THPM 18.8

A FAST TAGGED SORTER USED IN 100/10 MBPS ROUTERS §

Chua-Chin Wang† , *Hsin-Long Wu, & Shao-Ku Huang*‡

Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan

ABSTRACT

This paper presents a very fast circuit for a tagged sorter which has been integrated in 100M/10M bps routers to resolve the re-ordering problem of incoming packets. The utilization of both edges of the clock is presented, which can drastically improve the clock speed and the throughput.

1. INTRODUCTION

The performance of the applications with heavy I/O load such as routers might not be as fast as the their internal processing elements (PEs). One of the critical paths is to locate where the packet with maximal (or minimal) priority is [3]. It is often deemed as a sorting problem. In addition, the detection of whether the queue is full or empty is not efficiently resolved. A simple modification is presented in this paper, which not only improves the design up to 125 MHz, but also resolves the problem of detecting the queue's status without hardware penalty.

2. THEORY OF THE TAGGED SORTER

2.1. Operations of the tagged sorter

Many prior sorting algorithms were proposed to reduce the typical $O(n)$ complexity sorting problem to $O(\log n)$ [1]. When it comes to the hardware realization, many of them become either impractical or costly [3]. A tagged up/down sorter was proposed by Moore *et al.* [2]. The packets are assumed to be in a (key, data) pair format. Notably, all of the sorting elements contain infinity (or a sufficiently large number) initially. The functions of a tagged sorter unit are as follows.

Insertion : On the arrival of an input value on A_n, a copy of C_n is delivered to B_n, and D_n is transferred to A_{n+1}. Then, a comparison takes place between A_n and B_n. $\min(A_n, B_n) \rightarrow C_n$, and $\max(A_n, B_n) \rightarrow D_n$.

Extraction : C_n is either removed or transferred to B_{n-1}, D_n is copied to A_n, and C_{n+1} is transferred to B_n. Thus, a comparison also takes place between A_n and B_n. $\min(A_n, B_n) \rightarrow C_n$, and $\max(A_n, B_n) \rightarrow D_n$.

Ensured FIFO ordering : A simple method is to use a tag bit associated with each packet. When one packet arrives at the input port of the sorter, its tag bit is reset. The tag bit of a packet will be set once if the packet is moved to the right column, i.e., the output queue. However, if the packet is swapped to the left, the packet can be forced to be swapped back and keep its tag bit as set. If it is swapped to the left, then on the next cycle, regardless insertion or extraction, it will be compared with the right packet which was previously physically below it. Hence, a swap is required to maintain the order on the right column of the sorter.

2.2. Fast sorter implementation

Although Moore proposed one design of the sorter unit, his design not only has problems when given 100 Mbps inputs, but also the detection of critical boundary conditions was ignored, i.e., queue full and queue empty. Our proposed implementation is shown in Fig. 1, where only one crossbar switch is needed. However, the most important feature is that both edges of the clock are utilized to enhance the operation.

positive edge : process the insertion and extraction

$$\text{Insertion}: \quad A_n = L_{n-1}; \quad B_n = R_n \quad (1)$$

$$\text{Extraction}: \quad A_n = L_n; \quad B_n = R_{n+1} \quad (2)$$

negative edge : process the comparison and swap

$$\text{if } ((A_n.key < B_n.key) \bigcup A_n.tag == 1)$$

$$(L_n, R_n) = (B_n, A_n), \quad \text{else} \quad (L_n, R_n) = (A_n, B_n) \quad (3)$$

boundary conditions : Referring to Fig. 2, the queue empty condition can be monitored by $R_O.tag$, while the queue full situation can be judged by NANDing the bits in $L_O.key$.

3. PERFORMANCE SIMULATIONS

The result of the proposed implementation is shown in Table 1. The clock cycle time is drastically short-

§This research was partially supported by National Science Council under grant NSC 88-2219-E-110-001 and 89-2215-E-110-014, and Industrial Technology Research Institute under grant ITRI G4-88007-o and T2-89042-f.

†The contact author

‡Mr. Huang is currently the manager of mobile phone section in ITRI.

ened to 8 ns, which indicates that the sorter can correctly operates under a 125 MHz clock. Fig. 3 demonstrates the waveforms of our sorter given a 125 MHz clock. Fig. 4 is the physical layout of our design without pads.

4. CONCLUSION

The exploitation of both the edges of the clock leads to a more efficient implementation. Though there is a tradeoff regarding the chip area, the speed of the sorter possesses a higher priority owing to that most of the sorters are employed in heavy I/O trafficking applications, e.g., routers and ATM switches.

5. REFERENCES

[1]. Y.-C. Lin, "On balancing sorting on a linear array," *IEEE Trans. on Parallel and Distributed Systems,* vol. 4, no. 5, pp. 566-571, May 1993.

[2]. S. W. Moore, and B. T. Graham, "Tagged up/down sorter - a hardware priority queue," *The Computer Journal,* vol. 38, no. 9, pp. 695-703, Sep. 1995.

[3]. D. Picker, and R. D. Fellman, "A VLSI priority packet queue eith inheritance and overwrite," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,* vol. 3, no. 2, pp. 245-252, June 1995.

our implementation		
Area of an element	967 gates	378,754 μm^2
Area of the sorter	7736 gates	3,030,036 μm^2
Minimal cycle time	8 ns	

Table 1 : The features of the our implementation.

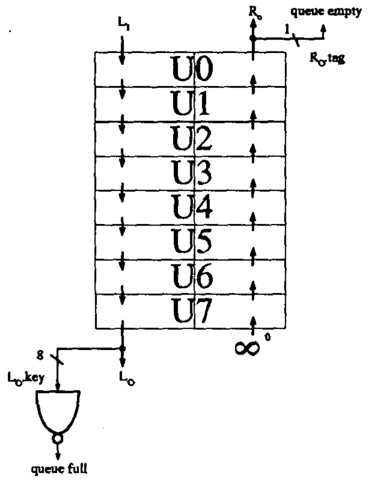


Figure 2: Proposed sorter (n=8)

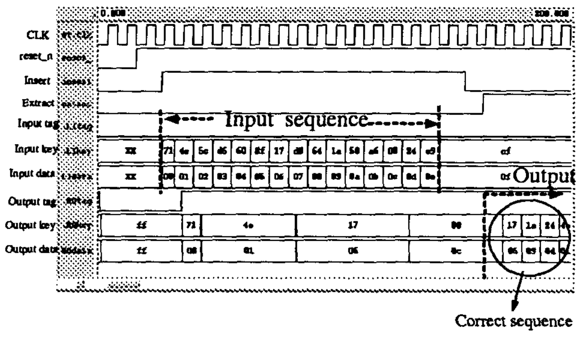


Figure 3: Results using the proposed design

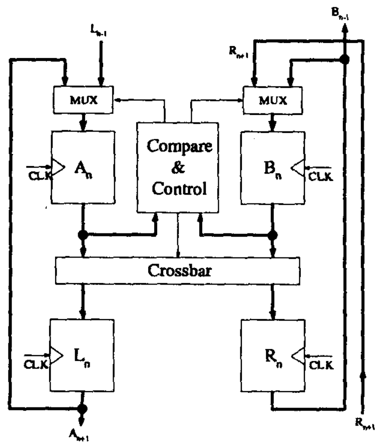
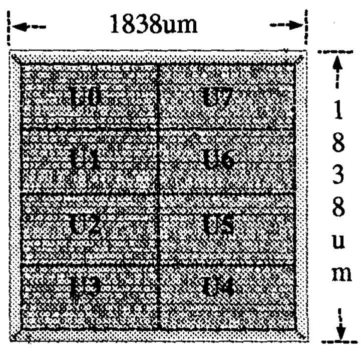


Figure 1: Proposed sorter unit design



Ui : sorter element, i=0,1...7

Figure 4: Chip layout (without pads)