Design of Low Power Asynchronous Parallel Adder

Benedicta Roseline. R[1] Kamatchi. S[2]
[1]M.E. Student (VLSI Design) [2]Assistant Professor
[1,2]Akshaya College of Engineering and Technology Coimbatore, India

Abstract— This paper describes an asynchronous parallel adder. It is based on Radix method for faster computation of sum and to reduce delay caused by carry chain. The computation has been carried out using parallel process. The aim of this work is to reduce the Power Delay Product (PDP) and Energy Delay Product (EDP) of an adder. We use two Full Adders (FA) in a single block and use a carry look-ahead technique to shorten the carry path within the radix-4 FA block. To obtain low area, the carry is generated first and then it is reused in sum generation. The adder is implemented using Tanner EDA v13 tool. The practicality and superiority of the proposed technique have been verified by simulations over other asynchronous adders.

Key words: Asynchronous Circuits, Radix Based Full Adder, Carry Look-Ahead Technique

I. INTRODUCTION

Synchronous circuits are used mostly for design of the adders although there is a strong interest in clockless/ asynchronous processors/circuits [1]. There is no assumption of quantization of time in asynchronous circuits.

Clockless circuits are free from problems that associated with clocked (synchronous) circuits. Asynchronous circuits are clockless and hence, in order to establish a pipeline in the absence of clocks, a request-acknowledgment handshaking protocol is used to control the logic flow in asynchronous circuits.

For small elements like bit adders, explicit handshaking blocks are expensive. Hence, using dual-rail carry propagation in adders it is implicitly and efficiently managed. Acknowledgment is given from a single-bit adder block by a valid dual-rail carry. Therefore, clockless adders are either based on full dual-rail encoding of all signals (more formally using null convention logic [2] which symbolically correct logic instead of Boolean logic) or pipelined operation using single-rail data encoding and dual-rail carry representation for acknowledgments. These constructs add robustness to circuit designs, however there is a significant overhead to the average case performance benefits of asynchronous adders.

This brief presents an asynchronous parallel adder using a radix Full Adder (FA). The basis of the design of this adder is a 24-bit transistor mirror FA. Therefore, it is feasible for VLSI implementation. To obtain an efficient area implementation, the carry is first generated on its critical path and reused in sum generation. For speed optimization, two full adders are joined for a radix-4 FA block. To shorten the carry path carry look ahead technique has been used.

This paper is organized as follows: Section II gives the description of asynchronous adders, Section III explains the existing method, Section IV details the proposed adder, Section V shows the CMOS implementation and simulation results and Section VI includes the conclusion and future scope.

II. BACKGROUND

Asynchronous circuits refer to logic circuits do not depend on an external clock for circuit operation. They depend on and/or engineer timing assumptions for the correct operation. The worst case bundled delay mechanism of synchronous circuits has been avoided by asynchronous adders, as early completion sensing in these circuits have the potential to run faster for dynamic data.

A. Single-Rail Data Encoding in Pipelined Adders

To enable the adder block and establish the flow of carry signals the asynchronous Req/Ack handshake can be used. For internal bitwise flow of carry outputs, a dual-rail carry convention is used. More than two logic values (invalid, 0, 1) can represent these dual-rail signals and therefore can be used to generate bit-level acknowledgment when a bit operation is completed. When all bit Ack signals are received (high), final completion is sensed. Examples include the carry-completion sensing adder [8] and a speculative completion adder is proposed in [9]. The adder in [8] is an of a pipelined adder, which uses full adder (FA) functional blocks adapted for dual-rail carry and the adder in [9] uses so-called abort logic and early completion to select the proper completion response from a number of fixed delay lines. The drawback is the expensive implementation of the abort logic due to high fan-in requirements.

B. Dual-Rail Encoding in Delay Insensitive Adders

Bundling constraints are asserted in Delay insensitive (DI) asynchronous adders. In presence of bounded but unknown gate and wire delays [2] they can work correctly. DI ripple carry adder (DIRCA) and DI carry look-ahead adder (DICLA) are examples for DI adders. The conventional CMOS RCA uses 28 transistors wheras DIRCA adder is presented in [8] uses 40 transistors per bit. The DICLA defines carry propagate, generate, and kill an equation in terms of dual-rail encoding [8] that is similar to CLA. In DICLA carry signals are organized as a hierarchical tree. Therefore for long carry chain, they can potentially operate faster.

III. EXISTING METHOD

A. Architecture

The architecture and theory behind Asynchronous Parallel Adder using recursive approach is presented.

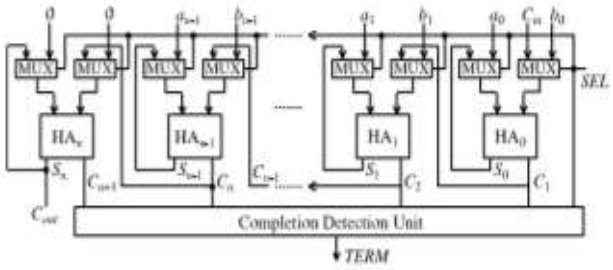

Fig. 1: General block diagram of Asynchronous parallel adder using recursions

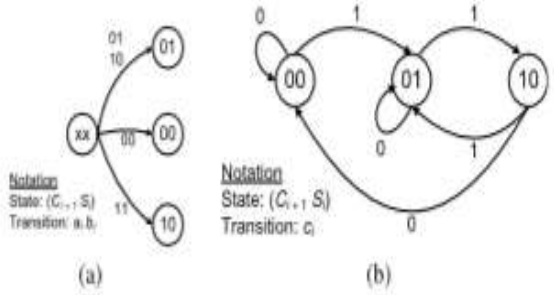

Fig. 2: State diagrams (a) Initial Phase (b) Recursive Phase

The general architecture of the adder is shown in Fig. 1. Req handshake signal is required for the selection input for two-input multiplexers from a single 0 to 1 transition denoted by SEL.

During the initial phase SEL = 0, the actual operands are initially selected and will switch to feedback/carry paths for iterative phase using SEL = 1. Multiple recursions occur through the feedback path and continue until all the carry values assume zero values to achieve completion.

B. State Diagrams

State diagrams are drawn for the initial phase and the Recursive phase in Fig. 2. Representation of each state is given by (C_{k+1} S_k) pair where C_{k+1}, S_k represent carry out and sum values, respectively, from the k^{th} bit adder block. The circuit merely works as a combinational HA operating in fundamental mode, during the initial phase. State (11) cannot appear, due to the use of HAs instead of FAs.

When SEL = 1, the feedback path through multiplexer block is activated. To complete the recursion, the carry transitions (C_k) are allowed as many times as needed.

This design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output.

This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states [4].

C. Recursive Formula for Binary Addition

Let $a_{n-1}a_{n-2} \ldots a_0$ and $b_{n-1}b_{n-2} \ldots b_0$ be two n-bit binary numbers with sum and carry denoted by $S_{n-1}S_{n-2} \ldots S_0$ and $c_nc_{n-1} \ldots c_0$ where 0^{th} bit represents the least significant bit.

Basic single bit adders are now discussed.

1) Single Bit Adders:

Single bit Half-Adder (HA) and Full-Adder (FA) are the fundamental building blocks for nearly all high-speed adders. A single bit HA for k^{th} bit addition is logically formulated as follows:

$$S_k = a_k \oplus b_k,$$
$$c_{k+1} = a_k b_k. \quad (1)$$

A single bit full adder implementation additionally takes consideration of the carry-in input from the preceding single bit unit and formulated as follows:

$$S_k = a_k \oplus b_k \oplus c_k,$$
$$c_{k+1} = a_k b_k + (a_k \oplus b_k)\, c_k. \quad (2)$$

The recursive binary addition formula for addition of A and B is presented as follows.

Let S_k^i and C_k^i be the Sum and Carry respectively for k^{th} bit at the i^{th} recursion. The initial condition for the addition operation can now be defined as follows:

$$S_k^0 = a_k \oplus b_k,$$
$$C_k^0 = a_k b_k. \quad (3)$$

The i^{th} iteration for the recursive addition can be found as follows:

$$S_k^i = S_k^{i-1} \oplus C_{k-1}^{i-1}, \quad (4)$$
$$C_k^i = S_k^{i-1} C_{k-1}^{i-1}, \quad (5)$$

The recursion is terminated at the j^{th} iteration when the following condition is met.

$$C_n^j C_{n-1}^j \Lambda C_0^j = 0, \quad (6)$$

Fast adder will now be designed using the formulae presented in equations (3)-(6).

At first the correctness of the recursive formulation will be proved inductively by the following observation and subsequent theorem.

2) Observation 1:

In a single bit adder with no carry in, the maximum obtainable result is 2.

3) Explanation:

It is obvious that the sum cannot exceed the maximum sum obtained by two highest possible operands and hence should be equal or less than 2.

The significance of this observation is that for individual k^{th} bit adder, the case of having S_k=1 and C_k=1 (decimal value of 3) is impossible as it will exceed the maximum of the sum of two inputs which is 2 (binary 10). Thus the only valid (S, C) forms by k^{th} bit adder are (0, 0), (0, 1) and (1, 0).

4) Theorem 1:

The recursive formulation of (3), (4), (5) and (6) will produce correct sum for any number of bits and will terminate at finite time.

5) Proof:

We prove the correctness of the algorithm by induction on terminating condition.

6) Basis:

For operands A, B such that $c_k^0 = 0$ for $\forall k$, k ε [0 . . . n], the proposed recursive formulation produces correct result in parallel by single bit computation time and terminates instantly as condition (6) is met.

7) Induction:

Assume $C_k^j \neq 0$ for $\exists k$. Let i be such a bit for which $C_i^j = 1$. First we show that it will be killed in the $(j+1)^{th}$ iteration and next we will show that it will be successfully transmitted to next higher bit in the $(j+1)^{th}$ iteration.

According to Observation 1, (S_i^j, C_i^j), (S_{i+1}^j, C_{i+1}^j) could be in any of (0, 0), (0, 1) or (1, 0) forms. As C_i^j=1, it implies that S_i^j=0. Hence, from equation (5), C_i^{j+1}=0 for any input condition between 0 to i−1 bits.

We now consider the next higher bit (S_{i+1}^j, C_{i+1}^j) at j^{th} iteration. By observation 1, it could be in any of (0, 0), (0, 1) or (1, 0) forms. In the $(j+1)^{th}$ iteration, the (0, 0) and (0, 1) forms from j^{th} iteration will correctly produce output of (1, 0) following equation (4) and (5) and hence carry will be absorbed in $(j+1)^{th}$ bit. For (1, 0) form, the carry is supposed to propagate through this bit level as the sum value is 1. By applying (4) and (5), we find C_{i+1}^{j+1}=1. Thus the carry propagation/killing will be correctly performed by i^{th} bit adder.

Finally, there is one extra bit adder block for carry out of the n-bit adder. This will have initial output (S_n^0, C_n^0)=(0, 0). Any carry chain is hence bound to end up at this bit and produce output (1, 0), if it is not already killed by any previous bit levels during earlier iteration(s).

Thus all the single bit adders will successfully kill or transfer the carries to the next level until being killed at the n[th] bit carry out block. This ensures that terminating condition is always reached by the recursive formulation.

8) Drawbacks

When SEL =1, the feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values. Due to multiple iterations, the delay gets increased and the speed of operation is slow. Hence the adder consumes more power.

IV. PROPOSED METHOD

A. 24-Bit Transistor Mirror FA

A popular topology is the mirror FA, which is used as the base topology for our designs. The carry is generated first since it is on the critical path, and then it is reused in the sum generation to obtain low area. Mirror FA that consists of 24 transistors is shown in Fig. 3 (a). Both FAs implement the same Boolean function, which is given in equation (1) and (2).

$$C_{out} = (A+B)C_{in} + AB \quad (1)$$

$$Sum = \overline{C_{out}}.A + \overline{C_{out}}.B + \overline{C_{out}}.C_{in} + A.B.C_{in} \quad (2)$$

B. Basic Radix-4 FA

The main approach in this work is to reduce the PDP and EDP of an adder and to reduce the delay due to carry propagation. We use the 24-transistor mirror FA as a base for our design, which is illustrated in Fig. 3(a). To optimize the speed, we joined two FAs in a single block and use the carry look-ahead technique to shorten the carry path within the radix-4 FA block. The transistor schematic is shown in Fig. 3(b). The total transistor count is 56. Boolean equations for the two carry and two sum gates of the carry-accelerated design are given below.

$$\overline{C_1} = \overline{(A_0 + B_0)C_{in} + A_0.B_0} \quad (3)$$

$$S_0 = \overline{C_1}.B_0 + \overline{C_1}.C_{in} + \overline{C_1}.A_0 + A_0.B_0.C_{in} \quad (4)$$

$$C_{out} = (A_0 + B_0)(A_1 + B_1)C_{in} + A_0.B_0(A_1 + B_1) + A_1.B_1 \quad (5)$$

$$S_1 = \overline{C_{out}}.\overline{B_1} + C_{out}.\overline{C_1} + C_{out}.\overline{A_1} + \overline{A_1}.\overline{B_1}.\overline{C_1} \quad (6)$$

The circuit for computing the least significant sum bit *S*0 is the same as that of a radix-2 FA. We use the carry look ahead technique to design the circuitry for the most significant carry bit *C*out. Equation (5) is similar to the equation for a CLA with the difference that generate and propagate signals are generated explicitly for the CLA. To implement the function that computes the most significant sum bit *S*1 with a single gate, we invert the inputs *A*1, B1 and *C*1.

V. CMOS IMPLEMENTATION AND SIMULATION

The tool used for simulation purpose is Tanner tool version 13.0. The design cycle for the development of electronic circuits includes an important Pre-fabrication verification phase.

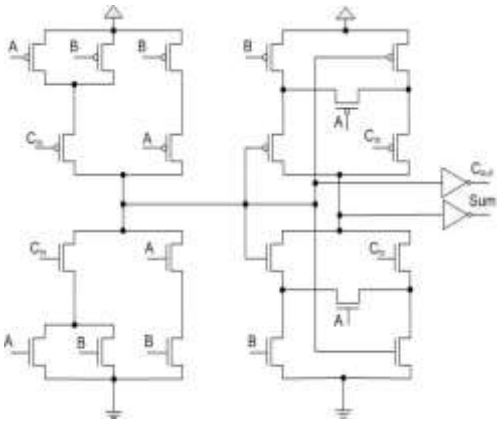

Fig. 3 (a): 24- bit transistor mirror FA

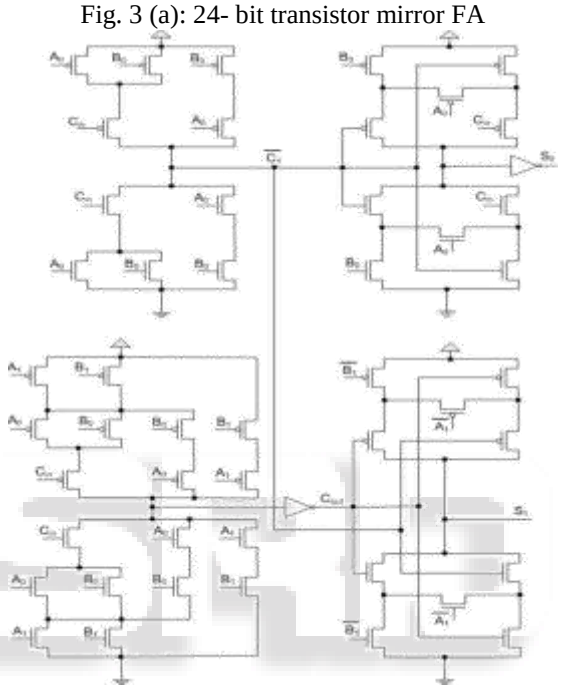

Fig. 3(b): 4-bit radix FA

A. 4-Bit Radix Adder

Radix method based 4-bit adder was designed in tanner tool and simulation is done.

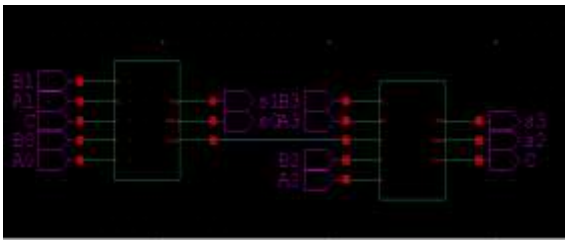

Fig. 4(a): Design of 4-bit Radix Adder

Output waveform for 4-bit radix adder is shown below.

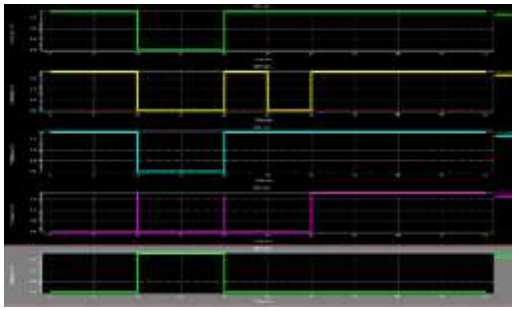

Fig. 4(b): Output waveform of 4-bit Radix Adder

B. 8-Bit Radix Adder

Radix method based 8-bit adder was designed and simulation is done.

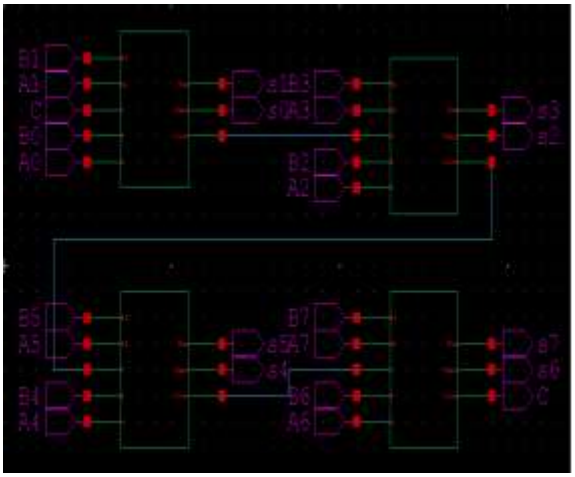

Fig. 5(a): Design of 8-bit radix adder

Output waveform for 8-bit radix adder is shown below.

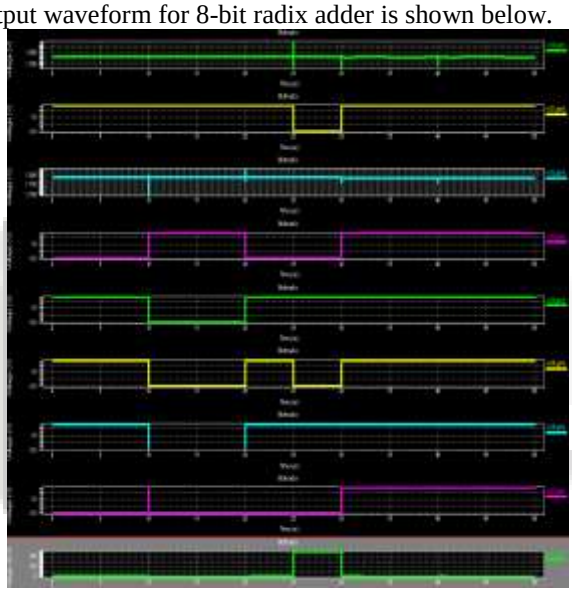

Fig. 5(b): Output waveform of 8-bit radix adder

C. 16-Bit Radix Adder

Radix method based 16-bit adder was designed in tanner tool and simulation is done.

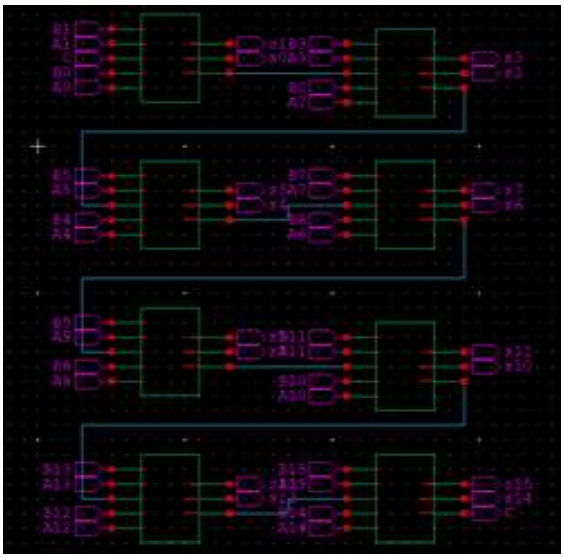

Fig. 6(a): Design of 16-bit Radix Adder

Output waveform for 16-bit radix adder is shown below.

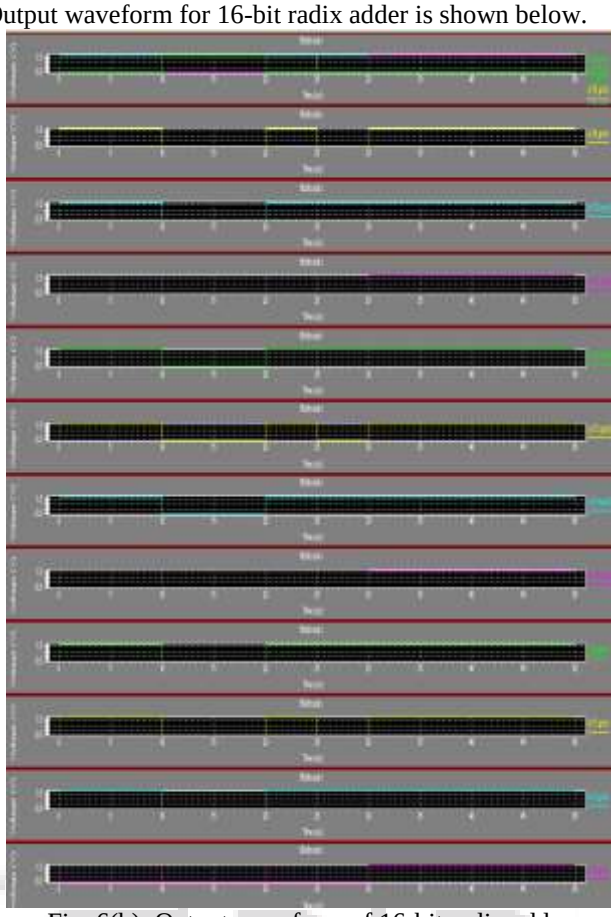

Fig. 6(b): Output waveform of 16-bit radix adder

D. Result Comparison

Table I gives the comparison results between both existing and proposed work's parameters.

Parameters	Existing work 16-bit Asynchronous Parallel Adder Using Recursive Approach	Proposed work 16-bit Asynchronous Parallel Adder Using Radix Adder
Average power	$5.8240 * 10^{-3}$ W	$5.9705 * 10^{-7}$ W
Maximum Power (Static)	$3.5777 * 10^{-2}$ W at $1.500 * 10^{-5}$ secs	$1.6677 * 10^{-2}$ W at $2.000 * 10^{-5}$ secs
Minimum Power (Dynamic)	$3.6648 * 10^{-5}$ W at $1.000 * 10^{-5}$ secs	$1.8541 * 10^{-7}$ W at $1.111 * 10^{-6}$ secs
Power Delay Product (PDP)	0.536 µWs	0.333 µWs
Energy Delay Product (EDP)	8.049 pWs	6.670 pWs
Static Current	1.9876 amps	0.9265 amps

Table 1: Parameter Analysis Comparison

VI. CONCLUSION

An efficient implementation of asynchronous parallel adder was presented. Initially, the theoretical foundation for a

single-rail wave-pipelined adder is established. Subsequently, the architectural design and CMOS implementations are presented. The drawback of asynchronous parallel adder based on recursive approach is overcome by using a radix method. An asynchronous parallel adder based on radix method has been proposed and it has great potential to decrease the delay and total power consumption of adder over the existing methods. Thus it is more suitable for adoption in fast adder implementation in high-performance processors.

REFERENCES

[1] D. Geer, "Is it time for clockless chips? [Asynchronous processor chips]," IEEE Comput., vol. 38, no. 3, pp. 18–19, Mar. 2005.
[2] J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design. Boston, MA, USA: Kluwer Academic, 2001.
[3] P. Choudhury, S. Sahoo, and M. Chakraborty, "Implementation of basic arithmetic operations using cellular automaton," in Proc. ICIT, 2008, pp. 79–80.
[4] M. Z. Rahman and L. Kleeman, "A delay matched approach for the design of asynchronous sequential circuits," Dept. Comput. Syst. Technol., Univ. Malaya, Kuala Lumpur, Malaysia, Tech. Rep. 05042013, 2013.
[5] M. D. Riedel, "Cyclic combinational circuits," Ph.D. dissertation, Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, USA, May 2004.
[6] R. F. Tinder, Asynchronous Sequential Machine Design and Analy- sis: A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems. San Mateo, CA, USA: Morgan, 2009.
[7] W. Liu, C. T. Gray, D. Fan, and W. J. Farlow, "A 250-MHz wave pipelined adder in 2-µm CMOS,"IEEE J. Solid-State Circuits, vol. 29, no. 9, pp. 1117–1128, Sep. 1994.
[8] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self-timed carry-lookahead adders," IEEE Trans. Comput., vol. 49, no. 7, pp. 659–672, Jul. 2000.
[9] S. Nowick, "Design of a low-latency asynchronous adder using spec- ulative completion," IEE Proc. Comput. Digital Tech., vol. 143, no. 5, pp. 301–307, Sep. 1996.
[10] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective. Reading, MA, USA: Addison-Wesley, 2005.
[11] C. Cornelius, S. Koppe, and D. Timmermann, "Dynamic circuit tech- niques in deep submicron technologies: Domino logic reconsidered," in Proc. IEEE ICICDT, Feb. 2006, pp. 1–4.
[12] M. Anis, S. Member, M. Allam, and M. Elmasry, "Impact of technology scaling on CMOS logic styles," IEEE Trans. Circuits Syst., Analog Digital Signal Process., vol. 49, no. 8, pp. 577–588, Aug. 2002.
[13] Mohammed Ziaur Rahman, Lindsay Kleeman, and Mohammad Ashfak Habib, "Recursive Approach to the Design of a Parallel Self-Timed Adder", IEEE Transactions in Very Large Scale Integration (VLSI) Systems. 1063-8210 © 2014

