Hindawi Publishing Corporation
VLSI Design
Volume 2008, Article ID 218565, 8 pages
doi:10.1155/2008/218565

Research Article

Delay Efficient 32-Bit Carry-Skip Adder

Yu Shen Lin and Damu Radhakrishnan

Department of Electrical and Computer Engineering, State University of New York, 1 Hawk Dr, New Paltz, NY 12561-2443, USA

Correspondence should be addressed to Damu Radhakrishnan, damu@engr.newpaltz.edu

Received 27 April 2007; Accepted 9 December 2007

Recommended by Jean-Baptiste Begueret

The design of a 32-bit carry-skip adder to achieve minimum delay is presented in this paper. A fast carry look-ahead logic using group generate and group propagate functions is used to speed up the performance of multiple stages of ripple carry adders. The group generate and group propagate functions are generated in parallel with the carry generation for each block. The optimum block sizes are decided by considering the critical path into account. The new architecture delivers the sum and carry outputs in lesser unit delays than existing carry-skip adders. The adder is implemented in 0.25 μm CMOS technology at 3.3 V. The critical delay for the proposed adder is 3.4 nanoseconds. The simulation results show that the proposed adder is 18% faster than the current fastest carry-skip adder.

1. INTRODUCTION

The ever-increasing demand for mobile electronic devices requires the use of power-efficient VLSI circuits. Computations in these devices need to be performed using low-power, area-efficient circuits operating at greater speed. Addition is the most basic arithmetic operation; and adder is the most fundamental arithmetic component of the processor. Depending on the area, delay and power consumption requirements, several adder implementations, such as ripple carry, carry-skip, carry look-ahead, and carry select, are available in the literature [1, 2]. The ripple-carry adder (RCA) is the simplest adder, but it has the longest delay because every sum output needs to wait for the carry-in from the previous full-adder cell. It uses $O(n)$ area and a delay of $O(n)$ for an n-bit adder. The carry look-ahead adder has $O(\log n)$ delay and uses $O(n \log n)$ area. On the other hand, the carry-skip adder and carry-select adders have $O(\sqrt{n})$ delay and use $O(n)$ area [3].

In this paper, we present the design of a low-power adder with less delay while using minimum hardware. The standard carry generate-propagate logic is used to reduce the critical delay of the adder while blocks of RCAs are used for lesser power consumption. In our design, the generate-propagate logic balances the delay and the number of inputs to the skip logic limits the critical path delay. By applying our design procedure, we speed up the adder by 18% when compared to the current fastest 32-bit adder [4]. In Section 2, we will discuss the previous work done in the area of high-performance adders. In Section 3, we present the design of our adder. Section 4 presents the design of a few basic CMOS cells used in the adder. In Section 5, we present the simulation results for our adder and compare it to other fast adders.

2. THEORETICAL BACKGROUND AND PREVIOUS WORK

The design of a carry-skip adder is based on the classical definition of generate and propagate signals as follows [1, 2]:

$$\begin{aligned} p_i &= X_i \oplus Y_i, \\ g_i &= X_i \cdot Y_i, \end{aligned} \tag{1}$$

where p_i is the propagate signal and g_i is the generate signal, and X_i and Y_i are the input operands to the ith adder cell. The carry out from the ith adder cell is expressed as

$$C_{i+1} = g_i + p_i C_i, \tag{2}$$

where C_i is the carry input to the ith cell.

Two signals, group generate and group propagate, are also defined in [1, 2] and are given by

$$G_{j:i} = g_j + p_j g_{j-1} + p_j p_{j-1} g_{j-2} + \cdots + p_j p_{j-1} p_{j-2} \cdots p_{i+1} g_i, \quad (3)$$
$$P_{j:i} = p_j p_{j-1} p_{j-2} \cdots p_i,$$

where $G_{j:i}$ and $P_{j:i}$ are group generate and group propagate signals from ith cell to jth cell, respectively. Then, the expression for carry out from the whole group is given by

$$C_{j+1} = G_{j:i} + P_{j:i} C_i. \quad (4)$$

Different adder implementations have been developed to optimize various design parameters. Most adder implementations tend to trade off performance and area. One of the earliest adder implementations of this kind was a regular parallel adder layout also known as the Brent-Kung adder '82 [5]. It is a variation of the basic carry look-ahead adder . They emphasized the need for regularity in VLSI circuits to reduce design and implementation costs. They use two types of processor cells: white processor and black processor. The black processor performs the associative concatenation defined in [5] and the white processor simply transmits the data. The adder delay was calculated in terms of the number of exclusive-or (XOR) operations performed while treating each XOR delay as one unit time. For an n-bit adder, the Brent-Kung adder has a delay of $O(\log n)$ and uses $O(n \log n)$ area.

Wei-Thompson'85 [6] proposed an area-time optimal adder design using three types of adder cells: black cells, white cells, and driver cells. The black and white cells are quite similar to the ones used in Brent-Kung adder. They divided the n-bit adder into ascending and descending halves so as to limit the number of bits in the final stage. The concentration of the maximum number of bits was in the middle of the adder and was defined as the height of the adder. The algorithm ends up in an unbalanced binary tree with a delay of $O(\log n)$ consuming an area $O(n \log n)$.

The ELM-adder design presented in [7] computes the sum bits in parallel; thereby reducing the number of interconnects. It implements an n-bit adder as a tree of processors to directly compute the sums in $O(\log n)$ time. The area used is $O(n \log n)$. The adder design was expressed in terms of standard cells, which do not compute carry for each stage. Instead, partial sums were computed for each stage.

Kantabutra'93 [8] presents the design of a one-levelcarry-skip adder using an approach that is very similar to that of Wei-Thompson. In contrary to Wei-Thompson's approach, this design ends up in a symmetrical binary tree of adders. The fan-in to the carry-skip logic increases linearly towards the middle of the adder. A two-level carry-skip adder is presented in [9], where the whole adder stage is divided into a number of sections, each consisting of a number of RCA blocks of linearly increasing length. These adders reduce the delay at the cost of an increase in area and less regular layout.

Nagendra'96 [3] did a survey of various adder designs and concluded that the ELM adder was superior in terms of area, power, delay, and power-delay product. RCA was concluded to have utilized the least power, but has the highest delay due to its carry chain. A variable-width carry-skip adder was shown to be superior to constant-width carry-skip adder, the advantage being greater at higher precisions.

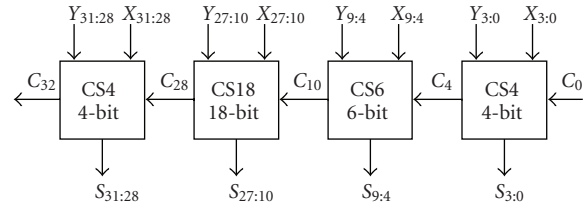

Figure 1: The 32-bit adder divided into 4 blocks [4].

A fully static carry-skip adder designed by Chirca'04 [4] achieved lower-power dissipation and higher performance. To reduce delay and power consumption, the adder is divided into variable-sized blocks that balance the inputs to the carry chain. The main principle behind this design was to utilize the lower blocks and make them work in parallel with higher blocks. This paper is a deviation from the tree approach presented in the ELM adder. A 32-bit adder implementation with a delay of 7 logic levels using carry-skip adders and ripple-carry adders was presented in [4]. This is shown in Figure 1. The logic-level delay defined in the paper is equivalent to the delay of a complex CMOS gate. Efficient and-or-invert (AOI) and or-and-invert (OAI) CMOS gates were used to reduce delay and power.

The 32-bit adder is divided into 4 adder blocks as shown in Figure 1. Carry-select adders were used in the final CS4 block, which significantly increases the hardware. The paper claims that the output will be ready with a delay of 7 logic levels, with the assumption that the critical delay path is the carry propagation path of C_{32} bit. But a closer examination of the previous block CS18 reveals that the 27th bit of the sum output will be available only after a delay of 9 logic levels.

3. NEW DESIGN FOR THE 32-BIT CARRY-SKIP ADDER

The 32-bit carry-skip adder design presented in this paper uses a combination of RCAs together with carry-skip logic (SKIP), carry-generate logic (CG), and group generate-propagate logic (PG). The complete adder is divided into a number of variable-width blocks. Both the carry generation and skip logic use AOI and OAI circuits. The width of each block is limited by the target delay T.

Each block is further divided into subblocks. A subblock may contain additional levels of subblocks in a recursive manner. The lowest-level subblock is formed by a number of variable width RCAs. The adder structure is described as follows:

$$\text{Block} \longrightarrow \langle \text{Block} \rangle * \langle \text{Block} \rangle \mid \langle \text{subblock} \rangle \mid \langle \text{RCA} \rangle,$$
$$\langle \text{subblock} \rangle \longrightarrow \langle \text{subblock} \rangle * \langle \text{subblock} \rangle \mid \langle \text{RCA} \rangle. \quad (5)$$

The 32-bit adder is divided into four blocks. A block diagram of the first three blocks (A_0, A_1, and A_2) is shown in Figure 2. The first block A_0 (LSB) is a full adder by itself.

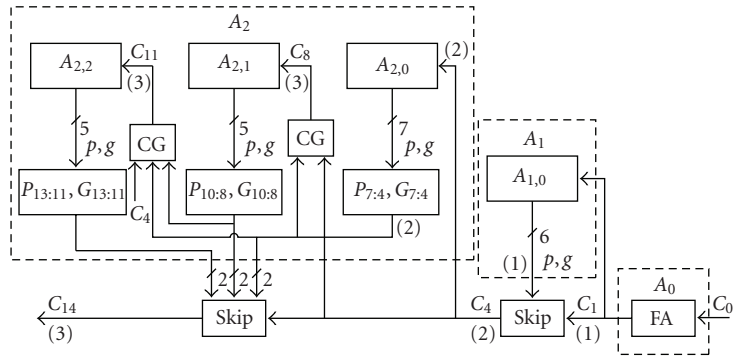

FIGURE 2: Block schematics for first three blocks of 32-bit adder.

The carry from the first block C1 is fed into the second block A_1 and is also fed into the skip logic. The generate and propagate functions (p,g) are generated separately for each full adder in one unit time, where one unit time is defined as the delay of a complex CMOS gate with at most three transistors connected in series from the output node to any supply rail. In Figure 2, the numbers shown in parenthesis represent the number of unit delays of the signal arrival times at the appropriate signal leads. Since the delay of a complex CMOS gate is quadratic on its stack height, in our design, the stack height is limited to 3. This implies that the maximum number of transistors (NMOS or PMOS) in any series connected path is 3. This also restricts the maximum number of inputs to the carry-skip logic to 7. On the other hand, when the generate-propagate outputs are used for group generation and group propagation (3) outputs, a stack height of 3 in the CMOS implementation will allow a 4-bit RCA.

The carry-generation delay from the skip logic is minimized by alternately complementing the carry outputs. Hence, the carry signals generated are $\overline{C_1}, C_4, \overline{C_{14}}$, and so forth. For the very first 1-bit block (A_0), the carry-generation logic is more important than the sum-generation logic since the overall delay of the adder is dependent on the carry from this block. Hence, this block is designed by minimizing the carry out delay as much as possible. The simplest expression of carry out from the LSB full adder is given by

$$\overline{C_1} = \overline{X_0 Y_0 + X_0 C_0 + Y_0 C_0}, \tag{6}$$

where X_0 and Y_0 are the operand bits and C_0 is the input carry. An AOI gate implements this.

The block A_1 in Figure 2 is implemented as a k-bit RCA. For any k-bit RCA, the total number of propagate and generate (p,g) outputs would be 2 k. These 2 k outputs together with the carry from the previous block are fed into carry-skip logic to generate the new carry signal. The fan-in restriction of 7 to the carry-skip logic therefore limits the number of bits in the RCA to 3. The carry out C_4 from skip logic for block A_1 is given by

$$C_4 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 C_1. \tag{7}$$

Since g's and p's can be best implemented in complementary form, we can rewrite C_4 as

$$C_4 = \overline{\overline{g}_3 (\overline{p}_3 + \overline{g}_2)(\overline{p}_3 + \overline{p}_2 + \overline{g}_1)(\overline{p}_3 + \overline{p}_2 + \overline{p}_1 + \overline{C}_1)}. \tag{8}$$

By inspection, C_4 can be implemented by an or-and-invert (OAI) gate and is available in 2 time units. The final Sum output S_3 from this 3-bit RCA will be available in 4 time units. The sum outputs for this RCA are generated either as $S_i = p_i \oplus C_i$ or $S_i = \overline{p_i \oplus \overline{C_i}}$ depending on the carry signal value (C_i *or* $\overline{C_i}$). The carry out C_2 and C_3 are implemented as $C_2 = \overline{\overline{g_1} \cdot (\overline{p_1} + \overline{C_1})}$ and $C_3 = g_2 + p_2 C_2$, respectively.

Now consider block A_2 in Figure 2. The delay of carry signal arriving at the input of the skip logic is 2 time units. This implies that the group generate-propagate (P,G) logic outputs feeding the skip logic must also be available in 2 time units. Hence, the inputs to the (P,G) logic must be available in 1 time unit. This implies that the inputs to the (P,G) logic must be the propagate and generate signals of the full adders. Block A_2 is divided into three subblocks $A_{2,0}$, $A_{2,1}$, and $A_{2,2}$ (in this case, each subblock is an RCA). The maximum width of each RCA is limited to 4 bits due to the fan-in restrictions imposed on the (P,G) block. The width of each RCA is also limited by the target delay T of the 32-bit adder. The width W of the first RCA is given as

$$W = T - D, \tag{9}$$

where D is the arrival delay of the carry output from the previous block. The width of all remaining higher order RCAs in the same block will be 1 bit less because of the delayed arrival times of their carry input by an additional time unit. The carry inputs C_8 and C_{11} to RCAs $A_{2,1}$ and $A_{2,2}$ are generated using AOI logic as follows:

$$\overline{C_8} = \overline{G_{7:4} + P_{7:4} C_4}, \tag{10}$$

$$\overline{C_{11}} = \overline{G_{10:8} + P_{10:8} G_{7:4} + P_{10:8} P_{7:4} C_4}. \tag{11}$$

For a target delay of 6 time units, the width of the first RCA in $A_2(A_{2,0})$ is 4 bits and the widths of the remaining

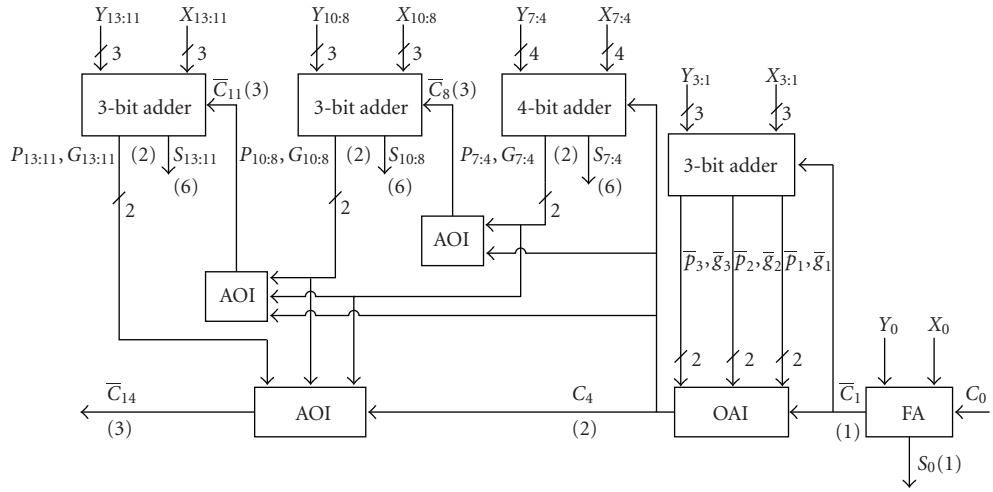

Figure 3: Detailed view of the first three blocks of 32-bit adder.

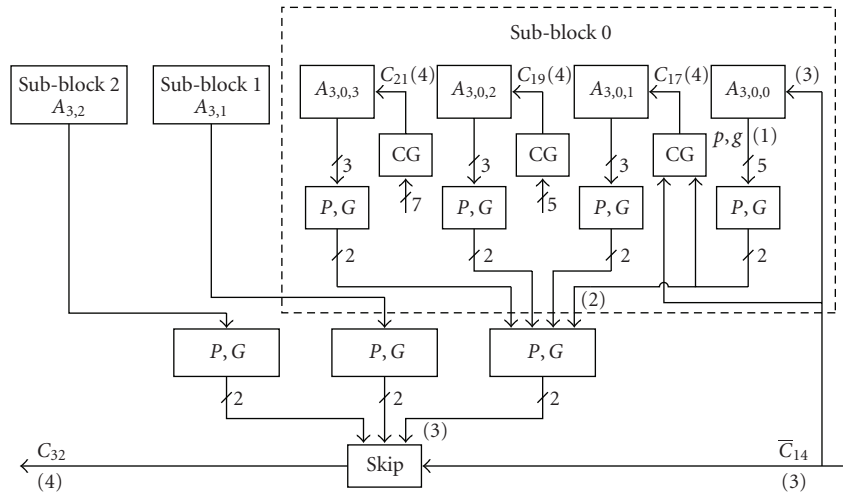

Figure 4: Block-3 of 32-bit adder with an expanded view of sub-block 0.

RCAs ($A_{2,1}$ and $A_{2,2}$) are each 3 bits. The number of RCAs in A_2 is limited to 3 due to the fan-in restriction of 7 on the skip logic. Each RCA in block A_2 also represents a subblock of A_2. The carry out C_{14} from the skip logic is implemented using AOI logic as

$$\overline{C_{14}} = \overline{G_{13:11} + P_{13:11} \cdot G_{10:8} + P_{13:11} \cdot P_{10:8} \cdot G_{7:4} + P_{13:11} \cdot P_{10:8} \cdot P_{7:4} \cdot C_4}. \quad (12)$$

A detailed block diagram of the first three blocks of the 32-bit adder (an expanded view of Figure 2) is shown in Figure 3. The three blocks together form a 14-bit adder.

Next let us consider the final block A_3 of the 32-bit adder. Block A_3 is divided into a number of subblocks. The maximum number of subblocks is again limited to 3 due to the fan-in restrictions on the skip logic. A block diagram of A_3 with an expanded view of subblock $0(A_{3,0})$ is shown in Figure 4. The subblock 0 is further divided into RCAs. The number of inputs to the CG logic increases, successively, by 2 for each RCA and is limited to a maximum of 7 in any subblock. Hence, the number of RCAs in any subblock is limited either by the number of inputs to the CG block or by the number of inputs to the (P, G) block. Therefore, subblock 0 can accommodate 4 RCAs. The carry input to the skip logic, as well as, to the first RCA $(A_{3,0,0})$ arrives in 3 time units. The propagate and generate signals (p and g) from each RCA will be available with a delay of 1 time unit. This implies that we can have two levels of (P, G) logic inside the block while satisfying the time delay constraints. Using (9), the width of the first RCA $(A_{3,0,0})$ is 3 bits, and the widths of the remaining RCAs are 2 bits each. Hence, the total width of subblock $0(A_{3,0})$ is 9 bits.

Figure 5 shows block A_3 with an expanded view of subblock $1(A_{3,1})$. The number of RCAs in $A_{3,1}$ is limited to 3 due to the condition stated earlier. The carry input C_{23} to the first RCA $(A_{3,1,0})$ of this subblock is given by

$$C_{23} = \overline{\overline{G_{22:14}}\left(\overline{P_{22:14}} + \overline{C_{14}}\right)}. \quad (13)$$

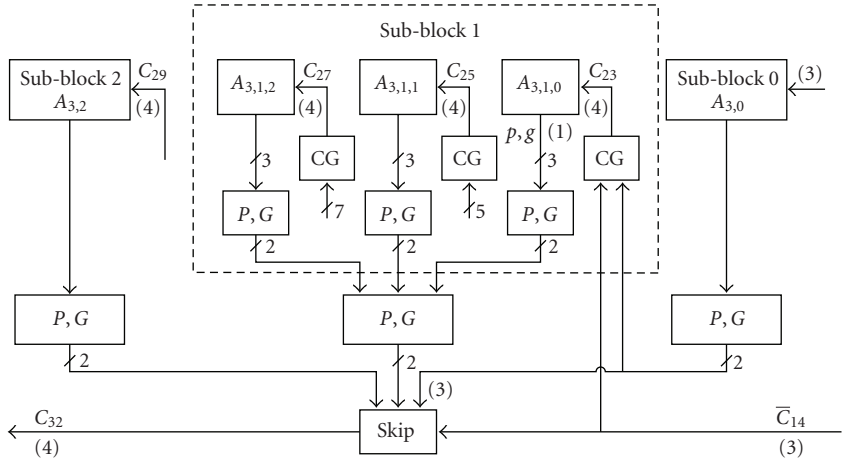

Figure 5: Block-3 of 32-bit adder with an expanded view of sub-block 1.

With an AOI logic implementation, C_{23} will be available in 4 time units, thereby limiting the length of the first RCA to 2 bits. The carry inputs C_{25} and C_{27} to the remaining RCAs in subblock $1(A_{3,1})$ are also available in 4 time units. Thus, the maximum width of subblock $A_{3,1}$ is 6 bits. The carry input C_{29} to the final subblock $2(A_{3,2})$ is given by

$$C_{29} = \overline{\overline{G_{28:23}}\left(\overline{P_{28:23}} + \overline{G_{22:14}}\right)\left(\overline{P_{28:23}} + \overline{P_{22:14}} + \overline{C_{14}}\right)}. \quad (14)$$

The maximum width of subblock $A_{3,2}$ can be calculated as 4 bits. This subblock can accommodate only 2 RCAs due to the fan-in limits of the CG blocks. Hence, the total width of block A_3 is 19 bits. By combining the 4 blocks A_0, A_1, A_2, and A_3 a 33-bit adder can be implemented. The width of subblock $A_{3,2}$ can be shortened to 3-bits for a 32-bit adder. The carry out C_{32} from the skip logic is given by

$$C_{32} = \overline{G_{31:29}(\alpha)(\beta)(\gamma)}, \quad (15)$$

where, $(\alpha) = \left(\overline{P_{31:29}} + \overline{G_{28:23}}\right)$, $(\beta) = \left(\overline{P_{31:29}} + \overline{P_{28:23}} + \overline{G_{22:14}}\right)$, $(\gamma) = \left(\overline{P_{31:29}} + \overline{P_{28:23}} + \overline{P_{22:14}} + \overline{C_{14}}\right)$.

An OAI logic implementation generates C_{32} in 4 time units. A detailed block diagram of block A_3 is shown in Figure 6. The final breakdown of the 32-bit adder into 4 blocks is shown in Figure 7. A reduction in hardware can be achieved by moving subblock $A_{3,2}$ from block A_3 and placing it as another block A_4. This will eliminate 1 carry generate logic (OAI) and $1(P, G)$ logic.

Although our adder has already achieved the 32-bit requirement, we still have room to extend the width further, while keeping the target delay the same. The schemes for the 5th and 6th blocks are shown in Figure 8. The fifth block A_4 is divided into three subblocks. The subblocks $(A_{4,0}, A_{4,1}, \text{and } A_{4,2})$ have the same structure as block A_3. Since the carry fed into the 5th block has 4 unit delays, the maximum width of the first RCA will be 2 bits. The remaining RCAs will be 1 bit each. Thus, the maximum width for the fifth block will be 20 bits. The first subblock $A_{4,0}$ (11 bits) is divided into subblocks of 5, 3, 2, and 1 bit. The subblock $A_{4,1}$ (6 bits) is divided into 3 subblocks of 3, 2, and 1 bit. Similarly, the final subblock $A_{4,2}$ (3 bits) is divided into subblocks of 2 and 1 bit. The first 5-bit subblock $(A_{4,0,0})$ consists of a 2-bit RCA and 3 individual full adders. Individual full adder cells form all other subblocks. The 6th block A_5 is a single bit full adder. Thus, the total width of the adder becomes 54.

Based on the adder design procedure, we can derive a formula for calculating the maximum number of full adders in every block. The following notations are used in the derivation. T Target delay of the n-bit adder in time units, $N(i)$ The number of RCAs in block i, $W(i, j)$ The width of RCA "j" in block i.

For any block i $(i \geq 2)$, the number of RCAs is defined by a recursive function $N(i)$. The recursive function is not valid for the blocks A_0 and A_1, and the values for $N(0)$ and $N(1)$ when used in the recursive function are assumed to be zero

$$N(i) = \sum_{1}^{i} i + N(i-1), \qquad N(0) = N(1) = 0. \quad (16)$$

The width of an RCA is defined in terms of the target delay. The width $W(i, j)$ of the RCA"j" in any block"i" is defined as

$$W(i,j) = \begin{cases} \min\ (4, T-i), & \text{for } j = 0, \\ \min\ (4, T-i-1), & \text{for } 1 \leq j \leq N(i)-1, \end{cases} \quad (17)$$

where $\min\ (a, b)$ is the minimum value among a and b.

The carry input to the first RCA of the block can be obtained directly from the previous carry-skip stage. Hence, the calculation of width for the first block is done differently from the others.

The maximum number of full adders $FA(i)$ in block i is given by

$$\begin{aligned} FA(i) &= \sum_{j=0}^{N(i)-1} W(i,j) \\ &= \sum_{j=1}^{N(i)-1} W(i,j) + W(i,0). \end{aligned} \quad (18)$$

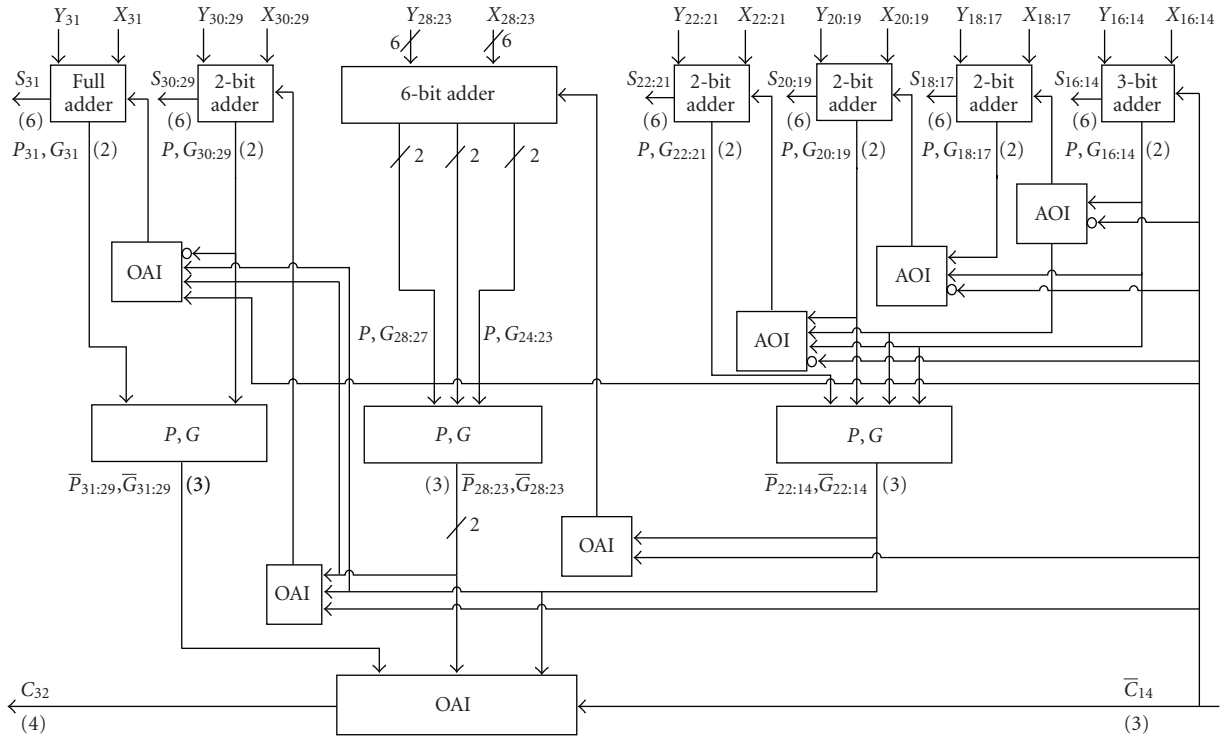

Figure 6: Detailed scheme for block A_3.

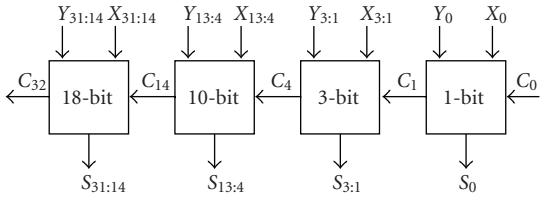

Figure 7: The proposed 32-bit adder.

Table 1: Maximum size of adders.

Target delay (T)-time units	4	5	6	7	8
Adder size (n)-bits	9	22	54	119	237

Table 1 lists the maximum adder size for a given target delay using our design procedure.

4. DESIGN OF BASIC CMOS CELLS

A few basic CMOS cells are used for the design of the adder stage. They are: AOI, OAI, and FA cells. Three different cells are used for AOI and OAI (3-input, 5-input, and 7-input). These cells are labeled as AOIn and OAIn, where n refers to the number of inputs to the cell. The 3-input and 5-input cells are implemented in a straightforward manner, and are given by the following Boolean expressions:

$$\text{AOI3: Out} = \overline{A + B \cdot C}, \tag{19}$$

where A, B, and C are the inputs for the gate. The 3-input OAI is expressed as

$$\text{OAI3: Out} = \overline{A \cdot (B + C)}. \tag{20}$$

The expressions for 5-input AOI and OAI are given as

$$\begin{aligned} \text{AOI5: Out} &= \overline{A + B \cdot (C + D \cdot E)}, \\ \text{OAI5: Out} &= \overline{A \cdot (B + C \cdot (D + E))}, \end{aligned} \tag{21}$$

where A, B, C, D, and E are the inputs to the cells. When the 7-input AOI and OAI cells are implemented in the above manner, the delay is prohibitive and hence we decided to implement them as a cascade connection of a number of smaller modules. Their corresponding Boolean expressions are given by

$$\begin{aligned} \text{AOI7: Out} &= \overline{A + B \cdot C + \overline{\overline{B \cdot D} + \overline{(E + F \cdot G)}}}, \\ \text{OAI7: Out} &= \overline{A \cdot (B + C) \cdot \left[\overline{(B + D)} \cdot \overline{(E \cdot (F + G))}\right]}, \end{aligned} \tag{22}$$

where A, B, C, D, E, F and G are the inputs to the cells. Since we reduce the stack height of the transistors connected in series from 4 to 3, the 7-input AOI and OAI cells will be speeded up and the propagation delay will be almost the same as the 5-input AOI and OAI. The full adder cell used in our design is the low-energy CMOS adder cell presented in [10].

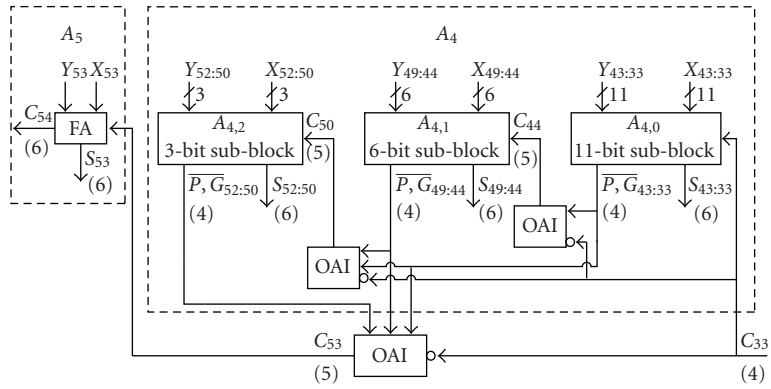

Figure 8: The schemes for the fifth and sixth blocks.

Table 2: Cell Characteristics.

	Delay (ns)	Average power (mw)
AOI3	0.267	0.128
OAI3	0.264	0.127
AOI5	0.527	0.15
OAI5	0.541	0.16
AOI7	0.578	0.33
OAI7	0.579	0.30
FA	0.604	0.34

Table 3: Adder comparison for delay, power, and power-delay product.

	Delay (ns)	Power (mw)	PDP (pJ)
32-bit adder (Chirca)	4.15	4.68	19.4
32-bit adder (Gayles)	4.39	3.26	14.3
Our adder (32-bit)	3.4	4.2	14.28
Our adder (54-bit)	4.3	8.6	36.98

5. SIMULATION

The adder was implemented using Tanner tools pro 11.03. L-edit was used to generate the layout and T-spice was used for performing the simulation. The generic 0.25 μm CMOS technology was used with 3.3 volts supply voltage. The different CMOS cells (AOI, OAI, and FA) were simulated for worst-case delays and the delays are tabulated in Table 2. From Table 2, it may be noted that the 5 and 7-input cell delays are comparable to that of the FA, while the 3-input cells have a much less delay. The average power was measured by feeding 10,000 random vectors at a frequency of 500 MHz and is also shown in Table 2.

For comparison purposes, we selected two other types of adders. They are (i) 32-bit carry skip-adder proposed in [4] and (ii) 32-bit multilevel carry-skip adder proposed in [11]. The first one is referred here as Chirca adder and the second one is referred as Gayles adder. These adders were compared with our 32-bit adder by measuring the critical path delays. To get a more realistic estimation of the delays involved, we laid out the complete 32-bit adder stages and performed TSPICE simulation. The simulation was carried out at a frequency of 100 MHz. The simulation results are shown in Table 3. These results show that our 32-bit adder has the minimum delay of 3.4 nanoseconds while Gayles adder exhibited a maximum delay of 4.39 nanoseconds. The Chirca adder had a delay of 4.15 nanoseconds. Thus, our design has a speedup of 18% and 22% compared to those of Chirca and Gayles adders, respectively. Our 32-bit adder was then extended to a 54-bit adder with marginal delay increase, and these simulation results are also included in Table 3. Even this 54-bit adder is found to be faster than the 32-bit Gayles adder.

The power consumption showed a marginal increase of power for our adder compared to Gayles adder while outperforming Chirca adder. Overall, our 32-bit adder achieved the lowest power-delay product.

6. CONCLUSIONS

In this paper, we presented a new 32-bit adder using carry-skip logic. The adder was implemented by dividing the adder into several blocks. The size of each block is limited by the delay of the carry-in signal and the final target delay. An algorithm is used to calculate the maximum size of the adder satisfying the target delay. The delay of a full adder is used as the unit of measurement in our analysis. The adder has been implemented by generating the layout with Generic 0.25 μm CMOS technology. The TSPICE simulations carried out at a frequency of 100 MHz and supply voltage of 3.3 V showed a critical path delay of 3.4 nanoseconds. The comparison results show that our adder is faster than Chirca and Gayles carry-skip adders. Overall our proposed adder is 18% and 22% faster compared to the Chirca and Gayles adders, respectively. Furthermore, a 54-bit adder implemented using our approach can operate almost at the same speed as a 32-bit Chirca adder or Gayles adder. Even though our adder has a marginal increase in power consumption compared to the Gayles adder, overall, we achieved the lowest power-delay product.

REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*, A. K. Peters, Natick, Mass, USA, 2nd edition, 2002.

[2] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford University Press, Oxford, UK, 2000.

[3] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 689–702, 1996.

[4] K. Chirca, M. Schulte, J. Glossner, et al., "A static low-power, high-performance 32-bit carry skip adder," in *Proceedings of the EUROMICRO Symposium on Digital System Design (DSD '04)*, pp. 615–619, Rennes, France, August-September 2004.

[5] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Transactions on Computers*, vol. 31, no. 3, pp. 260–264, 1982.

[6] B. W. Y. Wei, C. D. Thompson, and Y. F. Chen, "Time optimal design of a CMOS adder," in *Proceedings of the 19th Annual Asilomar Conference on Circuits, Systems, and Computers*, pp. 186–191, Pacific Grove, CA, USA, November 1985.

[7] T. P. Kelliher, R. M. Owens, M. J. Irwin, and T.-T. Hwang, "ELM-A fast addition algorithm discovered by a program," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1181–1184, 1992.

[8] V. Kantabutra, "Designing optimum one-level carry-skip adders," *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 759–764, 1993.

[9] V. Kantabutra, "Accelerated two-level carry-skip adders-a type of very fast adders," *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1389–1393, 1993.

[10] S. Goel, S. Gollamudi, A. Kumar, and M. Bayoumi, "On the design of low-energy hybrid CMOS 1-bit full adder cells," in *Proceedings of the 47th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '04)*, vol. 2, pp. 209–212, Hiroshima, Japan, July 2004.

[11] E. Gayles, R. M. Owens, and M. J. Irwin, "Low power circuit techniques for fast carry-skip adders," in *Proceedings of the 39th IEEE Midwest Symposium on Circuits and Systems*, vol. 1, pp. 87–90, Ames, Iowa, USA, August 1996.