

MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments

Jason Musicer and Jan Rabaey

Berkeley Wireless Research Center, Dept. of EECS, University of California at Berkeley

Abstract-- In this work, MOS Current Mode Logic (MCML) is analyzed for application to low power, mixed signal environments. A small MCML cell library is developed and optimized for several different performance requirements. The cells are then applied to the generation of pipelined CORDIC structures and compared with equivalent CMOS circuits. MCML CORDICs are designed which can operate from 125MHz to 310MHz with power consumption varying between 4.3mW and 18.6mW. These power results are up to 1.5 times less than CMOS CORDICs with equivalent propagation delays. Design was done in a 0.25 μ m standard CMOS process from ST Microelectronics.

I. INTRODUCTION

The recent development in VLSI technology has allowed rapid growth in the area of portable electronic devices. One of the limiting factors in the deployment of these devices is the battery life and power consumption of the circuitry. It is critical in future circuits that power be minimized beyond merely the constraints of packaging and heat dissipation. As device density increases, it is also extremely desirable to integrate analog and digital circuitry onto the same die. This integration has been delayed due primarily to the difficulty in designed high precision analog circuitry in the presence of digital noise.

A circuit style that seems to be promising in both reducing power consumption and providing an analog friendly environment is MOS Current Mode Logic (MCML). While bipolar CML, a derivative of ECL logic, has been used for years in high performance applications, it has become less desirable due to its high static power consumption and reliance on bipolar processing. In [1], MCML was analyzed and a 64-bit adaptively pipelined adder was developed and simulated. It was demonstrated in that paper that MCML could dissipate less power than equivalent CMOS circuitry as well as adjust for clock skew and environmental or process variations.

The uniqueness of this project is that MCML is analyzed using minimum or near-minimum sized transistors instead of the significantly larger designs in [1] and gives a much broader comparison with CMOS logic. It will be shown that area efficient MCML can actually consume significantly less power than

equivalent CMOS circuitry while maintaining many of the other benefits of traditional CML such as reduction in dI/dt effects, common mode noise immunity, and process and voltage variation immunity.

This paper begins with a discussion of MCML gate design. Examples of several gates will be given and design and simulation methodologies will be discussed. The second part of the paper discusses some system level issues of MCML design including peripheral control circuitry and effects of logic depth. The next section describes the CORDIC algorithm and implementation and gives the system level results. Finally, some conclusions are drawn and future work is presented.

II. MOTIVATION AND THEORY

The basic MCML gate structure is shown below in Fig.1. MCML gates are fully differential and steer current between the two pull up resistances. The total voltage swing, $\Delta V = I \times R$, is set by adjusting the resistance of the pull-up devices for a given current. It is important to note that the voltage swing is generally not rail to rail but in fact much less, of the order of several hundred millivolts.

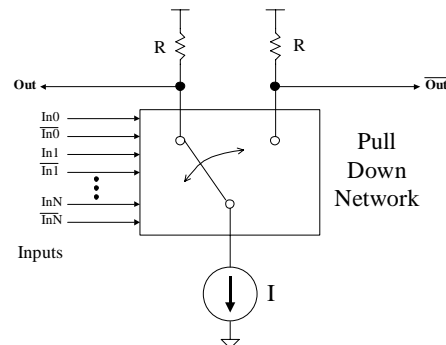


Figure 1 : Basic MCML Gate

With this simple model in mind, we can derive some basic properties for a circuit composed of MCML gates. For simplicity, let's assume that our circuit is a linear chain of N identical gates, all with load capacitance C . The total propagation delay will be proportional to:

$$D_{MCML} = NRC = \frac{N \times C \times \Delta V}{I}$$

where N is the total logic depth of the circuit.

While static CMOS gates tend to dissipate static and dynamic power, the current draw of MCML gates is constant over time and independent of switching activity. With this assumption, we can write expressions for power, power-delay, and energy-delay:

$$P_{MCML} = N \times I \times V_{dd}$$

$$PD_{MCML} = NIV_{dd} \times \frac{NC\Delta V}{I} = N^2 \times C \times \Delta V \times V_{dd}$$

$$ED_{MCML} = N^2 C \Delta V V_{dd} \times \frac{NC\Delta V}{I} = \frac{N^3 \times C^2 \times V_{dd} \times \Delta V^2}{I}$$

The delay, power, power-delay, and energy-delay for static CMOS logic are well known and approximated by [4]:

$$D_{CMOS} = \frac{N \times C \times V_{dd}}{\frac{k}{2} \times (V_{dd} - V_t)^\alpha}$$

$$P_{CMOS} = N \times C \times V_{dd}^2 \times \frac{1}{D_{CMOS}}$$

$$PD_{CMOS} = N \times C \times V_{dd}^2$$

$$ED_{CMOS} = N^2 \times 2 \times \frac{C^2}{k} \times \frac{V_{dd}^2}{(V_{dd} - V_t)^\alpha}$$

where k and α are process and transistor size dependent parameters.

One interesting property to note is that MCML circuits do not have a theoretical minimum to the energy-delay product whereas the CMOS circuits do [1]. A designer can arbitrarily reduce the ED product by increasing the current for a given C , V_{dd} , and ΔV . In reality, this is not possible for very large currents because the robustness of the circuitry will deteriorate if no other changes are made.

Possibly the most important conclusion from the above equations comes from the effect of logic depth, N . The performance of MCML gates *in relation* to CMOS decreases linearly with N . This is due to the fact that MCML consumes static power, even when not switching. It is very important therefore in MCML circuits to maintain a shallow logic depth. In slowly clocked circuits, CMOS will not consume as much power as MCML, but in circuits with high performance requirements, MCML can have significantly better power-delay or energy-delay.

Another interesting property is that the energy-delay is proportional to the square of the voltage swing. This fact encourages the use very low swing circuits. Once again, the limiting factor is the robustness of the circuitry.

For mixed signal environments, the constant current supplied by V_{dd} is extremely desirable. The di/dt effects are negligible in comparison to CMOS circuits and the current variation is theoretically 0. There will be some current change during switching due to non-idealities, but the change is less than 5% in circuits simulated. The circuits are also significantly more

robust against power supply noise due to their inherent common mode rejection.

III. CIRCUIT DESIGN

In order to give a fair analysis of the benefits of MCML in comparison to CMOS circuits, we first had to create a small yet diverse set of basic gates and test blocks for our experiments. We chose to compare the following basic circuit blocks: Inverter, NAND2, NOR2, MUX2, XOR2, XOR3, MUX4, Full Adder, latches and flip-flops. Examples of several MCML gates are given below in Fig. 2.

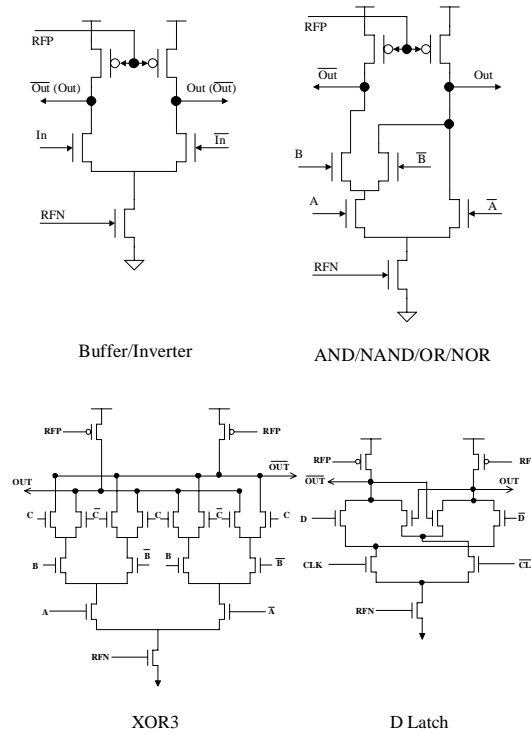


Figure 2 : MCML Gate Examples

Each MCML circuit has two control voltages, RFN and RFP. RFN is used to set the gate voltage of the NMOS current source and determines the current value. In general, the NMOS device of the current source has larger than minimum length. This is to provide higher output impedance for the current source and to reduce the effects of transistor length mismatch between the biasing and logic circuits. RFP determines the equivalent resistance of the pmos load devices. More will be said about how these control voltages are generated in the next section.

The CMOS versions of each block were optimized for low power. Traditional sizing rules were used in which the pmos devices were made twice as wide and all series transistors were made wider to achieve the same first order delays. Several gates were taken from the ST

Microelectronics standard cell library and dynamic C²MOS latches were used [2].

MCML circuits have much greater flexibility in design optimization than CMOS circuits. While CMOS circuits can be optimized by changing device sizes and VDD, MCML can be optimized by adjusting the voltage swing, current, VDD, and transistor sizes. Our optimization strategy went as follows: For several different currents ranging from 100nA to 100uA per gate, find the optimal device sizes, voltage swings and VDD's for minimum energy-delay product.

The limits of the optimization were set by a few measures of robustness. CMOS circuits tend to be characterized by their gain and noise margins. Since MCML circuits will produce much less noise than standard CMOS logic, we felt that noise margin and gain restrictions should be less severe. We set a lower limit on gain at 1.4 in order to achieve bistability in our latches and flip-flops and to create regenerative circuits. The lower bound on voltage swing was set at 300 mV to ensure signal integrity in the presence of thermal noise and device mismatch. The limit on lowering VDD was that the tail current source must stay in the saturation region.

With these limits on robustness, device sizes and voltage swings were chosen for a variety of currents. Transistors were chosen to be minimum sized whenever possible and voltage swings were kept as low as allowed for complete current switching. Beyond keeping the minimum robustness metrics, the circuits were optimized for energy-delay.

IV. SYSTEM LEVEL DESIGN

Now that the set of gates was designed and optimized, some system level decisions needed to be made. The first issues to be dealt with concern the generation of the two control voltages, RFP and RFN.

The RFP voltage determines the DC resistance of the PMOS loads. Since we would like the voltage swing to remain relatively constant for varying currents and process parameters, we use a feedback circuit similar to that used in [1]. A general Variable Swing Controller (VSC) is shown below in Fig.3:

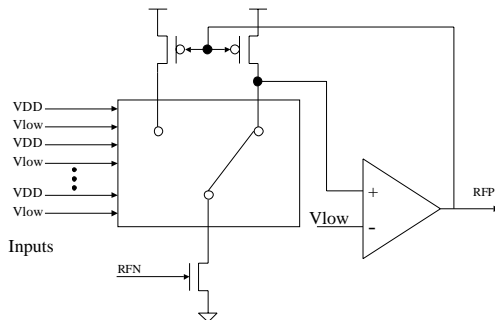


Figure 3 : Variable Swing Controller (VSC)

The VSC allows for a fixed voltage swing across a variety of currents and also provides an easy mechanism for trading off speed for power. As the current in the VSC changes (set by RFN), the opamp forces the value of the low output voltage to be Vlow by changing the gate voltage, and hence the resistance, of the pmos loads. The Vlow voltage can be generated by a resistor network or can come from off-chip. The RFP voltage is then broadcast to the rest of the MCML components on the chip.

The pull down network of the VSC should in general match the gate which the VSC is trying to model, but VSCs can be shared across different gates. The main difficulty in using a single VSC to set the RFP voltage for different types of gates is that the voltage swing of the gates will not track Vlow exactly. The number of VSCs used in a circuit can range from one to several, depending on the variety of gates used, the control needed over the voltage swing, and the amount of overhead tolerated in power and area. In general, if the voltage swing used is small and the overall block of logic is large, it is beneficial to have the fine precision from using multiple VSC's. If the voltage swing is larger than minimal or there is not much variety in topology of the blocks used, then a single VSC can be used.

The RFN signal determines the amount of current flowing in the current source and therefore determines the speed and power of the circuit. The simplest way to set this reference voltage is to use a current mirror. Alternatively, an adaptive pipelining system can be used [1] shown in figure 4:

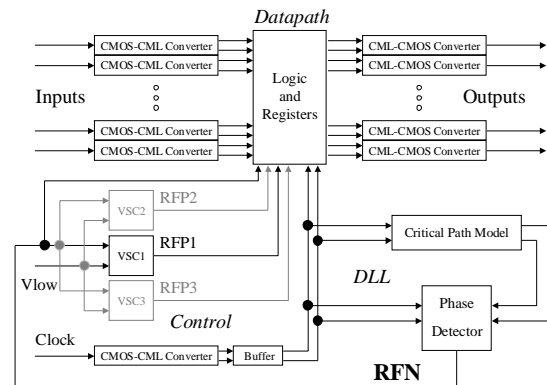


Figure 4 : Adaptively pipelined MCML system

The basic principle behind an adaptive pipeline is to use a Delay Locked Loop (DLL) to measure the delay through a model of the critical path of the circuit. If the critical path delay is greater than the required clock period, then the DLL increases the RFN voltage and thereby increases the current, speed and power of the circuit. If the delay is less than the required clock period, then RFN is decreased and less current is used. Single or multiple VSC's can be used to maintain a fixed voltage swing as the current varies. Multiple DLL's

could also be used if there are requirements for multiple RFN voltages to be generated.

The goal of the adaptive pipelining is to make the circuit timing insensitive to process, temperature, and voltage variations. For example, if a chip comes back from fabrication and happens to be near the slow process corner, the adaptively pipelined circuit will meet the same timing requirements as the chip near the fast process corner. The difference between the chips will be in the power consumption and not the timing.

In a standard CMOS design methodology, designers must always design for the worst case. This leads to using VDDs which are higher than required for the nominal case and therefore increases power consumption for all designs. With adaptive pipelining, designers can design for the nominal case for delay and instead, the power will vary. If multiple chips are used on a board, the average power consumption of all the chips should approach the nominal value. This technique can also improve the yield of circuits and allow for late changes in system clock frequency.

V. CORDIC ALGORITHM

In order to test many of the optimizations and analysis developed earlier, we felt it was necessary to design a complex block of logic using MCML. The target block of logic chosen was a pipelined CORDIC. The basic CORDIC algorithm is used for iteratively computing angles of vectors and for rotating vectors [5], [6]. While many different architectures have been proposed to increase the performance of CORDIC computation, the approach here was to compare CMOS and MCML implementations using standard design techniques.

In our implementation of the CORDIC, we decided to pipeline every iteration, both rotating and necessary scaling operations. For 8 bits of precision, there were a total of 14 pipeline stages, 8 for rotation and 6 for scaling. Another important feature to note is that additional bits are required in order to maintain precision during rotation. The total bit width of the stages is 12 bits for an 8 bit input and output.

A basic rotation stage of the CORDIC is shown in Fig. 5. The critical path is dominated by the delay of the ripple adder block. For large bit widths, it becomes highly beneficial to use carry-bypass or lookahead adders. For our implementation of the CORDIC, the adder is only 12 bits wide and the non-ripple topologies will not speed up the circuit very much. To reduce complexity with little loss in performance, simple ripple adders were used. In order to better understand the performance of the CORDIC design, we take a minute here to discuss the design methodology for ripple adders.

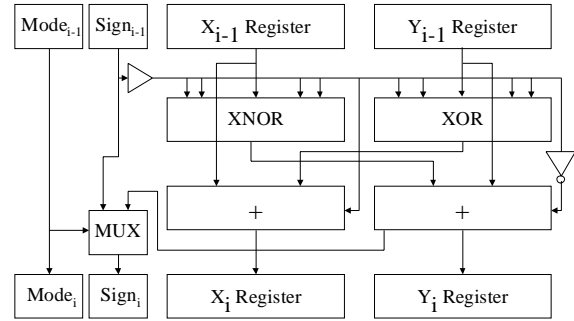


Figure 5 : CORDIC Pipeline Stage

VI. RIPPLE ADDER DESIGN

The logic equations for a full adder are well known to be an XOR3 for the sum and a 3 input majority vote for the carry. In nearly all CMOS ripple adders, an optimization is made which computes the propagate and generate signals in order to speed up the carry path. There is a similar possibility for MCML full adders and 2 implementations can be imagined. The first full adder uses two, three input gates, one for the sum and one for the carry. The second full adder used four smaller gates, one for each the generate, propagate, sum and carry.

While the second full adder will have a lower carry delay, it is not clear whether this improvement will compensate for the additional current required due to there now being 4 gates instead of 2. In fact, for small adders (< 16 bits), it is more efficient to use the first full adder implementation and this is the circuit used for the CORDIC:

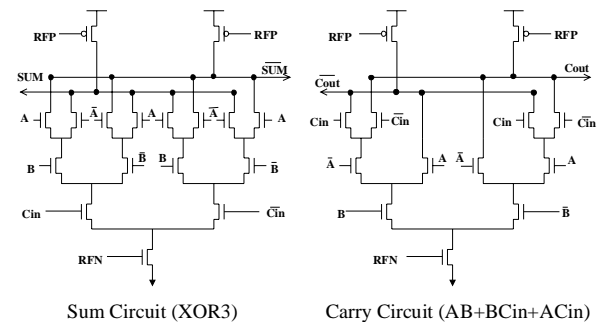


Figure 6 : MCML Full Adder

One interesting optimization which can be made is to have the sum and carry circuits of the full adder use different amounts of current. This requires 2 different VSCs, but as we have previously discussed, this may be an acceptable amount of overhead. If the sum and carry paths are allowed to have different currents, we can come up with a first order optimization for the best energy-delay product.

Let I_c be the current in the carry gate and I_s be the current in the sum gate. Also let N be the number of bits of the adder. If we assume a linear relationship between the current and delay, we can easily write that:

$$T_{p_{tot}} = tp_{abtoc} + (N-1) \times tp_{ctoc} + tp_{ctos}$$

$$T_{p_{tot}} = \frac{k_1}{I_c} + (N-1) \times \frac{k_2}{I_c} + \frac{k_3}{I_s}$$

For simplicity, assume $k_1 = k_2 = k_3 = k$. Also, let $I_c = r \times I_s$. Then,

$$T_{p_{tot}} = k \times \left(\frac{N+r}{I_c} \right)$$

We can also write the expressions for power, power-delay, and energy-delay:

$$P = \frac{(r+1)}{r} \times I_c \times VDD$$

$$PD = k \times VDD \times \frac{(r+1) \times (N+r)}{r}$$

$$ED = k^2 \times VDD \times \frac{(r+1) \times (N+r)^2}{r \times I_c}$$

If we plot the energy-delay as a function of r and normalize, we get a function which looks like:

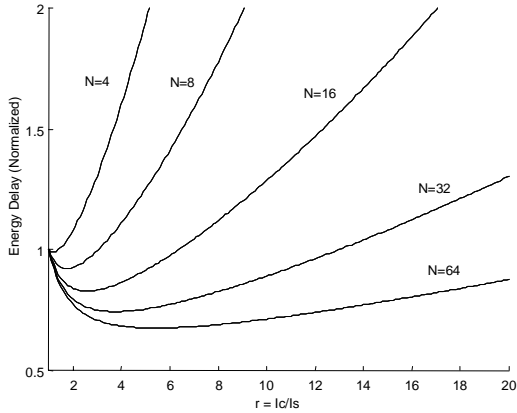


Figure 7 : Effects of Current Scaling on MCML Adders

Evident from the graph is that the potential benefits of current scaling are significant for large adders. For $N=4$, the optimal scaling for energy-delay is actually to use equal current in the sum and carry circuits. In the case of $N=64$, the energy delay for equal current is about 50% greater than with optimal scaling with $r = 5$. An estimate for the optimal point is at $r = \log_2(N)-1$.

These numbers are not completely accurate due to some of our previous assumptions about k , but letting $r = \log_2(N)-1$ is a good starting point. In reality, the lower current gates tend to also have smaller transistors so that the optimal r should be slightly higher than predicted.

Of course for large adders, other circuit techniques can be used such as carry-bypass or carry-lookahead, but this shows a general trend for the effects of logic depth and current scaling. In the CORDIC, $N=12$ so a ratio of $r=4$ is used.

VII. CORDIC RESULTS

Now that we have discussed the benefits of using multiple VSC's to allow different currents in the CORDIC ripple adders, we can look at the overall results. We designed 3 different MCML CORDICs, each optimized to a different performance level. For the high performance CORDIC, more current is used throughout to reduce delay. As a result of the increased current, slightly larger transistors, higher VDD, and higher voltage swing are required to maintain the desired DC properties. In the low performance mode, small currents are used and it can therefore utilize reduced voltage swing and VDD. The CMOS design is not optimized for different performance levels but it was rather simulated at 4 different values of VDD.

Besides using different performance levels, we can report two sets of results: one utilizing adaptive pipelining and the other one not. The non-adaptively pipelined results use the worst case clock frequency and the nominal power consumption. The worst case clock frequency assumes worst case process corner and +/- 10% variation in VDD for CMOS. Since the MCML circuits consume a constant amount of current, the variation on VDD will be much smaller and is assumed to be negligible but the worst case process corner is still used. The adaptively pipelined results use the nominal clock frequency and power consumption. All clock frequencies used have a 10% margin over the total critical path delay. The results are summarized in Figs. 8-10.

There are several important things to notice about the final data. The most important trend to realize is shown in Fig. 11 and concerns the relative performance of MCML to CMOS as a function of clock frequency. One can see that for high performance requirements, the energy-delay of MCML is significantly lower than for CMOS. As the performance requirement lessens, the gains from MCML also decrease until CMOS performs better than MCML. This agrees with our earlier analysis that MCML has no optimal point in the energy-delay but CMOS does. It is also interesting to note that MCML circuits can provide faster designs than possible in CMOS at maximum VDD.

The next thing to notice is the effect of process and voltage variations on CMOS and MCML. Even without adaptive pipelining, the MCML circuits have much more constant delays under varying conditions, especially at the low performance end. For a CMOS VDD = 1.0V, the process and voltage variation creates a difference of 80% in worst case delay from the nominal. The low

performance MCML variation is only about 7%. With adaptive pipelining, this variation reduces to 0. The large variations in CMOS significantly reduce the allowable clock frequency and hurt the energy-delay dramatically.

Nominal VDD (V)	2.5	2.0	1.5	1.0
Worst Case VDD (V)	2.25	1.8	1.35	0.9
Nominal Delay (ns)	2.71	3.38	5.01	12.1
Worst Case Delay (ns)	3.68	4.78	7.57	21.7
Clock Frequency (MHz)	250	190	120	40
Power (mW)	22.6	10.3	3.45	0.48
Power-Delay (pJ)	90.4	54.2	28.8	12.0
Energy-Delay (pJ*ns)	362	285	240	300

Figure 8 : CMOS CORDIC Results

Performance Level	High	Med.	Low
VDD (V)	1.1	1.05	1.0
Voltage Swing (V)	0.4	0.35	0.3
Worst Case Delay (ns)	3.29	4.86	7.84
Clock Frequency (MHz)	275	185	115
Power (mW)	18.6	9.00	4.33
Power-Delay (pJ)	67.6	48.6	37.7
Energy-Delay (pJ*ns)	246	263	328

Figure 9 : MCML CORDIC Results - No Adaptive Pipelining

Performance Level	High	Med.	Low
VDD (V)	1.1	1.05	1.0
Voltage Swing (V)	0.4	0.35	0.3
Nominal Delay (ns)	2.94	4.45	7.30
Clock Frequency (MHz)	310	200	125
Power (mW)	18.6	9.00	4.33
Power-Delay (pJ)	60	45.0	34.6
Energy-Delay (pJ*ns)	194	225	277

Figure 10 : MCML CORDIC Results - With Adaptive Pipelining

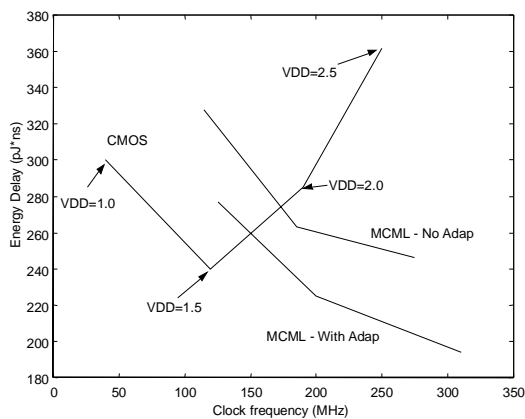


Figure 11 : MCML vs. CMOS CORDIC Performance

While actual area metrics are difficult to determine without circuit layout, we can use transistor count as a rough estimate. The MCML CORDICs have about

16,500 transistors and the CMOS CORDICs have around 11,500 transistors. The CMOS design has a much larger percentage of pmos devices and also will have more non-minimum sized devices than the MCML CORDIC. On the other hand, the MCML design is differential and will consume more area for routing. A completely accurate analysis of area resources can only be accomplished with layout.

The final result to examine is the actual supply current variation in MCML compared to CMOS. The supply current for the high performance MCML CORDIC varies from 16.4mA to 17.2mA for a net variation of +/- 0.4mA or 2.4%. The low performance CORDIC has current variation from 4.0mA to 4.4mA for a total of +/- 0.2mA or 4.8%. For comparison, the CMOS block has current which varies from 0 to 40mA for VDD=2.5V.

VIII. CONCLUSIONS

MOS Current-Mode Logic seems to be a promising alternative to standard CMOS design for high performance, low-power applications if used properly. While the complexity of design is much higher in MCML, it has been shown that significant reductions can be achieved in power-delay and energy-delay product of deeply pipelined, high performance computation.

MCML's other benefits include static current draw from the supplies, common mode noise rejection, insensitivity to process changes, and friendliness to neighboring analog circuit components.

REFERENCES

- [1] M. Mizuno, M. Yamashina, K. Furuta, H. Igura, H. Abiko, K. Okabe, A. Ono, H. Yamada, "A GHz MOS, Adaptive Pipeline Technique Using MOS Current-Mode Logic," IEEE Journal of Solid-State Circuits, Vol 31, No. 6, June 1996, p784-791.
- [2] Jan Rabaey, "Digital Integrated Circuits: A Design Perspective," Prentice Hall, 1996.
- [3] Dake Liu and Christer Svensson, "Trading Speed for Low Power by Choice of Supply and Threshold Voltages," IEEE Journal of Solid-State Circuits, Vol 28, No. 1, January 1993, p10-17.
- [4] Anantha P. Chandrakasan and Robert W. Broderson, "Minimizing Power Consumption in Digital CMOS Circuits," Proceedings of the IEEE, Vol 83, No. 4, April 1995, p498-523.
- [5] J.E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electron. Comput., vol. EC-8, p. 330-334, Sept. 1959
- [6] J. S. Walther, "A unified algorithm for elementary functions," in Proc. AFIPS Spring Joint Comput. Conf., 1971, p. 379-385.