A 16-bit High-Speed Low-Power Hybrid Adder

Assem Hussein*, Vincent Gaudet*, Hassan Mostafa† and Mohamed Elmasry*
*Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada N2L 3G1
Email: assem.hussein@uwaterloo.ca, vcgaudet@uwaterloo.ca, elmasry@uwaterloo.ca
†Dept. of Electronics and Electrical Communications Engineering, Cairo University, Cairo, Egypt
Email: hmostafa@uwaterloo.ca

***Abstract*—This paper presents the architecture of a hybrid adder based on the combination of the carry-lookahead and carry-select adder architectures. A 16-bit adder is designed as a case study to show the efficiency of the proposed architecture. This adder is implemented in a 0.18 μm CMOS technology using domino logic for most of the sub-circuits. The simulation results show that the 16-bit hybrid adder achieves a worst-case delay of 876.7 ps and has an average power consumption of 787.2 μW at 125°C for a throughput of 100 Mega operations per second in the slow-slow corner. The paper also demonstrates the efficiency of this architecture for higher number of bits.**

I. Introduction

One of the most substantial research areas in the design of VLSI systems is the design of power-efficient, high-speed and small-area datapath logic systems. Adders are basic logic elements that play a critical role in design and performance of different operations. Thus, there exists a high demand for high-speed and low-cost adders. Classic high-speed adders include carry-look ahead adders (CLAs) [1], carry-skip adders, carry-select adders (CSLAs) [2], etc. In the last 20 years, a lot of research was done to improve the speed and power consumption of these architectures. Hybrid adders are adders that use a combination of two or more of the previously described styles.

Carry-select adders and carry-lookahead adders are two of the fastest adder architectures [3]. Carry-lookahead adders generate the carries in parallel while carry-select adders implements the carry and sum generation for different carry scenarios in a parallel configuration to decrease the propagation time. Generally, in digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. A common approach to the design of hybrid adders is to choose one method for carry propagation and another method for sum calculation. In this hybrid architecture, a carry-select adder is chosen for the sum calculation and a carry-lookahead adder for the carry propagation. This idea was first presented in [4]. However, this adder has some important improvements mainly in the carry-select adder. Conventional carry-select adders are basically formed of blocks of ripple-carry adder pairs divided among grouped bits of the required number of bits. The CSLA is not area-efficient because it uses multiple pairs of Ripple Carry Adders (RCA). Instead, in this hybrid adder, the carry-select part uses a fast circuit like the Manchester carry chain shown in Fig. 1 to compute the carries for required blocks. Moreover, the group carry-in signal is not generated in a propagating manner as in the non-hybrid adder CSLA. Instead, they are generated by a carry-lookahead tree. The carry-lookahead adder is based on a spanning tree that computes the grouped carry propagates and carry generates. The group propagates and generates are implemented in Manchester carry chains. This hybrid adder is based on a regular architecture and is very efficient for generally large number of bits. As proof of concept, a 16-bit adder is implemented and simulated using this architecture in dynamic logic.

The rest of the paper is organized as follows. Section II describes the basic structure of the hybrid adder and the combined carry-lookahead/carry-select architecture. Section III discusses the circuit implementation techniques of the 16-bit adder. Section IV shows the simulation results. Finally, in Section V, a conclusion is presented along with the future work that can be done to improve the performance of the adder.

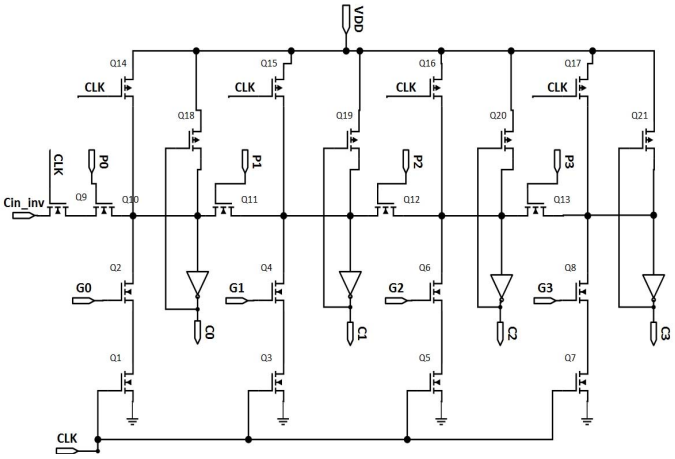

Fig. 1. Manchester Carry Chain.

II. Structure of the Hybrid Adder

The proposed hybrid adder is formed of a combination of a carry-select adder and a carry-lookahead adder. Each of the two adders is described alone and then the hybrid adder is presented afterwards. Before that, the two signals that will be used extensively afterwards have to be defined, the propagate (P) and generate (G) signals. 'P' and 'G' are initial signals that are used in carry and sum calculations. For adding two n-bit binary numbers, the propagate signal $P_i = A_i \oplus B_i$ and the generate signal $G_i = A_i B_i$. Then for the i^{th} bit, the carry

propagated to the next bit $C_i = G_i + (P_i \mathrm{C}_{i-1})$ and the resulted sum at the i^{th} bit position $S_i = P_i \oplus \mathrm{C}_{i-1}$.

A. Carry-select Adder

The architecture of the conventional carry-select adder is based upon generating all the propagate and generate bits as an initial step. Then the bits are divided into groups or blocks such that each group contains a number of bits. From each block, the target is to get the sum of the two input n-bits, where $S_i = P_i \oplus \mathrm{C}_{i-1}$. Since the P signals are already generated before, it is required to get the carries, $C_i = G_i + (P_i.\mathrm{C}_{i-1})$. Inside each block, two ripple carry adders are used in parallel. The first one assumes the input carry to be '0' and the other assumes that it is '1'. As a result, no need to wait in idle for the previous carry signal. After that, a multiplexer chooses between the result of the two sums; the sum assuming the input carry equals to zero and the sum assuming the input carry equals to one based on the carry of the previous block.

The CSLA uses several pairs of Ripple Carry Adders (RCA) to generate the initial sum and carry. Hence, the CSLA is not area-efficient. To improve the performance of the carry-select adders, Manchester carry chains are used to generate the carries for each block for the two cases; '0' carry input case and '1' carry input case. The internal design of the Manchester carry chains, shown in Fig. 1 is discussed in details in the implementation section. Manchester carry chains are capable of producing the carries of the next bits from the propagate and generate signals with no propagation of carries as in the ripple carry adder. Then, multiplexers are used to choose between the output carries in case of '0' input carry and '1' input carry instead of the sum signals. The output carry bits are then used to calculate the sums where $S_i = P_i \oplus \mathrm{C}_{i-1}$. There are two types of carry-select adders: linear carry-select adders and square root carry-select adders [5]. The square root carry-select adders have better overall speed improvement by a square root factor. This is because there is a waiting time in the carry-select adder because of the propagating carry between the blocks. In the square root adder, this waiting time is cancelled by assigning more bits to the later blocks than the starting blocks to cancel the waiting time for each block. However there are two problems with the square root carry-select adder. The first is that the design is no more regular, and the second is that the Manchester carry chains will not perform with larger number of bits because of the long chains of transistors. This is discussed in detail in Section III.

B. Carry-lookahead Manchester Chains

The carry-lookahead adder (CLA) is known to be one of the fastest adders [3]. It improves the speed in a logarithmic way by reducing the amount of time required to determine carry bits. The carry-lookahead adder calculates one or more carry bits before the sum, which decreases the time required to calculate the result of higher order bits. Carry-lookahead logic uses the propagate and generate signals to generate the carry signals required for all the bits before doing the sum operation. This is done on several levels where the concept of group propagate and generate signals is introduced. The group propagate and generate concepts depends on grouping the propagate and generate signals to form higher order signals that can produce the carry for a high order bit directly. This is done on several levels. For example, for a 4-bit adder,

$C_0 = G_0 + P_0 C_{in}$,

$C_1 = G_1 + P_1 C_0 = G_1 + P_1(G_0 + P_0 C_i n) = G_1 + P_1 G_0 + P_1 P_0 C_{in}$,

$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$,

$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$.

The group propagate and generate signals are presented as follows for a 4-bit adder:

$C_0 = G_0 + P_0 C_{in}$, $C_1 = \mathrm{G}_{1:0} + P_{1:0}\mathrm{C}_{in}$, $C_2 = \mathrm{G}_{2:0} + P_{2:0}\mathrm{C}_{in}$, $C_3 = \mathrm{G}_{3:0} + P_{3:0}\mathrm{C}_{in}$.

Where, $\mathrm{G}_{1:0} = \mathrm{G}_1 + G_0 P_1$, $\mathrm{G}_{2:0} = \mathrm{G}_2 + G_1 P_2 + G_0 P_1 P 2$, etc. In addition, higher order propagate and generate signals can be formed of lower order one, for example, $\mathrm{G}_{3:0} = (G_{3:2}, P_{3:2}).(G_{2:0}, P_{2:0})$.

In general, the i^{th} carry output is function in C_{in} rather than its preceding carry signal. Kogge-Stone [5] and Brent Kung [6] adders are two of the most famous fast adder architectures based on the carry-lookahead concept. Fig. 2 shows the circuit that calculates the grouped propagate and generate signals fast. It is a modified version of the Manchester carry chain circuit proposed before.

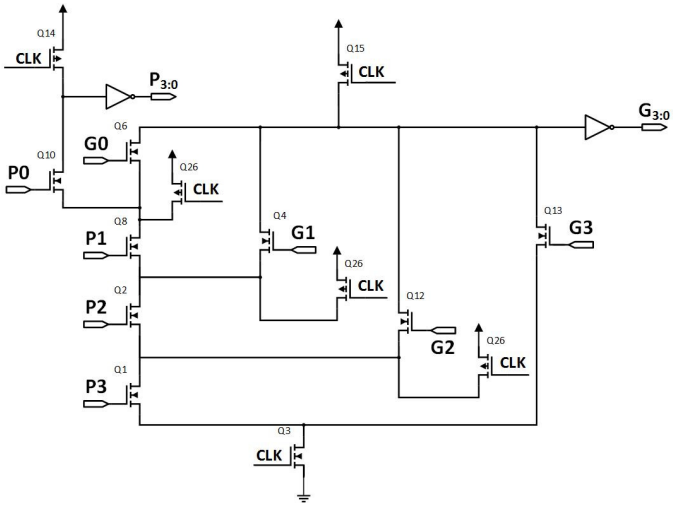

Fig. 2. Carry-lookahead Manchester Carry Chain.

C. The Hybrid adder

The hybrid adder used is shown in Fig. 3. The idea behind this adder is to combine the merits of the carry-select adder and the carry-lookahead adder. Thus, the main disadvantage of the carry-select adder is the need to propagate the carry through the blocks after getting it from the first block. The carry-lookahead adder can then be used to provide the carry signals to the carry-select blocks instead of propagating the carry between them. That's why the square root carry-select

adder is not having a major advantage here because the propagation time is already much reduced. The hybrid adder will now have two major parts. The first, is carry-lookahead Manchester chains responsible for providing only some of the carry signals. The second is a carry-select adder, that its block will be controlled by the carry signals of the Manchester chains. There are several ways to decide the size of each of them. The proposed 16-bit adder depends on a Manchester carry chain of 4 bits implemented in dynamic logic. The 4-bit restriction is just a chosen maximum, because if more bits are used, the circuit performance will decrease as the circuit will have too many transistors in series forming a long chain that has many performance issues. The Manchester chain circuit is used in the carry-select adder. Moreover, a modified version of the Manchester carry chain will be used on the carry-lookahead. As a result, the used blocks are of 4-bit max size in both of them. Based on that, the carry-select adder will be formed of four 4-bit carry-select blocks. If we decided to generate the carry for each one of them using the modified Manchester carry chain, we will need two levels of this MCC. As a result there will be some waste of time assuming that all the MCCs have the same delay. This is because that the carry-select adder will do nothing for a whole MCC delay waiting for the carries from the carry-lookahead adders. Consequently, the best option is to divide the circuit into two 8-bit carry-select adders, each composed of two 4-bit carry-select adder. The carry signal C8 will then be generated using the carry-lookahead Manchester chains.

This architecture is highly efficient with large number of bits, where parallelism really improves the design. However, it still improves performance for a 16-bit adder compared to stand-alone carry-select or carry-lookahead adders.

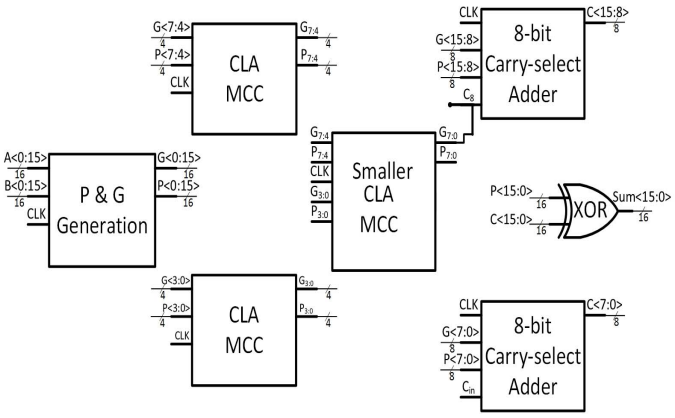

Fig. 3. Proposed Hybrid Adder.

III. Implementation

As shown in the previous section, the adder is composed of some blocks; the propagate and generate signals generation block, the carry-lookahead block and carry-select block. The main target of this adder is to achieve high speed low power consumption computations. That is why all the blocks were implemented in dynamic logic except for the multiplexers, which were implemented in transmission-gate logic. This is because multiplexers don't affect the critical path and they are very simple when they are implemented in transmission gate logic. However, there is still a problem with cascading same type dynamic CMOS circuits. That is why we used domino logic. This eliminates the problem by adding an inverter between the any dynamic logic circuits.

The rest of this section will be discussing the implementation techniques and sizing of each block.

A. Carry Propagate and Generate Block

This is the first block in the system. It is responsible for generating the carry propagate and generate signals. The carry generate signal $G_i = A_i.B_i$ which is basically a 2-input AND gate. The AND gate is implemented as a 2-input dynamic logic NAND gate cascaded by an inverter. The second signal is the carry propagate signal where $P_i = A_i \oplus B_i$. This was implemented as a dynamic logic XNOR gate cascaded by an inverter to make sure all our circuit is in domino logic.

B. Carry-lookahead Block

The carry-lookahead block is basically a modified carry-lookahead Manchester chain shown in Fig. 2. The target of this block is to generate C_8. Consequently, two MCC blocks are used. The first outputs $G_{3:0}$/$P_{3:0}$ and the second outputs $G_{7:4}$/$P_{7:4}$. Then ($G_{7:0}$,$P_{7:0}$)=($G_{7:4}$,$P_{7:4}$).($G_{3:0}$,$P_{3:0}$) and C_8=$G_{7:0}$+($P_{7:0}C_0$). Since C_0=0, (no carry signal is input from outside to the adder), then C_8 =$G_{7:0}$ This shows that no need to implement $P_{7:0}$ and as a result $P_{3:0}$ also. This is considered a special case in the architecture that will disappear for high-order adders. The sizing discussed here is for the general case modified MCC and the special case will just have the same sizing technique but with less number of transistors. The modified carry-lookahead Manchester chain circuit shows that the critical path will be a stack of five transistor. The ordering of the inputs will be such that late arriving signals will be assigned to transistors higher in the stack. Transistors lower in the stack will be given slightly higher width.

C. Carry-select Adder Block

The carry-select adder is composed of two 8-bit carry-select adders and multiplexers. Each composed of a two 4-bit carry-select adder. Each 4-bit carry-select adder is composed of the Manchester carry chain shown in Fig. 1. The critical path is defined by the stack of the five transistors in the Manchester carry chain. The remaining blocks are multiplexers, which are implemented in transmission gate logic. This is because transmission gate logic is very suitable for multiplexer design and also it is not affecting the critical path. At last, the final XOR gate which is used for sum calculation, is basically the same as the one proposed in the beginning but a bit faster.

IV. Simulation Results

Simulations were carried out to test the adder in a 0.18 μm technology. All adder inputs are buffered and all the outputs

TABLE I
DELAY, POWER CONSUMPTION AND POWER DELAY PRODUCT FOR DIFFERENT INPUT CASES

Input Case (in HEX)	Delay (ps) @SS 125°C	Delay (ps) @TT 27°C	Power @SS 125°C (uWatt)	Power @TT 27°C (uWatt)	Power delay product @SS 125°C (fJoules)	Power delay product @TT 27°C (fJoules)
A=FFFF, B=0001	0	0	833.76	824.6	0	0
A=FFFF, B=FFFF	876.7	587.4	787.2	787.1	690.2	462.3
A=AAAA, B=AAAA	828.4	559.8	448.8	448.9	371.8	251.3

are driving a load of 10fF. Moreover, this adder is implemented in domino logic. As a result, the system is operating on a clock which defines the throughput not the delay. In addition, as described before, our design is based upon regularity, so actually for an optimal design, this adder can achieve the same delay with nearly 3/4 of the simulated power consumption by removing some additional blocks as the carry-select adder of the first 4 bits. However, the design will not be as regular as before and we will not be able to estimate the results for higher order hybrid adders of the same architecture. For power calculations, the power depends on the clock of the system because of the dynamic logic used. The faster the clock used, the more power it burns, but the higher throughput achieved. For these simulations, a clock of 100 MHz is used.

In addition, based on the proposed architecture, the worst-case delay will always occur at the 16-th bit S_{15}, if this bit is equal to zero the previous bit is considered which is S_{14}. This is because of the carry-select architecture where there is a limited carry propagation inside. All the worst case delays, power consumptions and power delay products are tabulated in Table 1. Two input vectors that cover the worst case delay are discussed below.

The first input vectors are A="FFFFF" and B="0001". The output S should be equal to zero for all bits. This should introduce a worst-case delay. However, in this adder, this a best-case delay where there is no delay at all. This is because, all the outputs are pre-charged to zero, thus improving our worst-case delay. This is one of the merits of using dynamic logic. The logic was designed in this way to improve this particular worst-case delay by inverting the final dynamic logic circuit. However, there is still high power consumption, because the internal blocks still operate and switch. This is considered the worst-case in power propagation where power propagates through all the blocks.

The second input vectors are A="FFFFF" and B="FFFFF". Hence, all the outputs should go to one except the LSB S_0. The power in the previous case is higher than this case although the outputs don't switch in the previous case because there are some blocks that may switch in the previous case and not in this case. For example the generate signals switches in the previous case but do not switch in this case, etc.

TABLE II
COMPARISON BETWEEN THE PERFORMANCE OF SEVERAL 16-BIT ADDERS

Adder	CMOS Technology	Number of bits	Delay (ns)	Power (uWatt)	Power delay product (pJoules)
This work	0.18um	16 bits	0.8767	787.2	0.6902
16-bit Regular CSLA [7]	0.18um	16 bits	2.775	527.5	1.4638
16-bit modified CSLA [7]	0.18um	16 bits	3.048	471.8	1.438

Fig. 4 shows the power delay product for this case vs. the supply voltage. The curve shows that there exists an optimal point at V_{DD}≈1.4V that optimizes the power delay product.

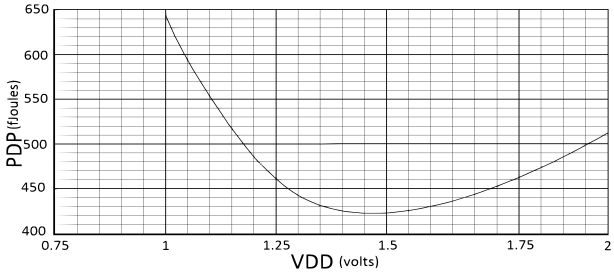

Fig. 4. Power Delay Product Vs V_{DD}.

Table II shows a a comparison between our proposed hybrid adder and other 16-bit carry-select adders. For the proposed hybrid adder, the reported numbers are of the second test case for the SS corner at 125°C.

V. CONCLUSION

A hybrid architecture composed of CLA and CLSA is presented in this paper. A 16-bit adder is implemented as a proof of concept. The simulation results shows that the 16-bit hybrid adder achieves a worst-case delay of 876.7 ps and has an average power consumption of 787.2 μW for a throughput of 100 Mega operations per second at 125°C in the SS corner. For future work, a 64-bit adder should be done using this architecture. It will show the merit of this design more and more. In addition, this system can be pipelined since it is implemented based on dynamic logic.

REFERENCES

[1] A. Welnberger and J. Smith, "A logic for high-speed addition," 1958.
[2] O. Bedrij, "Carry-select adder," *IRE Transactions on Electronic Computers*, no. 3, pp. 340–346, 1962.
[3] R. Uma, V. Vijayan, M. Mohanapriya, and S. Paul, "Area, delay and power comparison of adder topologies," *International Journal of VLSI design & Communication Systems (VLSICS) Vol*, vol. 3, p. 161, 2012.
[4] T. Lynch and E. Swartzlander, "A spanning tree carry lookahead adder," *Computers, IEEE Transactions on*, vol. 41, pp. 931–939, Aug 1992.
[5] J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.
[6] R. P. Brent and H. Kung, "A regular layout for parallel adders," *Computers, IEEE Transactions on*, vol. C-31, pp. 260–264, March 1982.
[7] B. Ramkumar and H. M. Kittur, "Low-power and area-efficient carry select adder," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 20, no. 2, pp. 371–375, 2012.