Design of Low Leakage Parallel Prefix Adder

[1]Nihal Kulkarni, [2]Suguri Charan Kumar
[1]Student, [2]Student,
[1]Electronics and Communication Engineering,
[1]GITAM University, Hyderabad, India

Abstract: The addition of two binary numbers is the basic and most often used arithmetic operation on microprocessors, digital signal processors and data processing applications in integrated circuits. Due to the advancement in CMOS technology, Parallel Prefix Adders came into picture and this is the most widely used adder. The battery life can be extended by reducing the chip size. This is the most reliable than the conventional adders. This adder is nothing but the extended performance of CLA. For the most recent CMOS feature sizes (90 nm and 65 nm) leakage power dissipation has become an overriding concern for the VLSI designers. The leakage power dissipation may come to dominate total power consumption. The leakage power increase is the increase in the sub-threshold leakage power. The sub-threshold conduction or drain current is the current that flows between source and drain of a MOSFET when the transistor is in the sub-threshold region that is for gate-to-source voltages below the threshold voltage. As drain voltage is increased, the depletion region of the p-n junction between the drain and the body increases in size and extends under the gate, so the drain assumes a greater portion of the burden of balancing depletion region charge leaving a smaller burden for the gate. As a result, the charge present on the gate retains charge balance by attracting more carriers into the channel, an effect equivalent to lowering the threshold voltage of the device.

Index Terms **– Carry Look-a-head adder (CLA), Most Significant bit, Kogge-Stone, Gray cell, Power, Switching.**

I. INTRODUCTION

Binary adders are one of the most essential logic elements within a digital system. In addition, binary adders are also helpful in units other than Arithmetic Logic Units (ALU), such as multipliers, dividers and memory addressing. Therefore, binary addition is essential that any improvement in binary addition can result in a performance boost for any computing system and, hence, help improve the performance of the entire system.

The fact that we save the carry-out word instead of using it immediately to calculate a final sum. The principal idea is that the carry has a higher power of 2 and thus is routed to the next column. Carry save adder is ideal to add several operands together. Thus, it can prevent time-consuming carry propagation and speed up computation. Effort the past shows that 16-bit CSA is the fastest adder within another adders.

The major problem for binary addition is the carry chain. As the width of the input operand increases, the length of the carry chain increases. Figure 1.1 demonstrates an example of an 8- bit binary add operation and how the carry chain is affected. This example shows that the worst case occurs when the carry travels the longest possible path, from the least significant bit (LSB) to the most significant bit (MSB). In order to improve the performance of carry-propagate adders, it is possible to accelerate the carry chain, but not eliminate it. Consequently, most digital designers often resort to building faster adders when optimizing a computer architecture, because they tend to set the critical path for most computations.

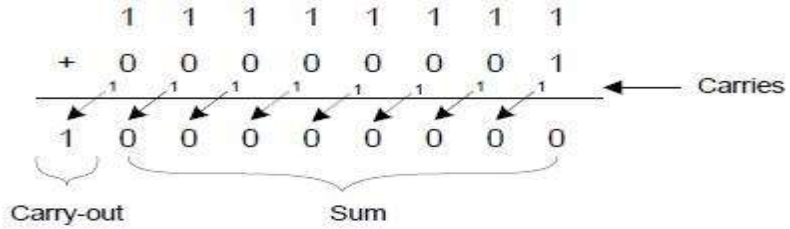


Figure 1.1 8-bit add operation

II. PARALLEL-PREFIX STRUCTURE

To resolve the delay of carry-look ahead adders, the scheme of multilevel-look ahead adders or parallel-prefix adders can be employed. The idea is to compute small group of intermediate prefixes and then find large group prefixes, until all the carry bits are computed. These adders have tree structures within a carry-computing stage similar to the carry propagate adder. However, the other two stages for these adders are called pre-computation and post-computation stages. In pre-computation stage, each bit computes its carry generate/propagate and a temporary sum. In the prefix stage, the group carry generate/propagate signals are computed to form the carry chain and provide the carry-in for the adder below.

$$G_{i:k} = G_{i:j} + P_{i:j} \cdot G_{j-1:k}$$

Pi:k = Pi:j . Pj-1:k

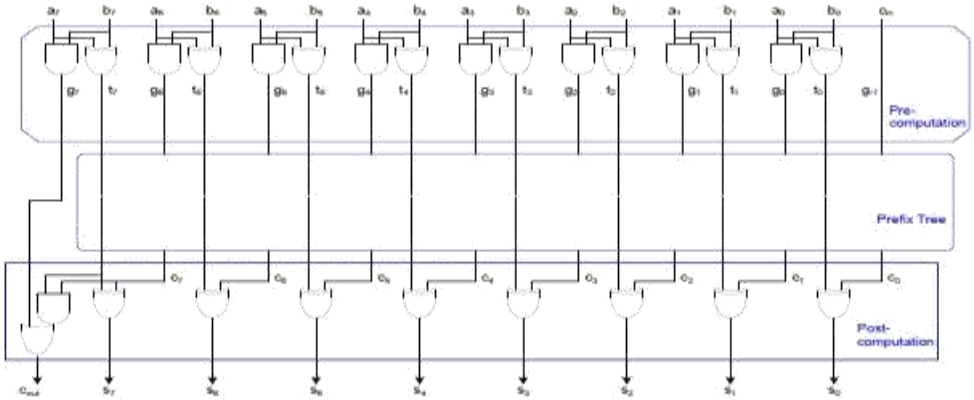


Figure 2.1 8-bit parallel prefix structure with carry save notation.

III. KOGGE-STONE ADDER STRUCTURE

Parallel-prefix adders, also known as carry-tree adders, pre-compute the propagate and generate signals. These signals are variously combined using the fundamental carry operator (fco).

$$(gL, pL) \circ (gR, pR) = (gL + pL \bullet gR, pL \bullet pR)$$

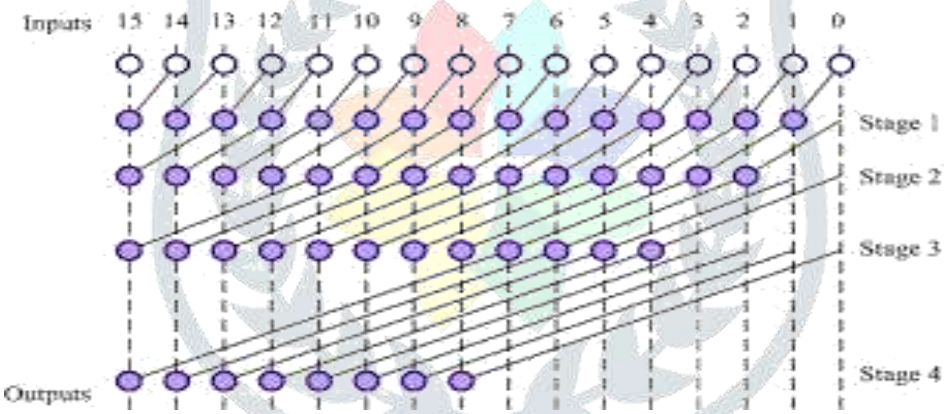


Figure 3.1 16-bit Kogge Stone Adder

It is readily apparent that a key advantage of the tree structured adder is that the critical path due to the carry delay is on the order of *log2N* for an N-bit wide adder. The arrangement of the prefix network gives rise to various families of adders. For this study, the focus is on the Kogge-Stone adder, known for having minimal logic depth and fan-out (see Figure 3.1). Here we designate BC as the black cell which generates the ordered pair in equation given. The Gray cell (GC) generates the left signal only. The interconnect area is known to be high, but for an FPGA with large routing overhead to begin with, this is not as important as in a VLSI implementation. The regularity of the Kogge-Stone prefix network has built in redundancy which has implications for fault-tolerant designs.

IV. DESIGN OF ADDER

The Kogge–Stone adder takes more area to implement than the Brent–Kung adder, but has a lower fan-out at each stage, which increases performance for typical CMOS process nodes. However, wiring congestion is often a problem for Kogge–Stone adders. The Lynch-Swartlzlander design is smaller, has lower fan-out, and does not suffer from wiring congestion; however, to be used the process node must support Manchester Carry Chain implementations. The general problem of optimizing parallel prefix adders is identical to the variable block size, multi-level, carry-skip adder optimization problem, a solution of which is found.

The time required to generate carry signals in this prefix adder is O(log n). It is the fastest adder with focus on design time and is the common choice for high performance adders in industry.

Enhancements to the original implementation include increasing the radix and sparsity of the adder. The radix of the adder refers to how many results from the previous level of computation are used to generate the next one. The original implementation uses radix-2, although it's possible to create radix-4 and higher. Doing so increases the power and delay of each stage, but reduces the number of required stages. The sparsity of the adder refers to how many carry bits are generated by the carry-tree. Generating every carry bit is called sparsity-1, whereas generating every other is sparsity-2 and every fourth is sparsity-4. The resulting carries are then used as the carry-in inputs for much shorter ripple carry adders or some other adder design, which generates the final sum bits. Increasing sparsity reduces the total needed computation and can reduce the amount of routing congestion.

V. WORKING AND IMPLEMENTATION

Kogge-Stone Adder has three processing stages for calculating the sum bits, they are:

1. Pre-processing stage
2. Carry generation (CG) network
3. Post-processing stage

The above steps involved in the operation of Kogge-Stone adder are as shown in below figure.

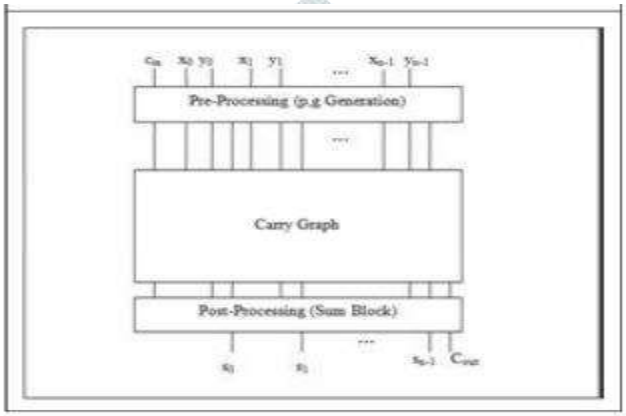


Figure 5.1 Block Diagram of KSA

1. Preprocessing

This step involves computation of generate and propagate signals corresponding to each pair of bits in A and B.

Pi = Ai xor Bi ...(1)
Gi = Ai and Bi ...(2)

2. Carry Generation Network

In this stage we compute carries corresponding to each bit. Execution of these Operations is carried out in parallel. After the computation of carries in parallel they are segmented into smaller pieces. It uses carry propagate and generate as intermediate signals which are given by the logic Equations (3 and 4):

CPi:j = Pi:k + 1 and Pk:j ……..(3)

CGi:j = Gi:k + 1 or (Pi:k + 1 and Gk:j)……....(4)

3. Post Processing

This is the final step and is common to all adders of this family (carry look ahead). It involves computation of sum bits.

Ci – 1 = (Pi and Ci) or Gig …… (5)
Si = Pi x or Ci – 1 ……. (6)

Carry look ahead network

- Generate (Pij, Gij) from (Gi, Pi) and (Gj, Pj)
- Pi:j = Pi:k+1 *and Pk:j*
- Gi:j = Gi:k+1 *or* (Pi:k+1 *and Gk:j*)

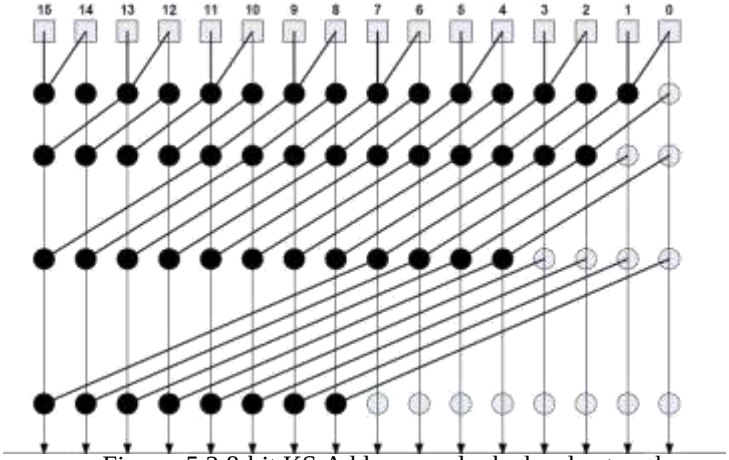


Figure: 5.2 8-bit KS Adder carry look ahead network

Sum and Carry waveforms

– Sum_i = pi *xor* Carryi-1
– C_i = Gi:0 *or* (Cin *and* Pi:0)

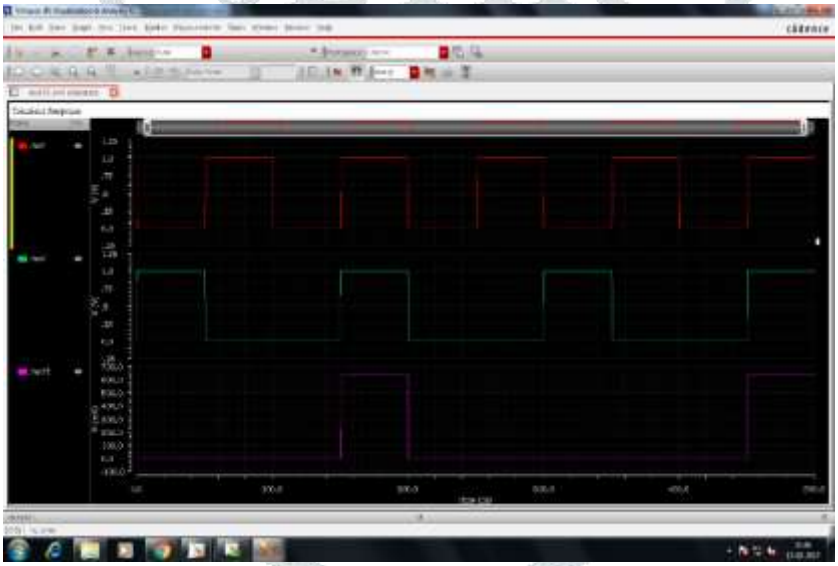


Figure 5.3 The Sum and Carry Waveforms.

VI. POWER LEAKAGE MINIMISATION

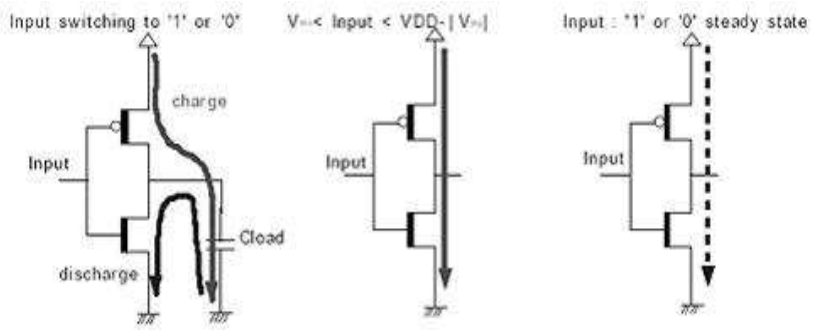
An integrated low power methodology requires optimization at all design abstraction layers as mentioned below.

1. System: Partitioning, Power down.
2. Algorithm: Complexity, Concurrency, Regularity.
3. Architecture: Parallelism, Pipelining, Redundancy, Data-Encoding.

The growing market of portable (e.g., cellular phones, gaming consoles, etc.), battery-powered electronic systems demands microelectronic circuits design with ultra-low power dissipation. As the integration, size, and complexity of the chips continue to increase, the difficulty in providing adequate cooling might either add significant cost or limit the functionality of the computing systems which make use of those integrated circuits. As the technology node scales down to 65nm there is not much increase in dynamic power dissipation. However, the static or leakage power is same as or exceeds the dynamic power beyond 65nm technology node. One of the techniques used is explained below

1. Dynamic Power Suppression

Dynamic/Switching power is due to charging and discharging of load capacitors driven by the circuit. Supply voltage scaling has been the most adopted approach to power optimization, since it normally yields considerable power savings due to the quadratic dependence of switching/dynamic power P-Switching on supply voltage V_{DD}. However, lowering the supply voltage affects circuit speed which is the major short-coming of this approach. So, both design and technological solutions must be applied to compensate the decrease in circuit performance introduced by reduced voltage.



a. Dynamic power b. Short-circuit current c. Leakage current

Figure 6.1 Components of Power in CMOS Circuits

Total Power dissipated in a CMOS circuit is sum total of dynamic power, short circuit power and static or leakage power. Design for low-power implies the ability to reduce all three components of power consumption in CMOS circuits during the development of a low power electronic product. In the sections to follow we summarize the most widely used circuit techniques to reduce each of these components of power in a standard CMOS design.

$$P_{total} = CLV_{DD2} + T_{sc}\, V_{DD}\, I_{peak} + V_{DD} I_{leakage}$$

VII. SIMULATION RESULTS

The below shown output waveform of the sum and carry bits resulting in the gate delays to 22 ns.

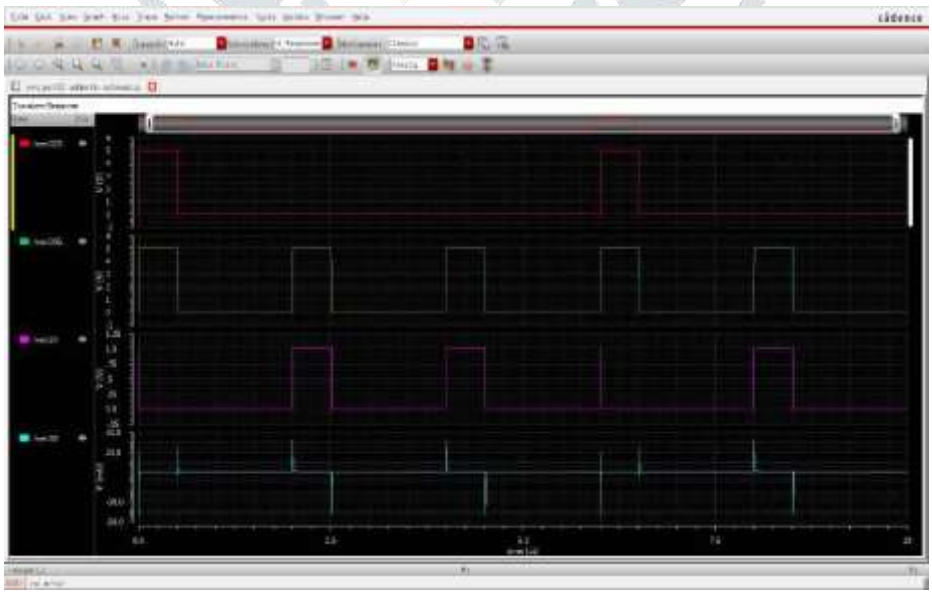


Figure 7.1 Output waveform of Sum and Carry

The simulation overview in the form of the logic gates is shown below

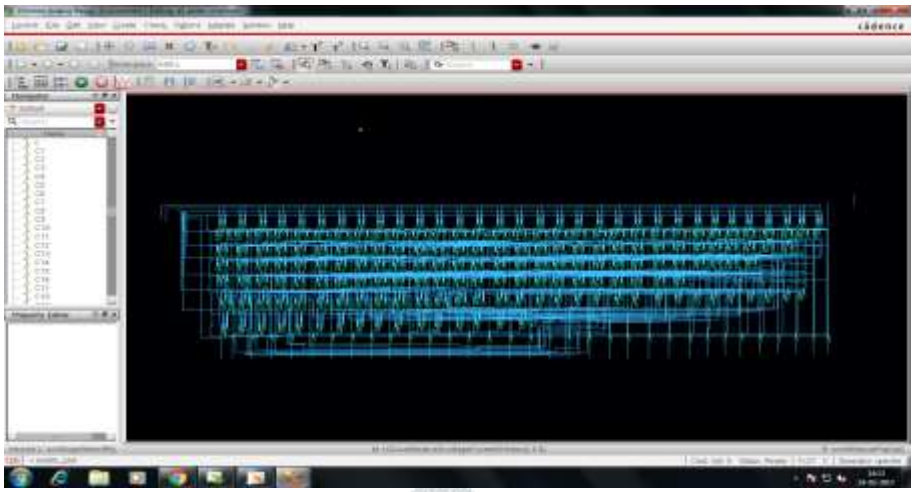


Figure 7.2 Simulation result of 32-bit KSA

VIII. FUTURE SCOPE

It was observed that due to the nature of Kogge-Stone prefix, the expected resource usage of Kogge-Stone adder will be greater comparing with Sklansky adder and it was justified by the results. It was also observed that multiple iterations of the same design's synthesis sometimes produce slightly different placement results in terms of logic resources usage and timing.

Pipelining by inserting a number of pipeline stages enhanced the designs and the results were analyzed. It turns out that the pipelining is not necessary improving the design speed. Both measured and simulation results from this study have shown that parallel-prefix adders are not as effective as the simple ripple-carry adder at low to moderate bit widths.

We have indications that the carry-tree adders eventually surpass the performance of the linear adder designs at high bit-widths, expected to be in the 128 to 256-bit range. This is important for large adders used in precision arithmetic and cryptographic applications where the addition of numbers on the order of a thousand bits is not uncommon. Because the adder is often the critical element which determines to a large part the cycle time and power dissipation for many digital signal processing and cryptographic implementations, it would be worthwhile for future FPGA designs to include an optimized carry path to enable tree-based adder designs to be optimized for place and routing

This would improve their performance similar to what is found for the RCA. We plan to explore possible FPGA architectures that could implement a "fast-tree chain" and investigate the possible trade-offs involved. The built-in redundancy of the Kogge-Stone carry-tree structure and its implications for fault tolerance in FPGA designs is being studied. The testability and possible fault tolerant features of the spanning tree adder are also topics for future research.

REFERENCES

[1] M. Morris Mano, "Computer System Architecture", Pearson Publications, 3rd Edition.

[2] M. Morris Mano and Michael D. Ciletti, "Digital Design", Pearson Education, Fourth Edition.

[3] R.P Brent and H. T Kung, "A regular layout for parallel adders," IEEE Trans, Computer vol C-31, pp 260-264,1982.

[4] T. R Padmanabhan and B. Bala Tripura Sundari, "Digital Design Through Verilog HDL", WILEY Publications, 2015 Edition.

[5] P. M Kogge and H. Stone, "A Parallel Algorithm for efficient solution of general class of recurrence relations," IEEE Trans on Computers, Vol C-22, No 8 , August 1973.

[6] P. Ndai , S. Lu, D. Somashekar and K. Roy, "Fine-Grained Redundancy in Adders", Int Sym on Quality Electronic Design, pp. 931-939, August 1992.