A Fast and Power Efficient Bit-Slice Design of a Two-bit Carry Skip Adder for Automated Layout Applications

Sirisha Nalmela and Krashna Nand Mishra

Abstract— **Rapid increase in design complexity and the requirement to reduce design time have resulted in a need for automation. Thus the design has been aimed to implement large operand adders while designing such automated layout tools. In this paper, 2-bit CSA bit-slice has been designed that can be used as basic building block of a general n-bit CSA adder, which accepts two n-bit inputs and produces the binary sum as output.**

The worst-case delay for two-bit slice after post-layout spice simulations is 0.93ns and power consumption is also reduced up to 11-18% in comparison to standard CMOS implementation exploiting signal correlations and resynchronization to minimize glitching . The adder is integrated into an area of 66912 µm^2 achieved by 1.2µm MOSIS process with 91.2% layout efficiency.

Index Terms— **Adder; Transistor-sizing; Glitching; Carry-Skip; Bit-slice;**

I. INTRODUCTION

Adder is one of the most widely used building blocks in all data processing (arithmetic) and digital signal processing architectures. In VLSI applications, Area and Power are very important factors that are taken into account while designing fast adders. But other than these, success of design also depends on some critical parameters like testability and ease of design. Carry-skip adder (CSA) [1] comes into the category of fast adder family and is considered good for its layout regularity and topological simplicity in terms of area and performance.

Several methods have already been suggested to optimal block distribution for carry-skip adder to reduce the worst case delay. So it is considered a good option for fast adders. Here, we haven't emphasized on the block level distribution, or particularly on the timing issues, but tried to address the transistor level implementation to reduce the glitching power component to a great extent at the cost of some additional area requirement to make it power effective also.

In this paper, the improved transistor level Implementation of CSA architecture will be presented and the circuit techniques for realizing different blocks used in two-bit CSA slice using transmission gate-pass transistor based logic will be described. Section III shows general architecture to implement carry skip adder. The behavior and transistor level circuit implementation of different blocks used in this slice is discussed in its sub-sections. Different aspects of bit-slice implementation and its utility for the implementation has been discussed and covered in section IV. Section V describes simulated results, and the conclusions are summarized in section VI.

II. A GENERAL ARCHITECTURE OF CSA

If we consider the carry generation for the operand bits of an adder, accepting two n-bit inputs, the following rules apply to each bit:

- If A_i#B_i for some bit i, we don't need to compute the value of C_{i+1}. Carry-in can be directly propagated to the next block.
- If A_i=B_i=1, this bit will generate carry irrespective of the value of C_{in} for this bit.
- If A_i=B_i=0, a carry would not be propagated by the bit location, i.e., carry out would be zero.

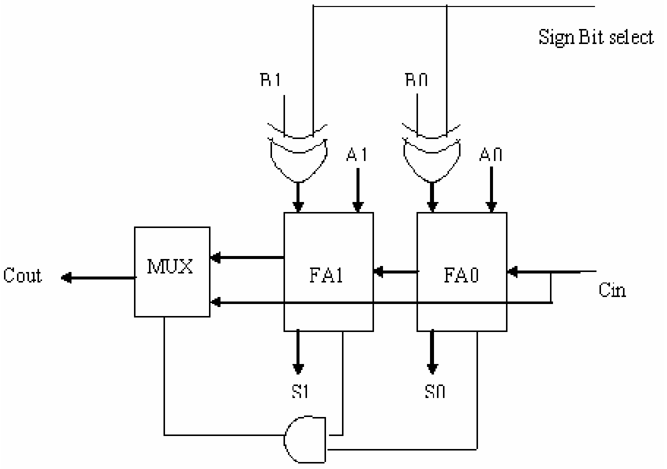

Fig. 1. Block Diagram of the Adder

A typical carry-skip adder breaks up the n bits into several blocks with sizes k1, k2,…, such that ∑ki = n. Every block also generates a block-carry-propagate signal that equals 1 if

---

Manuscript received June 21, 2006.

Sirisha Nalmela is with the Dhirubhai Ambani Institute of Information & Communication Technology, Gujrat, India (e-mail: n_sirisha@daiict.ac.in).

Krashna Nand Mishra is with the Dhirubhai Ambani Institute of Information & Communication Technology, Gujrat, India (e-mail: krashna_mishra@daiict.ac.in).

all stages internal to the block satisfying $A_i \# B_i$. This signal can be used to allow an incoming carry to skip all the stages within the block and generate a block-carry-out. It has been depicted in Figure 1, showing the AND gate output as block propagate signal. We further describe the operation of individual blocks in subsequent sub-sections.

A. Circuit Description: 1bit Adder Cell

Figure 2 describes the circuit implementation of 1-bit adder [2]. It uses transmission gate logic. This logic eliminates threshold loss effects, which result in loss of logic levels in NMOS and PMOS pass-transistors. Skip signal $(A \oplus B)$ is generated much before the sum is evaluated so that the block can be skipped if it satisfies the condition, without waiting for the sum bit to get computed. But the disadvantage is that while propagating the signal from primary input signal to node N1 or N2, it shows path delay mismatch (because of inverter in one path) corresponding to different input combinations. Consequently, it results in glitching at these nodes to produce high voltage spikes in the signal. Since the signal coming to transmission gate is complementary, two inverters have been introduced before the signal coming directly to compensate for the delay introduced by one inverter using concepts of Logical Effort [3].Tentative sizes are calculated using principles of logical effort and then the correct sizes are verified using simulations and layout is made and verified. It increased the transistor count but the benefit we derived is reduction of glitches to large extent and thus, less power consumed.

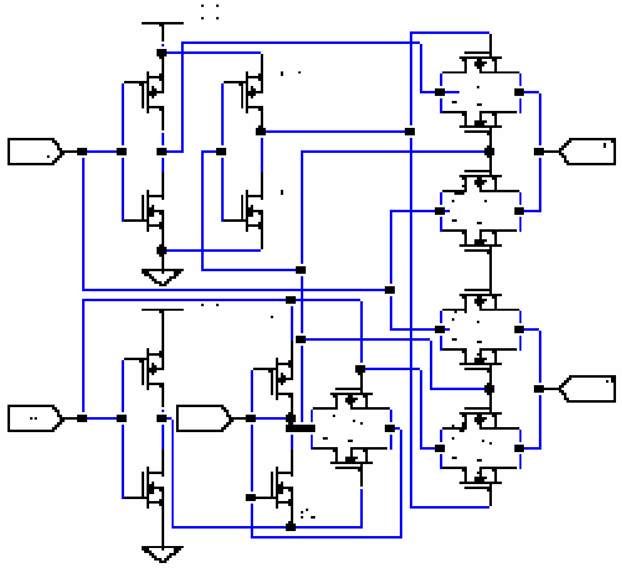

Fig. 2. Schematic Diagram of Transmission gate based Full Adder

B. Circuit Description: 5-Transistor MUX Cell &3-transistor AND gate

Figure 3(a) shows the implementation of MUX to select correct carry output based on carry-in available as carry out from the previous block. It is used as 'm' cell in the RCS architecture. M1 has been used as level restorer, which is sized so as not to beat the signal passing through the pass transistors during state transitions at the node that is to be restored. Also, it should be large enough to restore the voltage to Vdd quickly. Since pass transistor passes good '0', but a bad '1', the inverter has also been inversely ratioed to provide faster 1-> 0 transitions.

Figure 3(b) shows AND gate using one pass transistor and a transmission gate, which provides select input for the MUX.

C. Sign bit representation

In n-bit sign representation, MSB is used as sign bit; remaining (n-1)-bits can be used to represent a number. So for 4-bit binary numbers, numbers between –8 to 7 can be properly represented.

If the sum of two positive numbers exceeds the higher limit or the sum of two negative numbers exceeds the lower limit, sign bit error will occur.

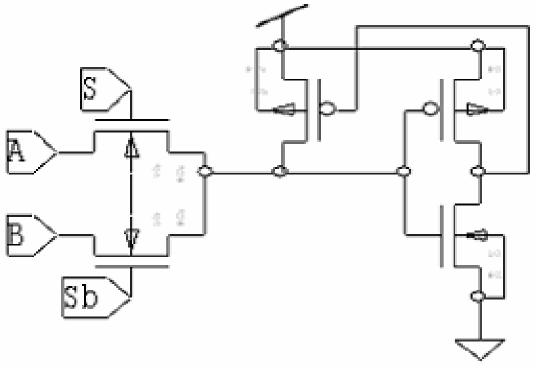

Fig. 3(a). Schematic Diagram of m-cell

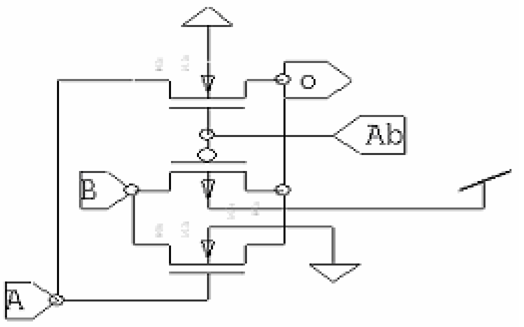

Fig. 3(b). Schematic Diagram of AND block

But in this particular architecture, 2's complement representation of both the numbers is not possible simultaneously; the only condition for which sign bit error can occur is if both the numbers are positive. If only one of the numbers is negative, then there would be no error provided the MSB for both numbers corresponds to sign bit. Here, for all those combinations in which the sum is exceeding the limit, two MSB's are complementary in nature. So we can take XOR of these two bits as a sign bit error signal and if it is '1', then there is sign bit error.

III. IMPORTANCE OF BIT SLICE ARCHITECTURES

Bit-slice [4] is a technique to implement a unit having modules in terms of slice or a group of a number of bits. Bit-slice processors usually consists of an ALU of 1, 2, 4 or 8 bits and control lines including carry or overflow signals usually internal to the CPU. For example, two 4-bit ALUs could be arranged side by side, with control lines between them, to form an 8-bit ALU slice units.

Similarly, we can have a library consisting of 2, 3 or 4-bit bit-slice cells and we can easily implement large operand size adders just by using a combination of them. Thus it can be very effective in automated layout applications and comes out very advantageous when used with bit-slice processors.

IV. CHARACTERIZATION PROCEDURE

For characterization, SPICE netlist with distinct capacitances for each cell has been extracted from the cell layout using Magic 7.0 Layout tool. The transistor models used are Level 3 model of 1.2µm MOSIS CMOS Technology.

A. Power Modeling

To characterize the power consumption of a CMOS circuit, a general method involves considering the power consumption of all possible input transition events in the look-up table. For larger adders, this approach can be very ineffective while calculating transition power for each entry in the look-up table. But since these bit-slices are going to be repeated further to implement large adders, we only need a systematic method to individually determine the contribution of bit-slice and it is stored in the layout database. And when it is required to estimate the total power consumption for large operand size adders, it can be used as an input for overall estimation.

We have estimated power consumption for these bit-slice cells, using the following method.

Since total power in CMOS circuits is dominated by the dynamic switching of circuit elements (i.e., charging and discharging of capacitances), dynamic short-circuit (or overlap) currents and static leakage are of less importance. So we can find out average power consumption per million transitions at the output from spice simulations as given below [5].

Input patterns are applied in such a way that all possible combinations are covered.

```
vA A 0 bit({00111000110101100} pw=5ns on=5 off=0 rt=0.1ns ft=0.1ns)
vB B 0 bit({01010110111100000} pw=5ns on=5 off=0 rt=0.1ns ft=0.1ns)
.tran 5n 85n
.power vVdd 0n 85n
```

From output file, average power consumed -> p watts

Input pulse is making 1 transition per 5 ns, i,e, 1 million transition per 0.005 sec. so average power consumed per million transition= 0.005*p

B. Delay Modeling

Propagation delay is determined by the cell and interconnection delays on the critical path.

Cell delay depends on the transistor level circuit implementation and related complexity of the cell. It is determined by the circuit topology and arrangements, thus resulting in logarithmic delay-to-area dependency.

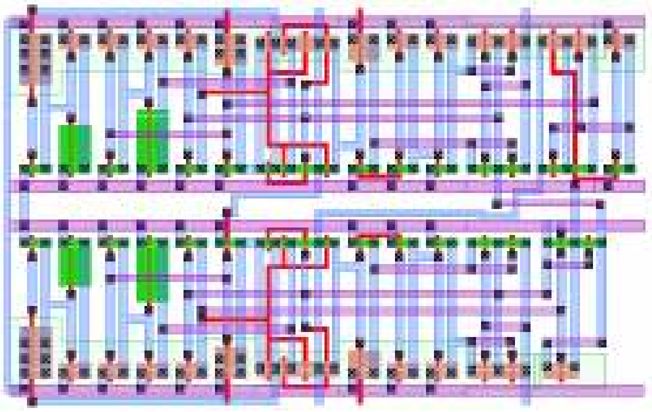

Fig. 4. Layout View of Two bit CSA Slice

Interconnection delay is the RC-delay of the wire. Usually this is almost negligible in comparison to cell delay and ignored in most of the cases.

C. Area Modeling

Although transistor count is not a proper method for area estimation, it can provide suitable guideline for area estimation for different design structures. Although, in cell-based design techniques, two components-Cell area and Wiring area can be formulated for area modeling:

Total cell area is roughly proportional to the number of transistors or gate equivalents contained in a circuit. This can be precisely determined by netlist generated after physical layout.

Wiring area is proportional to the total wire length that can be estimated from the number of nodes and the average wire length of a node or, more accurate, from the sum of cell fan-out and the average wire length of cell-to cell connections (i.e. accounts for the longer wire length of nodes with higher fan-out). The wire lengths also depend on circuit size, circuit connectivity (i.e., locality of connections), and layout topology, which are known only after circuit partitioning and physical layout.

V. SIMULATION RESULTS

Design has been implemented in 1.2 µm MOSIS technology using MAGIC 7.0 Schematic and Layout tool and spice simulations has been performed in LTSpice2.18

TABLE I

CHARACTERISTICS OF FOUR BIT TEST CIRCUIT

Attribute	Description
Process Technology	1.2 µm 5V MOSIS Level 3 Model
Architecture	Carry Skip Adder(CSA)
Worst Case Delay	0.93 ns
Area(Number of transistors/Colored area)	71/66912 λ^2
Routing Overhead	4554 λ^2
Power Consumption per million Transitions	4.6886 µW

VI. CONCLUSION AND FUTURE WORK

Effective use of circuit topologies and input reordering techniques has been considered to size the transistors to reduce the glitching at some nodes to make it power effective. Critical path delay has been reduced, resulting in improved timing performance.

Since the design is of bit-slice type, it can easily be extended to large operand size architectures. Thus it can work as a good input as adder structures for automated layout applications.

ACKNOWLEDGMENT

We are greatly indebted to Dr. Dipankar NagChoudhuri who guided us to work in this direction and motivated us to work on the problem.

REFERENCES

[1] S. Turrini, "Optimal Group Ditribution in carry-skip Adders", Digital Equipment Corporation, Western Research Laboratory, 100 Hamilton Avenue, Palo Alto, CA 94301.

[2] http://www.acsel-lab.com/Project/fast_adder/adder_designs.htm, Adder Designs.

[3] I. E. Sutherland, R. F. Sproull, and D. L. Harris. Logical Effort, Designing Fast CMOS Circuits. Morgan Kaufmann, 1999.

[4] T. Sridhar and J. P. Hayes, "Design of Easily Testable Bit-Sliced Systems", IEEE Transactions on Computers, vol. C30, issue 11, pp. 842-854, November 1981.

[5] Learning Material, "VLSI Subsystem Design", prepared at CEERI as part of the project, "Special Manpower development for VLSI design and related software" under Ministry of Information & communication Technology, India

Sirisha Nalmela received the Bachelor's degree in Information & Communication Technogy in 2006 and currently working as the member of UWB Solutions Group of R&D center in MindTree Consulting Pvt. Ltd., Banglore, India. Her research interests are VLSI digital circuit design, particularly in Low power Design domain and CAD Tool Development.

Krashna Nand Mishra received the Bachelor's degree in Information & Communication Technogy in 2006 and currently working as the member of Technical staff for Telecom Competency center in the Persistent Systems Pvt. Ltd, Pune, India. His research interests are VLSI analog & digital circuit design, particularly in Low power Design domain.