. Foster & F. D. Stockton, " Counting responders in an associative memory ", IEEE Trans. Computers, pp. 1580-1583, Dec. 1971.

[7.13] H. Kobayashi & H. Ohara, " A synthesizing method for large parallel counters with a network of smaller ones ", IEEE Trans. Computers, Vol. C-27, No. 8, pp. 753-757, August 1978.

[7.14] D.Hampel & R.O. Winder, " Threshold logic ", IEEE Spectrum, pp. 32-39, May 1971.

[7.15] S. Nakamura, "Algorithms for iterative array multiplication," IEEE Transactions on computers, Vol. C-35, No.8, August,1986.

[7.16] S. Nakamura, " A single chip parallel multiplier by MOS technology ", IEEE Trans. Computers, Vol. 37, N0. 3, pp. 274-282, March 1988.

[7.17] N. Takagi, H. Yasuura, S. Yajima, " High-speed VLSI multiplication algorithm with a redundant binary addition, " IEEE Trans. on Computers, Vol. C-34, no. 9, Sept. 1985.

[7.18] Y. Harata et al, " High-speed multiplier using a redundant binary adder tree ", Proc. IEEE ICCD, pp. 165-170, Oct. 1984.

APPENDIX A

**** SPICE model parameters for HP1X transistor of emitter size 1 x 5μm ****

```
.MODEL C1005 NPN
+IS=1.12E-16 BF=150 NF=1.08 VAF=15 IKF=3.65E-3 ISE=5F NE=2.2
+BR=1.0 NR=1.0 VAR=2.0 IKR=2M ISC=1F NC=2.6 RB=126
+IRB=14.3E-6 RBM=71 RE=28 RC=18 CJE=19.0F  VJE=.338 MJE=.289
+TF=9.0E-12 XTF=6.5 VTF=3.0 ITF=4.28E-3 CJC=51.8F VJC=.441 MJC=.20
+XCJC=.45 TR=.25N CJS=109.5F VJS=.33 MJS=.12 XTB=1.5 EG=1.11
+XTI=3.0 FC=.70
```

**** SPICE model parameters for HP1X transistor of emitter size 1 x 10μm ****

```
.MODEL C1010 NPN
+IS=2.65E-16 BF=230 NF=1.08 VAF=25 IKF=4.0E-3 ISE=5F NE=2.2
+BR=1.0 NR=1.0 VAR=2.0 IKR=2.0E-3 ISC=1F NC=2.6 RB=110
+IRB=20.0E-6 RBM=50 RE=5 RC=18 CJE=29.0F  VJE=.338 MJE=.289
+TF=8.0E-12 XTF=6.5 VTF=3.0 ITF=10.0E-3 CJC=29.0F VJC=.441 MJC=.20
+XCJC=.45 TR=.25N CJS=100.0F VJS=.33 MJS=.12 XTB=1.5 EG=1.11
+XTI=3.0 FC=.70
```

**** SPICE model parameters for HP1X transistor of emitter size 1 x 20μm ****

```
.MODEL C1020 NPN
+IS=5.35E-16 BF=240 NF=1.09 VAF=28 IKF=4.2E-3 ISE=15F NE=2.9
+BR=1.0 NR=1.0 VAR=2.0 IKR=2.5E-6 ISC=1F NC=2.6 RB=85
+IRB=34.0E-6 RBM=38 RE=4.8 RC=10 CJE=50.0F  VJE=.338 MJE=.289
+TF=7.0E-12 XTF=6.5 VTF=3.0 ITF=20.0E-3 CJC=49.0F VJC=.441 MJC=.20
+XCJC=.45 TR=.25N CJS=120.0F VJS=.33 MJS=.12 XTB=1.5 EG=1.11
+XTI=3.0 FC=.70
```

**** SPICE model parameters for HP1X transistor of emitter size 1 x 50μm ****

```
.MODEL C1050 NPN
+IS=2.0E-15 BF=200 NF=1.12 VAF=5.5 IKF=12.0E-3 ISE=59.6F NE=3.73
+BR=1 NR=1 VAR=2 IKR=12.9U ISC=69.9F NC=2.48 RB=65
+IRB=38U RBM=22 RE=4.5 RC=4 CJE=105F VJE=.384 MJE=.352
+TF=5.0P XTF=12 VTF=7 ITF=40M CJC=114F VJC=.554 MJC=.299
+XCJC=.45 TR=.25N CJS=260F VJS=.42 MJS=.18 XTB=1.5 EG=1.11
+XTI=3.0 FC=.81
```

APPENDIX B

```
Parameters=printfile,displayfile,initfile,scratch,coresize=1000 Kbytes
Computervision Corp.
CDS3000 version - Fault Free Simulator.

HILO MARK 3.1I.5  -  23-JUN-1989 13:33

(C) GenRad,  Inc.  1987
*dis ppgen


****{
    1 :
    2 : CCT  PPGEN (PP,XP,YP);
    3 : AND (0,0,0,0,0) G1 (PP,W4,W3);
    4 : BUF (144,144,0,0,0) G2 (W4,YP);
    5 : BUF (103,103,0,0,0) G3 (W3,XP);
    6 : WIRE W3;
    7 : WIRE W4;
    8 : WIRE PP;
    9 : WIRE YP;
   10 : WIRE XP;
   11 : .

****}
*dis ppgen1


****{
    1 : CCT  PPGEN1 (PP,XP,YP);
    2 : AND (0,0,0,0,0) G1 (PPLS,W4,W3);
    3 : BUF (144,144,0,0,0) G2 (W4,YP);
    4 : BUF (103,103,0,0,0) G3 (W3,XP);
    5 : BUF (34,34,0,0,0) G4 (PP,PPLS);
    6 : WIRE W3;
    7 : WIRE W4;
    8 : WIRE PPLS;
    9 : WIRE PP;
   10 : WIRE YP;
   11 : WIRE XP;
   12 : .

****}
*dis ppgen2


****{
    1 : CCT  PPGEN2 (PP,XP,YP);
    2 : AND (0,0,0,0,0) G1 (PPLS,W4,W3);
    3 : BUF (144,144,0,0,0) G2 (W4,YP);
    4 : BUF (103,103,0,0,0) G3 (W3,XP);
    5 : BUF (62,62,0,0,0) G4 (PP,PPLS);
    6 : WIRE W3;
    7 : WIRE W4;
    8 : WIRE PPLS;
    9 : WIRE PP;
   10 : WIRE YP;
   11 : WIRE XP;
```

```
  12 : .

****}
*dis exor2


****{
   1 : CCT  EXOR2 (A,B,Q);
   2 : BUF (122,122,0,0,0) BFA (W1,A);
   3 : BUF (181,181,0,0,0) BFB (W2,B);
   4 : AND (0,0,0,0,0) G1 (W5,W2,W3);
   5 : AND (0,0,0,0,0) G2 (W6,W4,W1);
   6 : OR (0,0,0,0,0) G3 (Q,W5,W6);
   7 : NOT (0,0,0,0,0) G4 (W3,W1);
   8 : NOT (0,0,0,0,0) G5 (W4,W2);
   9 : WIRE W3;
  10 : WIRE W4;
  11 : WIRE W5;
  12 : WIRE W6;
  13 : WIRE W1;
  14 : WIRE W2;
  15 : WIRE Q;
  16 : WIRE B;
  17 : WIRE A;
  18 : .

****}
*dis hadder


****{
   1 : CCT  HADDER (B,A,CARRY,SUM);
   2 : EXOR2 GT1 (A,B,SM);
   3 : AND (0,0,0,0,0) GT2 (CARRY,W4,W3);
   4 : BUF (26,26,0,0,0) GT5 (SUM,SM);
   5 : BUF (103,103,0,0,0) GT3 (W3,A);
   6 : BUF (144,144,0,0,0) GT4 (W4,B);
   7 : WIRE W3;
   8 : WIRE W4;
   9 : WIRE CARRY;
  10 : WIRE SUM;
  11 : WIRE B;
  12 : WIRE A;
  13 : WIRE SM;
  14 : .

****}
*dis fadder


****{
   1 : CCT  FADDER (SUM,CARRY,IPA,IPB,IPC);
   2 : BUF (122,122,0,0,0) BFA (W4,IPA);
   3 : BUF (200,200,0,0,0) BFB (W5,IPB);
   4 : BUF (266,266,0,0,0) BFC (W10,IPC);
   5 : BUF (41,41,0,0,0) BFS (SUM,SM);
   6 : NOT (0,0,0,0,0) G1 (W6,W4);
```

```
   7 : OR (0,0,0,0,0) G10 (SM,W14,W15);
   8 : AND (0,0,0,0,0) G11 (W20,W18,W17);
   9 : AND (0,0,0,0,0) G12 (W21,W19,W18);
  10 : AND (0,0,0,0,0) G13 (W22,W19,W17);
  11 : OR (0,0,0,0,0) G14 (CARRY,W20,W21,W22);
  12 : BUF (117,117,0,0,0) G15 (W17,IPA);
  13 : BUF (179,179,0,0,0) G16 (W18,IPB);
  14 : BUF (242,242,0,0,0) G17 (W19,IPC);
  15 : NOT (0,0,0,0,0) G2 (W7,W5);
  16 : AND (0,0,0,0,0) G3 (W8,W5,W6);
  17 : AND (0,0,0,0,0) G4 (W9,W7,W4);
  18 : OR (0,0,0,0,0) G5 (W11,W8,W9);
  19 : NOT (0,0,0,0,0) G6 (W13,W10);
  20 : NOT (0,0,0,0,0) G7 (W12,W11);
  21 : AND (0,0,0,0,0) G8 (W14,W10,W12);
  22 : AND (0,0,0,0,0) G9 (W15,W13,W11);
  23 : WIRE W6;
  24 : WIRE W7;
  25 : WIRE W5;
  26 : WIRE W4;
  27 : WIRE W8;
  28 : WIRE W9;
  29 : WIRE W12;
  30 : WIRE W13;
  31 : WIRE W14;
  32 : WIRE W15;
  33 : WIRE W10;
  34 : WIRE W11;
  35 : WIRE W18;
  36 : WIRE W17;
  37 : WIRE W19;
  38 : WIRE W20;
  39 : WIRE W21;
  40 : WIRE W22;
  41 : WIRE CARRY;
  42 : WIRE SUM;
  43 : WIRE IPC;
  44 : WIRE IPB;
  45 : WIRE IPA;
  46 : WIRE SM;
  47 : .

****}
*dis faddr1


****{
   1 : CCT  FADDR1 (SUM,CARRY,IPA,IPB,IPC);
   2 : BUF (122,122,0,0,0) BFA (W4,IPA);
   3 : BUF (200,200,0,0,0) BFB (W5,IPB);
   4 : BUF (266,266,0,0,0) BFC (W10,IPC);
   5 : NOT (0,0,0,0,0) G1 (W6,W4);
   6 : OR (0,0,0,0,0) G10 (SM,W14,W15);
   7 : BUF (47,47,0,0,0) BFLS1 (CARRY,CY);
   8 : BUF (60,60,0,0,0) BFLS2 (SUM,SM);
   9 : AND (0,0,0,0,0) G11 (W20,W18,W17);
  10 : AND (0,0,0,0,0) G12 (W21,W19,W18);
```

```
  11 : AND (0,0,0,0,0) G13 (W22,W19,W17);
  12 : OR (0,0,0,0,0) G14 (CY,W20,W21,W22);
  13 : BUF (117,117,0,0,0) G15 (W17,IPA);
  14 : BUF (179,179,0,0,0) G16 (W18,IPB);
  15 : BUF (242,242,0,0,0) G17 (W19,IPC);
  16 : NOT (0,0,0,0,0) G2 (W7,W5);
  17 : AND (0,0,0,0,0) G3 (W8,W5,W6);
  18 : AND (0,0,0,0,0) G4 (W9,W7,W4);
  19 : OR (0,0,0,0,0) G5 (W11,W8,W9);
  20 : NOT (0,0,0,0,0) G6 (W13,W10);
  21 : NOT (0,0,0,0,0) G7 (W12,W11);
  22 : AND (0,0,0,0,0) G8 (W14,W10,W12);
  23 : AND (0,0,0,0,0) G9 (W15,W13,W11);
  24 : WIRE W6;
  25 : WIRE W7;
  26 : WIRE W5;
  27 : WIRE W4;
  28 : WIRE W8;
  29 : WIRE W9;
  30 : WIRE W12;
  31 : WIRE W13;
  32 : WIRE W14;
  33 : WIRE W15;
  34 : WIRE W10;
  35 : WIRE W11;
  36 : WIRE W18;
  37 : WIRE W17;
  38 : WIRE W19;
  39 : WIRE W20;
  40 : WIRE W21;
  41 : WIRE W22;
  42 : WIRE CARRY;
  43 : WIRE SUM;
  44 : WIRE IPC;
  45 : WIRE IPB;
  46 : WIRE IPA;
  47 : WIRE SM,CY;
  48 : .

****}
*dis mult8x8


****{
   1 : cct multcsa (fp[0:15],a[0:7],b[0:7],gnd);
   2 : ppgen1 ppg0[0:7] ({fp[0],pp0[1:7]},a[0:7],b[0]);
   3 : ppgen ppg1[0:7] (pp1[0:7],a[0:7],b[1]);
   4 : ppgen2 ppg2[0:7] (pp2[0:7],a[0:7],b[2])
   5 :        ppg3[0:7] (pp3[0:7],a[0:7],b[3])
   6 :        ppg4[0:7] (pp4[0:7],a[0:7],b[4])
   7 :        ppg5[0:7] (pp5[0:7],a[0:7],b[5])
   8 :        ppg6[0:7] (pp6[0:7],a[0:7],b[6])
   9 :        ppg7[0:7] (pp7[0:7],a[0:7],b[7]);
  10 : hadder hadd0[1:7] (pp0[1:7],pp1[0:6],wc0[1:7],{fp[1],ws0[1:6]});
  11 : fadder fadd1[1:7]
({fp[2],ws1[1:6]},wc1[1:7],wc0[1:7],{ws0[1:6],pp1[7]},
pp2[0:6])
```

```
  12 :            fadd2[1:7]
({fp[3],ws2[1:6]},wc2[1:7],wc1[1:7],{ws1[1:6],pp2[7]},
pp3[0:6])
  13 :            fadd3[1:7]
({fp[4],ws3[1:6]},wc3[1:7],wc2[1:7],{ws2[1:6],pp3[7]},
pp4[0:6])
  14 :            fadd4[1:7]
({fp[5],ws4[1:6]},wc4[1:7],wc3[1:7],{ws3[1:6],pp4[7]},
pp5[0:6])
  15 :            fadd5[1:7]
({fp[6],ws5[1:6]},wc5[1:7],wc4[1:7],{ws4[1:6],pp5[7]},
pp6[0:6])
  16 :            fadd7[1:7]
(fp[8:14],{wcy[1:6],fp[15]},{gnd,wcy[1:6]},wc6[1:7],{ws6[1:6],
pp7[7]});
  17 : faddr1  fadd6[1:7]
({fp[7],ws6[1:6]},wc6[1:7],wc5[1:7],{ws5[1:6],pp6[7]},
pp7[0:6]);
  18 : wire fp[0:15],a[0:7],b[0:7],pp0[1:7],pp1[0:7],pp2[0:7],pp3[0:7];
  19 : wire pp4[0:7],pp5[0:7],pp6[0:7],pp7[0:7];
  20 : wire
ws0[1:6],ws1[1:6],ws2[1:6],ws3[1:6],ws4[1:6],ws5[1:6],ws6[1:6];
  21 : wire
wc0[1:7],wc1[1:7],wc2[1:7],wc3[1:7],wc4[1:7],wc5[1:7],wc6[1:7];
  22 : wire wcy[1:6],gnd
  23 : .

****}
*quit

HILO END OF RUN 23-JUN-1989 13:34
```

```
Parameters=printfile,displayfile,initfile,scratch,coresize=1000 Kbytes
Computervision Corp.
CDS3000 version - Fault Free Simulator.

HILO MARK 3.1I.5  -  24-JUN-1989 12:23

(C) GenRad,  Inc.  1987
*dis ppgen


****{
   1 : CCT  PPGEN (PP,XP,YP);
   2 : AND (0,0,0,0,0) G1 (PP,W4,W3);
   3 : BUF (144,144,0,0,0) G2 (W4,YP);
   4 : BUF (103,103,0,0,0) G3 (W3,XP);
   5 : WIRE W3;
   6 : WIRE W4;
   7 : WIRE PP;
   8 : WIRE YP;
   9 : WIRE XP;
  10 : .

****}
*dis exor2


****{
   1 : CCT  EXOR2 (A,B,Q);
   2 : BUF (122,122,0,0,0) BFA (W1,A);
   3 : BUF (181,181,0,0,0) BFB (W2,B);
   4 : AND (0,0,0,0,0) G1 (W5,W2,W3);
   5 : AND (0,0,0,0,0) G2 (W6,W4,W1);
   6 : OR (0,0,0,0,0) G3 (Q,W5,W6);
   7 : NOT (0,0,0,0,0) G4 (W3,W1);
   8 : NOT (0,0,0,0,0) G5 (W4,W2);
   9 : WIRE W3;
  10 : WIRE W4;
  11 : WIRE W5;
  12 : WIRE W6;
  13 : WIRE W1;
  14 : WIRE W2;
  15 : WIRE Q;
  16 : WIRE B;
  17 : WIRE A;
  18 : .

****}
*dis hadder


****{
   1 : CCT  HADDER (A,B,CARRY,SUM);
   2 : EXOR2 GT1 (A,B,SUM);
   3 : AND (0,0,0,0,0) GT2 (CARRY,W4,W3);
   4 : BUF (103,103,0,0,0) GT3 (W3,A);
   5 : BUF (144,144,0,0,0) GT4 (W4,B);
   6 : WIRE W3;
```

```
   7 : WIRE W4;
   8 : WIRE CARRY;
   9 : WIRE SUM;
  10 : WIRE B;
  11 : WIRE A;
  12 : .

****}
*dis fadder


****{
   1 : CCT  FADDER (CARRY,IPA,IPB,IPC,SUM);
   2 : BUF (122,122,0,0,0) BFA (W4,IPA);
   3 : BUF (200,200,0,0,0) BFB (W5,IPB);
   4 : BUF (266,266,0,0,0) BFC (W10,IPC);
   5 : NOT (0,0,0,0,0) G1 (W6,W4);
   6 : OR (0,0,0,0,0) G10 (SUM,W14,W15);
   7 : AND (0,0,0,0,0) G11 (W20,W18,W17);
   8 : AND (0,0,0,0,0) G12 (W21,W19,W18);
   9 : AND (0,0,0,0,0) G13 (W22,W19,W17);
  10 : OR (0,0,0,0,0) G14 (CARRY,W20,W21,W22);
  11 : BUF (117,117,0,0,0) G15 (W17,IPA);
  12 : BUF (179,179,0,0,0) G16 (W18,IPB);
  13 : BUF (242,242,0,0,0) G17 (W19,IPC);
  14 : NOT (0,0,0,0,0) G2 (W7,W5);
  15 : AND (0,0,0,0,0) G3 (W8,W5,W6);
  16 : AND (0,0,0,0,0) G4 (W9,W7,W4);
  17 : OR (0,0,0,0,0) G5 (W11,W8,W9);
  18 : NOT (0,0,0,0,0) G6 (W13,W10);
  19 : NOT (0,0,0,0,0) G7 (W12,W11);
  20 : AND (0,0,0,0,0) G8 (W14,W10,W12);
  21 : AND (0,0,0,0,0) G9 (W15,W13,W11);
  22 : WIRE W6;
  23 : WIRE W7;
  24 : WIRE W5;
  25 : WIRE W4;
  26 : WIRE W8;
  27 : WIRE W9;
  28 : WIRE W12;
  29 : WIRE W13;
  30 : WIRE W14;
  31 : WIRE W15;
  32 : WIRE W10;
  33 : WIRE W11;
  34 : WIRE W18;
  35 : WIRE W17;
  36 : WIRE W19;
  37 : WIRE W20;
  38 : WIRE W21;
  39 : WIRE W22;
  40 : WIRE CARRY;
  41 : WIRE SUM;
  42 : WIRE IPC;
  43 : WIRE IPB;
  44 : WIRE IPA;
  45 : .
```

```
****}
*dis modfadder


****{
   1 : CCT  MODFADDER (A0,A1,B0,B1,CY1,SUM1);
   2 : BUF (103,103,0,0,0) BUFA0 (W2,A0);
   3 : BUF (144,144,0,0,0) BUFB0 (W3,B0);
   4 : FADDER FA1 (CY1,W1,A1,B1,SUM1);
   5 : AND (0,0,0,0,0) G1 (W1,W3,W2);
   6 : WIRE W2;
   7 : WIRE W3;
   8 : WIRE W1;
   9 : WIRE CY1;
  10 : WIRE SUM1;
  11 : WIRE B1;
  12 : WIRE B0;
  13 : WIRE A1;
  14 : WIRE A0;
  15 : .

****}
*dis counter223


****{
   1 : CCT  COUNTER223 (A0,A1,B0,B1,CY1,SUM0,SUM1);
   2 : EXOR2 G1 (A0,B0,SUM0);
   3 : MODFADDER MFA1 (A0,A1,B0,B1,CY1,SUM1);
   4 : WIRE CY1;
   5 : WIRE SUM1;
   6 : WIRE SUM0;
   7 : WIRE B1;
   8 : WIRE B0;
   9 : WIRE A1;
  10 : WIRE A0;
  11 : .

****}
*dis mcelli0


****{
   1 : CCT  MCELLI0 (OPC1,OPC2,OPS0,OPS1,PS0,A0,A1,AI,B0,B1,BI);
   2 : COUNTER223 CNT1 (W1I,W1K2,W1J2,W1L2,W6,W7,W5);
   3 : COUNTER223 CNT2 (W5,W6A,PS0,GND,W10,OPS1,OPC1);
   4 : BUF (47,47,0,0,0) DRP1D1 (W6A,W6);
   5 : BUF (26,26,0,0,0) DRP1D2 (OPS0,W7);
   6 : BUF (47,47,0,0,0) DRP1D3 (OPC2,W10);
   7 : BUF (34,34,0,0,0) DRP1D4 (W1J2,W1J1);
   8 : BUF (34,34,0,0,0) DRP1D5 (W1K2,W1K1);
   9 : BUF (62,62,0,0,0) DRP2D1 (W1L2,W1L1);
  10 : PPGEN PPG1 (W1I,A0,BI);
  11 : PPGEN PPG2 (W1J1,AI,B0);
  12 : PPGEN PPG3 (W1K1,A1,BI);
  13 : PPGEN PPG4 (W1L1,AI,B1);
```

```
  14 : WIRE W6A;
  15 : WIRE W10;
  16 : WIRE W7;
  17 : WIRE W1I;
  18 : WIRE W6;
  19 : WIRE W5;
  20 : WIRE W1J1;
  21 : WIRE W1J2;
  22 : WIRE W1K1;
  23 : WIRE W1K2;
  24 : WIRE W1L1;
  25 : WIRE W1L2;
  26 : WIRE AI;
  27 : WIRE OPS1;
  28 : WIRE OPS0;
  29 : WIRE B1;
  30 : WIRE B0;
  31 : WIRE A1;
  32 : WIRE A0;
  33 : WIRE PS0;
  34 : WIRE OPC2;
  35 : WIRE OPC1;
  36 : WIRE BI;
  37 : SUPPLY0 GND;
  38 : .

****}
*dis mcelln1j


****{
   1 : CCT  MCELLN1J (OPC1,OPC2,OPS0,OPS1,PC1,A2J,A2J1,AN1,B2J,B2J1,BN1);
   2 : COUNTER223 CNT1 (W1I,W1K2,W1J2,W1L2,W9,W5,W8);
   3 : COUNTER223 CNT2 (W12,W11,W10,GND,OPC2,W15,OPC1);
   4 : BUF (26,26,0,0,0) DRP1D1 (W6,W5);
   5 : BUF (41,41,0,0,0) DRP1D2 (W10,W8);
   6 : BUF (47,47,0,0,0) DRP1D3 (W11,W9);
   7 : BUF (26,26,0,0,0) DRP1D4 (OPS0,W14);
   8 : BUF (26,26,0,0,0) DRP1D5 (OPS1,W15);
   9 : BUF (34,34,0,0,0) DRP1D6 (W1J2,W1J1);
  10 : BUF (34,34,0,0,0) DRP1D7 (W1K2,W1K1);
  11 : BUF (62,62,0,0,0) DRP2D1 (W1L2,W1L1);
  12 : HADDER HA1 (PC1,W6,W12,W14);
  13 : PPGEN PPG1 (W1I,A2J,BN1);
  14 : PPGEN PPG2 (W1J1,AN1,B2J);
  15 : PPGEN PPG3 (W1K1,A2J1,BN1);
  16 : PPGEN PPG4 (W1L1,AN1,B2J1);
  17 : WIRE W5;
  18 : WIRE W6;
  19 : WIRE W8;
  20 : WIRE W9;
  21 : WIRE W10;
  22 : WIRE W11;
  23 : WIRE W12;
  24 : WIRE W14;
  25 : WIRE W15;
  26 : WIRE W1I;
```

```
  27 : WIRE W1J1;
  28 : WIRE W1J2;
  29 : WIRE W1K1;
  30 : WIRE W1K2;
  31 : WIRE W1L1;
  32 : WIRE W1L2;
  33 : WIRE A2J1;
  34 : SUPPLY0 GND;
  35 : WIRE OPS1;
  36 : WIRE OPS0;
  37 : WIRE PC1;
  38 : WIRE B2J;
  39 : WIRE A2J;
  40 : WIRE OPC2;
  41 : WIRE OPC1;
  42 : WIRE BN1;
  43 : WIRE B2J1;
  44 : WIRE AN1;
  45 : .

****}
*dis mcellij


****{
   1 : CCT MCELLIJ
(OPC1,OPC2,OPS0,OPS1,PC1,PC2,PS0,PS1,A2J,A2J1,AI,B2J,B2J1
   2 : ,BI);
   3 : COUNTER223 CNT1 (W1,W3D,W2D,W4D,W7,W5,W6);
   4 : COUNTER223 CNT2 (W13,W15,W12,W10,W19,W17,OPC1);
   5 : BUF (47,47,0,0,0) DRP1D1 (W13,W11);
   6 : BUF (47,47,0,0,0) DRP1D2 (W15,W14);
   7 : BUF (26,26,0,0,0) DRP1D3 (OPS1,W17);
   8 : BUF (47,47,0,0,0) DRP1D4 (OPC2,W19);
   9 : BUF (34,34,0,0,0) DRP1D5 (W2D,W2C);
  10 : BUF (34,34,0,0,0) DRP1D6 (W3D,W3C);
  11 : BUF (53,53,0,0,0) DRP2D1 (W8,W5);
  12 : BUF (60,60,0,0,0) DRP2D2 (W9,W6);
  13 : BUF (79,79,0,0,0) DRP2D3 (W10,W7);
  14 : BUF (62,62,0,0,0) DRP2D4 (W4D,W4C);
  15 : FADDER FA1 (W11,PC1,PS1,W8,OPS0);
  16 : FADDER FA2 (W14,PS0,PC2,W9,W12);
  17 : PPGEN PPG1 (W1,A2J,BI);
  18 : PPGEN PPG2 (W2C,AI,B2J);
  19 : PPGEN PPG3 (W3C,A2J1,BI);
  20 : PPGEN PPG4 (W4C,AI,B2J1);
  21 : WIRE W5;
  22 : WIRE W6;
  23 : WIRE W7;
  24 : WIRE W8;
  25 : WIRE W9;
  26 : WIRE W10;
  27 : WIRE W11;
  28 : WIRE W12;
  29 : WIRE W13;
  30 : WIRE W14;
  31 : WIRE W15;
```

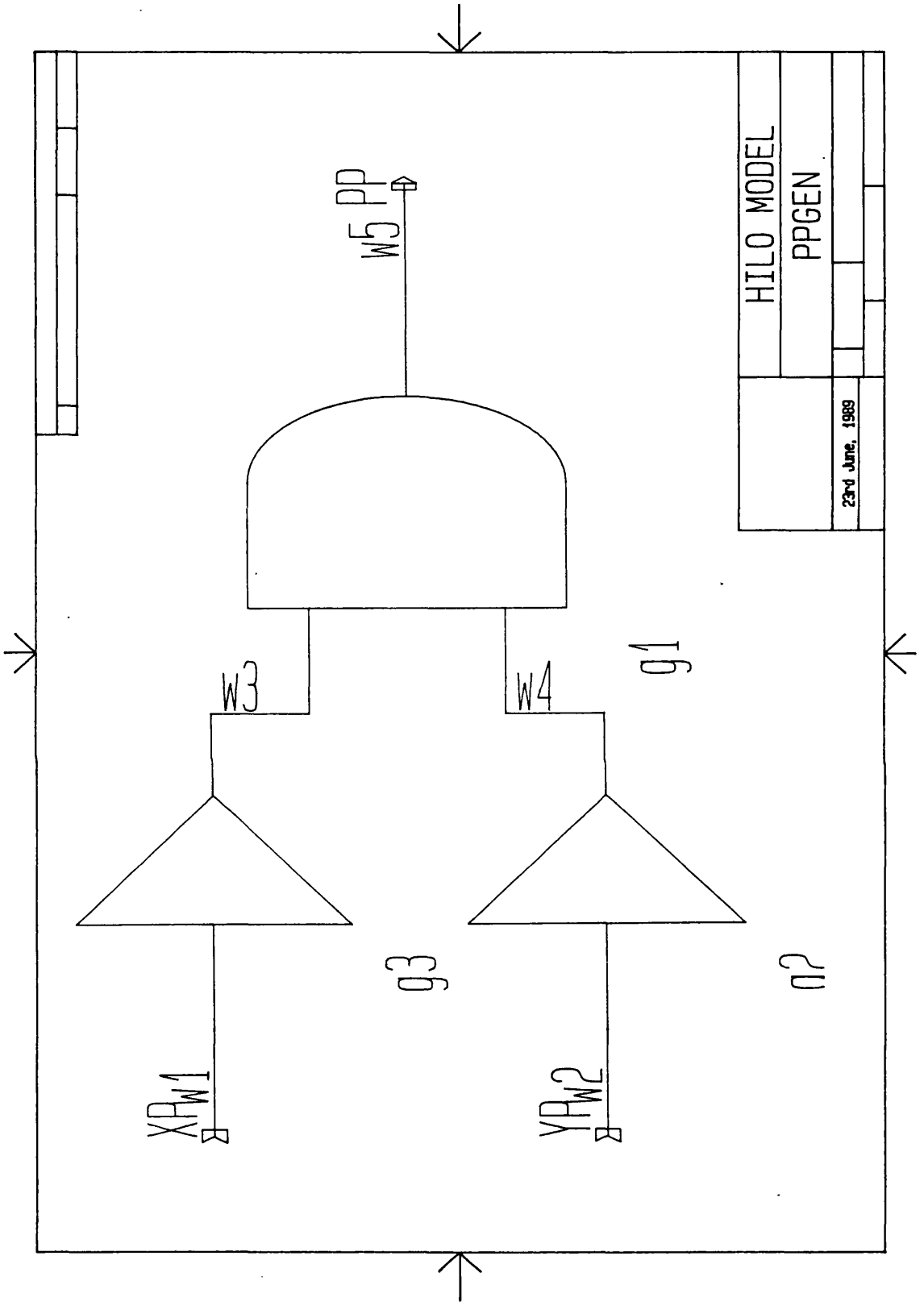
```
  32 : WIRE W17;
  33 : WIRE W19;
  34 : WIRE W1;
  35 : WIRE W2C;
  36 : WIRE W2D;
  37 : WIRE W3C;
  38 : WIRE W3D;
  39 : WIRE W4C;
  40 : WIRE W4D;
  41 : WIRE AI;
  42 : WIRE A2J1;
  43 : WIRE PC2;
  44 : WIRE OPS1;
  45 : WIRE PC1;
  46 : WIRE OPS0;
  47 : WIRE B2J;
  48 : WIRE A2J;
  49 : WIRE PS1;
  50 : WIRE PS0;
  51 : WIRE OPC2;
  52 : WIRE OPC1;
  53 : WIRE B2J1;
  54 : WIRE BI;
  55 : .

****}
*dis mcellii


****{
   1 : CCT  MCELLII (OPC1,OPS0,OPS1,PC1,PC2,PS0,PS1,AI,BI);
   2 : BUF (47,47,0,0,0) DRP1D1 (OPC1,W9);
   3 : BUF (41,41,0,0,0) DRP1D2 (OPS0,W7);
   4 : BUF (62,62,0,0,0) DRP2D1 (W1D,W1C);
   5 : FADDER FA1 (W4,PC1,PS1,W1D,W7);
   6 : FADDER FA2 (W9,W4,PS0,PC2,OPS1);
   7 : PPGEN PPG1 (W1C,AI,BI);
   8 : WIRE W4;
   9 : WIRE W7;
  10 : WIRE W9;
  11 : WIRE W1C;
  12 : WIRE W1D;
  13 : WIRE AI;
  14 : WIRE PC2;
  15 : WIRE OPS1;
  16 : WIRE PC1;
  17 : WIRE OPS0;
  18 : WIRE PS1;
  19 : WIRE PS0;
  20 : WIRE OPC1;
  21 : WIRE BI;
  22 : .

****}
*dis mult8x8
```

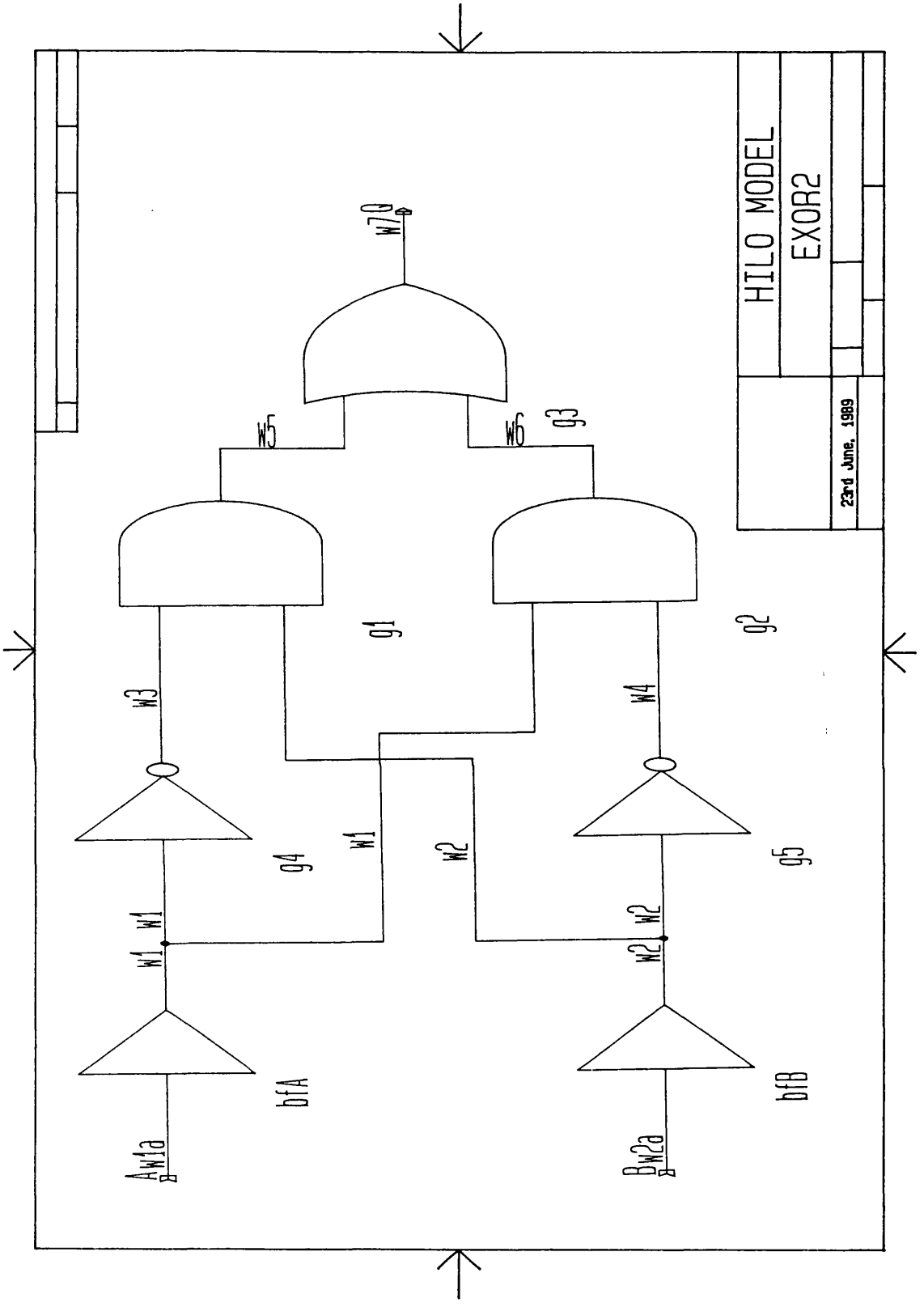
```
****{
   1 : CCT MULT223
(FP0,FP1,FP2,FP3,FP4,FP5,FP6,FP7,FP8,FP9,FP10,FP11,FP12,FP13
   2                                                                    :
,FP14,FP15,FP16,GND,N1J1,N1J2,N1J3,N1J4,A0,A1,A2,A3,A4,A5,A6,A7,B0,B1,B2
   3 : ,B3,B4,B5,B6,B7);
   4 : PPGEN C1 (FP0,A0,B0);
   5 : COUNTER223 C2 (W22A,W22B2,W12A2,W32A2,W122C1,FP6,FP7);
   6 : MCELLI0 C3 (W10C,W10D,FP1,FP2,W20A,A0,A1,A1,B0,GND,B1);
   7 : COUNTER223 C4 (W10C,W21A,W20B,W10D,W11C1,FP3,FP4);
   8 : HADDER C5 (W21B,W11C2,W12A1,FP5);
   9 : COUNTER223 C6 (W32,W42B1,W122C2,W32C2,W233C1,FP8,FP9);
  10 : MCELLI0 C7 (W20C,W20D,W20A,W20B,W30A,A0,A1,A2,B0,B1,B2);
  11 : MCELLII C8 (W21C,W21A,W21B,W20C,W20D,W31A,W30B,A2,B2);
  12 : HADDER C9 (W31B,W21C,W22B1,W22A);
  13 : COUNTER223 C10 (W43A,W43B2,W233C2,W53A1,W343C,FP10,FP11);
  14 : MCELLI0 C11 (W30C,W30D,W30A,W30B,W40A,A0,A1,A3,B0,B1,B3);
  15 : MCELLIJ C12
(W31C,W31D,W31A,W31B,W30C,W30D,W41A,W40B,A2,A3,A3,B2,GND,B3
  16 : );
  17 : COUNTER223 C13 (W31C,W42A,W41B,W31D,W32C1,W32A1,W32);
  18 : COUNTER223 C14 (W343C,W63B2,W53B2,W53C2,W454C,FP12,FP13);
  19 : MCELLI0 C15 (W40C,W40,W40A,W40B,W50A,A0,A1,A4,B0,B1,B4);
  20 : MCELLIJ C16
(W41C,W41D,W41A,W41B,W40C,W40,W51A,W50B,A2,A3,A4,B2,B3,B4);
  21 : MCELLII C17 (W42C,W42A,W42B,W41C,W41D,W52A,W51B,A4,B4);
  22 : HADDER C18 (W52B,W42C,W43B1,W43A);
  23 : COUNTER223 C19 (W454C,W64B2,W64A2,W73C1,FP16,FP14,FP15);
  24 : MCELLI0 C20 (W50C,W50D,W50A,W50B,W60A,A0,A1,A5,B0,B1,B5);
  25 : MCELLIJ C21
(W51C,W51D,W51A,W51B,W50C,W50D,W61A,W60B,A2,A3,A5,B2,B3,B5
  26 : );
  27 : MCELLIJ C22
(W52C,W52D,W52A,W52B,W51C,W51D,W62A,W61B,A4,A5,A5,B4,GND,B5
  28 : );
  29 : COUNTER223 C23 (W52C,W63A,W62B,W52D,W53C1,W53A,W53B1);
  30 : MCELLI0 C24 (W60C,W60D,W60A,W60B,W70A,A0,A1,A6,B0,B1,B6);
  31 : MCELLIJ C25
(W61C,W61D,W61A,W61B,W60C,W60D,W71A,W70B,A2,A3,A6,B2,B3,B6
  32 : );
  33 : MCELLIJ C26
(W62C,W62D,W62A,W62B,W61C,W61D,W72A,W71B,A4,A5,A6,B4,B5,B6
  34 : );
  35 : MCELLII C27 (W63C,W63A,W63B1,W62C,W62D,W73A,W72B,A6,B6);
  36 : HADDER C28 (W73B,W63C,W64B1,W64A1);
  37 : MCELLI0 C29 (W70C,N1J1,W70A,W70B,GND,A0,A1,A7,B0,B1,B7);
  38 : MCELLN1J C30 (W71C,N1J2,W71A,W71B,W70C,A2,A3,A7,B2,B3,B7);
  39 : MCELLN1J C31 (W72C,N1J3,W72A,W72B,W71C,A4,A5,A7,B4,B5,B7);
  40 : MCELLN1J C32 (W73C,N1J4,W73A,W73B,W72C,A6,A7,A7,B6,GND,B7);
  41 : BUF (47,47,0,0,0) DRP1D1 (W11C2,W11C1);
  42 : BUF (26,26,0,0,0) DRP1D10 (W64A2,W64A1);
  43 : BUF (34,34,0,0,0) DRP1D2 (W12A2,W12A1);
  44 : BUF (26,26,0,0,0) DRP1D2B (W32A2,W32A1);
  45 : BUF (47,47,0,0,0) DRP1D3 (W122C2,W122C1);
  46 : BUF (47,47,0,0,0) DRP1D4 (W233C2,W233C1);
  47 : BUF (41,41,0,0,0) DRP1D5 (W42B1,W42B);
  48 : BUF (34,34,0,0,0) DRP1D6 (W43B2,W43B1);
```

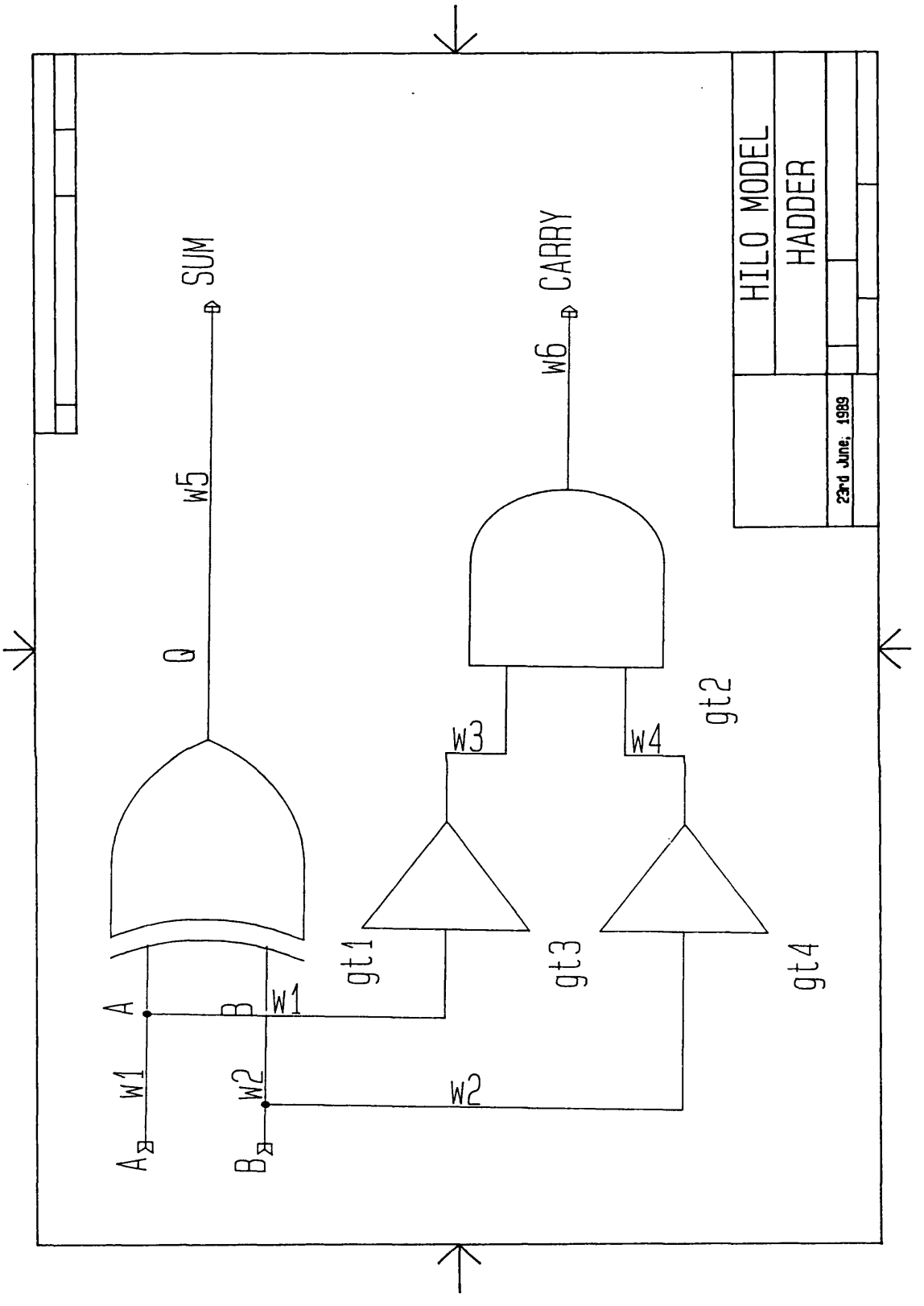
```
 49 : BUF (41,41,0,0,0) DRP1D7 (W63B2,W63B1);
 50 : BUF (41,41,0,0,0) DRP1D8 (W53B2,W53B1);
 51 : BUF (34,34,0,0,0) DRP1D9 (W64B2,W64B1);
 52 : BUF (79,79,0,0,0) DRP2D1 (W32C2,W32C1);
 53 : BUF (53,53,0,0,0) DRP2D2 (W53A1,W53A);
 54 : BUF (62,62,0,0,0) DRP2D2A (W22B2,W22B1);
 55 : BUF (79,79,0,0,0) DRP2D3 (W53C2,W53C1);
 56 : BUF (60,60,0,0,0) DRP2D5 (W73C1,W73C);
 57 : WIRE W70A;
 58 : WIRE W70B;
 59 : WIRE W70C;
 60 : WIRE W60A;
 61 : WIRE W60B;
 62 : WIRE W60C;
 63 : WIRE W60D;
 64 : WIRE W50D;
 65 : WIRE W50A;
 66 : WIRE W50B;
 67 : WIRE W50C;
 68 : WIRE W40A;
 69 : WIRE W40B;
 70 : WIRE W40C;
 71 : WIRE W40;
 72 : WIRE W30A;
 73 : WIRE W30B;
 74 : WIRE W30C;
 75 : WIRE W30D;
 76 : WIRE W20A;
 77 : WIRE W20D;
 78 : WIRE W20C;
 79 : WIRE W20B;
 80 : WIRE W10C;
 81 : WIRE W21A;
 82 : WIRE W10D;
 83 : WIRE W11C1;
 84 : WIRE W11C2;
 85 : WIRE W21B;
 86 : WIRE W21C;
 87 : WIRE W31A;
 88 : WIRE W31B;
 89 : WIRE W41A;
 90 : WIRE W31C;
 91 : WIRE W31D;
 92 : WIRE W42A;
 93 : WIRE W41C;
 94 : WIRE W41B;
 95 : WIRE W51A;
 96 : WIRE W51B;
 97 : WIRE W41D;
 98 : WIRE W51C;
 99 : WIRE W52A;
100 : WIRE W61A;
101 : WIRE W61B;
102 : WIRE W51D;
103 : WIRE W62A;
104 : WIRE W61C;
105 : WIRE W61D;
```

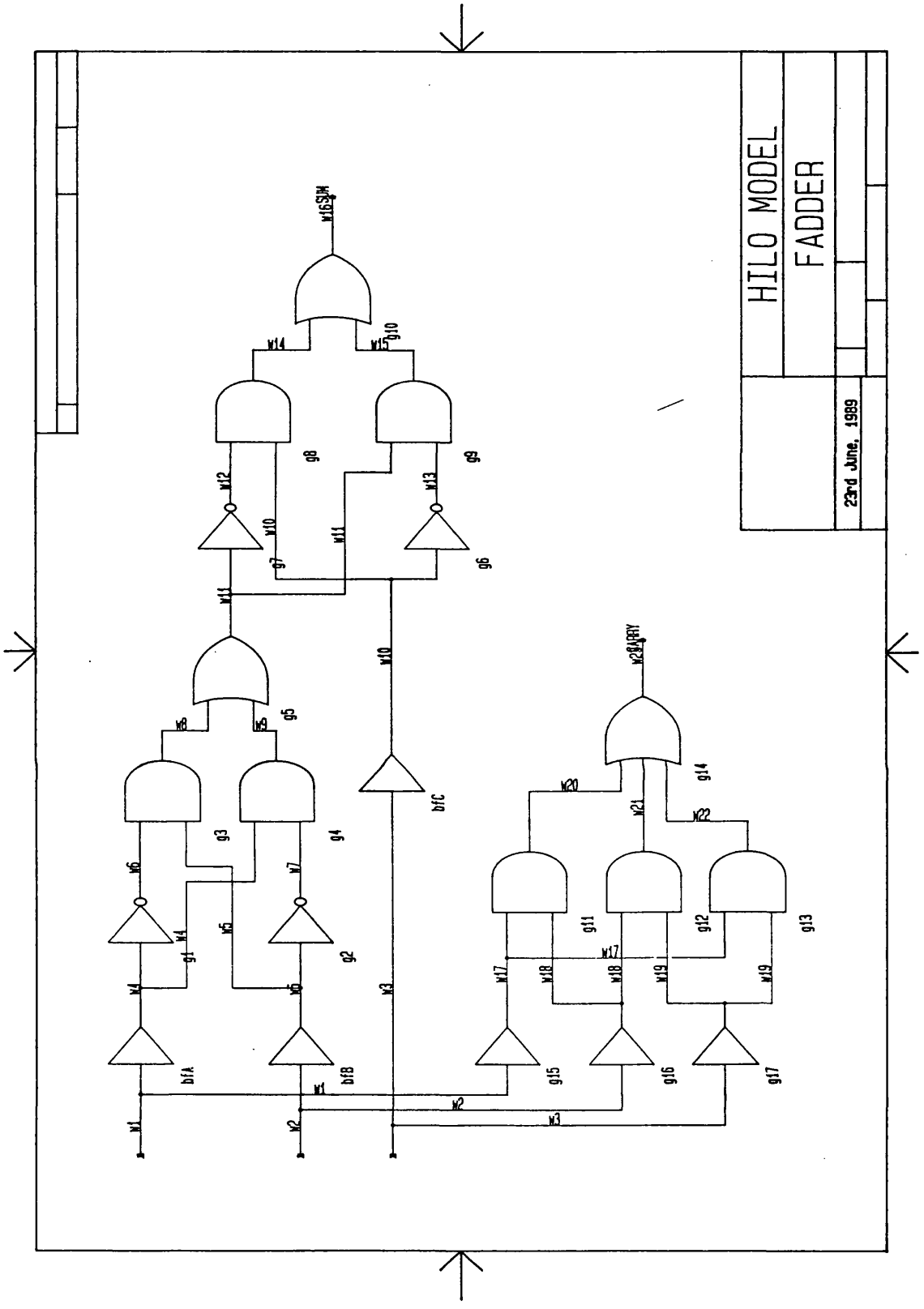
```
106 : WIRE W71A;
107 : WIRE W71B;
108 : WIRE W71C;
109 : WIRE W72A;
110 : WIRE W72B;
111 : WIRE W72C;
112 : WIRE W62D;
113 : WIRE W62C;
114 : WIRE W73A;
115 : WIRE W12A1;
116 : WIRE W22A;
117 : WIRE W52B;
118 : WIRE W42C;
119 : WIRE W52C;
120 : WIRE W52D;
121 : WIRE W62B;
122 : WIRE W63A;
123 : WIRE W12A2;
124 : WIRE W22B1;
125 : WIRE W22B2;
126 : WIRE W32A1;
127 : WIRE W32A2;
128 : WIRE W42B;
129 : WIRE W42B1;
130 : WIRE W32C1;
131 : WIRE W32C2;
132 : WIRE W43B1;
133 : WIRE W43B2;
134 : WIRE W53A;
135 : WIRE W53A1;
136 : WIRE W343C;
137 : WIRE W63B1;
138 : WIRE W63B2;
139 : WIRE W53C1;
140 : WIRE W53C2;
141 : WIRE W53B1;
142 : WIRE W53B2;
143 : WIRE W63C;
144 : WIRE W73B;
145 : WIRE W454C;
146 : WIRE W64B1;
147 : WIRE W64B2;
148 : WIRE W64A1;
149 : WIRE W64A2;
150 : WIRE W73C;
151 : WIRE W73C1;
152 : WIRE W32;
153 : WIRE W122C1;
154 : WIRE W122C2;
155 : WIRE W233C1;
156 : WIRE W233C2;
157 : WIRE W43A;
158 : WIRE N1J3;
159 : WIRE N1J2;
160 : SUPPLY0 GND;
161 : WIRE N1J1;
162 : WIRE FP9;
```

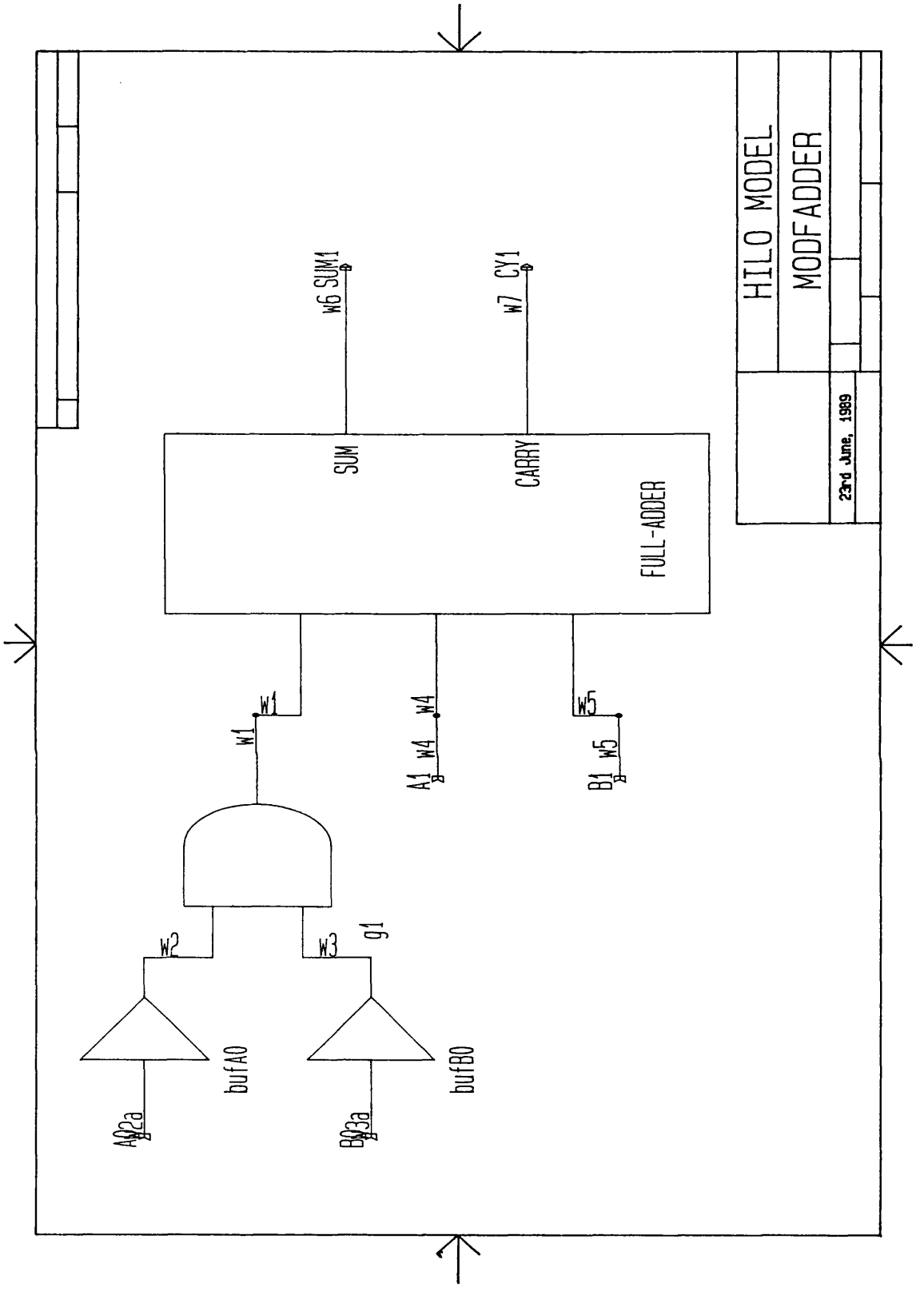
```
163 : WIRE FP8;
164 : WIRE FP7;
165 : WIRE FP6;
166 : WIRE FP5;
167 : WIRE B7;
168 : WIRE FP4;
169 : WIRE B6;
170 : WIRE FP3;
171 : WIRE A7;
172 : WIRE B5;
173 : WIRE FP2;
174 : WIRE A6;
175 : WIRE B4;
176 : WIRE FP1;
177 : WIRE A5;
178 : WIRE B3;
179 : WIRE FP0;
180 : WIRE A4;
181 : WIRE B2;
182 : WIRE A3;
183 : WIRE B1;
184 : WIRE A2;
185 : WIRE B0;
186 : WIRE A1;
187 : WIRE A0;
188 : WIRE FP16;
189 : WIRE FP15;
190 : WIRE FP14;
191 : WIRE FP13;
192 : WIRE FP12;
193 : WIRE FP11;
194 : WIRE FP10;
195 : WIRE N1J4;
196 : .

****}
*quit

HILO END OF RUN 24-JUN-1989 12:25
```

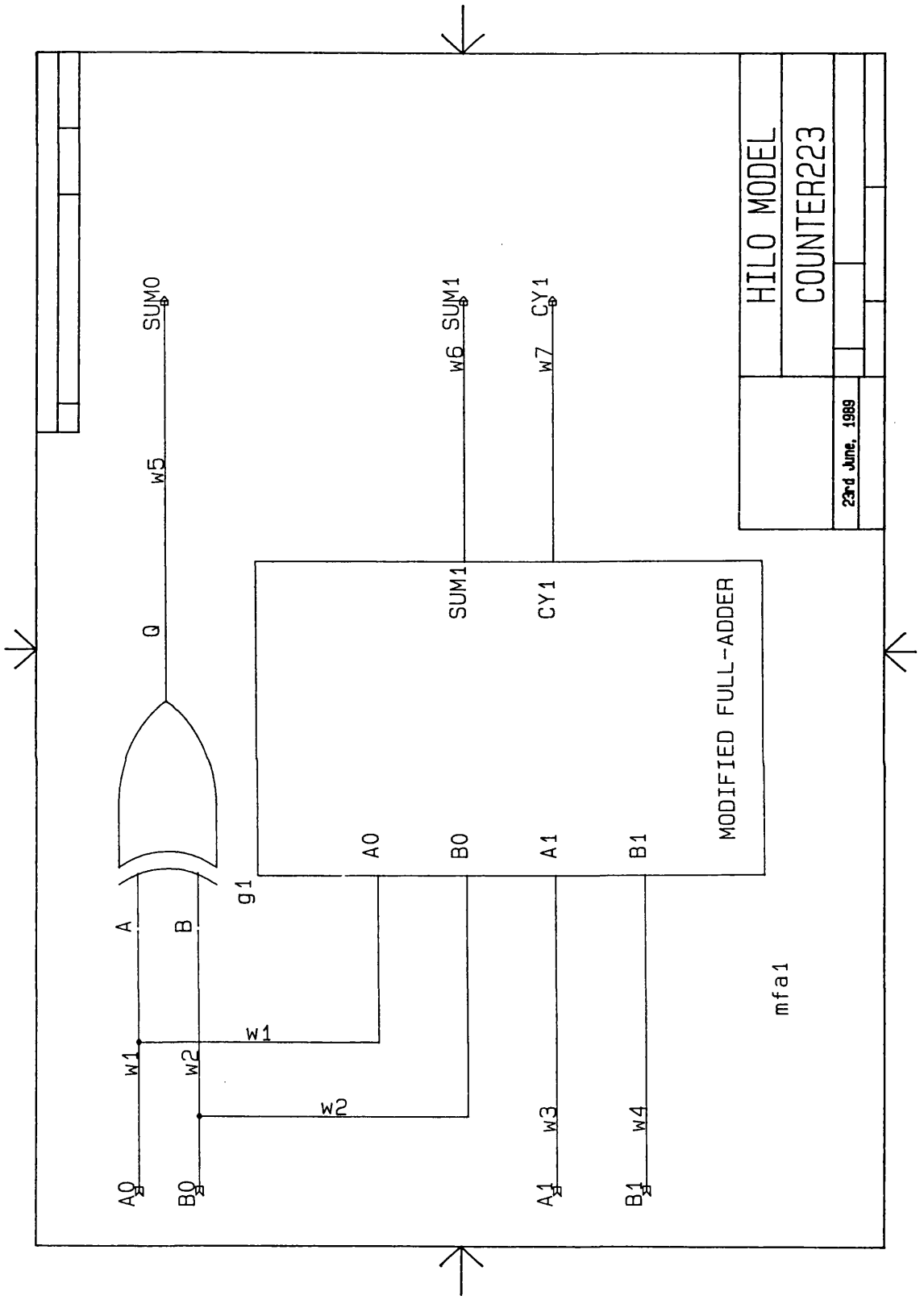

HILO MODEL
PPGEN
23rd June, 1989
W5 PP
W3
W4
g1
g3
g2
XPW1
YPW2

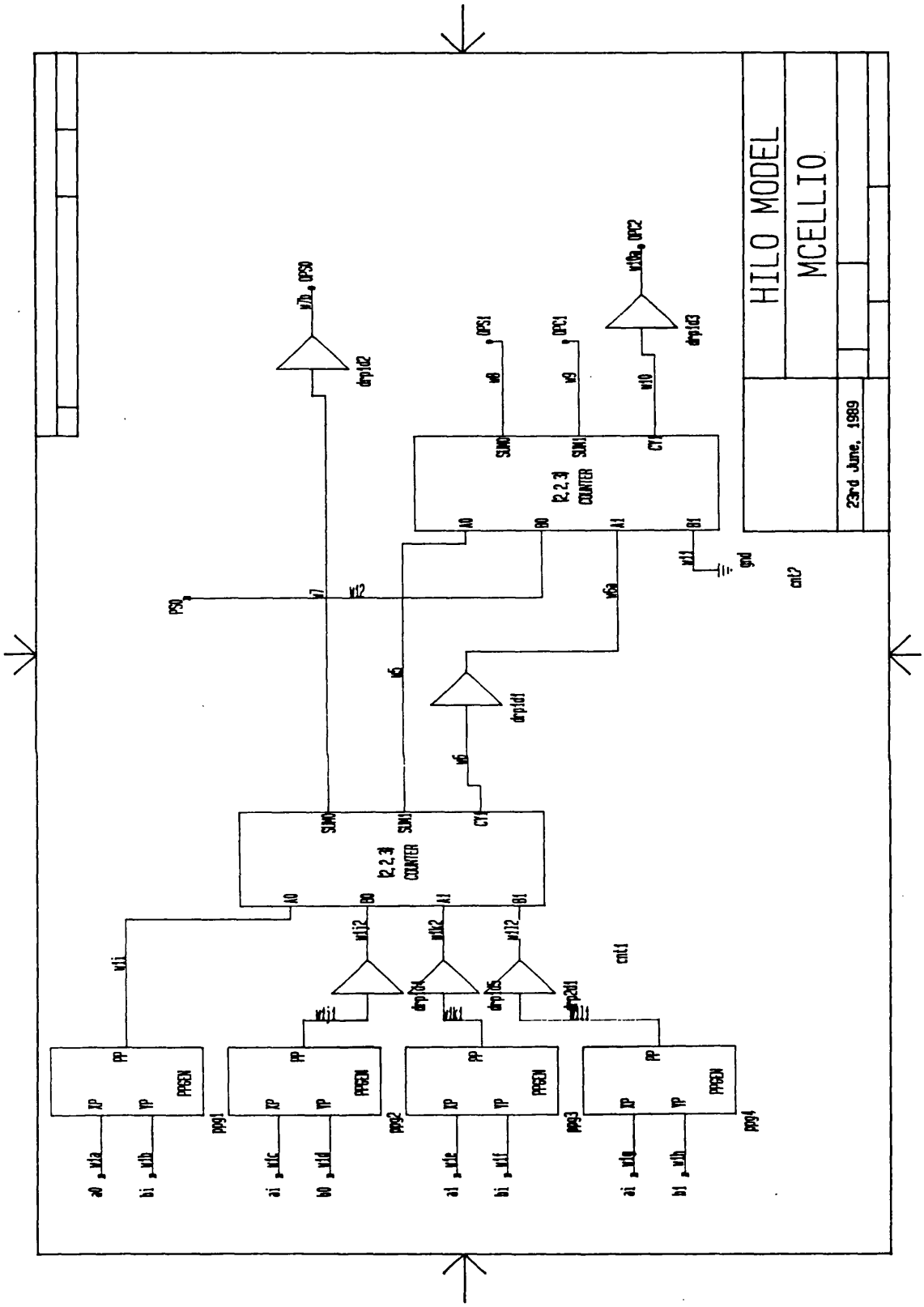

HILO MODEL
EXOR2
23rd June, 1989
w70
w5
w6
g3
g1
g2
w3
w4
g4
w1
w2
g5
w1
w1
w2
w2
bfA
bfB
Aw1a
Bw2a

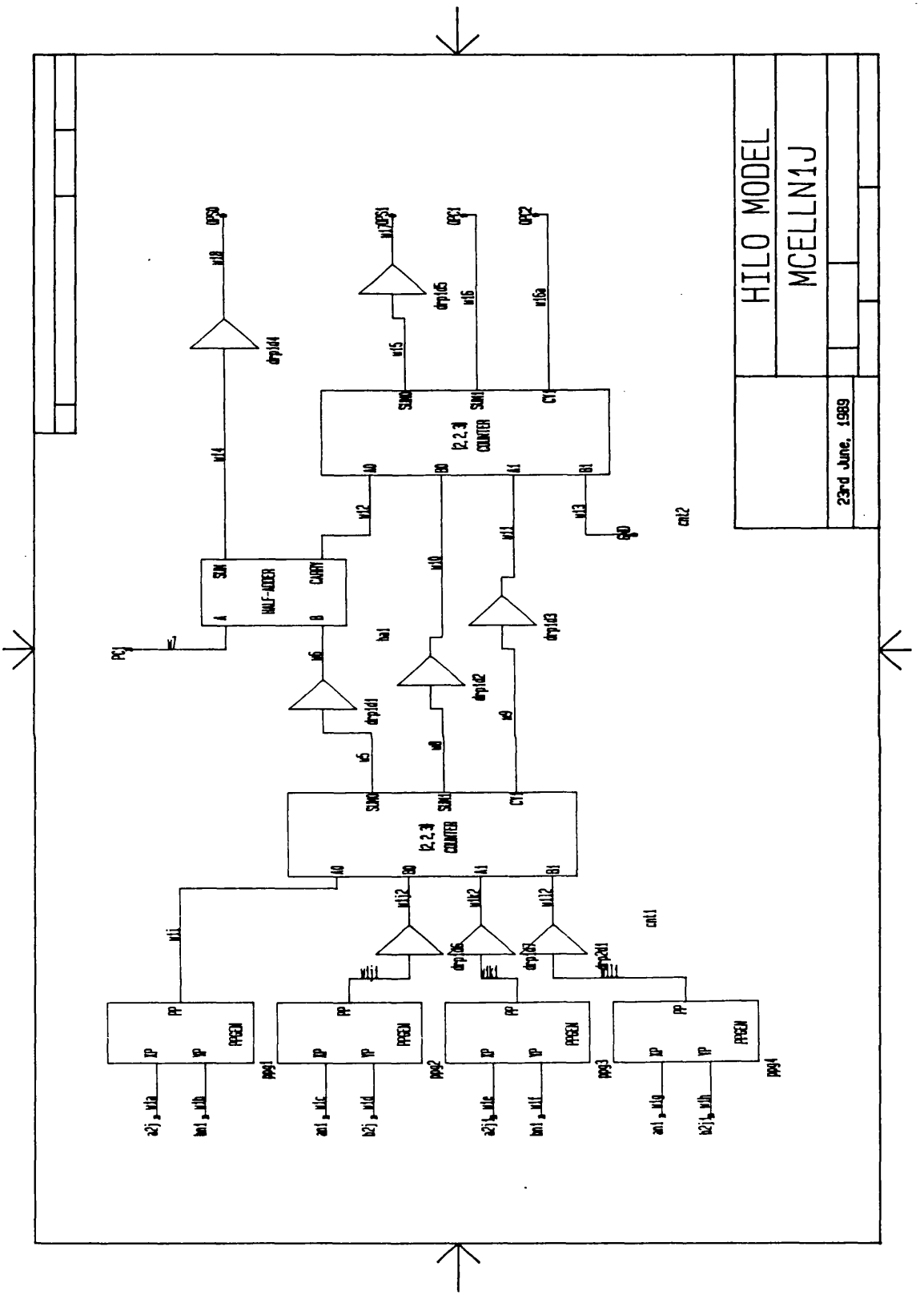

HILO MODEL
HADDER
23rd June, 1989
SUM
CARRY
w6
w5
Q
w3
w4
gt2
gt1
gt3
gt4
A
B
w1
w1
w2
w2
A
B

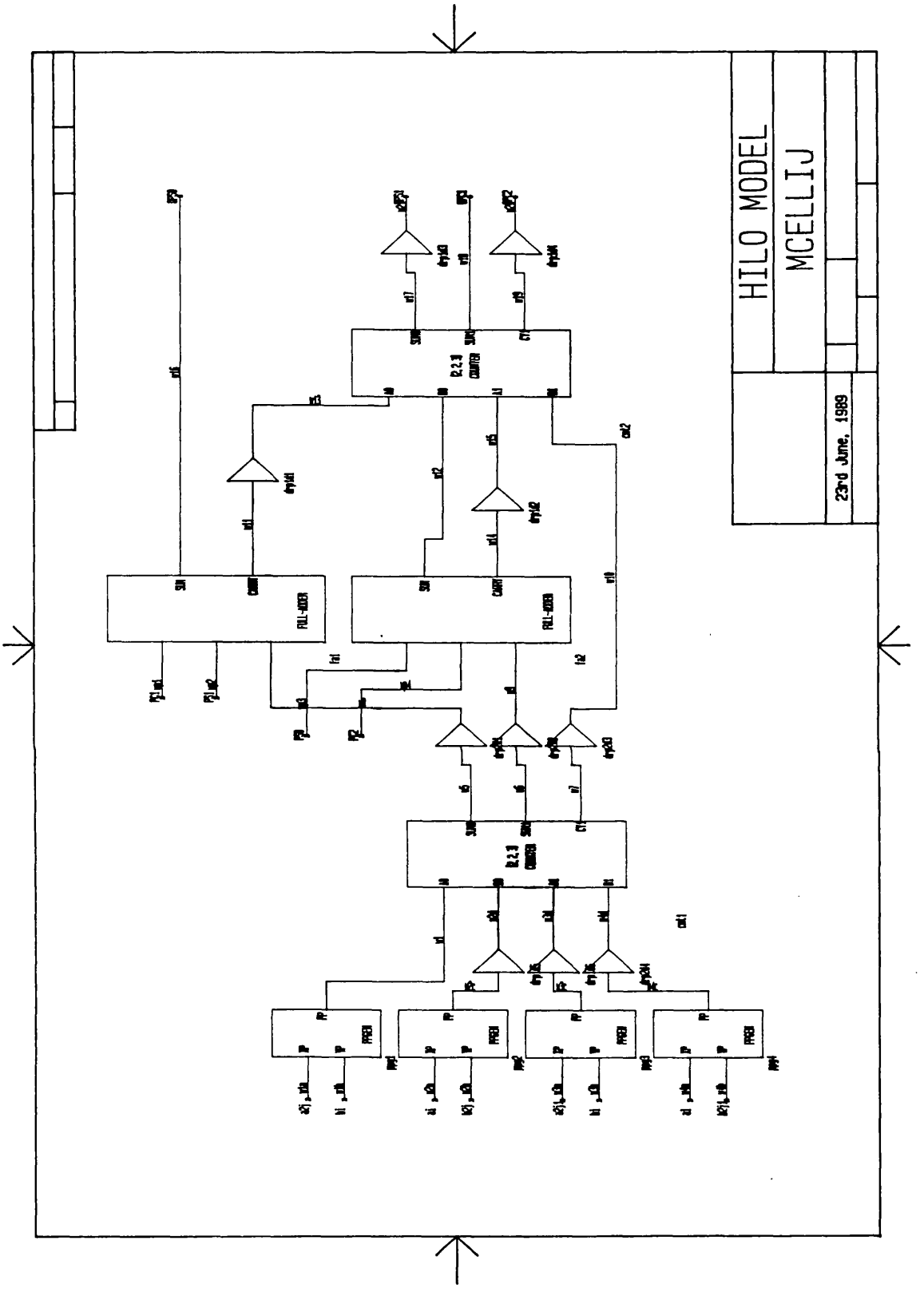

HILO MODEL
FADDER
23rd June, 1989
SUM
CARRY
g1
g2
g3
g4
g5
g6
g7
g8
g9
g10
g11
g12
g13
g14
g15
g16
g17
bfA
bfB
bfC
w1
w2
w3
w4
w5
w6
w7
w8
w9
w10
w11
w12
w13
w14
w15
w16
w17
w18
w19
w20
w21
w22

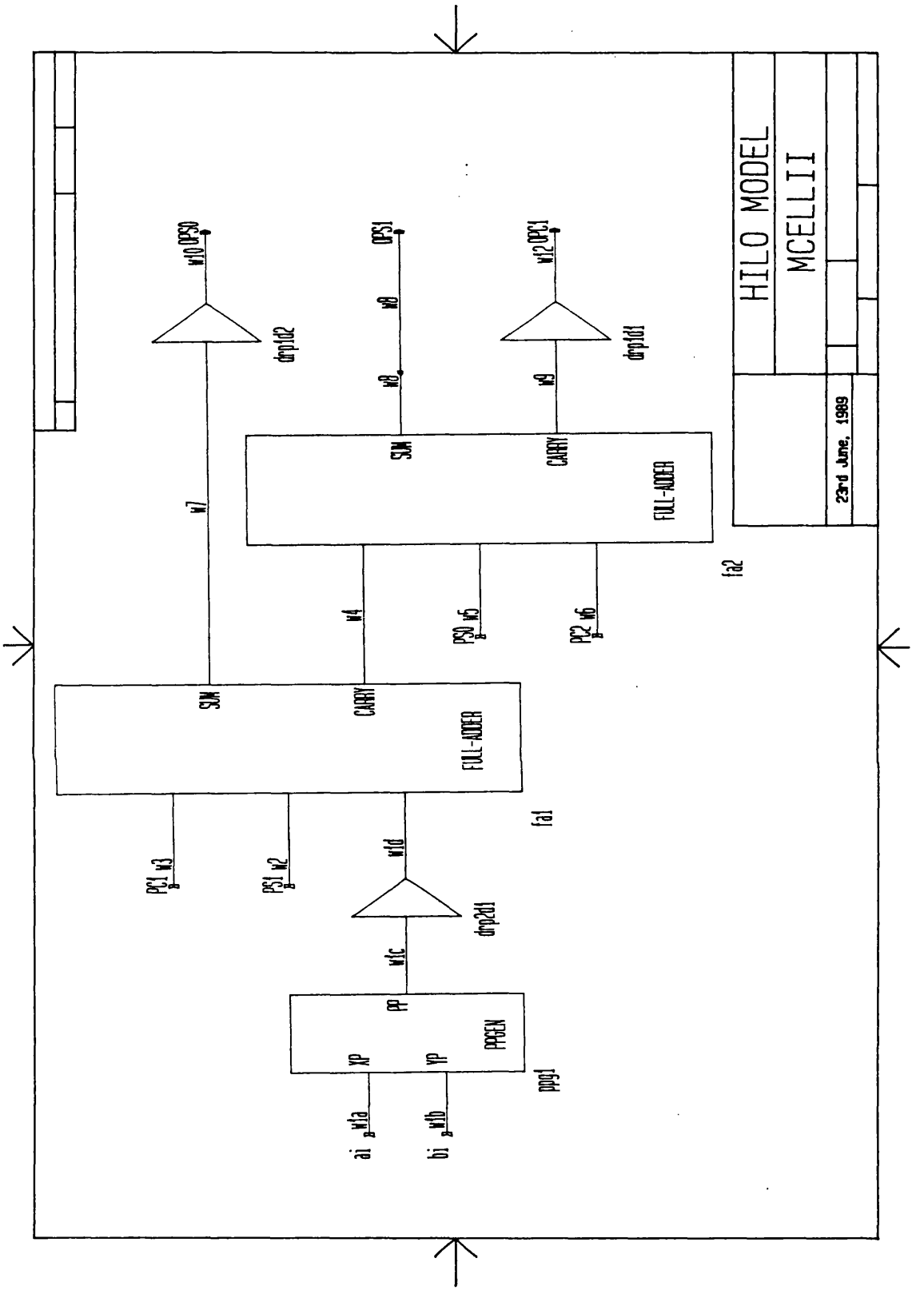

HILO MODEL
MODFADDER
23rd June, 1989
w6 SUM1
w7 CY1
SUM
CARRY
FULL-ADDER
w1
w1
w4
A1 w4
w5
B1 w5
w2
w3
g1
bufA0
bufB0
A0 2a
B0 3a

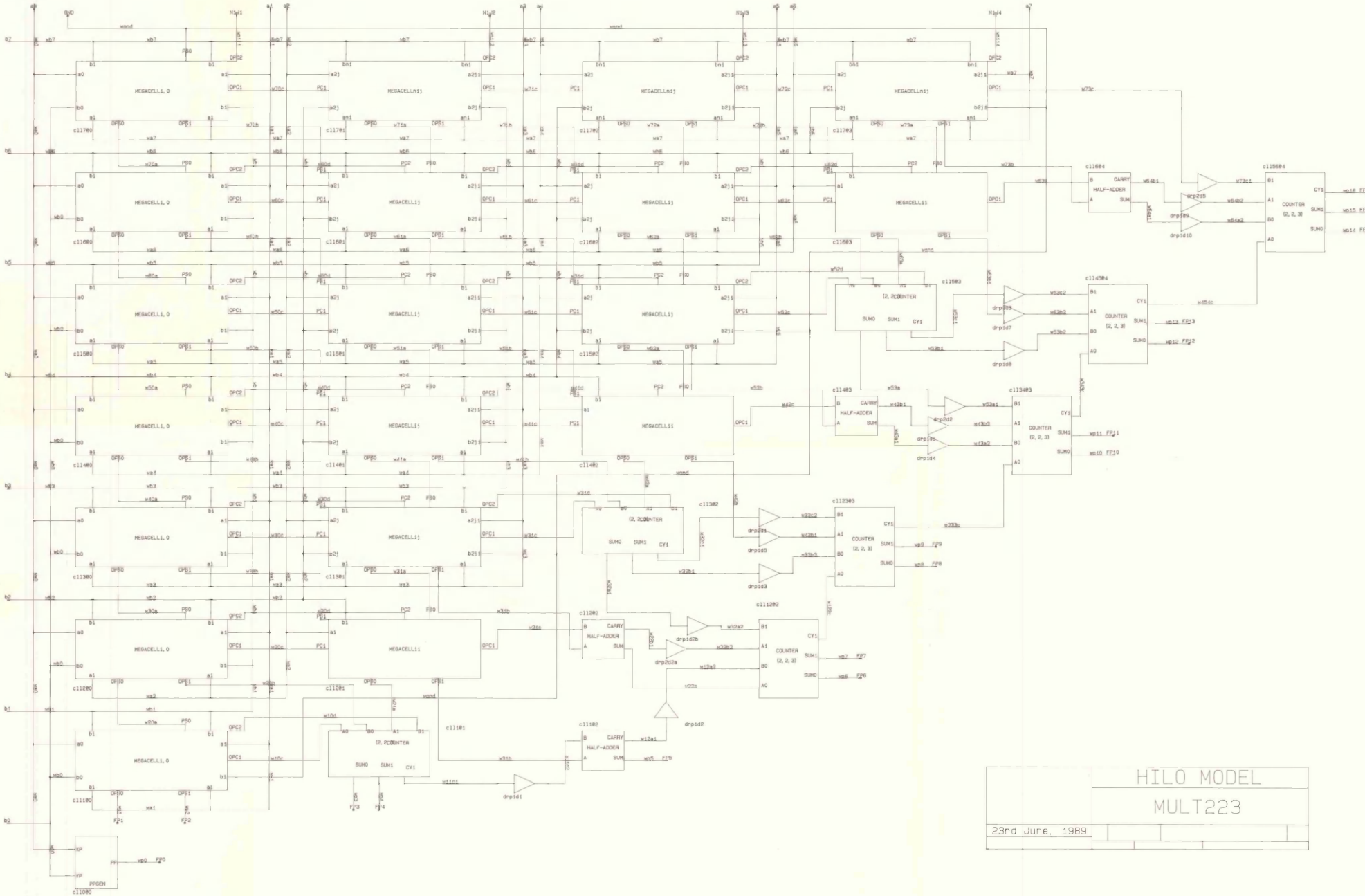

HILO MODEL
COUNTER223
23rd June, 1989
SUM0
w6 SUM1
CY1
w7
w5
Q
SUM1
CY1
MODIFIED FULL-ADDER
A0
B0
A1
B1
g1
A
B
w1
w2
w3
w4
mfa1


HILO MODEL
MCELLIO
23rd June, 1989
2,2,3 COUNTER
2,2,3 COUNTER
PARGEN
PARGEN
PARGEN
PARGEN
gnd
cnt1
cnt2


HILO MODEL
MCELLN1J
23rd June, 1989
OPS0
OPS1
OPS2
SUM0
SUM1
CY1
A0
B0
A1
B1
2,2,3 COUNTER
HALF-ADDER
SUM
CARRY
GND
cnt2
cnt1
ha1
PP
PP
PPGEN
PPGEN
PPGEN
PPGEN
ppg1
ppg2
ppg3
ppg4


HILO MODEL
MCELLIJ
23rd June, 1989


HILO MODEL
MCELLII
23rd June, 1989
w10 OPS0
OPS1
w12 OPC1
dmp1d2
w8
w8
w9
dmp1d1
SUM
CARRY
FULL-ADDER
fa2
w7
w4
PS0 w5
PC2 w6
SUM
CARRY
FULL-ADDER
fa1
PC1 w3
PS1 w2
w1d
dmp2d1
w1c
PP
XP
YP
PPGEN
ppg1
ai w1a
bi w1b


HILO MODEL
MULT223
23rd June, 1989