BINARY TREE SEARCH ARCHITECTURE FOR EFFICIENT IMPLEMENTATION OF ROUND ROBIN ARBITERS

C.E. Savin, T. McSmythurs, and J. Czilli

Cisco Systems
3000 Innovation Drive, Kanata, ON, Canada K2K 3J9
e-mail: {esavin, tmcsmyth, jczilli}@cisco.com

ABSTRACT

A new architecture for the implementation of round robin arbiters (RRAs) with one-hot encoded grant signals is introduced in this paper. The proposed architecture uses a binary tree search (BTS) mechanism in conjunction with a unit-weighted representation of the priority index. It is shown that the proposed BTS RRA architecture achieves very significant improvements on time-area complexity compared to the most efficient convetional RRA configuration to date.

1. INTRODUCTION

Priority encoders find extensive applications in computer systems with shared resources. When several blocks in a system require simultaneous access to a shared resource, a scheduling decision has to be made to allow a single block to use the resource. This decision is usually made by employing a priority encoding scheme, whereby a grant to access the shared resource is issued to the requesting block with highest precedence as defined by a priority sequence. In a fixed priority encoder (FPE), the priority sequence is static and can only be modified by explicitly changing the wiring of the encoder. On the other hand, a programmable priority encoder (PPE) implements a dynamical precedence scheme, where the priority sequence can be easily changed while in operation. Essentially, PPEs provide the core functionality of round robin arbiters (RRAs). As such, PPEs are frequently used in networking ASICs and FPGAs, to implement key functions like switch-fabric scheduling and queue/port selection in layer 2 and layer 3 switches [1]. With the constantly increasing size of the RRAs required in current networking devices, the problem of implementing time-area efficient and easily scalable PPEs and RRAs deserves special attention.

A review of the currently available solutions for the implementation of RRAs is provided in [1]. As highlighted in [1], the PPE architecture proposed in [2], with further improvements suggested in [1], stands out as the most efficient solution to date for the implementation of RRA arbiters. It should be noted here that while the original configuration proposed in [2] operates with binary-encoded grant signals, the architecture described in [1], targeted for high-speed performance, uses one-hot encoded grants. This paper addresses the problem of efficiently designing RRAs with one-hot encoded grant vectors.

A block-level diagram of the RRA architecture proposed in [1] is illustrated in Fig. 1. In this architecture, the core

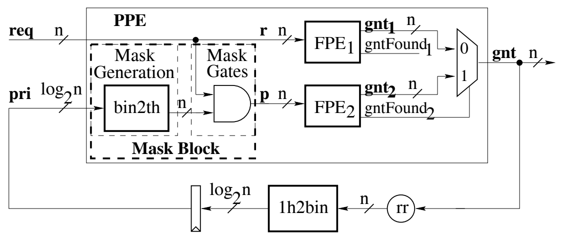


Fig. 1. Block diagram of an RRA with binary encoded priority/state vector and one-hot-encoded grant output.

PPE block comprises 2 FPE encoders operating in parallel. The input to FPE_1, $\mathbf{r}$, is the original request vector, $\mathbf{req}$, while the input to FPE_2 is a masked version of $\mathbf{req}$, designated in Fig. 1 as $\mathbf{p}$. As seen from Fig. 1, $\mathbf{p}$ is the output of a mask block whose inputs are the request signals, along with a priority index, $\mathbf{pri}$. The index $\mathbf{pri}$, also referred to as the round-robin state vector, points to the current highest-priority request-line. In operation, the mask block forces to 0 all elements $\mathbf{req}_j$ of the request vector $\mathbf{req}$, whose index j is strictly smaller than the decimal value of the binary encoded $\mathbf{pri}$, pri_{dec}, and leaves all other elements unchanged. The mask generation module on the mask block is a binary-to-thermo encoder, bin2th, whose operation is described in [1]. Essentially, the thermo-code $\mathbf{y}_{th}$ of a binary vector $\mathbf{x}$ is a unit-weighted representation of the decimal value of $\mathbf{x}$, x_{dec} (i.e., the sum of all the 1's in $\mathbf{y}_{th}$ is equal to the base-10 number whose binary code is $\mathbf{x}$). As an example, the 8-bit thermo-code of $\text{x}_{dec} = 5$, is $\mathbf{y}_{th} = 00011111$. The encoder FPE_1 finds the first nonzero element of the request vector $\mathbf{req}$ and generates a corresponding one-hot grant, $\mathbf{gnt}_1$. On the other hand, FPE_2 picks the first nonzero element of $\mathbf{req}$ (if any available) beyond (and including) $\mathbf{req}[pri_{dec}]$, and generates a corresponding n-bit one-hot encoded grant, $\mathbf{gnt}_2$. Each FPE block provides an output qualifier, $gntFound_j$, $j \in \{1, 2\}$, to indicate whether or not a grant is available at the output of FPE_j. The output multiplexer of the PPE uses $gntFound_2$ as a select signal, and takes $\mathbf{gnt} = \mathbf{gnt}_2$ if $gntFound_2 = 1$, and $\mathbf{gnt} = \mathbf{gnt}_1$ otherwise. It is to be noted that in the RRA architecture depicted in Fig. 1, the priority vector is a binary-encoded representation of the previous grant, right-shifted by 1 position (with the most significant bit of the grant vector assumed to be the rightmost bit). The shifting operation is achieved through suitable wiring connections in the rr block.

Hierarchical techniques employing binary tree mecha-

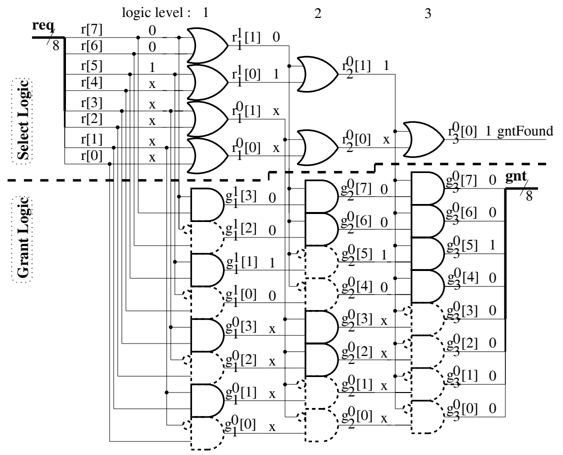


Fig. 2. Implementation of an 8-bit FPE block employing a BTS hardware algorithm.

nisms have long been known to provide simple, yet very efficient implementions of fixed priority encoders [3]. An example of a binary-tree search (BTS) implementation of an 8-bit FPE with one-hot grant output is illustrated in Fig. 2. The BTS FPE encoder depicted in Fig. 2. consists of 2 main modules, designated as select logic and grant logic. At each logic level ℓ, $\ell \in \{1,2\}$, the grant signals $g_\ell^k[i]$, with $k \in \{0,\ldots,\log_2 n - \ell - 1\}$, and $i \in \{0,\ldots,2^{\ell+1}-1\}$, are partitioned into a high priority subset, $H_\ell^k = \{g_\ell^k[i], i = 0,\ldots,2^\ell - 1\}$, and a low priority subset $L_\ell^k = \{g_\ell^k[i], i = 2^\ell,\ldots,2^{\ell+1}-1\}$. When a select signal $r_\ell^k[1]$ assumes a logic value of 1, there is at least one element in H_ℓ^k that is equal to 1, and therefore all elements in L_ℓ^k, gated with $\overline{r}_\ell^k[1]$, are forced to 0. On the other hand, if $r_\ell^k[1]$ is 0, no high priority grant is available in H_ℓ^k, and all grant signals in L_ℓ^k are propagated to the next level of logic, $\ell+1$. A simple numeric example is indicated on Fig. 2. With a request vector $\mathbf{req} = 001\mathrm{xxxxx}$, where x denotes a don't care logic value, the grant vector is, as expected, $\mathbf{gnt} = 00100000$.

By closely investigating the structure and the operation of the BTS FPE encoders used on an RRA device, this paper develops a new and more efficient BTS-based design for PPEs and RRAs with one-hot-encoded grant output. The proposed BTS architecture for the implementation of PPE encoders and RRA arbiters is introduced in Section 2, while Section 3 evaluates the performance advantages of the proposed BTS design.

2. PROPOSED ARCHITECTURE

The proposed PPE/RRA architecture is illustrated in Fig. 3, for a simple case of a block with 8-bit request and grant vectors. The core PPE block is depicted with continuous lines, while the additional logic and registers required to complete an RRA device are drawn using dotted lines. The proposed PPE architecture consists of 2 select logic modules, rSelect and pSelect, rpSelect multiplexers, and a grant logic unit. It can be easily recognized that the rSelect and pSelect modules in Fig. 3 are essentially the select logic sub-blocks of the FPE_1 and FPE_2 encoders in Fig. 1, when FPE_1 and FPE_2 are implemented using the BTS FPE design illustrated in Fig. 2. On the other hand, while the conventional PPE design depicted in Fig. 1 uses 2 complete FPE blocks, each with its own grant logic sub-module, the proposed PPE architecture requires only one grant logic module, shared between the rSelect and pSelect blocks by employing rpSelect multiplexers. Effectively, in the proposed design, the $(n-1)$ 2-bit rpSelect muxes, conveniently replace the $2n$-bit output mux in a conventional PPE design, allowing the rSelect and pSelect blocks in the proposed PPE architecture to share the same grant logic unit. In turn, this should significantly lower the gate-count requirements of the proposed PPE configuration compared to the gate-count of a conventional PPE design, as the bulk of a BTS FPE block is mainly concentrated in its grant logic sub-module. Further improvements on both chip area as well as latency are achieved with the proposed PPE design by using a unit-weighted priority/state vector, $\mathbf{pri}_{uw}$. Essentially, in the proposed PPE architecture, $\mathbf{pri}_{uw}$ is directly used as a mask vector. Thus, by employing a unit-weighted priority vector, the proposed RRA architecture eliminates the need for the 2 encoders denoted on Fig. 1 by 1h2bin and bin2th, therefore significantly improving on latency, at the expense of increasing the size of the state register from $\log_2 n$ bits to n bits. As shown in Section 3, the overall tradeoff is very clearly advantageous to the proposed RRA design.

It should be emphasized here that with the state vector represented using a unit-weighted encoding, it is very easy

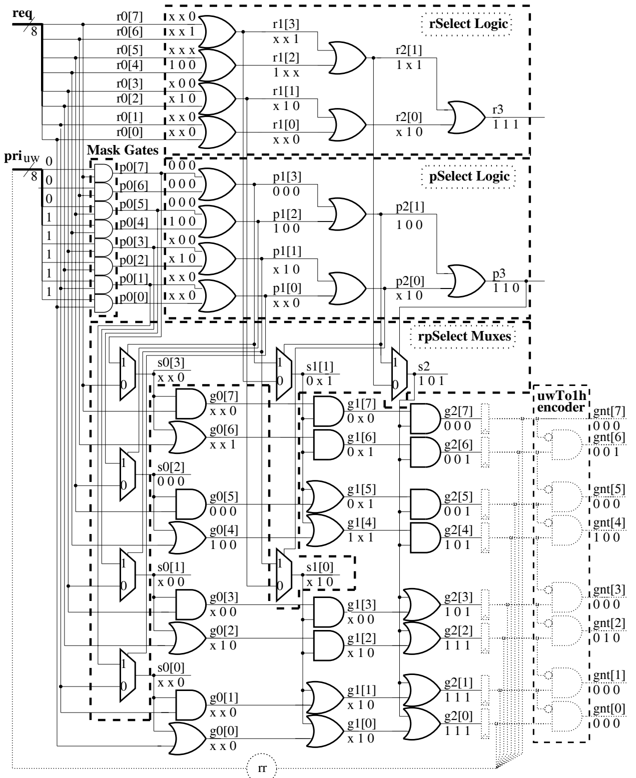


Fig. 3. Proposed implementation of an 8-bit PPE/RRA block employing a BTS algorithm and unit weighted priority/state vectors.

to generate the one-hot output grant, $\mathbf{gnt}[7:0]$, by employing a unit-weighted-to-one-hot (uwTo1h) encoder, as shown in Fig. 3. As seen from Fig. 3, a uwTo1h encoder requires only one level of AND gates. It should also be pointed out that the unit-weighted encoding used in the proposed architecture is slightly different from the thermo codes used in [1]. Denoting by $\mathbf{y}_{uw}$ the unit-weighted representation, as used in Fig. 3, of a decimal number x_{dec}, and designating the thermo code of x_{dec} by $\mathbf{y}_{th}$, we have that $\mathbf{y}_{uw} = \{\mathbf{y}_{th}[n-2:0], 1\}$, i.e., $\mathbf{y}_{uw}$ is a left shifted version of $\mathbf{y}_{th}$, with the least significant bit (assumed to occupy the right-most position) set to 1. Yet another consequence of using unit-weighted priority vectors, is that the grant logic module in Fig. 3 is slightly different from its counterpart in Fig. 2. Specifically, the AND gates depicted with dotted lines in Fig. 2 have been replaced in Fig. 3 with OR gates.

The proposed PPE/RRA design inherits a hierarchical structure that is characteristic of all BTS algorithms and architectures. As a result, a formal proof of correctness of the operation of the proposed design can be constructed using mathematical induction. However, due to limited space, a formal proof of correctness is not included in the present paper.

The operation of the proposed PPE architecture can be easily understood by using a few simple examples, indicated on the block diagram depicted in Fig. 3. With the priority index, $\mathbf{pri}_{uw}$, assuming a fixed value, say $\mathbf{pri}_{uw}[7:0] = 00011111$, the highest and lowest priority requests are, respectively, $\mathbf{req}[4]$ and $\mathbf{req}[5]$. Thus, when the request vector is $\mathbf{req}[7:0] = \mathrm{xxx1xxxx}$, the unit-weighted state vector assumes a value of $\mathbf{g2} = 00011111$, as expected, with $\mathbf{gnt}[4] = 1$. Whenever nonzero values are present in the masked request vector, $\mathbf{p0}$, a grant is always given to the active request of highest priority in $\mathbf{p0}$. For instance, in the case when $\mathbf{req} = \mathrm{xxx001xx}$, and thus $\mathbf{p0} = \mathrm{000001xx}$, the state vector is $\mathbf{g2} = 00000111$, and $\mathbf{gnt}[2] = 1$, as expected. On the other hand, when there is no active request line below (and including) the most significant logic-1 in $\mathbf{pri}_{uw}$, e.g., when $\mathbf{req} = \mathrm{01x00000}$, the masked request vector $\mathbf{p0}$ is all zeroes, and a grant is given to the highest priority request in the vector $\mathbf{req}$ itself. Thus, in the case when $\mathbf{req} = \mathrm{01x00000}$, the state vector is $\mathbf{g2} = 01111111$, and, as expected, $\mathbf{gnt}[6] = 1$.

3. PERFORMANCE ANALYSIS

In order to evaluate the improvements on chip area and timing that can be achieved by employing the proposed BTS PEE/RRA design compared to using the conventional architecture described in [1], estimates of gate counts and latencies have been determined for each building block of the proposed and conventional RRAs. These estimates were expressed as functions of the number of input ports (i.e., size of the $\mathbf{req}$ vector), n, and are listed in Table I for the components of a conventional RRA, and in Table II for the building blocks of the proposed RRA design. In Tables I and II, the gate count is stated as the number of 2-input gates, while the latency is expressed as number of levels of 2-input-gate logic. While most of the expressions for gate-counts and latencies listed in Tables I and II are fairly straightforward, the gate-count formulae for the bin2th and 1h2bin encoders require some further elaboration.

TABLE I
Estimation of Gate Count and Latency of a Conventional Implementation of an RRA Arbiter (Fig. 1)

Block	Gate Count (2-input gates)	Levels of logic
bin2th	$2(n-2) - 2(\log_2 n - 1)$	$\log_2 n - 1$
Mask Gates	n	1
Select Logic	$n-1$	$\log_2 n$
Grant Logic	$n \log_2 n$	$\log_2 n$
Output Mux	$3\,n$	2
1h2bin	$^*G_{1h2bin}$	$\log_2 n - 1$
Register	$7 \log_2 n$	N/A
RRA	$^{**}G_{rra}$	$3 \log_2 n + 1$

$^*G_{1h2bin} = 2(n-1) - n\,(2 + \log_2 n \,\%\, 2)\, 2^{-\lceil \log_2 n/2 \rceil}$
$^{**}G_{rra} = G_{1h2bin} + 8n + (2n+5)\log_2 n - 4$

TABLE II
Estimation of Gate Count and Latency of the Proposed Implementation of an RRA Arbiter

Block	Gate Count (2-input gates)	Levels of logic
Mask Gates	n	1
r/pSelect Logic	$n-1$	$\log_2 n$
rpSelect Muxes	$3(n-1)$	2
Grant Logic	$n \log_2 n$	$\log_2 n$
uwTo1h	$n-1$	1
Register	$7\,n$	N/A
RRA	$n\log_2 n + 14n - 6$	$\log_2 n + 4$

An efficient implementation of a binary-to-thermo encoder uses a hierarchical structure, as illustrated in Fig. 4a in the case of a bin2th encoder with 3-bit binary codes. It can be easily observed from the example illustrated in Fig. 4a, that, in general, for a bin2th encoder with $\log_2 n$-bit binary codes, the number of levels of logic is $\log_2 n - 1$, and there are $(2^{\ell+1} - 2)$ gates on each logic level ℓ. Thus, by adding together the number of gates on each level of logic, from $\ell = 1$ to $\ell = \log_2 n - 1$, the total gate-count of a bin2th encoder can be expressed as

$$\begin{aligned} G_{bin2th} &= (2^2-2) + (2^3-2) + \ldots + (2^{\log_2 n} - 2) \\ &= 2^2(1 + 2 + \ldots + 2^{\log_2 n - 2}) - 2(\log_2 n - 1) \\ &= 2^2(2^{\log_2 n - 1} - 1) - 2(\log_2 n - 1) \\ &= 2(n-2) - 2(\log_2 n - 1). \end{aligned} \tag{1}$$

Fig. 4c illustrates an efficient implementation of a one-hot-to-binary (1h2bin) encoder with $n = 16$, in which the output binary code is denoted by $\mathbf{b}[3:0]$. As seen from Fig. 4c, this encoder requires $(n/2 - 1)$ gates for each of the bits $\mathbf{b}[3]$ and $\mathbf{b}[0]$, plus $n/4$ additional bits for each of $\mathbf{b}[2]$ and $\mathbf{b}[1]$. It can be shown that, in general, when n is an even power of 2, the gate count of an 1h2bin encoder implemented using the algorithm illustrated in Fig. 4c is given by

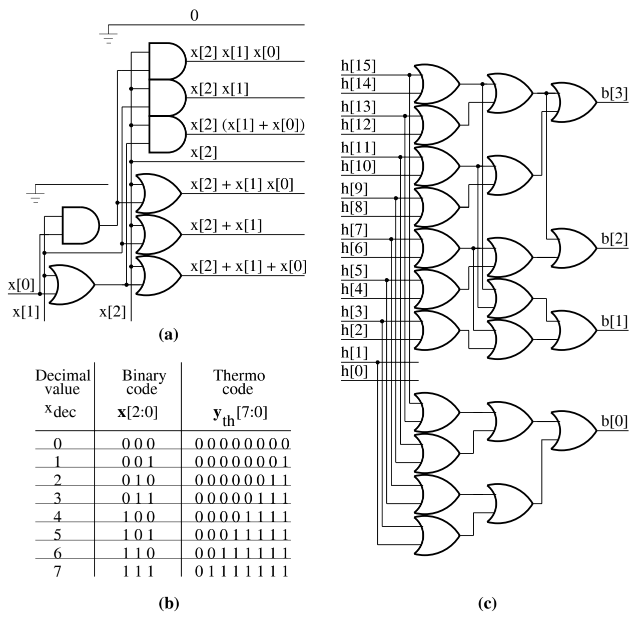
$$\begin{aligned} G_{even} &= 2 \cdot \left[\, (n/2-1) + n/4 + \ldots + n/2^{\log_2 n/2-1} \,\right] \\ &= 2n\,(1 - 2^{-\log_2 n/2}) - 2. \end{aligned} \tag{2}$$

On the other hand, when n is an odd power of 2, the gate count of an 1h2bin encoder with n inputs is

$$G_{odd} = 2n\,(1 - 2^{-\lceil \log_2 n/2 \rceil}) - 2 - n\, 2^{-\lceil \log_2 n/2 \rceil}, \tag{3}$$

where $\lceil x \rceil$ denotes the smallest integer larger than x. Assuming that n is a power of 2, and denoting by "$\log_2 n \,\%\, 2$" the rest of the integer division of $\log_2 n$ by 2 (i.e., $\log_2 n \,\%\, 2$ is equal to 1 when n is an odd power of 2, and 0 otherwise), the expressions (2) and (3) for G_{even} and G_{odd} can be merged together as

$$G = 2(n-1) \;-\; n\,(2 \;+\; \log_2 n \,\%\, 2)\, 2^{-\lceil \log_2 n/2 \rceil}. \tag{4}$$



Decimal value x_{dec}	Binary code $\mathbf{x}[2:0]$	Thermo code $\mathbf{y}_{th}[7:0]$
0	0 0 0	0 0 0 0 0 0 0 0
1	0 0 1	0 0 0 0 0 0 0 1
2	0 1 0	0 0 0 0 0 0 1 1
3	0 1 1	0 0 0 0 0 1 1 1
4	1 0 0	0 0 0 0 1 1 1 1
5	1 0 1	0 0 0 1 1 1 1 1
6	1 1 0	0 0 1 1 1 1 1 1
7	1 1 1	0 1 1 1 1 1 1 1

Fig. 4. Implementation (a) and truth table (b) of a binary-to-thermo encoder for 3-bit binary codes, and implementation (c) of a one-hot-to-binary encoder with 16 inputs.

The gate count and latency estimates for the conventional RRA arbiter depicted in Fig. 1, as functions of the number of request lines, n, are listed in the last row of Table I. Similarly, the gate count and latency expressions for the proposed RRA design are provided in the last row of Table II. As seen from Tables I and II, the gate counts for the registers were estimated using a typical number of 7 gates per bit, while the muxes were assumed to require 3 gates per bit and 2 levels of logic.

In order to compare the performance of the proposed RRA design with that of the conventional architecture described in [1], the latency expressions in the last rows of Tables I and II are evaluated in Table III for different values of n, i.e., 128, 256, 512 and 1024. The corresponding gate-counts are evaluated in Table IV. As seen from Tables III and IV, the proposed RRA architecture is two times faster, yet requires fewer gates, than the conventional RRA architecture described in [1].

TABLE III
Latency Comparison of Conventional and Proposed RRA Implementations, for Different Values of n

n	128	256	512	1024
Conventional [1]	22	25	28	31
Proposed RRA	11	12	13	14
Improvement	50%	52%	54%	55%

TABLE IV
Gate Count Comparison of Conventional and Proposed RRA Implementations for Different Values of n

n	128	256	512	1024
Conventional [1]	3,077	6,658	14,327	30,700
Proposed RRA	2,682	5,626	11,770	24,570
Improvement	13%	16%	18%	20%

4. CONCLUSION

A very efficient BTS architecture suitable for performance-driven implementation of RRAs, has been introduced in this paper. The proposed architecture achieves very important improvements on latency compared to conventional RRA designs, by employing a unit-weighted representation of the priority/state vector. It has been shown that the use of unit-weighted state vectors eliminates the need for a mask generation block, which is a standard component on conventional RRA devices, converting one-hot or binary-encoded state signals to a mask/unit-weighted format. Essentially, the proposed architecture is designed to take advantage of our observation that the conversion of unit-weighted state signals to a one-hot grant format is a straightforward operation, requiring only one level of logic, while a mask generation module is significantly more complex, with a latency of $\log_2 n$ levels of logic.

The proposed RRA architecture can also achieve significant improvements on gate-counts, compared to conventional RRA designs. The gate-count improvements stem from the elimination of the grant logic sub-module on one of the two FPEs used on a PPE device. Essentially, in the proposed RRA design, the FPE blocks use individual select-logic modules, but share the same grant-logic unit.

A performance analysis of the proposed architecture has shown that, for typical RRAs, with 128, 256, 512 and 1024 request lines, the new RRA design can achive more than 50% improvement on latency, compared to conventional RRA designs, yet requires at least 13% fewer gates.

5. REFERENCES

[1] P. Gupta and N. McKeown, "Designing and Implementing a Fast Crossbar Scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20-28, Jan.-Feb. 1999.

[2] T.L. Rodeheffer, *Rotating Priority Encoder Operating by Selectively Masking Input Signals to a Fixed Priority Encoder*, US patent 5,095,460, Mar. 1992.

[3] M. Parkin, "Writing Successful RTL Descriptions in Verilog," *Synopsys Methodology Notes*, Synopsys Inc., Jan 1994.