IEEE Int. Conf. Neural Networks & Signal Processing
Nanjing, China, December 14-17, 2003

LOGARITHMIC COMPLEXITY IMPLEMENTATION FOR INTEGER COMPARATOR

Haiping Sun, Jinqing Cai, Wei He, and Minglun Gao

Institute of VLSI Design
Hefei University of Technology
Hefei, Anhui, 230009, China

ABSTRACT

A logarithmic complexity implementation for the integer comparator is presented. The relation between unsigned integers to be compared and their comparison result is analyzed, and then a logarithmic complexity algorithm is inferred, which takes the advantage of both speed and area. The algorithm is also extended for comparing two signed integers in two's complement. The complexities of the basic algorithm in terms of time and hardware are given. The experimental results of several concrete schemes are also discussed.

1. INTRODUCTION

The integer comparison is frequently performed in integer unit to set/reset the program flags and/or to determine which branch is taken. The time for integer comparison instructions influences the cycle time of the integer unit.

The integer comparison is also a necessary step for the floating-point addition/subtraction. By this step, the base exponent, which is the exponent of the larger floating-point number, is picked up. Then the significand of the smaller floating-point number is shifted to be in the form of the base exponent. After the shift, the significand addition/subtraction is performed. Delay of comparison is a considerable part of the floating-point operation that affects its computation speed.

Implementation of the integer comparator is significantly complicated, since its result is dependent on each input bit. A straightforward method is based on the priority principle, by which a priority comparator is inferred [1]. The major drawback of the priority comparator is that the fan-in and fan-out are too large; as a result, the logic is too complex and slow to be applicable when the bit width of input strings to be compared is large.

In this paper, a logarithmic complexity algorithm to compare two integers is inferred from the relation between two input bit strings and their comparison result.

The paper is organized as follows. First, we analyze the relation between two input unsigned integers and their comparison result. Then an efficient and fast algorithm at the logarithmic complexity level is proposed, and circuits for unsigned integers are implemented in several concrete schemes. All the schemes are captured in Verilog HDL, and its logic is synthesized by using Synopsys' synthesis tool. Hereafter, the algorithm is extended to compare two signed integers. Lastly, the experimental data for the unsigned integer comparator are presented and discussed.

2. LOGARITHMIC COMPLEXITY ALGORITHM

2.1. Fundament

The integer comparator in the integer unit and the floating-point adder/subtracter unit is shown in Fig. 1, where the input bit-string A and B of the comparator represent the two integers to be compared.

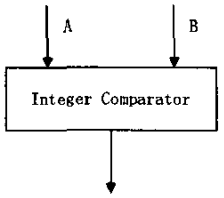

Fig. 1 Integer Comparator

Now we analyze the relation between the input bit strings and their comparison result. An assumption is made that the input bit strings are the length of 2^k bits. Table 1 presents some pairs of 8-bit strings with their comparison results.

As to the former three pairs, their comparison results are the same as that of their right-hand halves respectively. This is because that the left-hand halves are equal.

While as to the latter two pairs, their comparison results are the same as that of their left-hand halves respectively, instead of that of their right-hand halves. This is just because that the left-hand halves are unequal.

After in-depth studying, a conclusion may be reached:

Once the left-hand halves of the input strings are equal, the comparison result is that of their right-hand halves, otherwise it is that of their left-hand halves; and the rule is kept on the two smaller sub-strings recursively. This conclusion is very simple and easy to be understood and proved.

Table 1. Strings with their comparison result

Input String A	Input String B	Result
00001001	00001011	Less than
01001001	01001000	Greater than
11011011	11011011	Equal to
11010011	11111011	Less than
11010011	10010011	Greater than

2.2. Algorithm

The analysis of the relation between the input strings and their comparison result implies that the proposed comparison approach is a recursive process.

A two-bit attribute is attached to each pair of strings. One bit represents the flag *EQU* that denotes whether the two strings are equal. If they are equal, the flag *EQU* takes the value *TRUE*; otherwise, it takes *FALSE*. The other bit represents the flag *LESS*, which denotes whether the first integer is less than the second under the unequal condition. Flag *LESS* is valid only when the compared integers are unequal.

For clarity, the pair of inputs to be compared is denoted as $X=(a,b)$, its left-hand halves as $X_{left}=(a_{left},b_{left})$, its left-hand halves as $X_{right}=(a_{right},b_{right})$.

The equations for the recursive process, which are derived from the above conclusion, may be represented as:

$$EQU_X = EQU_{X_{left}} \cdot EQU_{X_{right}}$$

$$LESS_X = \begin{cases} LESS_{X_{right}} & EQU_{X_{left}} = TRUE \\ LESS_{X_{left}} & EQU_{X_{left}} = FALSE \end{cases}$$

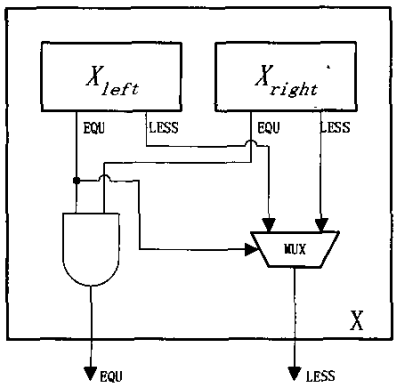

Fig. 2 Diagram for recursive equations

The initial condition for the recursive equations occurs when each of the pair $X=(a,b)$ is the length of one bit. The initial condition is given as:

$$EQU_X = \overline{a \oplus b}$$

$$LESS_X = \bar{a} \cdot b$$

The diagram for the recursive process and the initial condition is shown in Fig. 2 and Fig. 3 respectively. The recursive principle this paper applied was also applied by V.G. Oklobdzija [2] to implement leading zeros detector.

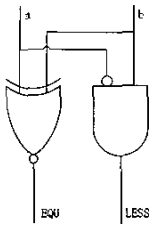

Fig. 3 Diagram for initial condition

2.3. Complexity Analysis

According to the assumption that n, which is the length of the input bit string, takes 2^k form, the comparator circuit has k-level logic, where $k=log_2 n$. So its time complexity is at $O(logn)$ level.

The structure of the initial level is distinctly different from its subsequent levels.

The initial level logic has 2^{k-1} blocks. Each block consists of an *XNOT* gate, an *AND* gate. The ith ($1 \le i \le k$) level logic has 2^{k-i} blocks. Each block consists of an *AND* gate and a two-way *MUX*. The whole LZD circuit consists of 2^k-1 *MUX*s, 2^{k+1}-1 *AND* gates, and 2^k *XNOR* gates.

2.4. Extension for Signed Integers

Typically, unsigned integers usually are represented in two's complement. In such a case, the most left bit (i.e., the most significant bit) is the sign bit. Value of 1 implies a negative integer; value of 0 implies a non-negative integer.

If two integers are with different sign-bits, the comparison is independent of their magnitudes, and the comparison result must be that the non-negative integer is greater than the negative one. So the comparison of signed integers is effectively performed on the magnitudes only when they have the same sign-bit. In two's complement representation, two integers are with the same sign-bit, either negative ones or non-negative ones, the bit-string for the larger integer is greater than the other one when the strings (including the sign-bit) are regarded as unsigned bit-string. Based on this fact, the circuit illustrated in Fig. 4 is inferred.

The output *EQU* of the whole comparator is the one of the bit-string comparator, while the output *LESS* is

chosen from the one of the bit-string comparator and the sign of integer *A* by the sameness of the two input integers. If the signs of two input integers are same, the final comparison result is the output of the bit-string comparator; otherwise it is the sign of integer *A*.

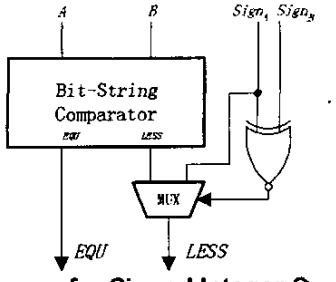

Fig. 4 Diagram for Signed Integer Comparator

3. EXPERIMENT

3.1. Experiment Setup

We have implemented several sets of prototypes of integer comparator of size 32, 64, and 128 bits in different schemes. To compare their performance, an experimental bench, as shown in Fig. 5, is constructed. All the schemes are captured in Verilog HDL. Their logics are synthesized by using Synopsys' tools. The target technology is Taiwan Semiconductor Manufacturing Company (TSMC) 0.25 m *slow* (the worst case) and *typical* libraries. Wide range clock frequencies, from 100 MHz to 500 MHz, are applied for logic synthesis. Comparator is a pure combinational logic circuit, but the input strings and their comparison result are clocked respectively in the bench. The time *T*, which is what we are interested in, is shown in Fig. 5.

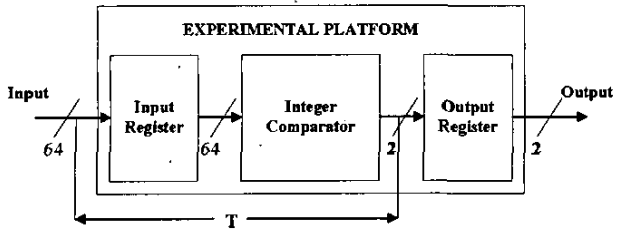

Fig. 5 Comparator experimental bench

3.2. Contrastive Schemes

Four contrastive schemes for unsigned integer comparison are proposed to explore the performance in terms of time and area.

The first scheme, called scheme *O*, is described as paragraphs above, where *LESS=1* indicates that the string *A* is less than the string *B*.

The second scheme, called scheme *I*, is an inverse encoding scheme against the scheme *O*, where *LESS=0* indicates that the string *A* is less than the string *B*.

The third scheme, called scheme *L0*, is an interleave encoding scheme with the mixture of the scheme *O* and *I*, one level using the original encoding scheme, the next level using the inverse encoding scheme, and vice versa. Within the scheme, at the initial level, *LESS=0* indicates that the string *A* is less than the string *B*.

The fourth scheme, called scheme *L1*, is similar to scheme *L0*. But at the initial level, *LESS=1* indicates that the string *A* is less than the string *B*.

The comprehensive experimental results are shown in Table 2. The unit of latency is nanosecond, and the unit of area is the square of micron. By studying these data, such a conclusion may be drawn now: Scheme *O* takes the overall optimal performance; under the condition of 250 MHz and 400 MHz frequencies, in terms of area, scheme *L1* is obviously advantageous.

Table 2. 64-bit unsigned integer comparator targeting *slow* library

Frequency	Scheme	Latency	Cell	Area
100M Hz	*O*	4.88	357	7581
	I	6.27	357	6238
	L0	5.37	363	6394
	L1	5.14	363	7684
250M Hz	*O*	3.60	405	8830
	I	3.66	439	7995
	L0	3.69	460	8266
	L1	3.64	416	8853
400M Hz	*O*	2.23	403	13467
	I	2.23	423	14031
	L0	2.40	395	13092
	L1	2.23	397	12851

4. CONCLUSION

In this paper, we have described a logarithmic complexity recursive algorithm to compare unsigned integers and its extension for signed integers. Several concrete schemes for unsigned integers are implemented with TSMC 0.25μm technology. The circuits are captured in Verilog HDL, and then the logic is synthesized by using Synopsys' tool. The resulting circuit has remarkable performance, which improves the critical path in both the integer unit and the floating-point adder.

5. REFERENCES

[1] M.M. Mano, *Computer System Architecture (Third Edition)*, Prentice Hall, 1993.

[2] V.G. Oklobdzija, "An Implementation Algorithm and Design of a Novel Leading Zero Detector Circuit", Conference record of the Twenty-Sixth Asilomar Conference on Signals, Systems and Computers, vol. 1, pp. 391-395, 1992.