Comments

Comments on "A TCAM-Based Parallel Architecture for High-Speed Packet Forwarding"

Yeim-Kuan Chang, *Member*, *IEEE Computer Soc.*, and Cheng-Chien Su

Abstract—This short report first indicates a design flaw in the contention resolver unit proposed in [1] and then proposes an improved design which is simpler and faster.

Index Terms—System design, contention resolver.

1 Introduction

Contention resolver is required in a parallel or distributed system to resolve the request conflicts in accessing system resources. A complete logic designed for the contention resolver in a TCAM-based parallel search engine is proposed in [1]. We will describe the design in [1] and its flaw and propose an improved design whose logic is simpler and faster.

Consider a generic parallel system that has M selectors and N search engines. Each search engine is prepended with a contention resolver (CR). After receiving a search key, each selector employs some selection criterion to select which search engine the received key will be sent to for processing. If selector x selects the search engine i, a one-bit request signal Req_x^i is set and other request signals Req_x^i for $j \neq i$ are reset. Each selector is given a priority called *hold priority* (*HP*) which is an $(M-1)$-bit value that is a continuous string of k 0s followed by a continuous string of $M-1-k$ 1s for $k=0$ to $M-1$ (e.g., 0011 for $M=5$). As a result, each CR (say CR^i) receives from every selector (say selector x) a set of input signals that are a search key $\overrightarrow{Key_x}$, an $(M-1)$-bit priority $\overrightarrow{HP_x}$, and a 1-bit request signal Req_x^i. Note that a variable of more than one bit wide is indicated by using an arrow on top and $\overrightarrow{Key_x}$ and $\overrightarrow{HP_x}$ have no superscript i because all of the CRs receive the same keys and hold priorities. CR allows only one of the contended requests with the highest priority to proceed and puts the other requesting selectors on hold. A selector that is put on hold will not receive a new search key in the next cycle, but will repeat the requesting process for the same key. To avoid starvation, the on-hold selectors must adjust their priorities as follows: Initially, a selector sets its HP value to zero, i.e., a continuous string of $M-1$ 0s. When a selector receives a *Hold* signal, its HP value is shifted left one bit and a "1" is added in its least significant bit (LSB). The function of the ith contention resolver (CR^i) is implemented by the equations in Table 1.

The highest *HP* value among all the requesting selectors for CR^i is computed in (1) and stored in $\overrightarrow{A^i}$. Signal B_x^i is designed in such a way that if $B_x^i = 1$, then selector x is a candidate selected by CR^i for processing its key $\overrightarrow{Key_x}$. Equations (2) and (3) are designed to guarantee that, when all HP values are zeros, B_x^i for the nonrequesting selectors will not become 1. Equation (2) converts the HP value by setting the LSB of $\overrightarrow{HP_x^i}$ to 1, keeping the other bits unchanged. The converted HP value is stored in $\overrightarrow{H_x^i}$. In (3), we set $B_x^i = 1$ if $\overrightarrow{H_x^i} = \overrightarrow{A_x^i}$ or $B_x^i = 0$ otherwise. Since we only need to take the requesting selectors into consideration for computing B_x^i, (2) obviously needs to be corrected as shown in (2)′.

Equations (4)-(7) determine the final winner (say selector x) among all of the candidate selectors by setting its S_x^i to 1. The search key ($\overrightarrow{Key_x}$) of the winning selector x is then transmitted to search engine i for processing. Let F^i be the vector of $F_M^i \ldots F_1^i$ that is assigned with the value of $B_M^i \ldots B_1^i$. Equation (4) is a conventional M-input priority filter that keeps the least significant set bit of $\overrightarrow{F^i}$ and clears all other bits. Since it is possible that all signals B_x^i for $x = 1 \ldots M$ are zeros when the highest HP is zero, they are useless for selecting the final winning selector. Instead, Req_x signals are used to find the winning selector by building another M-bit priority filter in (5). Equation (6) sets $b^i = 0$ if all B_x^i values are zeros or $b^i = 1$ otherwise. Finally, (7) sets $S_x^i = F_x^i$ if $b^i = 1$ or $S_x^i = G_x^i$ if $b^i = 0$. One special case is that, when no selector requests CR^i, all signals S_x^i for $x = 1 \ldots M$ are set to zeros.

Although the contention resolver designed in [1] seems to be working, we discover one condition that produces an incorrect result, described as follows: Suppose the highest priority among all requesting selectors for CR^i is $0 \ldots 01$ ($M-2$ zeros followed by a one). Let x be the selector whose ID is the smallest among all requesting and nonrequesting selectors with priority zero and y be the selector whose ID is the smallest among all requesting selectors with priority $0 \ldots 01$. If $x < y$, CR^i will set $S_x^i = 1$ because the converted priorities $\overrightarrow{H_x^i}$ and $\overrightarrow{H_y^i}$ have the identical value of $0 \ldots 01$. The correct answer should be $S_y^i = 1$. The example in Table 2 shows the incorrect result of $S_i^i = 1$ at cycle 1, while the correct one should be $S_4^i = 1$. This incorrect result will occur once every three cycles and, thus, CR^i processes only two requests every three cycles.

2 Proposed Contention Resolver Design

The main design flaw of the contention resolver in [1] is that the converted priority (i.e., $\overrightarrow{H_x^i}$) cannot be used to distinguish whether the request signal Req_x^i of a selector is set or not. As a result, a nonrequesting selector may be selected, as shown in the example of Table 2. This design flaw also slows down the speed of the contention resolver because the special case of all requesting selectors having a priority zero must be handled by some additional equations (i.e., (5)-(7)). To solve this design flaw, we

• *The authors are with the Department of Computer Science and Information Engineering, National Cheng Kung University, No. 1, Ta-Hsueh Road, 701 Tainan, Taiwan, ROC.*
E-mail: ykchang@mai.ncku.edu.tw, p7894104@ccmail.ncku.edu.tw.

Manuscript received 5 Sept. 2007; accepted 13 Nov. 2007; published online 8 Jan. 2008.
Recommended for acceptance by M. Gokhale.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2007-09-0450.
Digital Object Identifier no. 10.1109/TC.2008.13.

TABLE 1
Contention Resolvers

		CR Equations	comments
CR From [1]	(1)	$\vec{A}^i[j]=\sum_{x=1}^{M}\left(\overrightarrow{HP}_x[j]\cdot Req_x^i\right), j\in\{1,...,M-1\}$	requesting selector's highest HP
	(2)	$\vec{H}_x^i[1]=1$ and $\vec{H}_x^i[j]=\overrightarrow{HP}_x[j], j\in\{2,...,M-1\}$	Converted $\overrightarrow{HP}_x$
	(2')	$\vec{H}_x^i[1]=1$ and $\vec{H}_x^i[j]=\overrightarrow{HP}_x[j]\cdot Req_x^i, j\in\{2,...,M-1\}$	(2') corrects (2)
	(3)	$B_x^i=\overline{\sum_{j=1}^{M-1}\left(\vec{A}^i[j]\oplus\vec{A}_x^i[j]\right)}, x\in\{1,...,M\}$	B_x^i is set if $\vec{H}_x^i=\vec{A}^i$
	(4)	$F_x^i=\left(\prod_{y=1}^{x-1}\overline{B_y^i}\right)\cdot B_x^i, x\in\{1,...,M\}$	$F_x^i=1$ if $B_x^i=1$ and $B_y^i=0$ for all $y<x$ or $F_x^i=0$ otherwise.
	(5)	$G_x^i=\left(\prod_{y=1}^{x-1}\overline{Req_y^i}\right)\cdot Req_x^i, x\in\{1,...,M\}$	Similar to F_x^i.
	(6)	$b^i=\sum_{x=1}^{M}B_x^i$	$b^i=0$ if no $B_x^i=1$ or $b^i=1$ otherwise
	(7)	$S_x^i=b^i\cdot F_x^i+\overline{b^i}\cdot G_x^i, x\in\{1,...,M\}$	$S_x^i=1$ if selector x is selected or $S_x^i=0$ otherwise
Proposed CR	(8)	$\vec{A}^i[M]=1$ and $\vec{A}^i[j]=\sum_{x=1}^{M}\left(\overrightarrow{HP}_x[j]\cdot Req_x^i\right), j\in\{1,...,M-1\}$	Similar to (1), but augmented with a set bit on M^{th} bit
	(9)	$\vec{H}_x^i[M]=Req_x^i$ and $\vec{H}_x^i[j]=\overrightarrow{HP}_x[j], j\in\{1,...,M-1\}$	Converted $\overrightarrow{HP}_x$
	(10)	$B_x^i=\overline{\sum_{j=1}^{M}\left(\vec{A}^i[j]\oplus\vec{A}_x^i[j]\right)}, x\in\{1,...,M\}$	similar to equation (3)
	(11)	$S_x^i=\left(\prod_{y=1}^{x-1}\overline{B_y^i}\right)\cdot B_x^i, x\in\{1,...,M\}$	The same as (4)

TABLE 2
An Example in which Selectors 2 and 4 Continuously Request CR^i

		cycle 1				cycle 2				cycle 3			
	Selector x	4	3	2	1	4	3	2	1	4	3	2	1
CR from [1]	Req_x^i	1	0	1	0	1	0	1	0	1	0	1	0
	$\overrightarrow{HP}_x$	001	***	000	***	011	***	001	***	000	***	011	***
	$\vec{A}^i$	001				011				011			
	$\vec{H}_x^i$ from (2')	001	001	001	001	011	001	001	001	001	001	011	001
	B_x^i	1	1	1	1	1	0	0	0	0	0	1	0
	F_x^i	0	0	0	1	1	0	0	0	0	0	1	0
	G_x^i	0	0	1	0	0	0	1	0	0	0	1	0
	b^i	1				1				1			
	S_x^i	0	0	0	1	1	0	0	0	0	0	1	0
Proposed CR	$\overrightarrow{HP}_x$	001	***	000	***	000	***	001	***	001	***	000	***
	$\vec{A}^i$	1001				1001				1001			
	$\vec{H}_x^i$	1001	0***	1000	0***	1000	0***	1001	0***	1001	0***	1000	0***
	B_x^i	1	0	0	0	0	0	1	0	1	0	0	0
	S_x^i	1	0	0	0	0	0	1	0	1	0	0	0

add one more bit at the Mth bit position in $\vec{A}^i$ and $\vec{H}_x^i$ to make them M bits wide. Our design achieves the following four goals:

1. Nonrequesting selectors will not be selected.
2. Only the requesting selector with the highest priority will be selected.
3. If more than one selectors have the highest HP, the one with the lowest ID is selected.
4. No selector starves.

Table 1 lists the equations that implement the proposed contention resolver. We compute $\overrightarrow{A^i}$ in (8) by setting $\overrightarrow{A^i}[M] = 1$ and other $M-1$ bits of $\overrightarrow{A^i}$ to be the highest HP among all the requesting selectors. The converted HP values (i.e., $\overrightarrow{H_x^i}$) are computed in (9) by setting $\overrightarrow{H_x^i}[M] = Req_x^i$ and the other $M-1$ bits of $\overrightarrow{H_x^i}$ to be $\overrightarrow{HP_x}$. In (10), the signal B_x^i is set if $\overrightarrow{A^i}$ is equal to $\overrightarrow{H_x^i}$ or reset otherwise. In other words, if there is a bit position $j \in \{1, \ldots, M\}$ such that $\overrightarrow{A^i}[j] \neq \overrightarrow{H_x^i}[j]$, then B_x^i must be zero. Thus, the signal B_x^i corresponding to a nonrequesting selector x must be zero because $\overrightarrow{A^i}[M] = 1$ and $\overrightarrow{H_x^i}[M] = Req_x^i = 0$. There may be more than one requesting selector with the highest HP whose B_x^i will be set to 1. Therefore, we use (11) to select the one with the lowest ID among all of the selectors whose B_x^i is one. Because the selector that is on hold for $M-1$ cycles will be the only one having the highest HP of $1 \ldots 1$ (M 1s), it will be selected within M cycles and thus will not starve. The proposed contention resolver clearly uses less number of logic equations than that in [1]. The example in Table 2 shows that CR^i correctly processes one request every cycle.

To show the speed and cost advantage of the proposed contention resolver over the one in [1], we conduct simulations by using Verilog HDL synthesized in Synopsys design vision with the standard cell from the Artisan TSMC 0.18μm cell library. With the number of selectors (M) varying from 4 to 15, our results show that the proposed contention resolver needs only 53-71 percent of the chip area and 67-77 percent of the critical path delay needed in [1].

REFERENCES

[1] M. Akhbarizadeh, M. Nourani, R. Panigrahy, and S. Sharma, "A TCAM-Based Parallel Architecture for High-Speed Packet Forwarding," *IEEE Trans. Computers,* vol. 56, no. 1, pp. 58-72, Jan. 2007.