

# Cascaded Carry-Select Adder (C<sup>2</sup>SA): A New Structure for Low-Power CSA Design

Yiran Chen, Hai Li\*, Kaushik Roy and Cheng-Kok Koh

ECE School, Purdue University  
1285 EE Bldg, 465 Northwestern Ave.  
West Lafayette, IN 47906, USA  
+1 765 49-{41744, 42361, 63683}

{yc, kaushik, chengkok}@ecn.purdue.edu

\*Qualcomm Inc.

Building BB, 5775 Morehouse Drive  
San Diego, CA 92121, USA  
+1 858 84-57393

hail@qualcomm.com

## ABSTRACT

In this paper we propose a novel low-power Carry-Select Adder (CSA) design called Cascaded CSA (C<sup>2</sup>SA). Based on the prediction of the critical path delay of current operation, C<sup>2</sup>SA can automatically work with one or two clock-cycle latency and a scaled supply voltage to achieve power improvement. Post-layout simulations of a 64-bit C<sup>2</sup>SA in 180nm Technology show that C<sup>2</sup>SA can operate at a lower supply voltage, attaining 40.7% energy saving, while maintaining a similar (average) Latency Per Operation (LPO) compared to standard CSA.

## Categories and Subject Descriptors

B.7.1 [Integrated Circuits]: Types and Design Styles – microprocessors and microcomputers, VLSI (very large scale integration)

**General Terms:** Design

**Keywords:** Low-power, Carry-select adder

## 1. INTRODUCTION

Smaller feature dimension of transistors and higher level of integration have produced faster speed, although, with increased power dissipation and power density. High power dissipation raises temperature, degrades system reliability, and introduces high cost for heat sinks. Numerous power management techniques targeting different components of power have been proposed in the past few years. Examples of such techniques include clock-gating [1], gated-ground [2], supply voltage scaling [3], and logic and architectural level techniques [4].

The relationship between dynamic/leakage power consumptions and the supply voltage ( $V_{DD}$ ) can be summarized as [3]:

$P_{Dyn} \propto V_{DD}^2$ ,  $P_{S\_leak} \propto V_{DD}^3 \sim V_{DD}^4$  and  $P_{G\_leak} \propto e^{V_{DD}}$ . Here  $P_{Dyn}$ ,

$P_{S\_leak}$  and  $P_{G\_leak}$  denote dynamic power, sub-threshold leakage power and gate leakage power, respectively. Therefore, both dynamic and leakage power can be drastically reduced by scaling down the supply voltage [3].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISLPED'05, August 8-10, 2005, San Diego, California, USA.  
Copyright 2005 ACM 1-59593-137-6/05/0008...\$5.00

The required  $V_{DD}$  for logic operations is mainly determined by the required operation frequency and the *critical* path delay. Usually a margin is reserved to take into account any uncertainties in circuit/device parameters and environmental factors. However, such a worst-case-based  $V_{DD}$  selection overestimates the actual required  $V_{DD}$ : the combination of worst-case conditions is rare.

It is noted that in an ALU, only a few operations can reach the critical delay, which is determined by some specific operands. In a 64-bit Ripple-Carry Adder (RCA), for example, the longest carry propagation delay occurs only when the carry-out signal ( $C_o$ ) of the first bit adder propagates through the last bit adder, i.e.,  $A<63:0> = \text{fff\_fff\_fff\_fff}$  (hex) and  $B<63:0> = 0000\_0000\_0000\_0001$  (hex). However, if  $A<63:0> = \text{fff\_0000\_fff\_fff}$  (hex) and  $B<63:0> = 0000\_0000\_0000\_0001$  (hex),  $C_o$  of the 33rd bit adder does not rely on its carry-in signal ( $C_{in}$ ) from the 32nd bit adder. Consequently, the total operation latency equals the delay of the carry propagation from the first bit adder through the 32nd bit adder. In [5], by checking the possible longest carry propagation length based on the input vector, RCA operations are classified as long- or short-latency ones. When a long-latency operation occurs, the  $V_{DD}$  is raised to a higher level to satisfy certain timing requirement.

Carry-Select Adder (CSA) provides a well-balanced choice between the slow speed of RCA and the large area occupied by Carry Look-Ahead Adder (CLA). In this paper, we propose a novel low-power Carry-Select Adder (CSA) structure: Cascaded CSA (C<sup>2</sup>SA). By distinguishing between the short- and long-latency operations, C<sup>2</sup>SA works with variable latencies (1 or 2 cycles). Our design allows more aggressive  $V_{DD}$  scaling (under the same timing constraint), or extra timing margin to tolerate the process parameter and environmental variations (under the same energy budget), while closely maintaining the same Average Latency Per Operation (ALPO) compared to standard CSA.

Our experiments on a prototype 64-bit C<sup>2</sup>SA show 40.7% and 44.4% total power savings in 180nm and 70nm technologies, respectively, under scaled  $V_{DD}$ 's. No ALPO-based performance loss is introduced and only around 3.97% area overhead is incurred with respect to the standard CSA.

The remaining sections are organized as follows: Section 2 describes the structure and functionality of C<sup>2</sup>SA; Section 3 presents the corresponding performance and power analysis; Section 4 concludes the paper.

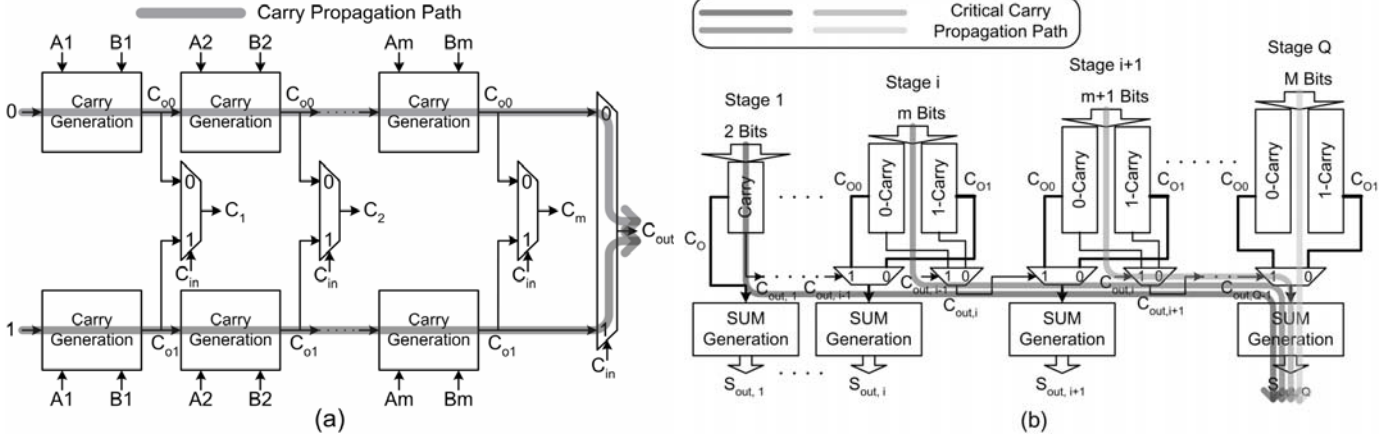


Fig. 1 Carry propagation in CSA (a) Carry-select stage (b) Square-root CSA

## 2. CASCADED CSA DESIGN

### 2.1 Input-vector-dependent Latency of CSA

In a standard square-root CSA, the adder is composed of several carry-select stages (Fig. 1). A carry-select stage includes carry generation, a multiplexer (MUX) for carry selection and a sum generation block. In a carry-select stage  $i$ , two carry-out signals  $C_{00}$  and  $C_{01}$  of every bit are pre-calculated by assuming the carry-in signal of stage  $i$ ,  $C_{in}$  is 0 or 1, respectively. One of  $C_{00}$  and  $C_{01}$  is selected by  $C_{in}$  that comes from the previous stage  $i-1$  and the carry-out signal of the last bit in stage  $i$  is transferred to the next stage  $i+1$ .

Usually square-root CSA is carefully designed to ensure that in each carry-select stage, i.e., stage  $i+1$  in Fig. 1(b),  $C_{00}$ ,  $C_{01}$  of the last bit adder and  $C_{out,i}$  arrive at the input of MUX at the same time. Hence, the critical data paths begin from the first bit adder of every stage, cross the MUXs of the sequential stages and ends at the sum generation block of the last stage (Fig. 1(b)). The longest delay of CSA  $D_{CSA}$  is determined by [6]:

$$D_{CSA} = D_{setup} + m_0 D_{carry} + Q D_{mux} + D_{sum} \quad (1)$$

where  $D_{carry}$ ,  $D_{mux}$  and  $D_{sum}$  are the delays of carry generation circuit, MUX and sum generation circuit, respectively.  $m_0$  is the number of input bits in the first carry-select stage.  $Q$  is the number of carry-select stages.  $D_{setup}$  is the setup time, such as the delays for the generation of the intermediate signals G (Generation) and P (Propagation) [6]. The row "Std. CSA" in Table 1 gives an example of a well-designed standard 64-bit CSA [7]. The total number of input bits of carry-select stage increases with the increase of stage number.

The delay of CSA heavily depends on the input vectors:

**Case 1:** In an RCA-based carry-select stage, the longest carry propagation length is determined by the input vectors (see Section 1). An intermediate signal P (Propagation) can be adopted to indicate the behavior of carry propagation in an RCA: For the  $k$ th bit adder in an  $m$ -bit RCA, the propagation signal is  $P_k = A_k \oplus B_k$  [6],  $k = 1 \dots m$ , where  $A_k$  and  $B_k$  are the inputs of the  $k$ th bit adder. When  $P_k = '0'$  ( $A_k = B_k$ ), the carry-out signal of the  $k$ th bit adder is determined only by  $A_k$  and  $B_k$  and the carry propagation in RCA is "broken" at the  $k$ th bit adder under such condition.

**Case 2:** The carry propagation through the MUX chain of carry-select stages is also determined by the input vectors. For example,

when  $A_k = B_k$  for any  $k = 1 \dots m$  in carry-select stage  $i$ , the carry-out signal  $C_{out,i}$  of carry-select stage  $i$  is determined only by the input bit pairs of stage  $i$ , regardless the carry-in signal  $C_{out,i-1}$  from stage  $i-1$ . Hence, the carry propagation through the MUXs of the carry-select stages is "broken" at the MUX of stage  $i$ . We point out that in such a situation, the critical path delay of a well-designed standard square-root CSA may *not* change. It is because of the existence of the critical paths that begin from the first bit adder of stage  $j$  ( $j \geq i$ ), cross the MUXs of the sequential stages and end at the sum generation of the last stage.

### 2.2 Proposed $C^2$ SA Structure

Based on the observation of case 2 in Section 2.1, a prototype 64-bit  $C^2$ SA design with 13 carry-select stages is proposed in the row " $C^2$ SA" in Table 1. The carry-select stages of  $C^2$ SA can be divided into two separate parts: The first part starts at stage 1 with 2 bits, and the second part starts at stage 8 with 3 bits. In each part, the total number of input bits of carry-select stage increases with the increase of the stage number. Each carry-select stage is designed as an RCA. Compared to the standard CSA with 10 carry-select stages,  $C^2$ SA with 13 carry-select stages has a longer critical delay, which begins from the first bit adder of stage 1 and ends at the sum generation of stage 13. A Carry Length Detect Circuit (CLDC) is designed to detect long- and short-latency operations, based on the input vectors (Fig. 2(a)).

In our prototype 64-bit  $C^2$ SA, The input of CLDC is the input bit pairs of carry-select stage 8 and 9:  $A <37:31>$  and  $B <37:31>$ . If  $A_i = B_i$ , for any  $i = 31 \dots 37$ , the carry-out signal of stage 9,  $C_{out,9}$  is determined only by the input bit pairs of stage 8 and 9, regardless the carry-in signal  $C_{out,7}$  from stage 7. The current operation is detected as a short-latency operation and the signal  $S\_LAT$  is kept at 'high'. However, when  $A_i \neq B_i$ , for all  $i = 31 \dots 37$ , the negation of signal  $S\_LAT$  indicates the occurrence of a long-latency operation and triggers a  $\overline{GATING}$  signal to disable the clock switching in the following clock cycle (Fig. 2(b)). The stretched clock phase guarantees the completion of long-latency operations.

In CLDC design, the row of XOR gates generates the propagation signal P for each sing-bit adder. Such P-generation circuit may not be required if the single-bit adder itself can provide propagation signal P. For example, if carry-select stage is designed with cascaded static adders [6], the generated intermediate P signals can be reused by CLDC directly.

**Table 1. The numbers of input bits of carry-select stages in 64-bit standard CSA and C<sup>2</sup>SA**

Stage Number	1	2	3	4	5	6	7	8	9	10	11	12	13
Std. CSA	2	2	3	4	6	7	8	9	11	12	/	/	/
C <sup>2</sup> SA	2	2	3	4	6	7	7	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>

If there are multiple carry-select stages involved in CLDC (i.e., stage 8 and 9 in C<sup>2</sup>SA in Table 1), we just send all related P signals:  $P_{i,k}$  ( $i=8, k=1\dots3$  and  $i=9, k=1\dots4$ ) to CLDC.

### 2.3 The functionality of C<sup>2</sup>SA

Without loss of generality, we assume there are  $N_C$  carry-select stages: stage  $L, L+1\dots$  and  $L+N_C-1$ , involved in CLDC.

When a short-latency operation is detected, the carry propagation in C<sup>2</sup>SA is divided into two parts: one from carry-select stage 1 through stage  $L+N_C-1$  and another one from carry-select stage  $L$  through the last stage  $Q$ . The longest latency of all short-latency operations  $L_S$  is:

$$L_S = \text{Max}(D_{\max 1}, D_{\max 2}) \quad (2)$$

where  $D_{\max 1}$  is the longest delay from carry-select stage 1 through  $L+N_C-1$  and  $D_{\max 2}$  is the longest delay from carry-select stage  $L$  through  $Q$ .

Similarly, the longest delay of all long-latency operations  $L_L$  is the maximum delay of the C<sup>2</sup>SA. We have:

$$L_S < D_{\text{std}} < L_L \quad (3)$$

Here  $D_{\text{std}}$  is the longest delay of a standard CSA. Because of the simplicity of CLDC, the delay of S\_LAT signal generation  $T_G$  and the delay of D-Flip-flop  $T_D$  are much shorter than  $L_S$ .

The classification of operations allows C<sup>2</sup>SA to work in low power mode. For example, when short-latency operations occur,

$V_{DD}$  can be scaled down until  $L_S$  equals  $D_{\text{std}}$  under original  $V_{DD}$ . Consequently, original clock period, which is mainly determined by  $D_{\text{std}}$  and the data setup/hold time of CSA under original  $V_{DD}$ , is maintained. However, both dynamic and leakage powers are reduced due to scaled down  $V_{DD}$ .

The existence of long-latency operation results in the extended execution time of CSA, and consequently, power and performance overhead. However, our analysis in Section 3 shows that the probability of long-latency operations is quite small and introduces little power and performance overheads.

## 3. PERFORMANCE/POWER ANALYSIS

### 3.1 Detection of Long/short-latency Operation

For random inputs A and B, the probability of  $P = 1$  (or  $P = 0$ ) is  $1/2$ . As is evident from the above discussion, the long-latency operation occurs only when all P signals in CLDC = 1. Hence, for random inputs, the probability of long-latency operations  $\text{Pr}_L$  is:

$$\text{Pr}_L = \prod_{i=1}^{N_C} \frac{1}{2^{m_i}} \quad (4)$$

Here  $N_C$  is the total number of carry-select stages in CLDC while  $m_i$  is the total number of input bits of stage  $i$ .

Obviously, increasing the numbers of input bits involved in CLDC reduces the probability of long-latency operation, and hence, decreases the performance overhead. However, it may also increase  $L_S$  and the power overhead introduced by CLDC.

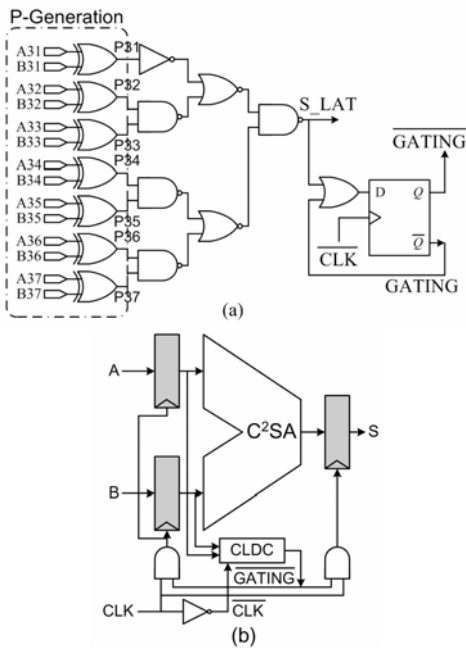
In our 64-bit C<sup>2</sup>SA design, the input bits of stage 8 and 9 (3 bits for stage 8 and 4 bits for stage 9, respectively – see Table 1) are used for the latency detection in CLDC. Based on Eq. (4), the probability of long-latency operation is only  $1/(2^3 \cdot 2^4) \approx 0.78\%$ .

If  $L_S < L_L < 2 \cdot L_S$ , the long-latency operation can be completed within two clock cycles, which are determined by  $L_S$ . The extra clock cycle required by long-latency operations results in IPC-based (instruction per cycle-based) performance loss. For random inputs, the IPC-based performance loss of C<sup>2</sup>SA equals the long-latency operation possibility (0.78% in our 64-bit C<sup>2</sup>SA design).

### 3.2 C<sup>2</sup>SA with $V_{DD}$ Scaling

To evaluate our C<sup>2</sup>SA design, a 64-bit standard CSA and C<sup>2</sup>SA (C<sup>2</sup>SA-180nm) were designed with TSMC 180nm technology. Every carry-select stage is designed as an RCA. Mirror adder [6] and inverter elimination scheme in carry path [8] are adopted in RCA for delay optimization. The multiplexers are also carefully sized. We note that our C<sup>2</sup>SA structure can be easily extended to other carry-select stage structures, for example, RCA with static adders or CLA. The layout implementation (Fig. 3) shows that C<sup>2</sup>SA introduces only 3.97% area overhead with respect to the standard CSA design.

The corresponding timing parameters of standard CSA and C<sup>2</sup>SA are summarized in Table 2.



**Fig. 2 CLDC design in C<sup>2</sup>SA**

**(a) CLDC Schematic (b) Functionality of CLDC**



Fig. 3 Layouts of C<sup>2</sup>SA-180nm and standard CSA

Table 2. Timing parameters of C<sup>2</sup>SA and standard CSA

Tech.	$D_{conv}$ (ns)	$L_L$ (ns)	$L_S$ (ns)	$T_G$ (ns)	$T_D$ (ns)
180nm	2.709	3.459	2.068	0.2188	0.2021
70nm	0.8864	1.1107	0.6546	0.0813	0.103

To evaluate the effectiveness of our design in scaled technologies, we also designed a 64-bit C<sup>2</sup>SA and the corresponding standard CSA in BPTM [9] 70nm technology (C<sup>2</sup>SA-70nm).

Since the long-latency operation requires two clock cycles, we introduce Average Latency Per Operation (ALPO),  $L_A$  to evaluate the performance of C<sup>2</sup>SA. Here  $L_A = L_S \cdot (1 + Pr_L)$ . The reciprocal of  $L_A$  is the Instruction Per Second (IPS), which accurately describes the throughput of C<sup>2</sup>SA. Fig. 4 shows  $L_L$ ,  $L_S$  and  $L_A$  tracking  $V_{DD}$  scaling for C<sup>2</sup>SAs in both 180nm and 70nm technologies.

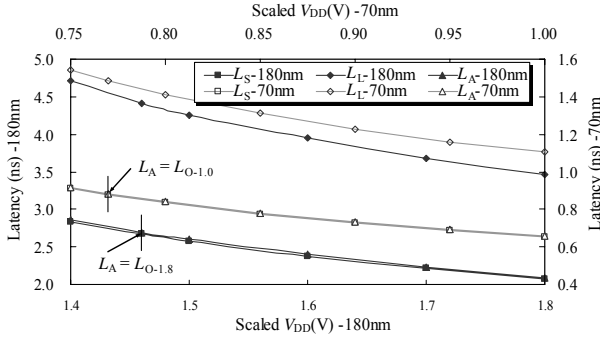


Fig. 4 Performance of C<sup>2</sup>SA under scaled  $V_{DD}$ 's

For C<sup>2</sup>SA-180nm, when  $V_{DD}=1.8V$ , the normal operation latency ( $L_S$ ) is 23.7% less than the one of standard CSA ( $L_{O-1.8}$ ). Also, when  $V_{DD} = 1.46V$ , C<sup>2</sup>SA offers the same performance as the standard CSA with  $V_{DD}=1.8V$  because  $L_A = L_{O-1.8}$ . The simulation results also show that  $L_S < L_L < 2 \cdot L_S$  under all considered  $V_{DD}$ 's. Similar results can be drawn for C<sup>2</sup>SA-70nm.  $L_A = L_{O-0.7}$  when  $V_{DD} = 0.77V$ .

We used NanoSim to simulate the energy consumptions of the standard CSA and C<sup>2</sup>SA with  $2^{14} = 16384$  random inputs. In our simulation, both dynamic energy and static energy are considered. The comparisons between the energy consumptions of C<sup>2</sup>SA under the scaled  $V_{DD}$ 's and the one of standard CSA under original  $V_{DD}$ 's (1.8V for 180nm Technology and 1.0V for 70nm Technology, respectively) are shown in Fig. 5. All results are normalized over the energy consumptions of standard CSA's under original  $V_{DD}$ 's, as depicted by bar "Std.-180n/70n"

Essentially, compared to the standard CSA design, 40.7% and 44.4% total energy savings are respectively achieved by C<sup>2</sup>SA-180nm and C<sup>2</sup>SA-70nm under the scaled  $V_{DD}$ 's, without any ALPO-based performance loss.

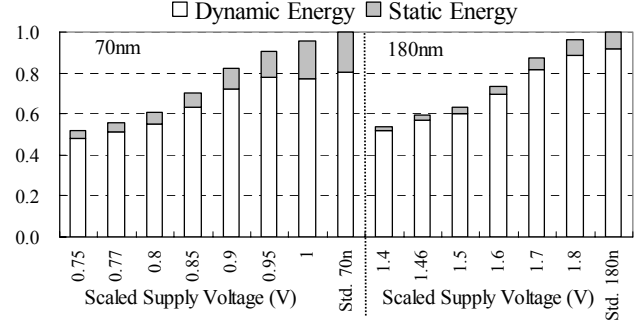


Fig. 5 Energy consumptions of standard CSA and C<sup>2</sup>SA  
(a) 180nm Technology (b) 70nm Technology

## 4. CONCLUSION

A novel low-power Carry-Select Adder (CSA) structure – Cascaded CSA (C<sup>2</sup>SA) – is proposed. The auxiliary carry length detection circuit (CLDC) enables C<sup>2</sup>SA to execute the short- or long-latency operations by one or two clock cycles, respectively. Our prototype 64-bit C<sup>2</sup>SA in 180nm technology offers a 40.7% energy saving while maintaining the same throughput, under a scaled  $V_{DD}$ . The incurred area overhead is about 4%.

## 5. ACKNOWLEDGMENTS

This research was supported in part by Semiconductor Research Corporation under contract 1122.001.

## 6. REFERENCES

- [1] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy. Deterministic clock gating for microprocessor power reduction. *In Proceedings of the 9th Int'l Symp. on High Performance Computer Arch.*, pp. 113-124, Feb. 2003.
- [2] A. Agarwal, H. Li, and K. Roy, A Single-Vt Low-Leakage Gated-Ground Cache for Deep Submicron, *In IEEE Journal of Solid-State Circuits*, Vol.38-2, pp. 319-328, Feb. 2003.
- [3] H. Li, C.-Y. Cher, T. N. Vijaykumar, and K. Roy, VSV: L2-Miss-Driven Variable Supply-Voltage Scaling for Low Power, *In Proceeds of the 36th IEEE/ACM Int'l Symp. on Microarchitecture*, pp. 19-28, Dec. 2003.
- [4] D. Drnst, et. al. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. *In Proceedings of the 36th IEEE/ACM Int'l Symp. on Microarchitecture*, pp. 7-18, Dec. 2003.
- [5] H. Suzuki, W. Jeong, K. Roy, Low Power Adder with Adaptive Supply Voltage, *In Proceedings of the 21st Int'l Conf. on Computer Design*, pp. 103-106, Oct. 2003.
- [6] J. M. Rabaey, Digital Integrated Circuits: a design perspective, *Prentice Hall*, 1996.
- [7] Y. Kim and L. S.-Kim, A Low Power Carry Select Adder with Reduced Area, *In Proceedings of 2001 IEEE Int'l Symp. on Circuits and Systems*, Vol. 4, pp. 218-221, May 2001.
- [8] H. Morinaka, et. al. A 64 bit carry look-ahead CMOS adder using Modified Carry Select, *In Proceeds of the IEEE Custom Integrated Circuits Conf.*, pp. 585-588, May 1995.
- [9] <http://www-device.eecs.berkeley.edu/~ptm>.