# Design of Parallel Prefix Tree Structure for CMOS Comparator

[1]A. Ramesh, [2]A.Srinivas Gupta, [3]B. Rajasekhara Reddy

[1]Student of VLSI Design, [2]Assistant Professor, [3]Assistant Professor
E.C.E Department, Narasaraopeta Institute of Technology, Narasaraopeta,
Email: [1]aluriramesh4@gmail.com, [2]anpsgupta@gmail.com, [3]rajasekhar.reddy61@gmail.com

**Abstract—This paper proposes a comparator design using conventional digital CMOS cells featuring wide-range and high-speed operation. The Comparison is most basic arithmetic operation that determines whether one number is greater than, less than or equal to the other number. Our comparator uses a novel scalable parallel prefix structure that leverages the comparison outcome of the MSB, proceeding bitwise towards LSB only when the comparison bits are equal. This comparator is composed of locally interconnected CMOS gates with a maximum fan-in of five and fan-out of four, independent of comparator bandwidth.**

**Comparator is most fundamental component that performs comparison operation. Comparison between modified and existing 8-bit binary comparator designs is calculated by simulation performed at 90nm technology in DSCH, Microwind Tool The main advantages of our proposed design are high speed and power efficiency, maintained over a wide range.**

**Index Terms—High-speed arithmetic, high-speed wide-bit comparator architecture, parallel prefix tree structure.**

## I. INTRODUCTION

Comparator is a basic arithmetic unit that compares the magnitude of two binary numbers, say A and B, and produces output bits: A>B or A<B or A=B. It is an important data-path element for any general purpose architecture as well as an essential device for application-specific and signal processing architectures. Comparators are also used in sorting networks which play an important role in areas such as parallel computing, multi-access memories and multiprocessing.

Comparator forms a fundamental component of processors and digital systems. For processors, in order to achieve high throughput with fast clock rates, it is necessary that such devices have less delay. Consequently, the designing of high speed comparator architecture becomes a relevant and essential research topic. Previously published comparator implementations having serial and parallel architecture can both be found in literature. The serial architecture is suitable for short inputs (i.e. when both the inputs have lesser number of bits). For longer inputs (say, 32 bit, 64 bit inputs), the circuit complexity and the combinational delay increase drastically. As a result, parallel approach is generally preferred for comparators with longer inputs. The comparator designs presented in this paper are based on parallel approach.

## II. COMPARATOR ARCHITECTURAL OVERVIEW

The comparison resolution module in Fig. 1 (which depicts the high-level architecture of our proposed design) is a novel MSB-to-LSB parallel-prefix tree structure that performs bitwise comparison of two N-bit operands A and B, denoted as AN−1, AN−2, . . ., A0 and BN−1, BN−2, . . ., B0, where the subscripts range from N–1 for the MSB to 0 for the LSB. The comparison resolution module performs the bitwise comparison asynchronously from left to right, such that the comparison logic's computation is triggered only if all bits of greater significance are equal. The parallel structure encodes the bitwise comparison results into two N-bit buses, the left bus and the right bus, each of which store the partial comparison result as each bit position is evaluated, such that

if Ak > Bk, then left k = 1 and right k = 0

if Ak < Bk , then left k = 0 and right k = 1

if Ak = Bk , then left k = 0 and right k = 0.

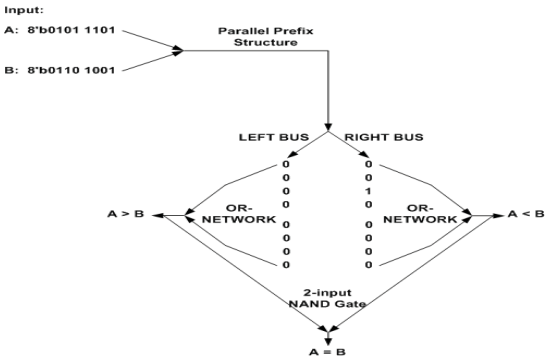



Fig. 2. Example 8-b comparison

In addition, to reduce switching activities, as soon as a bitwise comparison is not equal, the bitwise comparison of every bit of lower significance is terminated and all such positions are set to zero on both buses, thus, there is never more than one high bit on either bus. The decision module uses two OR-networks to output the

final comparison decision based on separate OR-scans of all of the bits on the left bus (producing the L bit) and all of the bits on the right bus (producing the R bit). If LR = 00, then A = B, if LR = 10 then A > B, if LR = 01 then A < B, and LR = 11 is not possible.

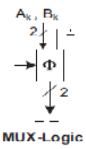
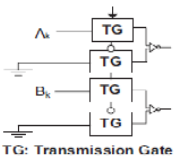
An 8-bit comparison of input operands A = 01011101 and B = 01101001 is illustrated in Fig. 2. In the first step, a parallel prefix tree structure generates the encoded data on the left bus and right bus for each pair of corresponding bits from A and B. In this example, A7 = 0 and B7 = 0 encodes as left7 = right7 = 0, A6 = 1, and B6 = 1 encodes as left6 =right6 = 0, and A5 = 0 and B5 = 1 encodes left5 = 0 and right5 = 1. At this point, since the bits are unequal, the comparison terminates and a final comparison decision can be made based on the first three bits evaluated. The parallel prefix structure forces all bits of lesser significance on each bus to 0, regardless of the remaining bit values in the operands. In the second step, the OR-networks perform the bus OR-scans, resulting in 0 and 1, respectively, and the final comparison decision is A > B.

We partition the structure into five hierarchical prefixing sets, as depicted in Fig. 3, with the associated symbol representations in Tables I and II, where each set performs a specific function whose output serves as input to the next set, until the fifth set produces the output on the left bus and the right bus.

TABLE I - SYMBOL NOTATION AND DEFINITIONS

| Symbol (Cells) | Definition |
|---|---|
| *N* | Operand bitwidth |
| *A* | First input operand |
| *B* | Second input operand |
| *R* | Right bus result bit |
| *L* | Left bus result bit |

TABLE II - LOGIC GATE REPRESENTATIONS FOR SYMBOLS USED IN FIG. 3

| Symbols (Cells) | Logic Gate | Maximum Fan-In/Fan-out And (Transistor Counts) |
|---|---|---|
|  |  | 2 / 4 (12) |
|  |  | 4 / 4 (8) |
|  |  | 5 / 1 (20) |
|  MUX-Logic |  TG: Transmission Gate | 3 / 2 (12) |

All cells (components) within each set operate in parallel, which is a key feature to increase operating speed while minimizing the transitions to a minimal set of leftmost bits needed for a correct decision. This prefixing set structure bounds the components' fan-in and fan-out regardless of comparator bitwidth and eliminates heavily loaded global signals with parasitic components, thus improving the operating speed and reducing power consumption. Additionally, the OR-network's fan-in and fan-out is limited by partitioning the buses into 4-b groupings of the input operands, thus reducing the capacitive load of each bus.

## III. COMPARATOR DESIGN DETAILS.

In this section, we detail our comparator's design (Fig. 3), which is based on using a novel parallel prefix tree (Tables I and II contain symbols and definitions). Each set or group of cells produces outputs that serve as inputs to the next set in the hierarchy, with the exception of set 1, whose outputs serve as inputs to several sets

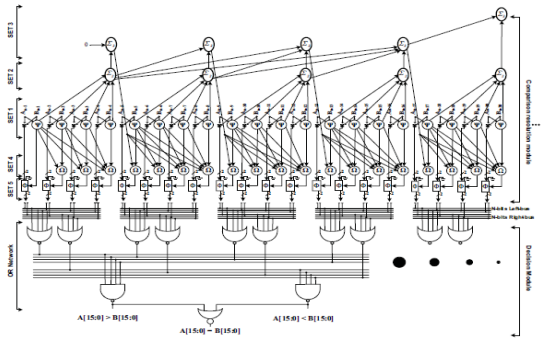


Fig. 3. Implementation details for the comparison resolution module (sets 1 through 5) and the decision module.

Set 1 compares the N-bit operands A and B bit-by-bit, using a single level of N -type cells. The N-type cells provide a termination flag Dk to cells in sets 2 and 4, indicating whether the computation should terminate.

Set 2 consists of ∑2-type cells, which combine the termination flags for each of the four Ψ-type cells from set 1 (each ∑2-type cell combines the termination flags of one 4-b partition) using NOR-logic to limit the fan-in and fan-out to a maximum of four. The ∑2-type cells either continue the comparison for bits of lesser significance if all four inputs are 0s, or terminate the comparison if a final decision can be made.

Set 3 consists of ∑3-type cells, which are similar to

∑2-type cells, but can have more logic levels, different inputs, and carry different triggering points. A∑3-type cell provides no comparison functionality; the cell's sole purpose is to limit the fan-in and fan-out regardless of operand bit width. To limit the ∑3-type cell's local interconnect to four, the number of levels in set 3 increases if the fan-in exceeds four. Set 3 provides functionality similar to set 2 using the same NOR logic to continue or terminate the bitwise comparison activity. If the comparison is terminated, set 3 signals set 4 to set the left bus and right bus bits to 0 for all bits of lower significance.

Set 4 consists of Ω-type cells, whose outputs control the select inputs of Ω -type cells (two-input multiplexors) in set 5, which in turn drive both the left bus and the right bus. For an Ω-type cell and the 4-b partition to which the cell belongs, bitwise comparison outcomes from set 1 provide information about the more significant bits in the cell's Ω type cells.

The number of inputs in the Ω-type cells increases from left to right in each partition, ending with a fan-in of five. Thus, the Ω type cells in set 4 determine whether set 5 propagates the bitwise comparison codes. Table III shows a sample 16-b comparison to clarify (5) using (1)–(4). Set 5 consists of N Ω -type cells (two-input, 2-b-wide multiplexers). One input is (Ak, Bk) and the other is hardwired to "00." The select control input is based on the Ω-type cell output set 4. We define the 2-b as the left-bit code (Ak) and the right-bit code (Bk), where all left-bit codes and all right-bit codes combine to form the left bus and the right bus, respectively. The output F1,0 k denotes the "greater-than," "less-than," or "equal to" final comparison decision Essentially, the 2-b code F1,0 k can be realized by OR-ing all left bits and all right bits separately as shown in fig2.

## IV. PROPOSED 8 BIT COMPARATOR:

In this section, The Proposed comparator design is same as the decision module. The modify the comparison resolution module as shown in fig.4

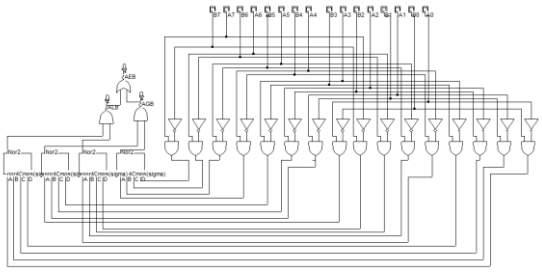


Fig. 4 Design of Proposed 8 Bit Comparator using

## V. SIMULATION RESULTS

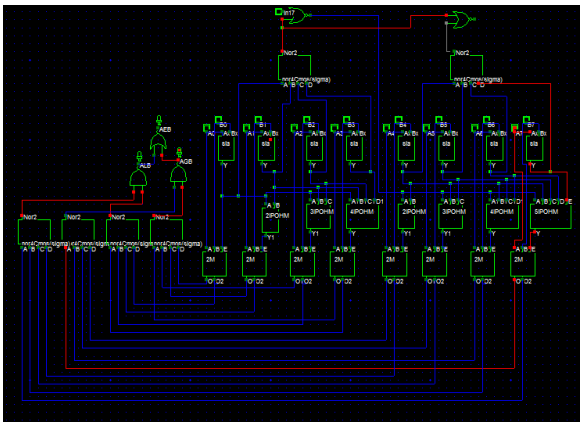


Fig. 5.1 Design of 8 Bit Comparator Using a Parallel Prefix Tree using in DSCH Tool.

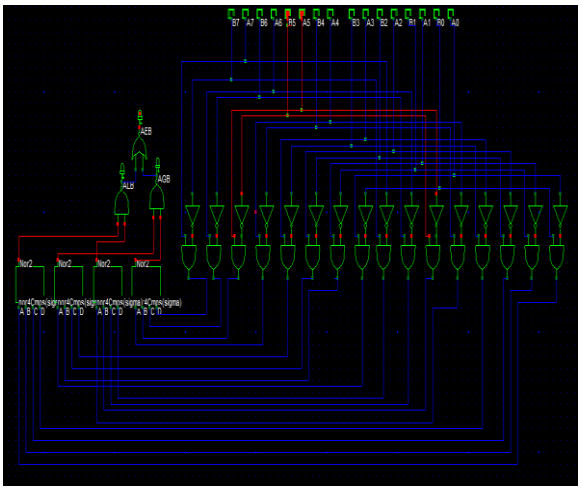


Fig. 5.2 Design of Proposed 8 Bit Comparator using DSCH Tool

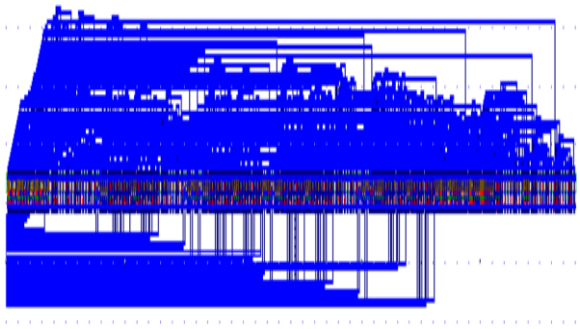


Fig. 5.3 Design of 8 Bit Comparator Using a Parallel Prefix Tree layout using in Microwind Tool in 90nm.

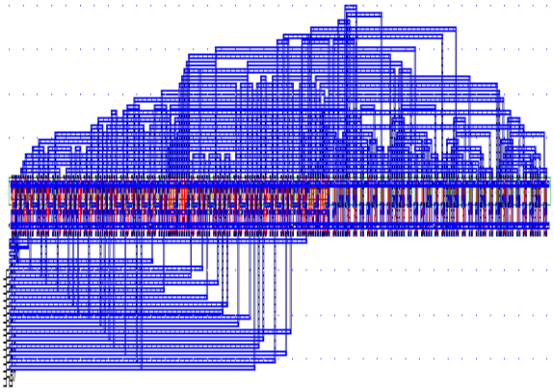


Fig. 5.4 Design of Proposed 8 Bit Comparator Using a layout using in Microwind Tool in 90nm.

TABLE III- COMPARATOR WITH 8 BITS AT DIFFERENT TECHNOLOGY

| | Scalable 8 bit Comparator | | | Proposed 8 bit Comparator | | |
|---|---|---|---|---|---|---|
| Technology | 0.18µm 1.95 V | 0.120µm 1.95 V | 90µm 1.95 V | 0.18µm 1.95 V | 0.120µm 1.95 V | 90µm 1.95V |
| Area | 38211 | 8796.5 | 6108.7 | 12322 | 2838.8 | 1971.4 |
| Power | 1.342 mW | 0.193 mW | 93.67 µW | 0.217 mW | 20.633 µW | 16.716 µW |

## V. CONCLUSION

In this paper, we presented a scalable high-speed low-power comparator using regular digital hardware structures consisting ABDEL-HAFEEZ et al.:

SCALABLE DIGITAL of two modules: the comparison resolution module and the decision module. These modules are structured as parallel prefix trees with repeated cells in the form of simple stages that are one gate level deep with a maximum fan-in of five and fan-out of four, independent of the input bit width. This regularity allows simple prediction of comparator characteristics for arbitrary bit widths and is attractive for continued technology scaling and logic synthesis.

## VI. REFERENCES

[1] H. J. R. Liu and H. Yao, High-Performance VLSI Signal Processing Innovative Architectures and Algorithms, vol. 2. Piscataway, NJ: IEEE Press, 1998.

[2] Y. Sheng and W. Wang, "Design and implementation of compression algorithm comparator for digital image processing on component," in Proc. 9th Int. Conf. Young Comput. Sci., Nov. 2008, pp. 1337–1341.

[3] H. Suzuki, C. H. Kim, and K. Roy, "Fast tag comparator using diode partitioned domino for 64-bit microprocessor," IEEE Trans. Circuits Syst. I, vol. 54, no. 2, pp. 322–328, Feb. 2007.

[4] A. H. Chan and G. W. Roberts, "A jitter characterization system using a component-invariant Vernier delay line," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 1, pp. 79–95, Jan. 2004.