"VLSI circuit design - Study on high speed multi-bit adder architectures"

In the Very Large Scale Integration (VLSI), high-performance digital signal processing (DSP) systems require both efficient methods for processing data as well as for building architectures that support faster operation, and low power dissipation to enable data to be processed in real-time [1]. As the demand for higher performance processors grows, there is a continuing need to improve the performance of arithmetic units and to increase their functionality. Furthermore, the selection of the initial topology expected to yield a desired performance in the allotted power budget is the most important step taken.

In VLSI design process, addition is the pivotal operation. More specifically addition is the most commonly used arithmetic operation and also the speed-limiting element to make faster VLSI processors. The design of high-speed, low-power and area efficient binary adders always receives a great deal of attention. Hence, developing efficient adder architecture (from the standpoint of timing, area, and power) is crucial to improving the efficiency of the design.

Adder architectures:

The arrangement of the prefix network gives rise to various families of adders. There exist various architectures for the carry calculation part [2]. Tradeoffs in these architecture involves
• Area of the Adder
• Its depth
• The fan out of the nodes
• The overall wiring network.

Parallel Prefix Adders are the most important elements used in arithmetic operations of many processors. Their regular structure and fast performance makes them particularly attractive for VLSI implementation. Over the years several classical topologies for parallel prefix adder have been proposed, since the structure of the prefix network determines the type of the prefix adder. The classical parallel prefix adder structures that have been introduced optimize the logic depth size area, fanout and interconnect count of the logic circuits. The main difference between the full adder and parallel prefix adder is that in the full adder, summation and carry calculation is done in the same, one bit block but in the prefix adder, summation and carry calculation are separated from the bit block and all calculation is treated as a whole in the carry graph. The carry graph uses the prefix circuit and this is the origin of the name, "Prefix Adder". Parallel Prefix Adders have been established as the most efficient technique for speeding up binary addition as it includes the implementation of logic functions which determine whether groups of bits will generate or propagate a carry.

A Parallel Prefix Addition is generally a three step process [3] [4]: The first stage of the computation is called as pre-processing, involves the creation of generate and propagate signals for the input operand bits. The second stage in the prefix addition is termed as prefix computation and involves the

generation of carry signals. Namely, this stage is responsible for creation of group generate and groups propagate signals. The final stage in the prefix addition is termed as post-processing and involves the generation of sum bits with the propagate signals of the operand bits and the preceding stage carry bit.

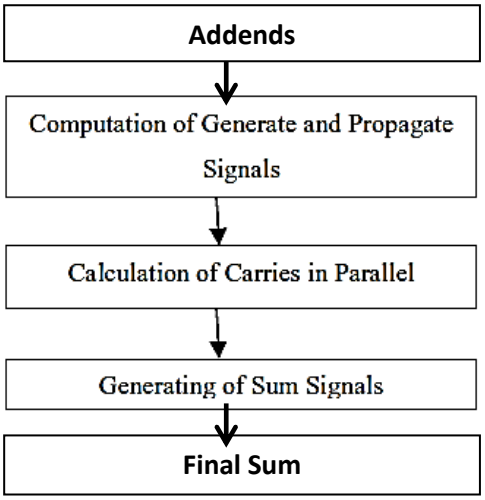


Fig. 1 Three basic step process generally involved in the construction of Parallel Prefix Adders.

The first stage and last stage are intrinsically fast because they involve only simple operations on signals local to each bit position. The intermediate stage embodies long distance propagation of carries, so the performance of the adder depends on the intermediate stage. This is the reason that prefix adders are distinguished by the structure of their intermediate stages.

In a prefix adder this part is constructed of nodes which perform the prefix operation '•'. In logical terms, the prefix operator consists of an AND gate and an AND-OR gate. The carry into any bit position can be computed as a chain of prefix operations. A parallel prefix adder computes multiple sub-terms in parallel by exploiting the associativity property.

It is worth mentioning that Parallel Prefix Adders are variations of the Carry Lookahead adder (CLA) [2]. The difference between a Carry Lookahead adder and a Parallel Prefix Adder is found in the second stage of the "Addition three step process", which is responsible for the generation of the carry signals of the binary addition. A carry-lookahead adder system estimates whether a carry will be generated before it actually computes the sum. There are multiple schemes of doing this, so this is the reason why there is no just "one" circuit that constitutes a look-ahead adder. Carry Look Ahead adders are considered fast, but they required more area comparing to other adder architectures.

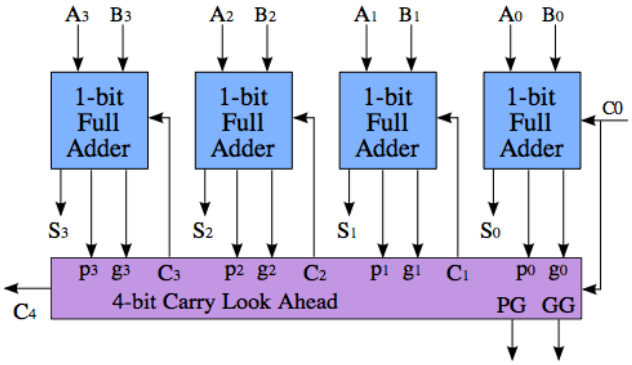


Fig. 2 Carry Lookahead adder (CLA)

Among the plethora of adder architectures known in the literature, when high performances are mandatory, parallel-prefix trees are generally preferable. Optimizing a parallel-prefix tree architecture and its transistor-level implementation for a specific design is not trivial since the designer has to choose the radix-of the carry tree, namely the number of carries grouped in each step of the computation, the tree architecture and the logic style. All these choices are crucial for both speed and power. In fact, higher radices determine a lower number of stages needed in the tree to compute the output carry signals, but they require more complex gates.

For example at a given radix r, dense architectures, such as the Kogge-Stone tree, reach the minimum logic depth, but they require a large number of gates and consume a large amount of power. On the contrary, other sparse trees, do not assure obtaining the minimum logic depth, but they save hardware resources and power [5].

The Kogge–Stone adder is a parallel prefix form Carry Lookahead adder and is widely considered one of the fastest adder designs possible. It is the common design for high-performance adders in industry.

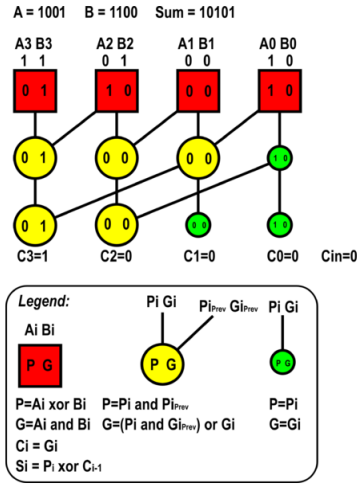


Fig. 3 A 4-bit Kogge–Stone adder architecture. Each vertical stage produces a "propagate" and a "generate" bit. The culminating generate bits, namely the carries are produced in the last stage, and these bits are go through “XOR process” with the initial propagate after the input to produce the sum bits.

The Kogge-Stone tree reaches the minimum logic depth using a very regular structure with uniform fanout, but has a drawback in that it is very dense, thus requiring a large number of gates and wires.

The Kogge-Stone scheme uses the property of limiting the lateral logical fanout at each node to unity, but at the cost of a dramatic increase in the number of lateral wires at each level. This is because there is massive overlap between the prefix sub-terms being pre-computed [6]. On the contrary, sparse trees, like the Brent-Kung, do not assure obtaining the minimum logic depth, but they save hardware resources and power, and, in any case, allow high-speed performances to be reached.

Circuit design style - Dynamic logic styles:

The combinational logic is implemented with dynamic circuits that can offer the desirable high speed operation. The selection of the initial topology that will allow the acquisition of the desired performance in the acceptable power budget is the most important step taken. In this way, high quality designs can be achieved, in terms of performance, energy consumption and area, with respect to alternative dynamic design styles.

Logic style significantly affects delay and energy. For example, it is proven in the literature [5] [7] that parallel-prefix trees realized using dynamic domino logic achieve higher speed performances at the expense of consumed energy; whereas, using static logics lowers power consumption, but sacrifices computational speed.

Static CMOS circuits achieve low-power consumption and high robustness and are preferable when the performance target is relaxed, and limiting the energy consumption is the main objective. Conversely, in high-performance designs, dynamic circuits are typically required. Dynamic circuit design techniques can provide high speed operation at lower silicon area requirements, compared to full static CMOS designs. Due to these features dynamic circuits are widely used to speed up critical units in high performance CMOS microprocessors.

According to the typical dynamic design style, the gate output is periodically precharged to high through a single pMOS control transistor (precharge transistor). This phase in the circuit operation is called precharge phase. Then, in between the precharge phases, an n-network is exploited to calculate the gate response according to the input data. In case that a logic low value is required at the output an active path in the n-network discharges the output while in case that logic high value is required no path is formed in the n-network to discharge the output which simply remains charged to VDD. This phase in the circuit operation is called evaluation phase. An additional nMOS control transistor (evaluation transistor) isolates the n-network from the ground and ensures that no discharge path is formed through the n-network during the precharge phase. During the evaluation phase the pMOS precharge transistor is inactive and no low to high transitions can take place at the output. This implies that during the evaluation phase if an input combination discharges the output, the latter will remain discharged regardless of the input combinations that may follow within the same evaluation phase. Consequently, it must be ensured that only a single and valid input combination is applied during each evaluation phase.

Dynamic logic uses a clock signal in its implementation of combinational logic circuits. The clock signal is used to drive the control transistors -synchronize transistors in sequential logic circuits- forming the two circuit operating phases. Therefore, dynamic logic circuit requires two phases where the output is driven high or low during distinct parts of the clock cycle. The first phase, when Clock is low, is the setup phase or the precharge phase and the second phase, when Clock is high, is the evaluation phase. Dynamic logic requires a minimum clock rate fast enough that the output state of each dynamic gate is used or refreshed before the charge in the output capacitance leaks out enough to cause the digital state of the output to change, during the part of the clock cycle that the output is not being actively driven.

Advantages of Dynamic Logic design:

- When properly designed, can be over twice as fast as static logic.
- It uses only the faster N transistors, which improve transistor sizing optimizations.
- Faster than static logic because static logic has twice the capacitive loading, higher thresholds, and uses slow P transistors.
- Avoiding pFETs where possible is one of the main goals of Dynamic Logic, due to speed.
- It may be the only choice when increased processing speed is needed.

Disadvantages of Dynamic Logic design:

- If the clock speed is too slow, the output will decay too quickly to be of use.
- Also, the output is only valid for part of each clock cycle, so the device connected to it must sample it synchronously during the time that it is valid.
- In general, dynamic logic greatly increases the number of transistors that are switching at any given time, which increases power consumption over static CMOS.

The most frequently used dynamic design logic styles are Domino logic and Multiple-output domino logic (MODL), which is a CMOS-based evolution of the dynamic logic techniques based on either PMOS or NMOS transistors. Multiple-output domino logic (MODL) is a dynamic CMOS logic in which complex gates can have multiple outputs for producing multiple functions. Domino logic allows a rail-to-rail logic swing. It was developed to speed up circuits by inserting an ordinary static inverter between stages in order to cascade dynamic logic gates. In Domino logic cascade structure of several stages, the evaluation of each stage ripples the next stage evaluation, similar to a domino falling one after the other. Once fallen, the node states cannot return to "1" (until the next clock cycle) just as dominos, once fallen, cannot stand up, justifying the name Domino CMOS Logic. It contrasts with other solutions to the cascade problem in which cascading is interrupted by clocks or other means.

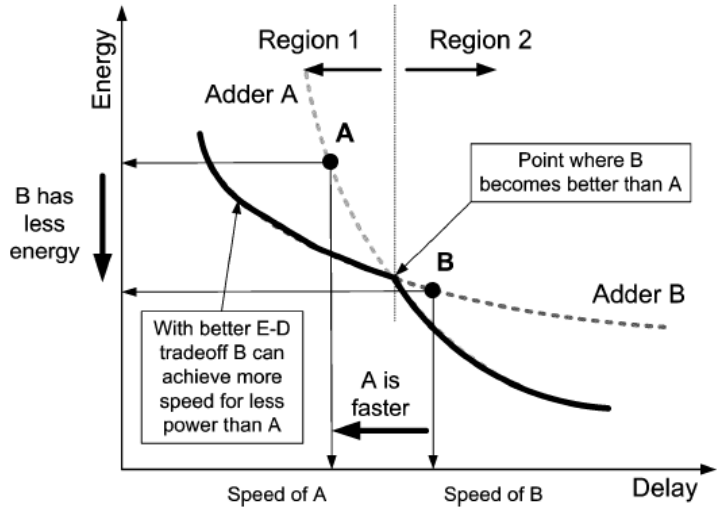
Performance and power consumption issues:

Low power consumption and high performance integrated circuits have been the main targets of the recent research on VLSI design. However, these two design criteria are often in conflict and it is observed that improving one particular aspect of the design may constrain the other. Concerns about energy consumption have forced digital designers to develop techniques for improving energy efficiency.

While the delay difference between different circuit families is apparent, the delay difference between topologies using the same circuit family is relatively small making it difficult to know which design can stretch further in the energy-delay space. The concept of comparing VLSI adders mainly based on their energy-delay characteristics stems from a need to make appropriate selection at the beginning of the design process. High-speed adders architectures based on the Carry Lookahead (CLA) principle remain dominant, since the carry delay can be improved by calculating each stage in parallel.

Energy is also important because if too much power is used to achieve a target delay, hot spots can be created Several approaches have been proposed to improve energy efficiency like the proper selection of circuit family and prefix; reducing the number of logic stages without increasing gate count; reducing switching activity; reducing the number of logic gates; and reducing the wiring complexity. This section presents the approaches for the optimal construction of high-performance VLSI adders in a given technology [8].

Considering that two adders A and B are compared against each other based on delay only. However, such a comparison provides an incomplete and potentially misleading picture. If we consider that energy can be traded for delay it is clear that further analysis is needed [9].



As the curves show, adder B has more room for delay improvement and can achieve lower energy in the high-performance region (Region1).

On the other hand, if lower computational energy is the design objective, adder A is the better choice as it can achieve lower energy in the low-performance region (Region 2).

Fig.4 Hypothetical energy-delay dependencies of two designs, A and B, optimized under the same constraints.

Recently, the main research concern in VLSI design, focus mainly in improving the already existing conventional structures in order to gain in terms of performance and power consumption. Efficient methods for obtaining higher performance combined with the lowest possible power consumption may be achieved by many different manners:

One could be a new clever design inspired by the conventional topologies and based on the combination and compositions of already existing structures introducing this way a new more efficient topology.

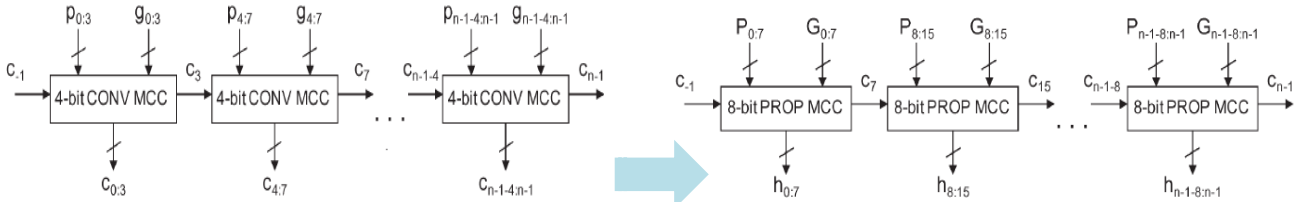
It is also possible to acquire the desirable results in terms of power consumption and performance by ameliorating the function of an existing topology by changing the placement of some elements in the topology or/and by improving the standard processes of the specific structure by introducing extra operations.

In their work Costas Efstathiou and his co-workers [10], are analyzing an implementation of an *8-bit Manchester carry chain (MCC) adder in multi-output domino CMOS dynamic logic style.*

The Manchester carry chain adders can efficiently be designed in CMOS logic and are considered to be the most efficient and widely accepted design approach to construct dynamic (domino) carry-lookahead adder (CLA) architecture with a regular, fast, and simple structure adequate for implementation in VLSI. The recursive properties of the carries in Manchester carry chain adder have enabled the development of multi-output domino gates which have shown area–speed improvements with respect to single-output gates. The Manchester carry chain generates all the carries in parallel, using an iterative shared transistor structure. In practice, the carry-lookahead adder length is limited to four in order to cut down the number of series-connected transistors.

The novel approach of this work indicates a high-speed double carry chain adder and more specifically an 8-bit adder module which is composed of two independent carry chains with length limited to 4 bits. These chains have the same length (measured as the maximum number of series-connected transistors) as the 4-bit Manchester carry chain adders. Consequently, the carries of the case in study are computed by the two independent 4-bit carry chains. More specifically the even and odd carries of this adder are computed separately by the two independent 4-bit carry chains. Thus, one chain computes the even carries, while the other chain computes the odd carries. The groups of even and odd new carries are computed in parallel by the two different carry chains in multi-output

domino CMOS logic. This separation allows the implementation of the carries by the two independent 4-bit carry chains.



Conventional 4-bit MCC adder module

Novel 8-bit MCC adder module

Fig. 5 Comparison between conventional and proposed MCC adder module

Implementation of wider adders based on the use of the proposed 8-bit adder was performed. The novel design wider adders were compared to their corresponding adders based on the standard 4-bit Manchester carry chain adder module. Therefore, the proposed novel design technique has been applied for 8-bit, 16-bit, 32-bit and 64-bit adders in multi-output domino logic. In order to evaluate the speed performance of the novel and the conventional design 8-bit, 16-bit , 32-bit and 64-bit adders have been designed according to this carry chain principle and simulated using SPECTRE in a standard 90-nm CMOS technology (with VDD = 1 V).
The conventional 8-bit, 16-bit, 32-bit and 64-bit Manchester carry chain adders are designed by cascading two, four, eight, and sixteen 4-bit MCC adder modules, respectively and the novel Manchester carry chain adders of 16-bit, 32-bit and 64-bit are designed by cascading two, four, and eight of the proposed 8-bit Manchester carry chain adder modules, respectively.

Considering that $a = a_{n-1}a_{n-2} \cdots a_1a_0$ and $b = b_{n-1}b_{n-2} \cdots b_1b_0$ represent the two binary numbers to be added and their sum is given by the equation: $s = s_{n-1}s_{n-2} \cdots s_1s_0$ the following equation describes the computation of the carry signals in binary addition: $c_i = g_i + z_i \cdot c_{i-1}$ By the expanded form of the previously mentioned equation the calculation for each carry bit c_i can be expressed as follows:

$$c_i = g_i + z_ig_{i-1} + z_iz_{i-1}g_{i-2} + \cdots + z_iz_{i-1} \cdots z_1g_0 + z_iz_{i-1} \cdots z_0c_{-1}.$$

The terms $g_i = a_i \cdot b_i$ symbolize the carry generate term while z_i represents the carry propagate term. The term c_{-1} is the input carry. The sum bits of the adder are defined as $s_i = p_i \oplus c_{i-1}$ and since Manchester carry chain adders are EXCLUSIVE OR adders the carry propagate signal is defined as $z_i = p_i = a_i \oplus b_i$

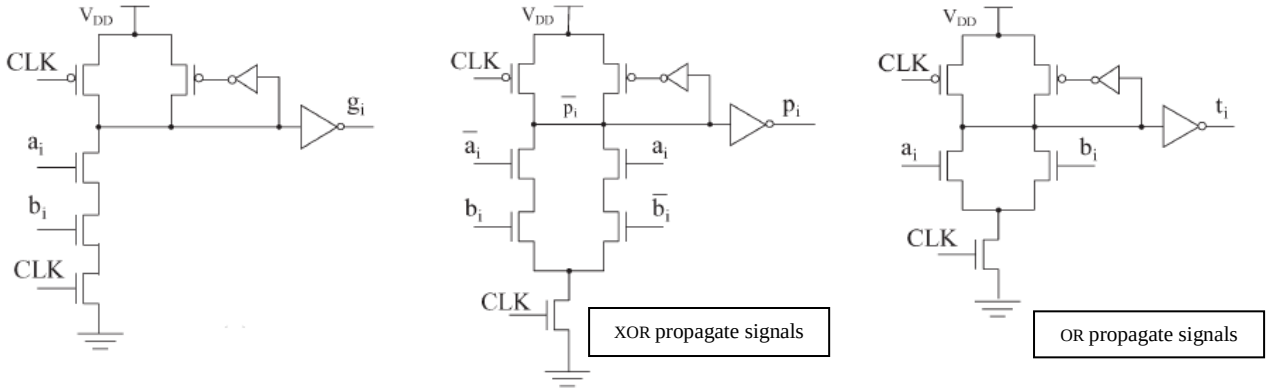


Fig. 6 Implementation of the generate and the two types of propagate signals in domino CMOS logic.

Table. 1 Computation of Even and Odd Carry and estimation of new carry.

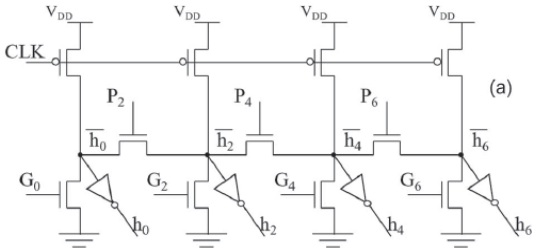
Even Carry Computation and new carry h_i estimation.

for i = 0 $\rightarrow h_0 = g_0 + c_{-1}$

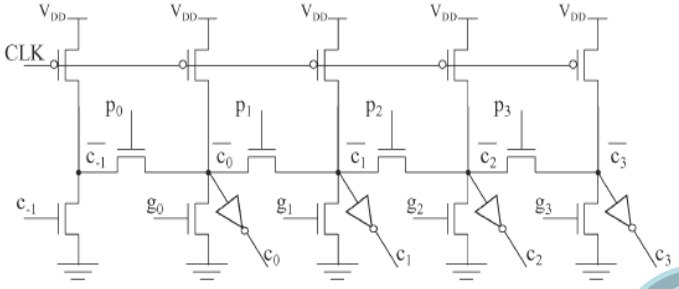
for i = 2 $\rightarrow h_2 = g_2 + g1 + p_2p_1t_0(g_0 + c_{-1})$

for i = 4 $\rightarrow h_4 = g_4 + g_3 + p_4p_3t_2\ (g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1}))$

for i = 6 $\rightarrow h_6 = g_6 + g_5 + p_6p_5t_4 \times (g_4 + g_3 + p_4p_3t_2\ (g_2 + g_1 + p_2p_1t_0(g_0 + c_{-1})))$



Even carry chain



Conventional domino 4-bit MMC

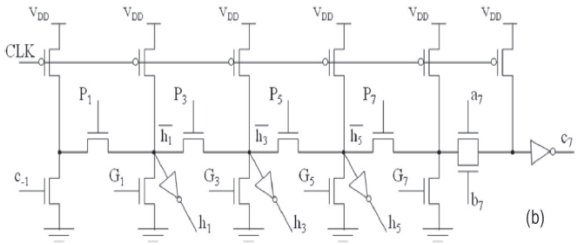
Odd Carry Computation and new carry h_i estimation.

for i = 1 $\rightarrow h_1 = g_1 + g_0 + p_1p_0c_{-1}$

for i = 3 $\rightarrow h_3 = g_3 + g_2 + p_3p_2t_1(g_1 + g_0 + p_1p_0c_{-1})$

for i = 5 $\rightarrow h_5 = g_5 + g_4 + p_5p_4t_3(g_3 + g_2 + p_3p_2t_1\ (g_1 + g_0 + p_1p_0c_{-1}))$

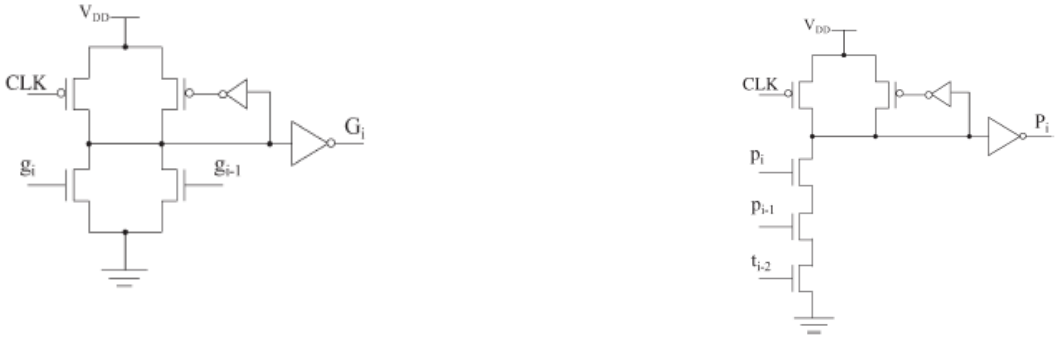
for i = 7 $\rightarrow h_7 = g_7 + g_6 + p_7p_6t_4 \times (g_5 + g_4 + p_5p_4t_3) \times (g_3 + g_2 + p_3p_2t_1(g_1 + g_0 + p_1p_0c_{-1}))$.



Odd carry chain

Table. 2 Estimation of the new generate and propagate signals $G_i = g_i + g_{i-1}$ and $Pi = p_i \cdot p_{i-1} \cdot t_{i-2}$

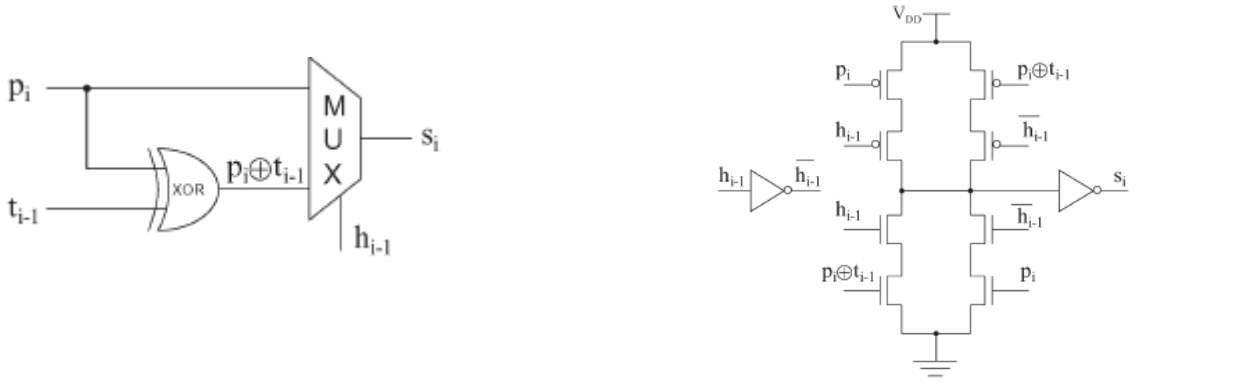
Equations for the new carries:	
for even values of i:	for odd values of i:
$h_2 = G_2 + P_2G_0$	$h_1 = G_1 + P_1c_{-1}$
$h_4 = G_4 + P_4G_2 + P_4P_2G_0$	$h_3 = G_3 + P_3G_1 + P_3P_1c_{-1}$
$h_6 = G_6 + P_6G_4 + P_6P_4G_2 + P_6P_4P_2G_0$	$h_5 = G_5 + P_5G_3 + P_5P_3G_1 + P_5P_3P_1c_{-1}$
	$h_7 = G_7 + P_7G_5 + P_7P_5G_3 + P_7P_5P_3G_1 + P_7P_5P_3P_1c_{-1}$



Generate signals **Propagate signals**

Fig. 7 Domino CMOS implementation for the new signals

The new and the conventional carries are linked by the following relation: $c_{i-1} = t_{i-1} \cdot h_{i-1}$ therefore the sum can be calculated through the equation: $s_i = p_i \oplus (t_{i-1} \cdot h_{i-1})$. According to the literature this equation can be expressed $s_i = \overline{h_{i-1}} \cdot p_i + h_{i-1} \cdot (p_i \oplus t_{i-1})$ For the implementation of the sum signals, the domino chain is terminated, and static CMOS technology is used. In order to calculate the sum a 2→ 1 multiplexer is introduce since it can selects either p_i or $p_i \oplus t_{i-1}$ according to the value of h_{i-1}.



Sum bit implementation. Static CMOS implementation of the 2→1 multiplexer.

Fig. 8 Implementation of a 2→1 multiplexer for the sum calculation

It is worth mentioning that no extra delay is introduced by the use of the proposed carries for the computation of the sum bits. This is because of the fact that XOR gate introduces equal delay with a 2 → 1 multiplexer and both terms p_i and $p_i \oplus t_{i-1}$ are computed faster than h_i. The speed performance

is significantly improved with respect to that of the standard Manchester carry chain adders' topology. The simulation results, for the carry propagation delays, depict that the novel design introduced in the work of Efstathiou and his co-workers, provides a performance improvement of 4.73% over the conventional design for the 8-bit adder architecture. Similarly the performance improvements of the novel design over the conventional design for the 16-bit adder, the 32-bit adder and the 64-bit adder are also significant indicating values of 23.08%, 30.05% and 35.08% respectively. However, in all cases previously mentioned, the average energy consumption for a computation is increased by 43.4% for the novel design with respect to the conventional design. In addition the area overhead is 49.9%, due to the extra gates that are required for the implementation of the t_i and the new generate (G_i) and propagate (P_i) signals. Thus, it can be concluded that the novel technique can be ideally used in the design of arithmetic circuits where high performance is required at the expense of power consumption.

In their work Zaher Owda and his collaborators [11] introduced *a three phase dynamic circuit design style* that provides the ability to implement pipelines without the use of memory elements. The proposed scheme is demonstrated on a *Kogge-Stone adder design*. The pipeline operation and the memory elements elimination provide very high speed dynamic circuit realizations. Furthermore a pre-evaluation phase hidden inside the pre-charge phase of each gate is introduced. This fact provides significant speed improvements at reduced power consumption. Usually, two clock signals (Clkn and Clkp) of equal period are used to drive each one of the two control transistors and provide the three operating phases, precharge, evaluate and memory of equal time duration (called phase time). Each gate level, that is a pipeline stage, is passing continuously through the three phases in that order. Thus, for the case under study the data scheduling is realized with the appropriate clock signals which are required at each level. Namely, three clock signals (Clk1n, Clk2n and Clk3n) and their complements (Clk1p complement of Clk3n, Clk2p complement of Clk1n and Clk3p complement of Clk2n) the two clock signals used at level L=i+1 are the clock signals used at level L=i delayed by one third of the clock period (or equivalently a phase time).

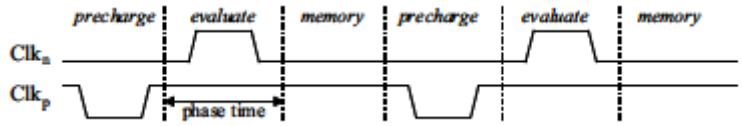
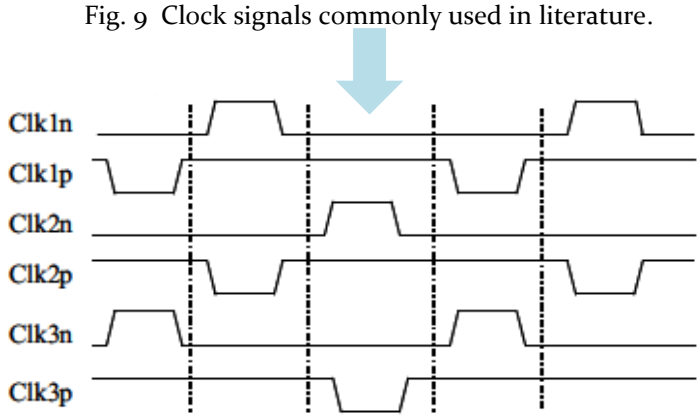


Fig. 9 Clock signals commonly used in literature.



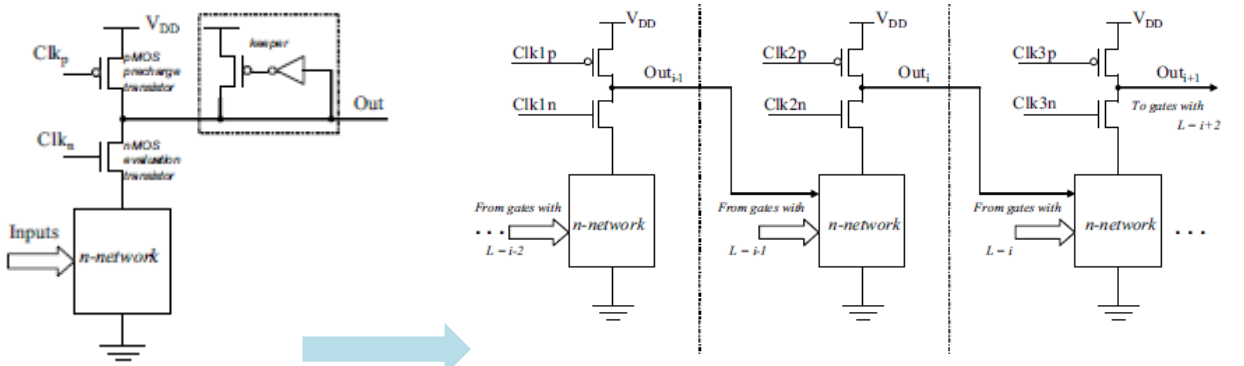
Level	Clock signal for nMOS evaluation transistor	Clock signal for pMOS precharge transistor
L mod 3 = 1	$Clk1_n$	$Clk1_p$
L mod 3 = 2	$Clk2_n$	$Clk2_p$
L mod 3 = 0	$Clk3_n$	$Clk3_p$

Fig. 10 The three phases clocking implemented in the novel design

Furthermore, in this work instead of placing this transistor between the n-network and the ground, the transistor is placed between the n-network and the output.

In the precharge phase the pMOS precharge transistor of the gate is activated and the output is precharged to high. The nMOS evaluation transistor is inactive and ensures that there is not any discharging path though the nMOS network. The output precharge operation does not depend on the input values of the n-network and can be completed regardless of these values.

The evaluation phase is analogous to the evaluation phase in a dynamic design. The pMOS precharge transistor is inactive and the nMOS evaluation transistor is active. The input values of the n-network during this phase, actually determine the response value of the gate at the end of this phase. According to the proposed design style, a high value at the output of the gate at the beginning of the evaluation phase is required, while valid and stable values are assumed at the inputs of the n-network during the whole phase time. During the evaluation phase, the inputs of the gate are stable since they are outputs of a level in the memory phase. during the precharge and evaluation phases of a level (where its output may change) the following level in the pipeline is at the memory and precharge phases respectively where there is not any constraint on the inputs' status (to be stable or not).Finally in the memory phase both pMOS and nMOS control transistors are inactive and the output will retain the state (logic low or high). The memory phase is after the evaluation phase and ensures that for a phase time the input values calculated during the evaluation phase will remain stable. This phase does not normally exist in typical dynamic gates. During the memory phase the input values of the n-network must not affect the output of the gate.



The proposed dynamic gate topology

Pipeline construction along with the selection of the appropriate clock signals for each gate, according to its level in the design.

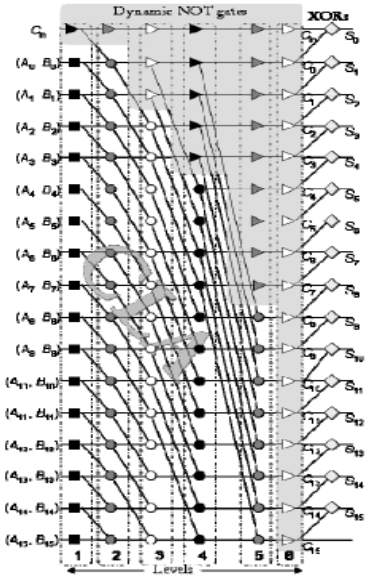
Fig. 11 Dynamic gate topologies

According to the proposed design technique, in case that we need to connect the output of a gate at level L=i as input to a gate at level L=i+k we have to add k levels in-between.

In case that k is even, the solution is to use a dynamic NOT gate for each one of the k intermediate levels. Since the number of the added NOT gates is even we do not alter the functionality of the circuit. In case that k is odd, the solution is to use a dynamic NOT gate for each one of the k-1 intermediate levels plus a dynamic buffer (dynamic NOT gate followed by a static NOT gate). Once again, since the number of the added NOT gates is even we do not alter the functionality of the circuit.

In a standard 180nm CMOS technology (VDD=1.8V), 16-bit Kogge-Stone adders have been designed, using for the implementation of their carry lookahead unit the dynamic design techniques shown in the following table. At the Kogge-Stone adders each gate level is fed by the same pair of clock signals.

Table. 3 Proposed dynamic gates for the CLA unit design.	
1st level	NOR and NAND
2nd and 4th levels	NOR and OR-NAND
3rd and 5th levels	NAND and AND-NOR



Level	Operation
1st	$\overline{G_{1:0}} = A \cdot B$ and $\overline{P_{1:0}} = A + B$
2nd and 4th	$G_{j:s} = \overline{\overline{G_j} \cdot (\overline{P_j} + \overline{G_s})}$ and $P_{j:s} = \overline{\overline{P_j} + \overline{P_s}}$
3nd and 5th	$\overline{G_{j:s}} = G_j + (P_j \cdot G_s)$ and $\overline{P_{j:s}} = P_j \cdot P_s$

Fig. 12 The 16-bit Kogge-Stone adder architecture. Each line inside, except the primary inputs carries a pair of generate/propagate signals (Gj, Pj).The square symbol at the first level represents the calculation of the generation/propagation signals by the primary inputs. Moreover, each circle at the rest levels represents a "dot" operation between two pairs of generate/propagate signals (Gj, Pj)·(Gs, Ps).

A major improvement of the new topology with respect to is the ability to exploit the precharge phase of a gate as a pre-evaluation phase. That part of the evaluation operation in each level is hidden inside its precharge phase, increasing circuit performance, as it is analyzed next. However, its main advantage is the ability to implement pipelines without the need of memory elements. A disadvantage of the proposed approach is the need of additional clock signals. However the generation of these signals is a one-time cost that does not increase with circuit complexity.

The worst case gate propagation delay time in the evaluation phase of the proposed dynamic design is 44.80ps, and comparing to other conventional designs mentioned in Litterature the gate propagation delay was reduced. In addition it was noticed an improvement of the proposed design style over previous conventional designs in terms of clock frequency and a reduction of the dynamic energy consumption. Last but not least, the silicon area, estimated by the sum of the transistor widths was also reduced.

According to the results obtained from the novel techniques' testing high quality designs can be achieved, in terms of performance, energy consumption and area, with respect to alternative dynamic design styles existing in literature for conventional structures. More specifically for both cases of

novel designs the speed performance was improved while the propagation delay time was significantly reduced.

Concerning the average energy consumption for a computation it was depicted that for the first case which combines and composes a new topology consisting of already existing structures, energy consumption is increased as well as the area overhead due to the extra gates that are required for the implementation. These facts make the novel technique ideal for applications of design of arithmetic circuits where high performance is required at the expense of power consumption. On the contrary the second novel technique presented, showed improvements also in terms of energy consumption and design area.

References:

[1] Cheng, K-H; S-W, Cheng "Improved 32-bit Conditional Sum Adder for Low-Power High-Speed Applications" *J. Inf. Sci. Eng.* 2006, 22, 975-989.

[2] Annapurna P. Bai, Vijaya M. Laxmi "Design of 128- bit Kogge-Stone Low Power Parallel Prefix VLSI Adder for High Speed Arithmetic Circuits" *International Journal of Engineering and Advanced Technology (IJEAT)* 2013, Volume-2, Issue-6

[3] Konstantinos Vitoroulis Asim J. Al-Khalili "Performance of Parallel Prefix Adders implemented with FPGA technology" Circuits and Systems, 2007.

[4] K.Nehru A.Shanmugam S.Vadivel "Design of 64-Bit Low Power Parallel Prefix VLSI Adder for High Speed Arithmetic Circuits" Computing, International Conference on Communication and Applications (ICCCA), 2012

[5] Perri S, Lanuzza M, Corsonello P., "Design of high-speed low-power parallel-prefix adder trees in nanometer technologies" Int. J. Circ. Theor. Appl. (2012)

[6] Knowles S. "A Family of Adders" 15th IEEE Symposium on Computer Arithmetic Proceedings, 2001.

[7] Perri S, Corsonello P. "Efficient Implementations of Radix-4 Parallel-Prefix Trees" CENICS The Fourth International Conference on Advances in Circuits, Electronics and Micro-electronics 2011.

[8] Zeydel B, Baran D, Oklobdzija,V, "Energy-Efficient Design Methodologies: High-Performance VLSI Adders" *journal of solid-state circuits*, 45, 6, E 2010.

[9] Oklobdzija V, Zeydel B, Dao H, Mathew S, Krishnamurthy R. "Comparison of High-Performance VLSI Adders in the Energy-Delay Space" *Transactions on very large scale integration* (VLSI) *systems*, 13, 6, 2005

[10] Efstathiou C, Owda Z, Tsiatouhas Y, "New High-Speed Multioutput Carry Look-Ahead Adders" T*ransactions on circuits and systems*—II*: express briefs,* 60, 10, 2013

[11] Owda Z, Tsiatouhas Y, Haniotakis T. "High Performance and Low Power Dynamic Circuit Design" 01/2011; DOI:10.1109/NEWCAS.2011.5981329