1. N.PRASANNA 1.H.SUMITHA,ASSISTANT PRO

2. DEPT.OF E.C.E IN SRIT,ANANTAPUR

Design of High Speed CMOS Comparator Using Parallel Prefix Tree

ABSTRACT: This paper Presents a new comparator design is proposed by using parallel prefix tree. Energy efficient and high speed operation of comparators is needed for high speed digital circuits. The comparison outcome of the most significant bit, proceeding bitwise toward the least significant bit only when the compared bits are equal. In this project a 16, 32, 64 bit comparator architectures is designed by using parallel prefix structure. This project evaluates the successful results as per requirement and specifications. In existing system, the parallel prefix structure is designed for 16, 32 and 64 bit architectures and the reports from the Xilinx tool concludes that for every bit range doubles the delay, memory, LUT and power has not doubled up to the mark. But In the proposed design of my project, each and every element in the parallel prefix structure will be replaced by universal logic (multiplexer). By performing this modification in the architecture will leads to reduction in POWER CONSUMPTION and DELAY .

Keywords: Parallel prefix tree structure high fan in,high fan out, Bitwise competition logic(BCL).

1. INTRODUCTION

A high speed comparator is a very basic and useful arithmetic component of digital systems. Comparator is a major fundamental element in most digital circuits. The main advantages of this design are high speed and power efficiency, maintained over a wide range. Comparators are key design element for a wide range of applications like parallel testing, signature analyzer, built- in self- test circuits, graphics and image/signal processing. The design of high-speed, low power, and area-efficient comparators. comparison is a fundamental operation digital processors. There are several approaches to designing CMOS comparators, each with different operating speed, power consumption, and circuit complexity. The digital comparator place an important role which compares two input voltage and generates which is greater/lesser or equal.

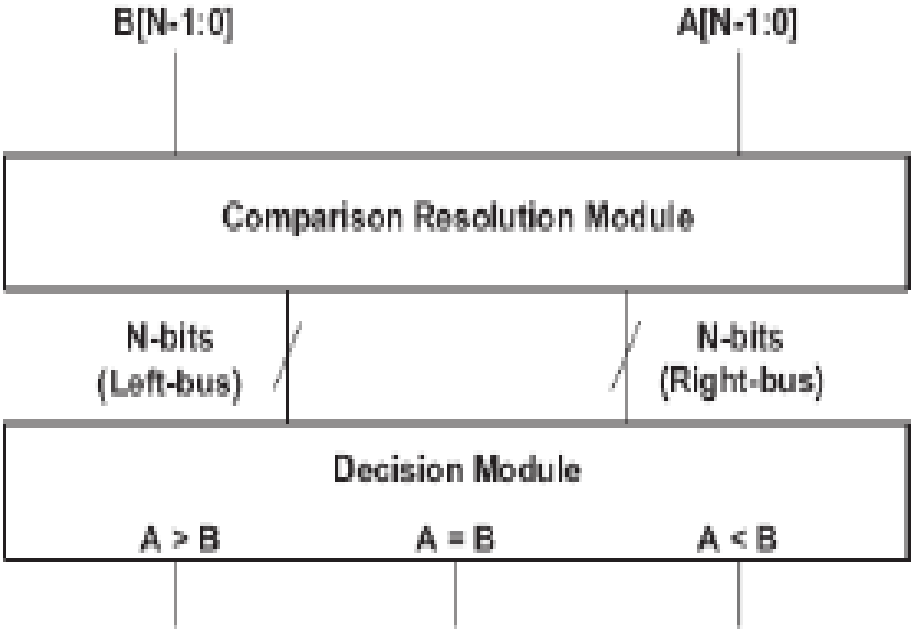
Comparator designs improve scalability and reduce comparison delays using a hierarchical prefix tree structure composed of 2-b comparators. These structures require log2N comparison levels, with each level consisting of several cascaded logic gates. However, the delay and area of these designs may be prohibitive for comparing bit operands. The prefix tree structure's area and power consumption can be improved by leveraging two-input multiplexers(instead of 2-b comparator cells) at each level and generate-propagate logic cells on the first level (instead of 2-b adder cells), which takes advantage of one's complement addition. Using this logic composition, a prefix tree requires six levels for the most common comparison bitwidth of 64 bits, but suffers from high power consumption.

One can implement the comparator by flattening the logic function directly. For the comparators with longer inputs, circuit complexity increases drastically, and the operating speed is degraded accordingly. Another way to designing the comparator is employing a parallel adder. In this approach, the adder becomes the major factor limiting the operating speed. One design uses all-N transistor (ANT) circuits to compensate for high fan in with high pipeline throughput. Furthermore, the structure can perform only “greater-than” or “less-than” comparisons and not equality. To improve the speed and reduce power consumption, several designs rely on pipelining and power-down mechanisms to reduce switching activity with respect to the actual input operands’ bit values. A 64-bcomparator requires only three pipeline cycles using a multiphase clocking scheme. However, such a clocking scheme may be unsuitable for high-speed single-cycle processors because of several heavily loaded global clock signals that have high-power transition activity. Additionally, race conditions and a heavily constrained clock jitter margin may make this design unsuitable for wide-range comparators. An alternative architecture leverages priority encoder magnitude decision logic with two pipelined operations that are triggered at both the falling and rising clock edges to improve operating speed and eliminate long dynamic logic chains. Other architectures use a multiplexer-based structure to split a 64-b comparator into two comparator stages: the first stage consists of eight modules performing 8-b comparisons and the modules’ outputs are input

into a priority encoder and the second stage uses an 8-to-1 multiplexer to select the appropriate result from the eight modules in the first stage. Similarly, other energy-efficient designs leverage schemes to reduce switching activity. Compute-on demand comparators compare two binary numbers one bit at a time, rippling from the most significant bit (MSB) to the least significant bit (LSB). The outcome of each bit comparison either enables the comparison of the next bit if the bits are equal, or represents the final comparison decision if the bits are different. Thus, a comparison cell is activated only if all bits of greater significance are equal. Although these designs reduce switching activity, they suffer from long worst case comparison delays for wide worst case operands. To reduce the long delays suffered by bitwise ripple designs, an enhanced architecture an algorithm that uses no arithmetic operations. This scheme detects the larger operand by determining which operand possesses the leftmost 1 bit after pre-encoding before supplying the operands to bitwise competition logic (BCL) structure. The BCL structure partitions the operands into 8-b blocks and the result for each block is input into a multiplexer to determine the final comparison decision. Due to this BCL-based design's low transistor count, this design has the potential for low power consumption, but the pre-encoder logic modules preceding the BCL modules limit the maximum achievable operating frequency. In addition, special control logic is needed to enable the BCL units to switch dynamically in a synchronized

fashion, thus increasing the power consumption and reducing the operating frequency.

This structure consists of two basic modules: Comparison resolution module and decision module. The comparison resolution module divides two input N- bit arrays to be compared into two busses namely left bus and right bus each of N bits wide respectively. The decision module in turn decides whether equal, less than or greater than relationship exists between applied inputs for comparison. To overcome some of the drawback present in the above designs (such as higher power consumption, multi cycle computation, unsuitable custom structures for scaling, irregular VLSI structures, and irregular transistors), parallel prefix structure based comparator design provides fast, scalable, wide range, and power efficient algorithm.

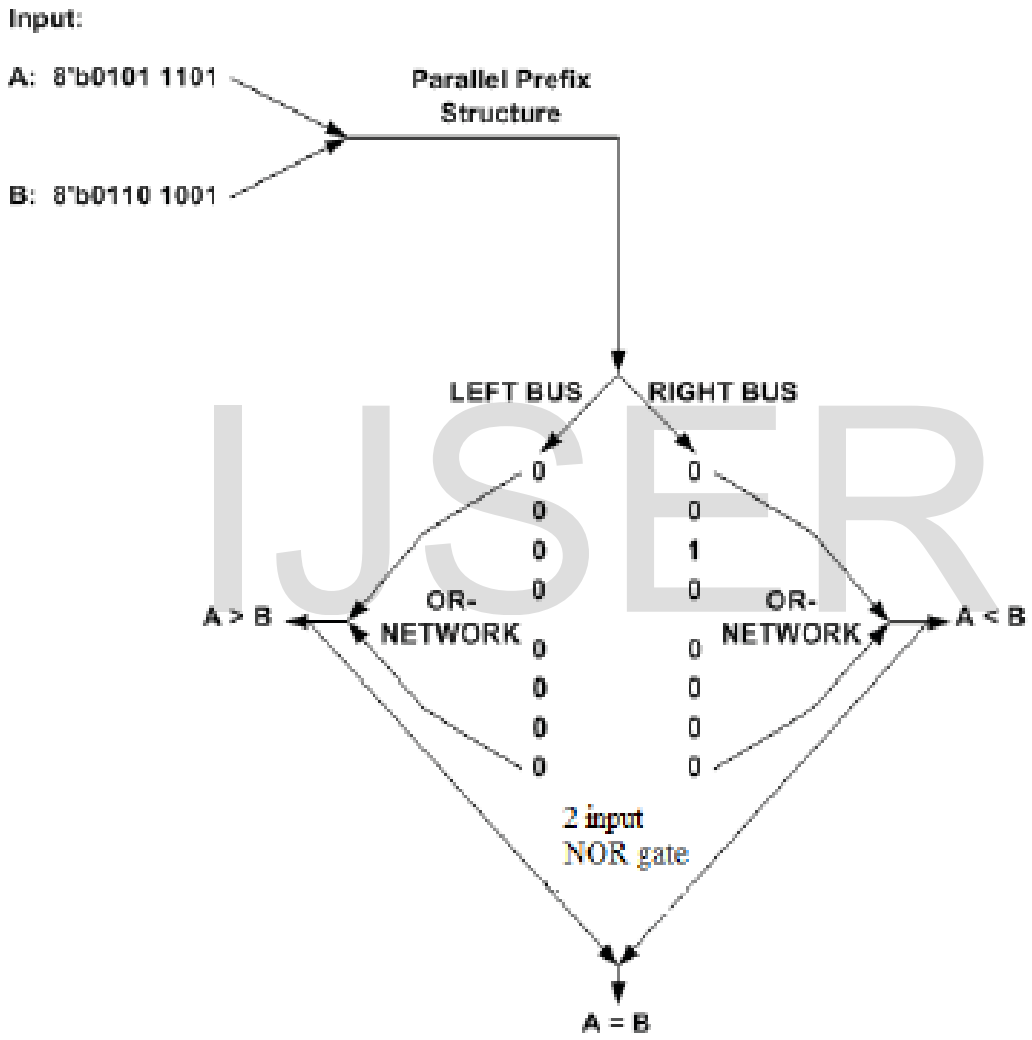


.

Fig 1: Block diagram of our comparator architecture, consisting of a comparison resolution module connected to a decision module

The comparison resolution module is a novel MSB-to- LSB parallel prefix tree structure that performs the bitwise comparison of two N bit operands (A & B) entered into the comparator. The parallel structure encodes the bitwise comparison results to two N bit buses called left bus and right bus. The bitwise comparison of equal bits sets _0 in both the buses. If the bitwise comparison of unequal bits occur, any of the buses (A or B) sets to _1 and the bitwise comparison stops immediately by setting _0 in the remaining bits present in the buses. The decision module produces the result of comparison of the input operands based on the signals from the left and right buses. The possible results from the decision module are (i) both are equal (A= B), (ii) A is greater than B (A>B), (iii) A is lesser than

Figure 2: Example 8-b Comparison



Let the two 8-bit binary numbers be A and B. A = 0101 1101 and B = 0110 1001. 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.

We partition the structure into five hierarchical prefixing sets, in Figure 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. The below symbols are usually used in implementation.

Table 1: 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
$\prod$	Bitwise AND
$\sum$	Bitwise OR
T{*}	Logic function of cell type*
COMP{*}	Complement function of set*

Each symbol is represented by the corresponding logic gates. The symbol will perform the operation represented by the logic

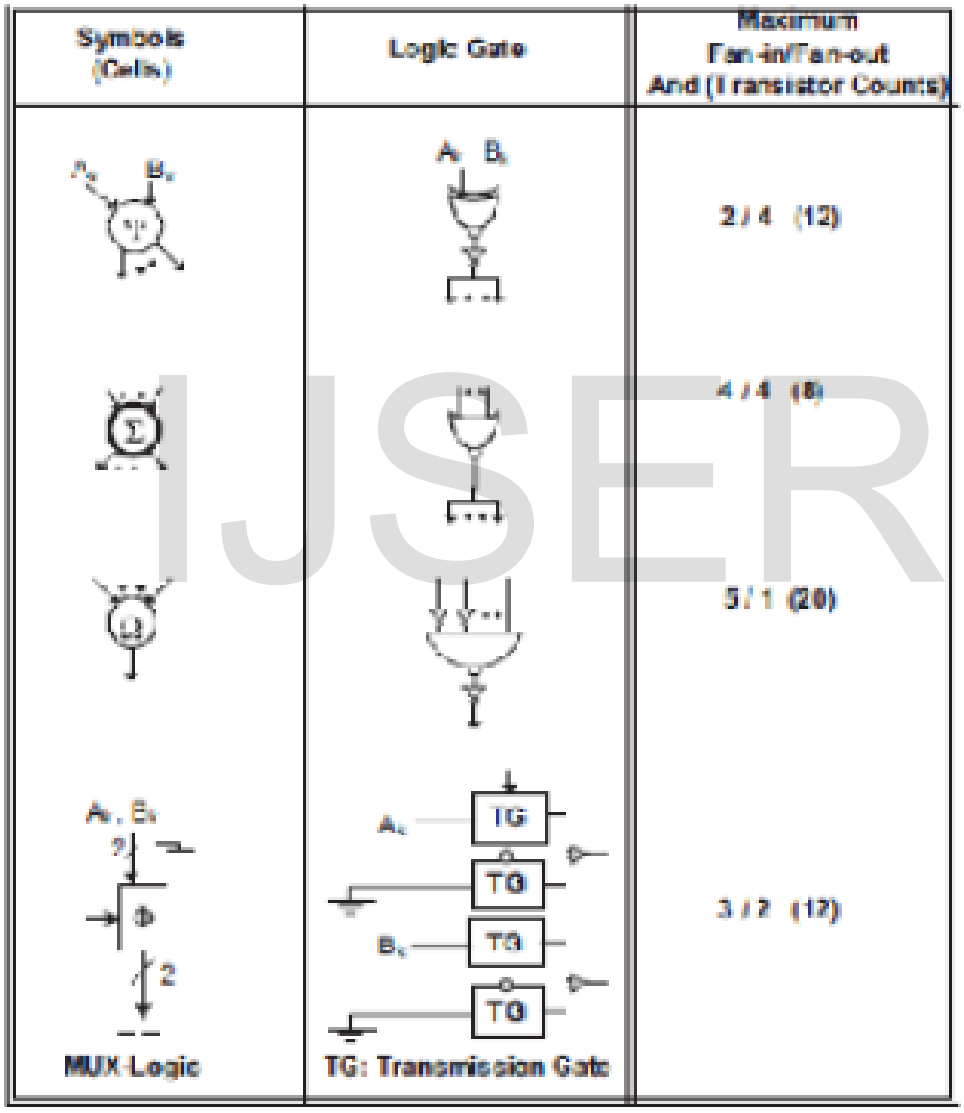
gate and maximum fan in and fan outs are indicated as 2/4 i.e., the maximum number of inputs are 2 and the maximum number of outputs are 4. These symbols are used to implement the several sets of operations. All cells (components) within each set operate in parallel, which is a key feature to increase 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 bit width 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.

COMPARATOR DESIGN DETAILS

We partition the structure into five hierarchical prefixing sets, where as each set performs a exact function whose output serves as input to the next set, in hope of the fifth set produces the output on the left bus and the right bus Every part of cells components within each set operate in parallel were as it's a key feature to increase operating speed while minimizing the transitions to a minimal set of left most bits needed for a correct decision. This prefixing set structure bounds the components fan-in and fan-out regardless of comparator bit-width and eliminates heavily loaded global signals with parasitic

components, thus improving the operating speed and reducing power consumption.

Figure 3: Logic Gate Representations for Symbols



Set 1 compares the N -bit operands A and B bit-by-bit, using a single level of N * Type cell. The *Termination flag Dk to cells

in sets 2 and 4, indicating whether the computation should terminate. These cells compute (where $0 \leq k \leq N-1$) .

ARCHITECTURE OF 16-BIT COMPARATOR USING PARALLEL PREFIX TREE

In comparison resolution module four sets are used and each set performs different gate operations. In set1 the XOR operation is performed with A and B inputs and the output of the gates is D which is 16 down to 0. Set2 perform the NOR operation. The set1 output is given as input for set2 and each gate has 4 inputs with one output. Set3 is similar to set2 (XOR operation). The inverted inputs are applied to NAND gate and its output is also inverted. In decision module set5 performs the multiplexer operation.

For an Ω type cell and the 4-b partition to whichthe cell belongs, bitwise comparison outcomes from set 1 provide information about the more significant bits in the cell's Ω type cells, Set 5 consists of N Φ -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 from 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 Φ-type cells compute (where $0 \leq k \leq N-1$).compute (where $0 \leqslant k \leqslant N-1$).The

$$\Phi = f_k^{1,0} = y_k \times m_k + \bar{y}k \times (00)$$

$$F_k^{1,0} \begin{cases} 00, & \text{for } A_k = B_k \\ 01, & \text{for } A_k < B_k \\ 10, & \text{for } A_k > B_k. \end{cases}$$

Essentially, the 2-b code can be realized by OR-ing all left bits and all right bits separately, as shown in the decision module, using an OR gate network in the form of NOR NAND gates yielding a more optimum gate structure 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 Φ-type cells compute (where $0 \leq k \leq N - 1$. From left to right, the first four Σ3-type cells in set 3 combine the 4- b partition comparison outcomes from the one, two, three, and four 4-b partitions of set 2. Since the fourth Σ3-type cell has a fan-in of four, the number of levels in set 3 increases and set 3's fifth Σ3-type cell combines the comparison Out comes of the first 16 MSBs with a fan-in of only two and fan out of one.

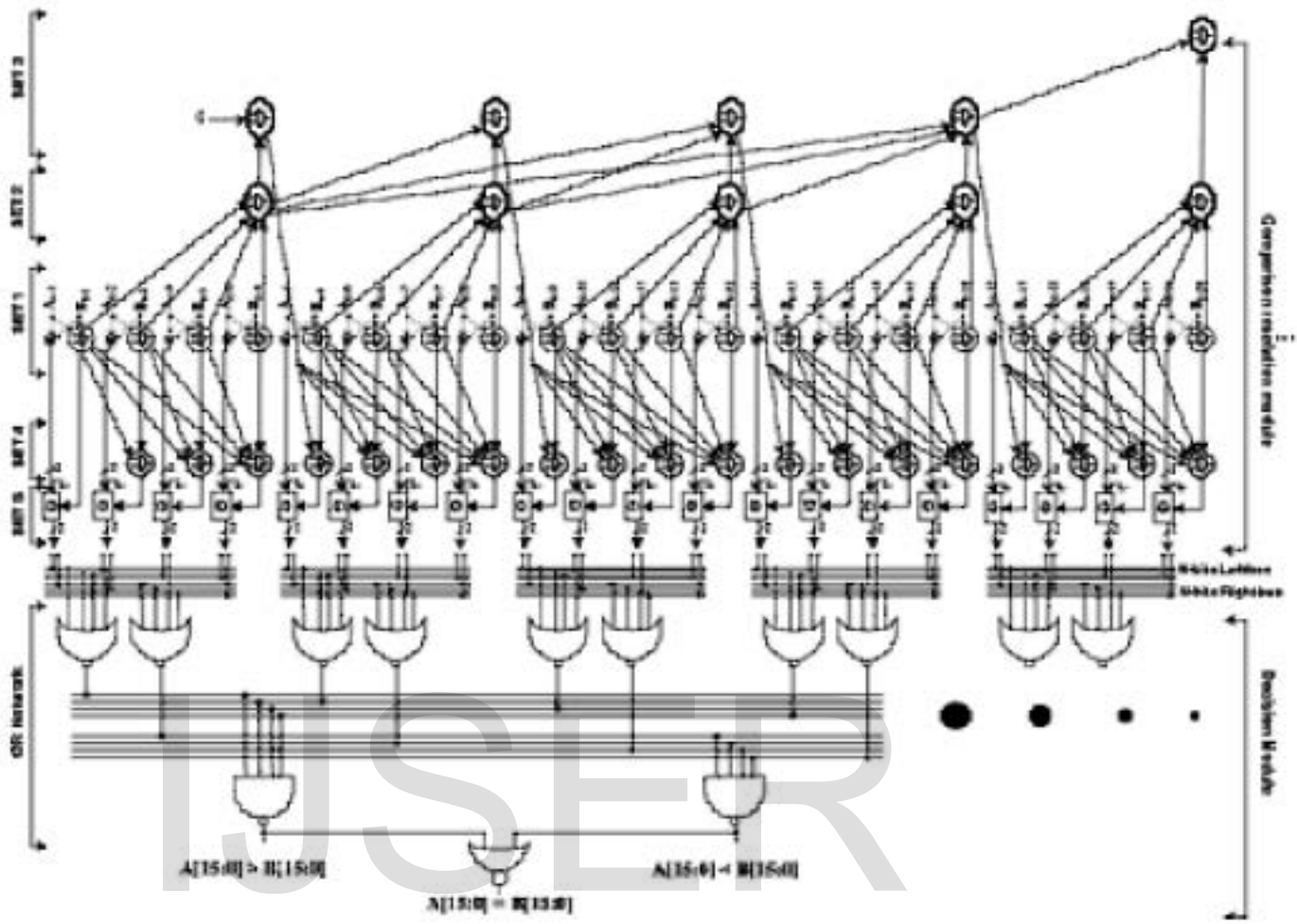


Fig. 3. Implementation details for the comparison resolution module (sets 1 through 5) and the decision module.

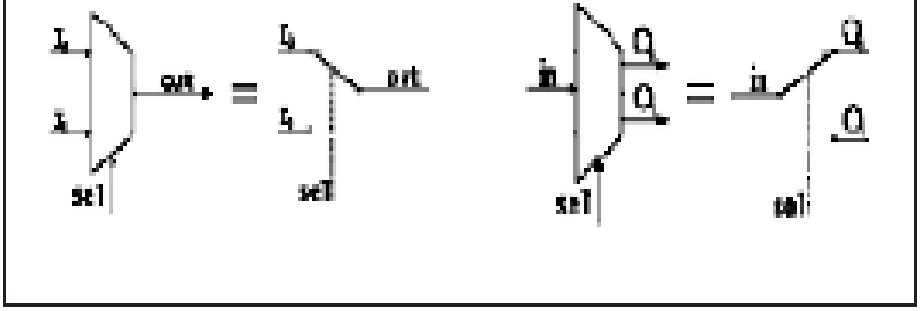
PROPOSED ARCHITECTURE

Replacing With Multiplexer Logic

In this project the switching logic and the main block design is carried out by using mux logic to perform low power operations because , In electronics, a multiplexer is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to

the output. A multiplexer is also called a data selector. An electronic multiplexer can be considered as a multiple-input single ouput'input,single out.

Figure 5: Switch



.

ADVANTAGES BY USING MULTIPLEXR BASED IN PARALLEL PREFIX TREE

Low power consumption by replacing the needed logics by multiplexer, because multiplexer operates at very low power switching transitions compared to another logical gates. Low delay compared to normal based comparator, less area and less LUT compared to existing system.

CONCLUSION

In this project a Design Of High Speed CMOS Comparator Using Parallel Prefix Tree using regular digital hardware structures consisting of two modules: the comparison resolution module and the decision module. This regularity allows simple prediction of comparator characteristics for arbitrary bit widths and is attractive for continued technology Scaling and logic

synthesis. These modules are structured as parallel prefix trees by using a normal flow.

Family	Existed one using logic gates 16 bit	Proposed one using mux 16bit
Spartan=3 (XC3S50) (PQ208) Speed=-4	LUT=57 Time Delay=21.774ns	LUT=47 Time Delay=18.739ns

Future work will include additional circuit optimizations to further reduce the power dissipation by adapting dynamic and analog implementations for the comparator resolution module and a high-speed zero-detector circuit for the decision module. Given that our comparator is composed of two balanced timing modules, the structure can be divided into two or more pipeline stages with balanced delays, based on a set structure, to effectively increase the comparison throughput.

REFERENCES

[1] S. Abdel-Hafeez, "Single rail domino logic for four-phase clocking scheme," U.S. Patent 6 265 899, Oct. 20, 2001.

[2] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, San Mateo, CA: Morgan Kaufmann, 2004.

[3] J. P. Uyemura, *CMOS Logic Circuit Design*, Norwood, MA: Kluwer, 1999.

[4] J. E. Stine and M. J. Schulte, "A combined two's complement and floating-point comparator," in *Proc. Int. Symp. Circuits Syst.*, vol. 1. 2005, pp. 89–92.

[5] S.-W. Cheng, "A high-speed magnitude comparator with small transistor count," in *Proc. IEEE Int. Conf. Electron., Circuits, Syst.*, vol. 3. Dec. 2003, pp. 1168–1171.

[6] 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.

[7] B. Parhami, "Efficient hamming weight comparators for binary vectors based on accumulative and up/down parallel counters," *IEEE Trans. Circuits Syst.*, vol. 56, no. 2, pp. 167 171, Feb. 2009.

[8] 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.

[9] D. R. Lutz and D. N. Jayasimha, "The half-adder form and early branch condition resolution," in *Proc. 13th IEEE Symp. Comput. Arithmetic*, Jul. 1997, pp. 266–273.

[10] J. Hensley, M. Singh, and A. Lastra, "A fast, energy efficient zcomparator," in *Proc. ACM Conf. Graph. Hardw.*, 2005, pp. 41–44.

[11] V. N. Ekanayake, I. K. Clinton, and R. Manohar, "Dynamic significance compression for a low-energy sensor network asynchronous processor," in *Proc. 11th IEEE Int. Symp. Asynchronous Circuits Syst.*, Mar. 2005, pp. 144–154.

[12] H.-M. Lam and C.-Y. Tsui, "High-performance single clock cycle CMOS comparator," *Electron. Lett.*, vol. 42, no. 2, pp. 75–77, Jan. 2006.

[13] J.-Y. Kim and H.-J. Yoo, "Bitwise competition logic for compact digital comparator," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2007, pp. 59-62.