IEEE International Symposium on Circuits and Systems, Volume II, pp. 785-788, May 2004, Vancouver, Canada

GIGAHERTZ-RANGE MCML MULTIPLIER ARCHITECTURES

Venkat Srinivasan, Dong Sam Ha, and Jos B Sulistyo

VTVT (Virginia Tech VLSI for Telecommunications) Laboratory,
Bradley Department of Electrical and Computer Engineering,
Virginia Tech, Blacksburg, VA 24061, USA

E-mail: {vsriniva, ha, jsulisty}@vt.edu
http://www.ee.vt.edu/~ha/

ABSTRACT

In this paper, we present three digital multiplier architectures capable of operating in the gigahertz range, based on MOS Current Mode Logic (MCML) style. A small library of MCML logic gates consisting of NAND/AND, XOR/XNOR, (3x2) counter (full adder), [4:2] compressor, and master-slave flip-flop were designed and optimized for high-speed operation. Using these gates, we propose three different 8-bit MCML binary-tree multiplier architectures and compare their performance in terms of latency, throughput (number of multiplications per second) and power consumption. According to our simulation, the fastest multiplier targeting for TSMC 0.18 μm CMOS technology attains a throughput of 4.76 GHz or 4.76 Billion multiplications per second and a latency of 3.8 ns.

1. INTRODUCTION

The increasing demand for fast arithmetic units in floating point co-processors, graphic processing units and DSP chips has shaped the need for highly integrated, high-speed multipliers. Traditionally multiplier architectures fall in into one of the following two categories, viz. array multipliers and tree multipliers. The latency of array multipliers is a linear function of the word length of the multiplier, *O(n)*, whereas in the case of tree multipliers, the latency is a logarithmic function of the word length, *O[log(n)]*. Hence, tree structures require fewer numbers of stages for partial product reduction compared to array structures and are more suitable for high-speed multiplier designs. To enhance the throughput, we pipelined our multipliers by inserting a register stage after every compressor cell.

The ability to build logic gates that operate at a high speed, while dissipating relatively small power, makes MOS current mode logic (MCML) a promising technique for designing gigahertz-range arithmetic circuits. Our high-speed pipelined tree multipliers exploit several attractive features of (MCML) as described later. A small library of MCML logic gates consisting of NAND/AND, XOR/XNOR, 3x2 Counter (Full Adder), [4:2] Compressor and Flip-flop form the core components of our multipliers, and they were designed and optimized for high-speed operation. We propose three 8-bit MCML multiplier architectures, a 3-2 tree architecture with a ripple carry adder, a 4- tree architecture with a ripple carry adder, and a 4-2-tree architecture with a carry look-ahead adder.

Section 2 covers basics of MCML. In this section, we also discuss various MCML design metrics and tradeoffs involved in MCML gate design. Section 3 describes design of various MCML gates for our library, discusses optimization techniques adopted for the design and also provides simulation results. In Section 4, we present our three 8-bit MCML multiplier architectures. In section 5, we compare the performance of the proposed multiplier architectures and present simulation results. Finally, Section 6 summarizes our research.

2. MOS CURRENT MODE LOGIC (MCML)

The operation of an MCML gate may be understood with the help of a basic structure of an MCML gate, shown in Figure 1 [1]. It consists of a load resistors R_L, a differential pull-down network (PDN) with complementary sets of inputs and outputs, and a constant current source I_{CS}.

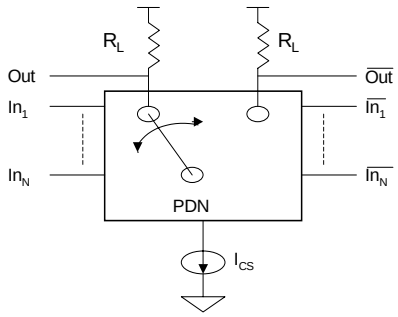

Figure 1: Basic Structure of an MCML Gate

The differential inputs (complementary sets) are applied to the pull down network (PDN). The PDN has a tree-like differential structure, similar to a Differential Cascode Voltage Switch (DCVS) family [2]. The output and its complement are available at the two arms as indicated in the figure. The PDN is grounded through a constant current source I_{CS}, which is usually an NMOS transistor. The voltage swing at the output and its complement is $\Delta V = I_{CS}R_L$ and is controlled by setting the value of the current source I_{CS} and the effective value of R_L, which is usually a PMOS transistor. The voltage swing is in the range of

a few hundred mV and is a crucial leverage factor in high-speed MCML gate design. Every MCML gate has two bias voltages, RFP and RFN. The value of RFP is set to achieve the desired load resistance. The value of the load resistance can also be controlled by the dimensions of the PMOS transistor. RFN biases the current source transistor and helps in fixing the desired current. The width of the current source transistor is usually large to make the transistor robust, to decrease the mismatch effects, and to enable a future reduction in V_{DD} [3].

The equations for the total propagation delay, power dissipation, and power delay product of an MCML logic circuit and its CMOS counterpart are shown in Table 1 [1].

Table 1: Performance of MCML and Its CMOS Counterparts

Parameter	MCML Logic Style	Conventional CMOS Logic
Propagation Delay (τ)	$\tau_{MCML} = \frac{C_L \times \Delta V \times N}{I_{SC}}$	$\tau_{CMOS} = \frac{C_L \times V_{DD} \times N}{\frac{\beta}{2} \times (V_{DD} - Vt)^2}$
Power Dissipated	$PD_{MCML} = V_{DD} \times I_{SC} \times N$	$PD_{CMOS} = N \times C_L \times V_{DD}^2 \times f$
Power Delay Product (PDP)	$PDP_{MCML} = N^2 \times C_L \times \Delta V \times V_{DD}$	$PDP_{CMOS} = N \times C_L \times V_{DD}^2$

As can be seen from Table 1, the delay of an MCML logic circuit varies linearly with voltage swing ΔV and is independent of the supply voltage V_{DD}, in contrast to conventional CMOS logic circuits. The power dissipation of an MCML logic circuit varies linearly with the supply V_{DD} and is independent of the operating frequency, whereas power dissipation of conventional CMOS circuits depends linearly on operating frequency and has a square-law dependence on supply voltage. Since the delay of an MCML gate depends linearly with ΔV and is independent of supply $V_{DD,}$ the delay can be effectively minimized by lowing the voltage swing ΔV, while maintaining the supply voltage. Further, as the power dissipation is independent of the operating frequency, MCML circuits may be operated at high speeds without increasing the power dissipation, which is in contrast to conventional CMOS circuits [1]. Important design issues for MCML logic circuits include the need for shallow logic depth and signal regeneration. For in-depth comparison between MCML and CMOS logic styles, the reader is referred to [1].

3. MCML GATE LIBRARY

Our library of MCML logic gates consists of NAND/AND, XOR/XNOR, (3x2) counter (or full adder), [4:2] compressor, and master-slave flip-flop. To be able to operate the gates in the gigahertz range, minimization of delays expressed in Table 1 was our main objective. A three-step approach was adopted for this purpose. First, the maximum current through the logic transistors per unit width was determined through simulations, which is in the order of a few hundred microamperes. After fixing the bias voltages RFP and RFN and using the current value, the dimensions of the load transistors and the current source transistor were determined through further simulations. Optimizing the PMOS load transistor is one of the most crucial and challenging tasks in an MCML gate design, involving fine tradeoffs between non-linearity and signal strengths. Whereas a high W/L ratio improves the delay by decreasing the resistance, it increases the non-linearity of the resistance leading to degradation in the output voltage swings [3].

Figure 2 shows a transistor level circuit diagram of an MCML NAND/AND gate and a D-latch. A positive edge master-slave flip-flop is designed using two such D-latches and is used for pipelining purposes for the proposed multipliers.

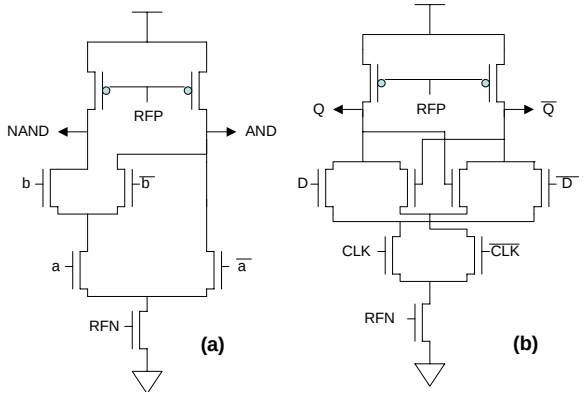

Figure 2: (a) NAND/AND Gate (b) D-Latch

An MCML full adder and a [4:2] compressor are shown in Figure 3. The full adder or the (3x2) counter consists of an XOR3 gate or a sum circuit as shown in Figure 3(a), and a majority function or the carry circuit is shown in Figure 3(b). The MCML XOR3 gate shown in Figure 3(a) is based on the DCVSL design proposed by Chu and Pulfrey in [2]. The XOR3 design reduces the number of transistors by two compared to the BDD design proposed by Musicer in [1]. Our simulation results confirmed that the former design is faster than the latter one with the added advantage of less area. The [4:2] compressor is designed using two full adders as shown in Figure 3(c). It is a special form of a (5, 3) counter with one carry entering and one leaving the compressor column [4]. The (3x2) counters and [4:2] compressors are used in our 3-2-tree and 4-2-tree MCML multipliers, respectively.

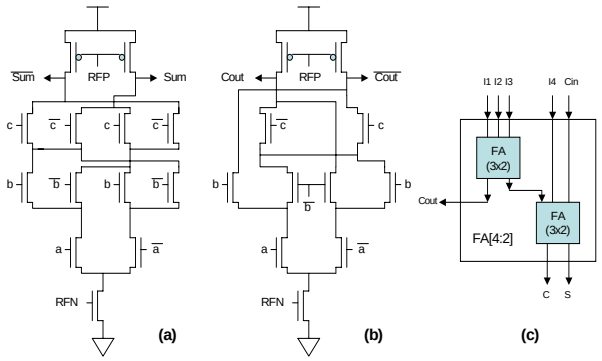

Figure 3: (a) XOR3 Gate (Sum) (b) Majority Function (Carry) (c) [4:2] Compressor

Three different delay models, intrinsic delay, FO4 delay, and actual delay, were considered. The intrinsic delay is the critical path delay of a gate without any load. The FO4 delay is the propagation delay with a fan-out of four, in which a gate drives four identical gates. The FO4 delay parameter often serves as a

benchmark for gates of different logic families and/or processing technologies. The actual delay is the critical delay of a gate when it is embedded in our proposed MCML multipliers. The actual delay determines the speed of the proposed multiplier architectures. It should be noted that the delay of the flip-flop is calculated as the sum of its setup time and the clock-to-q delays. Another important performance metric is power consumption. We estimated average power by applying all input combinations for a gate with at most three inputs and by random input patterns for a gate with larger number of inputs.

The gates were laid out with TSMC 0.18 µm CMOS as the target technology, and SPICE simulation was performed to estimate the performance. Table 2 shows the performance for our MCML library gates.

Table 2: Performance of MCML Library Gates

GATE	Intrinsic Delay (ps)	FO4 Delay (ps)	Actual Delay (ps)	Average Power (mW)	Area (μm^2)
NAND/AND	**53.9**	**104.6**	**62.9**	**0.45**	**86.7**
Master-Slave FF	**49**	**74**	**55**	**0.26**	**206.3**
XOR3 (Sum of FA)	**57**	**113**	**66.4**	**0.45**	**147.1**
Maj Function (Carry of FA)	**76**	**138**	**88**	**0.45**	**147.1**
[4:2] Compressor	**148**	**238**	**159**	**1.49**	**558.1**

4. PROPOSED MULTIPLIER ARCHITECTURES

A tree structure is a good choice for high-speed multipliers and has been employed in the proposed MCML multiplier architectures. This is due to the logarithmic reduction of partial products in tree multipliers in contrast to a linear reduction in array multipliers [4]. The latency for tree multipliers is a logarithmic function of the word length *O[log(n)]*, whereas the latency for array multipliers is a linear function *O(n)*.

The performance of a multiplier is usually measured in terms of its latency and throughput. The throughput of a multiplier is an important metric for applications such as high-speed digital signal processing and other computation intensive circuits, whereas latency is more significant for applications such as advanced microprocessor architectures. The block diagram of the architectures of the three proposed 8-bit MCML multipliers, viz. 3-2-tree architecture with a ripple carry adder, 4-2-tree architecture with a ripple carry adder and 4-2-tree architecture with a carry look-ahead adder are shown in Figure 4. In order to avoid clutter, the complementary signals are not shown in Figure 4.

4.1. 3-2-Tree MCML Multiplier with Ripple Carry Adder

The main components of the 8-bit 3-2 tree MCML multiplier shown in Figure 4(a) are: a 64-bit partial product generator, the 14 3-2-tree slices, the deskew registers, and a 13-bit ripple carry adder. The partial product generator consists of 64 MCML AND/NAND gates. There are seven full adders in one 3-2 tree slice that are pipelined using flip-flops. Each tree slice reduces eight partial products to two outputs (sum and carry) in four clock cycles as shown in the block diagram. In general, a 3-2 tree design requires $\log_{1.5}[N/2]$ stages to reduce N partial products in to sum and carry outputs. In contrast, a 4-2 tree design needs $\log_2[N/2]$ stages for the same reduction. 14 such tree slices are used for the reduction of 64 partial products into sum and carry outputs that arrive at the same time. A ripple carry adder is used for parallel addition of these sum and carry bits. It is pipelined by inserting a flip-flop between *every full adder*. Since all inputs arrive at the same time, they should be delayed successively and applied to the adder. This delay balancing at the inputs and outputs of the ripple carry adder is achieved with the help of deskewing registers. Deskewing registers are columns of flip-flops used to insert appropriate delays at the input and the output. This ensures that all the outputs of the ripple carry adder appear at the same clock cycle. Thus, the total latency (number of delay stages) in the multiplier is 18: one clock cycle is attributed to the partial product generator, four clock cycles are attributed to the 3-2-tree slice and 13 attributed to the ripple carry adder.

4.2. 4-2-Tree MCML Multiplier with Ripple Carry Adder

The 4-2-tree MCML multiplier architecture, shown in Figure 4(b), is designed in a similar manner to the 3-2-tree architecture except for the use of [4:2] compressors. 13 4-2 tree slices are used for the reduction of 64 partial products. Unlike the previous architecture, a 14-bit ripple carry adder (RCA), which is used for

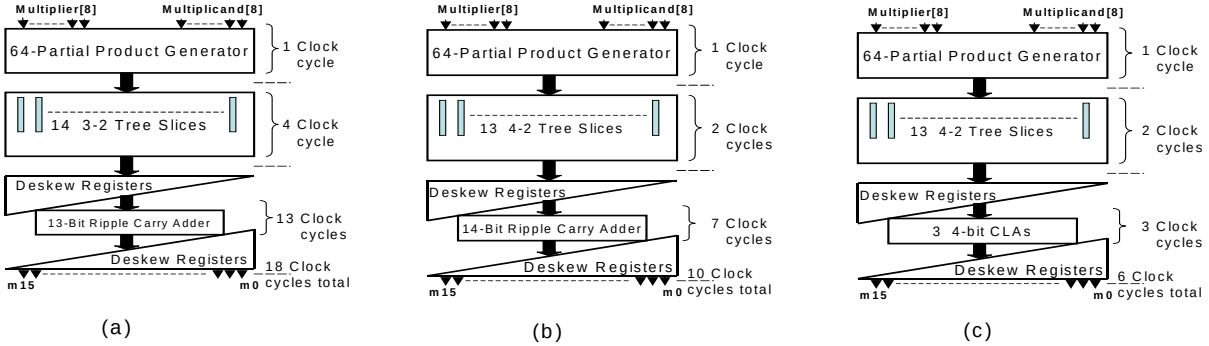

Figure 4: (a) 3-2-Tree multiplier with an RCA (b) 4-2-Tree multiplier with an RCA (c) 4-2-Tree multiplier with a CLA

the parallel addition of the sum and carry bits, is pipelined by inserting a flip-flop between *every two full adders*. The total latency or the number of delay stages for the multiplier is ten.

4.3. 4-2-Tree Multiplier with Carry Look-ahead Adder

The architecture of the 4-2-tree MCML multiplier using a carry look-ahead adder (CLA), shown in Figure 4(c), is the same to the above multiplier except the type of the adder used. In this design, a CLA, consisting of three 4-bit CLAs, is pipelined by inserting one flip-flop between *every 4-bit CLA*, and it is used for the parallel addition of the sum and carry outputs from the 13 4-2-tree slices. The total latency or the number of delay stages in the multiplier is six.

5. PERFORMANCE OF MCML MULTIPLIERS

The performance of the three 8-bit MCML multipliers proposed in this paper is summarized in Table 3.

Table 3: Performance of the Three Proposed MCML Multipliers

Multiplier	Clock (ns)	Throughput (GHz)	Latency (Clock cycle)	Latency (Absolute Delay)	Power (mW)	Area (mm2)
[I] 3-2 Tree with RCA	0.21	4.76	18	3.78	261.20	0.15
[II] 4-2 Tree with RCA	0.30	3.33	10	3.00	124.47	0.06
[III] 4-2 Tree with CLA	0.50	2.00	6	3.00	106.20	0.04

The 3-2 tree MCML multiplier with a RCA, called Architecture I, achieves the highest throughput among the three, but it incurs the largest latency and the area. It operates with a throughput of 4.76 GHz and dissipates 261 mW of power. The high throughput of Architecture I is due to the high degree of pipelining achieved, while the high area is due to the large number of flip-flops used for delay balancing. The high-degree of pipelining is also the cause for the high latency clock cycle count in Architecture I. On the other hand, the 4-2-tree MCML multiplier with a CLA called Architecture III has the least latency clock cycle count because of the least degree of pipelining and also has the smallest area among the three architectures. However, it results in the lowest throughput of 2 GHz. It is interesting to note that a 4-2 tree multiplier with a RCA (Architecture I) is faster than its CLA version (Architecture III), but it consumes more power and less area-efficient. This is because of the shorter critical path of the pipelined RCA. The performance of the 4-2 tree MCML multiplier with an RCA, called Architecture II, lies between the two architectures, Architecture I and Architecture III.

A fair comparison of various multiplier architectures with the proposed designs is difficult. Multipliers are often designed with different technologies and performance goals. We selected four fastest multipliers designs available in open literature for comparison with the proposed architectures. Table 4 shows the throughputs of four multiplier architectures. It should be noted that all the four designs adopted CMOS technology. The fastest multiplier among the four designs is the one proposed by Intel [7], which achieves the throughput of 1.5 GHz. It can be observed that all proposed architectures achieve higher throughputs than the designs found in contemporary literature.

Table 4: Comparison of Multiplier Architectures

Designs	Year	FO4 Delay (ps)	Feature Size (μm)	Word Length	Throughput (GHz)
[5]	2000	216	0.6	8	0.625
[6]	2001	126	0.35	32	0.444
[7]	2002	41.4	0.13	54	1.538
[8]	2002	41.4	0.13	59	1.000

6. SUMMARY

In this paper, we propose three gigahertz-range multiplier architectures using MOS Current Mode Logic (MCML) style, involving tradeoffs among throughput, latency, power dissipation and area. The 3-2 tree MCML multiplier with an RCA operates with a maximum throughput of 4.76 GHz (4.76 Billion multiplications per second) and a latency of 3.78 ns. The 4-2-tree architecture with a RCA operates with a throughput of 3.3 GHz and a latency of 3 ns and the 4-2-tree architecture with a CLA operates with a throughput of 2 GHz and a latency of 3 ns. So a designer may choose an appropriate architecture considering his/her need.

7. REFERENCES

[1] J. Musicer, "An Analysis of MOS Current Mode Logic for Low Power and High Performance Digital Logic" A research project at the University of California, Berkeley, 2000.

[2] K.M. Chu and D.I. Pulfrey, "Design and Procedures for Differential Cascode Voltage Switch Circuits", IEEE Journal of Solid-State Circuits, vol. Sc-21, No. 6, December 1986.

[3] J.Musicer and J.Rabaey, "MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments", Proc., ISLPED 2000, pp. 102-107, 2000.

[4] M R Santoro, "Design and Clocking of VLSI Multipliers", Technical Report No. CSL-TR-89-397, October 1989, Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305.

[5] J S. Wang, P H Yang, and D Sheng, "Design of a 3-V 300-MHz Low-Power 8-b×8-b Pipelined Multiplier Using Pulse-Triggered TSPC Flip-Flops", IEEE Journal of Solid-State Circuits, vol. 35 issue 4, Apr 2000, pp. 583 –592.

[6] M Olivieri, "Design of Synchronous and Asynchronous Variable-Latency Pipelined Multipliers", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9 issue 2, Apr 2001, pp. 365 –376.

[7] Intel Corp, *Intel Pentium 4 Optimization Guide*, Intel Corp, Santa Clara, CA, 2002.

[8] R Rogenmoser, L O'Donnell, and S Nishimoto, "A Dual-issue Floating-Point Coprocessor with SIMD Architecture and Fast 3D Functions", Digest of Technical Papers, 2002 IEEE International Solid-State Circuits Conference, 2002, vol. 1, pp. 414 –415.