Logical Effort Based Design Exploration of 64-bit Adders Using a Mixed Dynamic-CMOS/Threshold-Logic Approach

Peter Celinski, Said Al-Sarawi, Derek Abbott
Centre for High Performance Integrated Technologies & Systems (CHiPTec)
The Department of Electrical and Electronic Engineering, The University of Adelaide
SA 5005, Australia.
celinski@eleceng.adelaide.edu.au

Sorin Cotofana, Stamatis Vassiliadis
Computer Engineering Group
Electrical Engineering Department
Delft University of Technology
Mekelweg 4, 2628 CD Delft
The Netherlands.

Abstract

This paper presents the design exploration of CMOS 64-bit adders designed using threshold logic gates based on systematic transistor level delay estimation using Logical Effort (LE). The adders are hybrid designs consisting of domino and the recently proposed Charge Recycling Threshold Logic (CRTL). The delay evaluation is based LE modeling of the delay of the domino and CRTL gates. From the initial estimations, we select the 8-bit sparse carry look-ahead/carry-select scheme. Simulations indicate a delay of less than 5 FO4, which is 1.1 FO4 or 17% faster than the nearest domino design.

1 Introduction

Addition is one of the most critical operations performed by VLSI processors. Adders are used in floating-point arithmetic units, ALUs, memory addressing and program counter-update. The main requirements are speed, power dissipation and low area.

Threshold logic (TL) was introduced over four decades ago, and over the years has promised much in terms of reduced logic depth and gate count compared to conventional logic-gate based design. Lack of efficient CMOS realizations has meant that TL has thus far had little impact on VLSI. Efficient TL gate realizations have recently become available, and a small number of applications based on TL gates [2, 1, 9, 10] have demonstrated its ability to achieve high operating speed and significantly reduced area.

To date no large scale arithmetic building blocks for processors have been designed using TL. We address this issue by proposing a high speed 64-bit TL based adder.

The delay analysis method used in in this work enables the comparison of various adder topologies based on logical effort. Another motivator for this approach is the desire to avoid the common and largely unsatisfactory presentation of circuit performance results commonly found in the literature in the form of delay numbers with insufficient information to allow comparison across different process technologies and loading conditions.

This paper presents the design of and critical path delay evaluation of a high speed hybrid CRTL-domino adder. The aim is to use a relatively quick method to determine fast 64-bit adder topologies without venturing into CAD tool complexity, taking into account the advantages and limitations of CRTL. Simulations are used to verify the accuracy of the delay model estimate. The proposed 64-bit adder has a simulated critical path delay of 5.3 FO4 in a 0.35 μm process.

We begin in Section 2 by giving a brief overview of threshold logic. This is followed by a description of CRTL in Section 3. Section 4 briefly reviews Logical Effort and the delay model for CRTL gates is developed in Section 5. The circuit design examples are presented and evaluated in Section 6. Finally a conclusion and suggestions for future work are given in Section 7.

2 Overview of Threshold Logic

Threshold logic emerged in the early 1960's as a generalized theory of switching logic and includes conventional Boolean logic as its subset. A threshold logic gate is functionally similar to a hard limiting neuron without learning capability. The gate takes n binary inputs x_1,x_2,...,x_n and produces a single binary output Y, as shown in Fig. 1.

The Boolean function computed by such a gate is called a threshold function and it is specified by the gate threshold T and the weights w_1,w_2,...,w_n, where w_i is the weight associated with the i^{th} input variable x_i. The output y is

Proceedings of the IEEE Computer Society Annual Symposium on VLSI Emerging Trends in VLSI Systems Design (ISVLSI'04)
0-7695-2097-9/04 $20.00 © 2004 IEEE

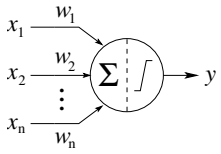

Figure 1. Threshold Gate Model

given by (all operators algebraic):

$$y = \begin{cases} 1, & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq T \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A TL gate can be programmed to realize many distinct Boolean functions by adjusting the threshold T and/or the weights w_i. For example, an n-input TL gate with $T = n$ will realize an n-input AND gate and by setting $T = n/2$, the gate computes a majority function. This versatility means that TL offers a significantly increased computational capability over conventional AND-OR-NOT logic.

3 Charge Recycling Threshold Logic

The realization for CMOS threshold gates presented in [2] and used in the design of TL circuits in this work is now described. Fig. 2 shows the circuit structure. The sense amplifier (cross coupled transistors M1-M4) generates output Q and its complement Q_b. Precharge and evaluate is specified by the enable clock signal E and its complement $\bar{E}$. The inputs x_i are capacitively coupled onto the floating gate ϕ of M5, and the threshold is set by the gate voltage t of M6. The potential ϕ is given by

$$\phi = \frac{\sum_{i=1}^{n} C_i x_i}{C_{tot}}, \tag{2}$$

where C_{tot} is the sum of all capacitances, including parasitics, at the floating gate of M5. Weight values are thus realized by setting capacitors C_i to appropriate values. Typically, in CMOS technology these capacitors are implemented between the polysilicon 1 and polysilicon 2 layers where available, or other dedicated linear layers for linear capacitors.

The enable signal E controls the precharge and activation of the sense circuit. The gate has two phases of operation, the equalize phase and the evaluate phase. When $\bar{E}$ is high the output voltages are equalized. When E is high, the outputs are disconnected and the differential circuit (M5-M7, M10, M11) draws different currents from the formerly equalized nodes Q and Q_b. The sense amplifier is activated after the delay of the enable inverters and amplifies the difference in potential now present between Q and Q_b, accelerating the transition. In this way the circuit structure determines whether the weighted sum of the inputs, ϕ, is greater or less than the threshold, t, and a TL gate is realized. Transistors M10 and M11 turn off the differential circuit after evaluation is completed to reduce the power dissipation. The gates may be pipelined in a self-timed manner as described for the Asynchronous Sense Differential Logic family in [5]. Extensive Monte Carlo, varied voltage and temperature operating point and process corner variation simulations have shown the gate operates reliably at high speed [2]. In addition, a series of test gates with fan-in ranging from 8 to 64 were fabricated in a 0.35 μm process, as shown in Fig. 3 and correct functionality was verified for all gates, however precise delay measurement at high clock frequency is the subject of ongoing work.

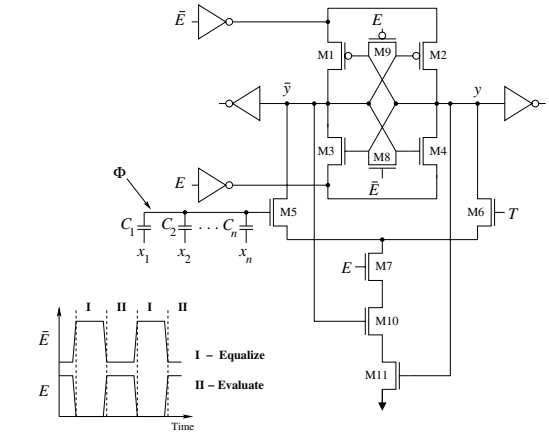

Figure 2. The CRTL gate circuit and Enable signals.

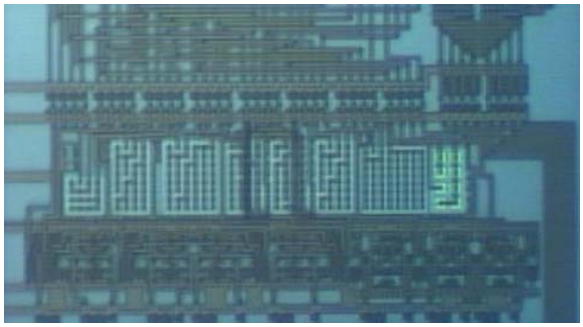

Figure 3. Chip micrograph showing CRTL gates with 8 to 64 inputs.

4 Logical Effort

Logical effort (LE) is a design methodology for estimating the delay of CMOS logic circuits, implementing a given logic function [12]. Logical effort is based on a reformulation of the conventional RC model of CMOS gate delay which separates the effects on delay of gate size, topology, parasitics and load. The relative simplicity of the method compared to other delay modeling techniques and sufficient accuracy allow it to be used early in the design process to evaluate alternative circuits.

The total delay of a gate, d, is comprised of two parts, an intrinsic parasitic delay p, and an effort delay, f, driving the capacitive load. The parasitic delay is largely independent of the transistor sizes in the gate, since wider transistors which provide increased current have correspondingly larger diffusion capacitances. The effort delay in turn depends on two factors, the ratio of the sizes of the transistors in the gate to the load capacitance and the complexity of the gate. The former term is called *electrical effort*, h, and the latter is called *logical effort*, g.

Electrical effort is defined as

$$h = \frac{C_{out}}{C_{in}}, \tag{3}$$

where C_{out} and C_{in} are the gate load capacitance and input capacitance, respectively. The logical effort, g, characterizes the gate complexity, and is defined as the ratio of the input capacitance of the gate to the input capacitance of and inverter that can produce equal output current. Alternatively, the logical effort describes how much larger than an inverter the transistors in the gate must be to be able to drive loads equally well as the inverter. By definition an inverter has a logical effort of 1.

The delay of a single logic gate can be expressed as

$$d = gh + p. \tag{4}$$

This delay is in units of τ, which is, the delay of an inverter driving an identical copy of itself, without parasitics. This normalization enables the comparison of delay across different technologies. The product gh is called the gate or stage effort.

The path delay, D, is the sum of the delays of each of the gate stages in the path, d_i, and consists of the *path effort delay*, D_F, and the *path parasitic delay*, P,

$$\begin{aligned} D &= \sum d_i \\ &= D_F + P \\ &= \sum g_i h_i + \sum p_i. \end{aligned} \tag{5}$$

It can be shown that the path delay is minimized when each stage in the path bears the same stage effort and the minimum delay is achieved when the stage effort is

$$f_{min} = g_i h_i = F^{1/N}, \tag{6}$$

where F is the path effort.

This leads to the main result of logical effort, which is the expression for minimum path delay

$$D_{min} = NF^{1/N} + P. \tag{7}$$

The accuracy of the delay predicted by Equation (4) can be improved by calibrating the model by simulating the delay as a function of load (electrical effort) and fitting a straight line to extract τ, the inverter parasitic delay, p_{inv}, and the logical effort, g. We will use this technique to develop a calibrated logical effort based model for the delay of the CRTL gates.

5 Modeling CRTL Delay

We begin by providing a set of assumptions which will simplify the analysis, a proposed expression for the worst case delay of the CRTL gate and a derivation of the model's parameters.

5.1 Notation and Assumptions

The TL gate is assumed to have n logic inputs (fanin), the total number of gate inputs connected to logic one is denoted by N, and T is the threshold of the gate. The potential of the gate of transistor M6, t, in Fig. 2 is given by $t = \frac{T}{n} \times V_{dd}$. In the worst case, the voltage ϕ in Equation (2) takes the values $\phi = t \pm \frac{\delta}{2}$ where δ is given by $\delta = \frac{V_{dd}}{n}$. This expresses the worst case (greatest delay) condition where the difference between ϕ and t is minimal, ie. the step voltage generated by the sum of inputs with respect to the threshold voltage is smallest.

The gate inputs are assumed to have unit weights, ie. $w_i = 1$, since the delay depends only on the value of N and T. Also, without loss of generality, we will assume positive weights and threshold, since negative weights may easily be accommodated in the differential structure of the gate by using a network of input capacitors connected to the gate of M6. Since the gate is clocked, we will measure delay from the clock E to Q_i-Q_{bi}. Specifically, delay will be measured as the average of the 50% point on two falling transitions of E to the 50% points on the corresponding falling and rising edges of Qi and Q_{bi}. Generally, the delay will depend on the threshold voltage, t, the step size, δ, and the capacitive output load on Q_i and Q_{bi}. To simplify the analysis, we will fix the value of t at 1.5 V. This value is close to the required gate threshold voltage in typical circuit applications. Therefore the worst case delay depends only on the fan-in and gate loading, and allows us to propose a model based on expressions similar to those for conventional logic based on the theory of logical effort.

5.2 Formulation of the Model and Parameter Extraction

The delay of the CRTL gate may be expressed as Equation (8). This delay is the total delay of the sense amplifier and the buffer inverters connected to Q and Q_b, and depends on the load, h, and the fanin, n, as follows

$$d_{E\rightarrow Qi} = \{g(n)h + p(n)\}\tau. \quad (8)$$

The load, h, is defined as the ratio of load capacitance on Q_i (we assume the loads on Q_i and Q_{bi} are equal) and the CRTL gate unit weight capacitance. Both logical effort and parasitic delay in Equation (8) are a function of the fanin.

The delay parameters for the industrial 0.35 μm process used to obtain the simulation results presented here are τ=40 ps, p_{inv}=1.18 and FO4=207 ps.

The values of g and p in Equation 8 were extracted by linear regression from simulation results for a range of fanin from $n = 2$ to 60 while the electrical effort was swept from $h = 0$ to 20. By fitting a curve to the parameters g and p, CRTL gate delay may be approximated in closed form by

$$d_{E\rightarrow Qi} = \{(0.002n + 0.34)h + \ln(\mathrm{n}) + 1.6\}\tau. \quad (9)$$

In order to use Equation (9), it is necessary to compensate for the parasitic capacitance at the floating gate of M5. This effective fanin, n_{eff}, is given by

$$n_{eff} = \left\{ \frac{\sum_{i=1}^{n} C_i + C_p}{\sum_{i=1}^{n} C_i} \right\} n_0 \ , \quad (10)$$

where n_0 is the number of inputs to the gate and n_{eff} is the value used to calculate the delay. Typically, for a large fanin CRTL gate, by far the major contribution to the parasitic capacitance will be from the bottom plate of the floating capacitors used to implement the weights. In the process used in this work, this corresponds to the poly1 plate capacitance to the underlying n-well used to reduce substrate noise coupling to the floating node.

6 64-bit Adder Design and Critical Path Delay Estimation

The delay estimation based on logical effort has been carried out for a number of high speed adders [3, 8], including dynamic Kogge-Stone (D-KSA), dynamic carry look-ahead (D-CLA), dynamic Ling/conditional-sum (D-LCNSA) and Intel's Quarternary (D-QTA) [6]. We extend this work to include CRTL based adders. For completeness, we also include comparison with the HP Ling adder [7], Harris' [4] adder and the Output Prediction Logic adder developed by Sechen [11].

6.1 64-bit Adder Architecture

The selection of the adder architecture is heavily influenced by the availability of fast high fan-in CRTL gates. This leads us to use CLA (carry look-ahead) and CSA (carry-select) blocks. The adders described in [7] and [4] are based on 4-bit CLA blocks, which is usually the optimal trade off between the depth of the CLA tree and the number of series transistors in a CMOS gate. The carries in these adders are generated at 16-bit boundaries, requiring 16-bit sub-adders for carry-select blocks.

Increasing the number of bits handled by a CLA block to 8-bits results in fewer logic levels and a more regular design and layout [11]. This is impractical in conventional CMOS logic, since it requires 8 series transistors. We can, however, take advantage of the wide AND gates in CRTL. We obtain the regular structure shown in Fig. 4. In this scheme, the 64-bit input addends are divided into eight 8-bit blocks, and it has $\log_8 64$, or two levels of carry look-ahead. The Kogge-Stone scheme generates carries for each bit position, so no carry select is needed. The 4- and 8-bit block versions have depth $\log_4 64$=3 and $\log_8 64$=2, respectively. However, they consist of many more CLA blocks with significantly increased wiring and fanouts.

The structure of the proposed adder is a sparse carry prefix tree. In the first layer, the bitwise propagate and generate signals, p_i, g_i, are formed, followed by the computation of eight pairs of 8-bit group generate and propagate signals P_j^{j-7}, G_j^{j-7} in the second layer. These are then assimilated in the global carry look-ahead block to generate the sum selection carries, c_7, c_{47}, ..., c_{55}, which select pre-computed 8-bit sums. These 8-bit adders are also based on carry look-ahead.

The CLA equations may be written as given as Equations (11)-(14). Each CLA level consists of an AND and an OR gate, which requires significantly lower sum of weights in the CRTL implementation that a single gate AND-OR implementation. The six stage critical path of the 64-bit adder consists the domino-OR2 to generate p_7 (despite the lower logical effort and parasitic delay, this gate has a higher fanout than g_0), AND8 and OR8 to generate G_7^0, AND8 and OR8 to generate c_{55} in the global CLA block and a 2:1 MUX to select the sum.

The bitwise propagate and generate signals are computed as follows

$$p_i = a_i + b_i \quad (11)$$
$$g_i = a_i \cdot b_i \ , \quad (12)$$

and from these the 8-bit block group propagate and generate signals are given by

$$P_7^0 = p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \quad (13)$$
$$G_7^0 = g_7 + p_7 \cdot g_6 + p_7 \cdot p_6 \cdot g_5 + \ \ldots$$

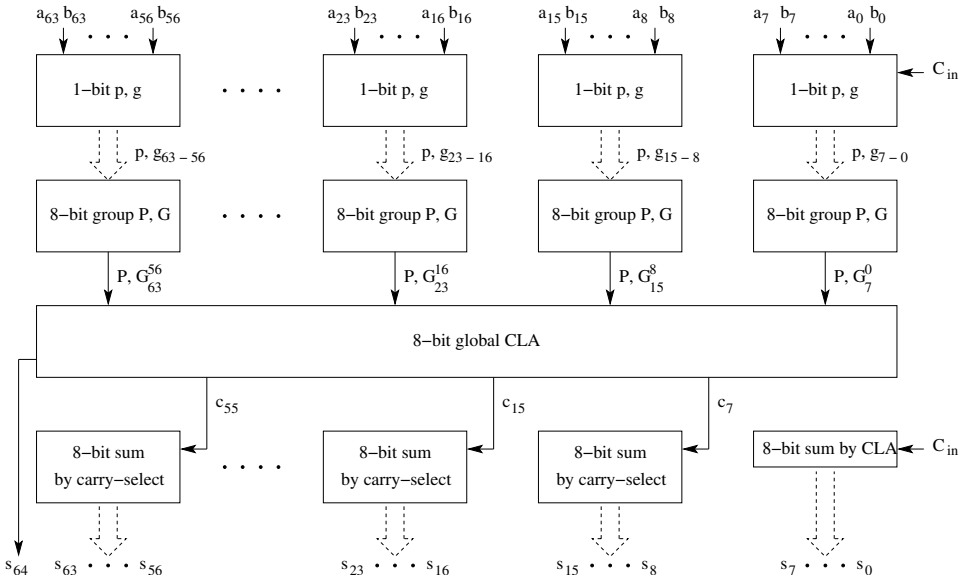

Figure 4. 64-bit adder block diagram.

$$+\ p_7 \cdot p_6 \cdot p_5 \cdot p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0\ . \quad (14)$$

Finally the 8-bit block carry outputs are given by

$$c_7 = G_7^0\ . \quad (15)$$

A similar expression to Equation (14) may be written for generating the global look-ahead carries.

6.2 Delay Estimation and Comparison

In addition to the delay model for CRTL discussed earlier, in order to evaluate the adder delay it is necessary to characterize the domino gates using HSPICE simulation of the gate delay for various output loads, according to the LE rules. Characterization of domino gates considers only the one transition of interest, which is the falling transition for the dynamic pull down and rising transition for the hi-skew static inverter. This is repeated for each of the gates, and the results are shown in Table 1. Note that the dynamic gates listed consist of the pull down path only, excluding the static inverter.

The delay of the critical path, s_{63}, dyn-OR2 → CRTL-AND8 → CRTL-OR8 → CRTL-AND8 → CRTL-OR8 → MUX2 is calculated using Equations (5) and (9) and Table 1. For the 8-input CRTL gates we ues an n_{eff} value of 10. In addition, we must consider the fan-out of 7 of the dyn-OR2 gate (which drives 7 unit weight CRTL inputs). The other gates have a unity electrical effort. From Equation 7 the optimized delay of the two stage dyn-OR2 gate is therefore given by

$$\begin{aligned} d_{\mathrm{OR2,min}} &= NF^{1/N} + P \\ &= 2\{g_{\mathrm{NOR2}} \times g_{\mathrm{HS-Inv}} \times h_{\mathrm{HS-Inv}}\}^{0.5} \\ &\quad + p_{\mathrm{NOR2}} + p_{\mathrm{HS-Inv}} \\ &= 2\{0.3 \times 0.7 \times 7\}^{0.5} + 1.4 + 1 \\ &= 4.8\tau\ . \end{aligned} \quad (16)$$

Table 1. Normalized LE parameters of various gates in 0.35 μm technology, in units of $\tau = 40$ ps.

Gate Type	LE, (g)	Parasitic delay, p
Inverter	1	1.18
Hi-skew Inverter	0.7	1
dyn-NAND2	0.4	1.8
dyn-NOR2	0.3	1.4
2:1 static MUX	1.13	2.6

From this the critical path delay is calculated as follows

$$\begin{aligned} d_{s_{63}} &= d_{\mathrm{OR2,min}} + 4 \times d_{\mathrm{CRTL10}} + (gh+p)_{\mathrm{MUX2}} \\ &= 4.8 + 4 \times 4.26 + 1.13 + 2.6 \\ &= 25.6\tau \\ &= 4.9\ \mathrm{FO4}\ . \end{aligned} \quad (17)$$

The proposed adder consists of 3653 transistors and 342 unit capacitors. The critical path was also simulated, including wiring capacitance estimations based on traversed CRTL and domino cell pitch, and the extracted gate layouts

Table 2. Comparison of high speed 64-bit adders.

64-bit Adder	# Stages	Tech. μm	LE FO4	Sim. FO4
D-CLA [3]	14	0.18	11.1	13.6
D-LCNSA [3]	9	0.18	9.0	9.5
Intel D-QTA [6]	10	0.10	8.3	-
D-HCA [8]	10	0.10	8.26	-
D-KSA [3]	6	0.18	6.2	7.4
HP mod. Ling [7]	4	0.5	-	7
Harris [4]	-	0.6	-	6.4
OPL [11]	8	0.25	-	2.9
→ This Work	6	0.35	4.9	5.3

and the critical path delay thus obtained was 5.3 FO4. Note that the 207 ps FO4 delay is a very slow process corner for a drawn channel length of 0.4 μm, and is the fastest we had available, ([11] similarly reports 162 ps for the 0.25 μm process used in that work). It is therefore not surprising that the 930 ps delay for the 0.5 μm process reported in [7] has a FO4 delay less than ours, especially if a faster process corner was used.

The FO4 delay comparison with eight other dynamic high speed adders is shown in Table 2, with the logical effort estimate and simulated or measured delay values listed where available. The comparison suggests a significant delay speed improvement of almost 1.1 FO4 or 17% compared to Harris' agressive domino design. The OPL adder is included for completeness to acknowledge other novel circuit techniques, it has the significant drawback of requiring 8 clock phases which has significant power dissipation issues, in addition to the reduced noise margin of OPL gates. Table 2 also shows that delay is related to but not proportional to the number of gate levels on the critical path, so comparing delay estimates based on this simple metric is inconclusive.

7 Conclusions and Future Work

A high speed 64-bit adder based on a hybrid carry lookahead/carry-select scheme using Charge Recycling Threshold Logic and conventional domino logic has been proposed. The worst case critical path delay was shown to be significantly improved compared to previously proposed domino high-speed adders. The results show that by combining TL and conventional CMOS logic with the appropriate architectural strategy, relatively fast arithmetic circuits may be achieved.

The work presented here leaves a number of unresolved questions. The important issue of power dissipation has not been addressed. Power dissipation may be traded for delay and the energy-delay curves for adders may cross [8] which implies that single point delay comparisons such as in Table 2 are not always meaningful. The energy-delay dependency of the proposed adder is currently under investigation. The results presented here suggest that the substantial delay improvement over domino justifies the added design complexity of CRTL.

References

[1] P. Celinski, S. D. Cotofana, and D. Abbott. Area efficient, high speed parallel counter circuits using charge recycling threshold logic. In *Proc. IEEE International Symposium on Circuits and Systems*, volume 5, pages 233–236, May 2003.

[2] P. Celinski, J. F. López, S. Al-Sarawi, and D. Abbott. Low power, high speed, charge recycling CMOS threshold logic gate. *IEE Electronics Letters*, 37(17):1067–1069, August 2001.

[3] H. Dao and V. G. Oklobdzija. Application of logical effort techniques for speed optimization and analysis of representative adders. In *Proc. Thirty-Fifth Asilomar Conferenceon Signals, Systems and Computers, 2001*, volume 2, pages 1666–1669, November 2001.

[4] M. Horowitz. EE271 class notes (adders). Stanford University, http://eeclass.stanford.edu/ee371/.

[5] B.-S. Kong, J.-D. Im, Y.-C. Kim, S.J-Jang, and Y.-H. Jun. CMOS differential logic family with self-timing and charge-recycling for high-speed and low-power VLSI. *IEE Proceedings Circuits, Devices and Systems*, 150(1):45–50, February 2003.

[6] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar. A 4 GHz 130nm address generation unit with 32-bit sparse-tree adder core. In *Symposium on VLSI Digest of Technical Papers*, pages 126–127. IEEE, 2002.

[7] S. Naffziger. A sub-nanosecond 0.5μm 64b adder design. In *International Solid State Circuits Conference Digest of Technical Papers*, pages 362–363. IEEE, 1996.

[8] V. Oklobdzija, B. Zeydel, D. Hoang, S. Mathew, and R. Krishnamurthy. Energy-delay estimation technique for high-performance microprocessor VLSI adders. In *Proceedings. 16th IEEE Symposium on Computer Arithmetic*, pages 272–279, June 2003.

[9] H. Özdemir, A. Kepkep, B. Pamir, Y. Leblebici, and U. Çiliniroğlu. A capacitive threshold-logic gate. *IEEE JSSC*, 31(8):1141–1149, August 1996.

[10] M. Padure, S. Cotofana, and S. Vassiliadis. A low-power threshold logic family. In *Proc. IEEE International Conference on Electronics, Circuits and Systems*, pages 657–660, 2002.

[11] S. Sun, L. McMurchie, and C. Sechen. A high-performance 64-bit adder implemented in output prediction logic. In *Proceedings Conference on Advanced Research in VLSI*, pages 213–222, 2001.

[12] I. E. Sutherland, R. F. Sproull, and D. L. Harris. *Logical Effort, Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.