Dynamic-Priority Arbiter and Multiplexer Soft Macros for On-Chip Networks Switches

Giorgos Dimitrakopoulos
Electrical and Computer Engineering Dept.
Democritus University of Thrace, Xanthi, Greece

Emmanouil Kalligeros
Information and Communication Systems Eng. Dept.
University of the Aegean, Samos, Greece

***Abstract*—On-chip interconnection networks simplify the integration of complex system-on-chips. The switches are the basic building blocks of such networks and their design critically affects the performance of the whole system. The transfer of data between the inputs and the outputs of the switch is performed by the crossbar, whose active connections are decided by the arbiter. In this paper, we design scalable dynamic-priority arbiters that are merged with the crossbar's multiplexers. The proposed RTL macros can adjust to various priority selection policies, while still following the same unified architecture. With this approach, sophisticated arbitration policies that yield significant network-throughput benefits can be implemented with negligible delay cost relative to the standard round-robin policy.**

I. Introduction

Large systems-on-chip (SoCs) and chip multiprocessors (CMPs), incorporating tens to hundreds of cores, create a significant integration challenge. Designers react to these challenges mostly with architectural solutions either at the switch or the network level [1]. Often, switch architecting takes for granted the characteristics of the main building blocks of the switch and without any modifications, tries to re-organize them in a more efficient way [2], [3], [4]. Although such pure high-level design has produced highly efficient switches, the question on how better the switch would be if better building blocks were available remains to be investigated.

In this paper, we try to partially answer this question by explicitly targeting the design from scratch of new RTL soft macros that can handle concurrently arbitration and multiplexing and, hence, simplify the design of low-latency and high-radix switches. The proposed design is based on a generic architecture for a dynamic priority arbiter (DPA) and multiplexer, as shown in Fig. 1. It consists of the arbitration logic, the pointer update logic and the per-output multiplexer. Apart from resolving any conflicting requests for the same resource, it should guarantee that this resource is allocated fairly to the contenders [5]. Fairness is achieved by changing appropriately the priority of each input at runtime. The width of the priority state associated with each input depends on the complexity of the priority selection policy. A single priority bit per input is enough for simple round-robin policy [6], while for more complex weight-based policies [7], multi-bit priority state is needed.

Both simple round-robin as well as more complex weight-based selection policies that offer much better throughput [7] can be implemented following the proposed unified hardware architecture. Practically, the transition from simple round-robin to much more efficient weight-based policies, such as first-come-first-served or shortest packet first, is achieved with insignificant cycle time overhead.

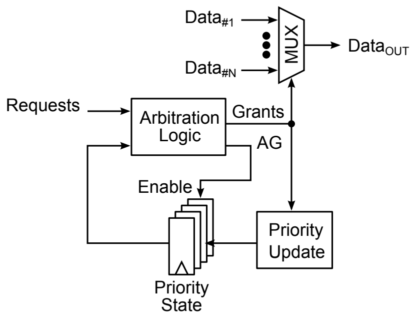


Fig. 1. The block diagram of a generic DPA that controls a multiplexer.

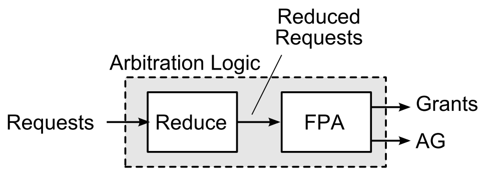


Fig. 2. The ogranization of the proposed two-step arbtration logic.

II. Unified Design Approach for DPAs

The unified approach for the design of DPAs that employ either round-robin or more complex weight-based selection policies is based on a two-step algorithm that is graphically depicted in Fig. 2. The first step transforms the requests and the corresponding input priority state to a new reduced request vector that involves only equal-priority requests. The second step involves a fixed priority arbiter (FPA), *i.e.*, a priority encoder, that operates on the reduced request vector granting the first active request.[1]

A. Round-robin arbiters

Round-robin arbitration logic scans the input requests in a cyclic manner beginning from the position that has the highest priority. The priority vector P that indexes the request with the highest priority, consists of N bits that follow the thermometer code. For example, in the case of an 8-port round-robin arbiter,

[1]In an FPA, the request of position 0 (rightmost) always has the highest priority and the request of position N-1 the lowest.

if position 3 has the highest priority, vector P is equal to 11111000 (MSB-to-LSB). The priority is diminishing in a cyclic manner to positions 4,5,6,7,0,1,2, giving to input 2 the lowest priority to win a grant. As shown in the example of Fig. 3, the priority vector splits the input requests in two segments. The high-priority (HP) segment consists of the requests that belong to high priority positions where $P_i = 1$, while the requests, which are placed in positions with $P_i = 0$, belong to the low-priority (LP) segment. The operation of the round-robin arbiter is to give a grant to the first active request (scanning right to left) of the HP segment and, if not finding any, to close the cycle by giving a grant to the first active request of the LP segment.

We can avoid this cyclic search by treating each input request R_i and the corresponding priority bit P_i as a 2bit unsigned number with value equal to $2R_i + P_i$ (the request R_i is assumed to be the most-significant bit of the two). An example of such arithmetic symbols for a request and priority vector are shown in Fig. 3. From the 4 possible arithmetic symbols, the symbols that represent an active request are either 3 (for the HP segment) or 2 (for the LP segment). On the contrary, symbols 1 and 0 denote, respectively, an inactive request and thus they are both mapped to symbol 0.

According to round-robin arbitration policy and the example priority vector of Fig. 3, the arbiter should start looking for an active request from position 3 and grant the one that lies on position 4, which is the first (rightmost) request of the HP segment. This operation is equivalent to granting the first maximum symbol found when searching from right to left. Therefore, the introduction of the arithmetic symbols practically transformed the cyclic round-robin arbitration to an acyclic sorting operation.

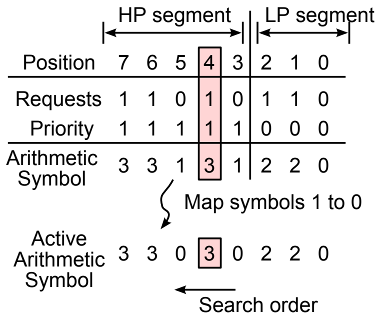


Fig. 3. The translation of the request and the priority vector to the corresponding arithmetic symbols that removes the cyclic priority trasnfer.

The request on position 4 needs to fight for a grant only with the requests with equal maximum symbols. Therefore, our first goal is to generate a reduced request vector that involves only the requests that are associated with the maximum arithmetic symbol. The requests that correspond to smaller weights should be filtered out. The reduced request vector for the arithmetic symbols 33030220 of the example of Fig. 3 would become equal to 11010000, having a 1 only in the positions that correspond to symbol 3, which is the largest. Then, using an FPA driven by the reduced request vector, as shown in Fig. 2, suffices to identify the rightmost active request.

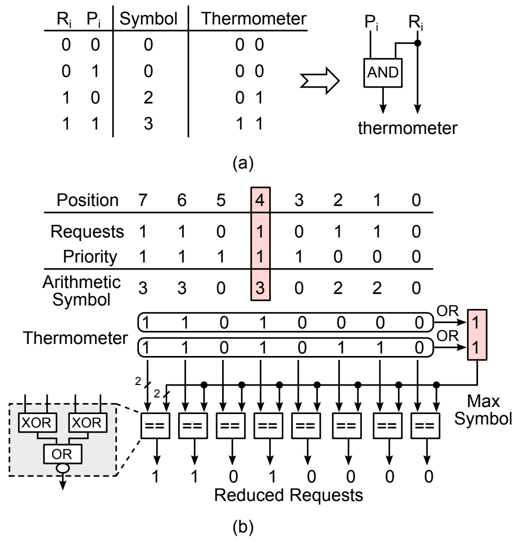


Fig. 4. The computation of the maximum weight and the derivation of the reduced request vector that contains only equal maximum-priority requests.

From the implementation viewpoint, the bottleneck operation is the transformation of the input requests and the priority vector to a reduced request vector that would be later handled by the FPA. This transformation involves finding first the maximum weight and then marking the positions that have a weight equal to the maximum. Maximum symbol identification can be easily performed if we encode the symbols as thermometer codewords [see Fig. 4(a)]. Then, the maximum symbol is the one with the largest number of consecutive 1s.

Therefore, in our case, we need two N-input OR gates to compute the maximum arithmetic symbol as shown in Fig. 4(b). Then, we identify the positions that have the same priority as the maximum symbol, by employing a 2-bit equality comparator at each position. The output of each 2-bit comparator corresponds to a bit of the reduced request vector that is later passed to the FPA.

B. Dynamic priority arbiters with arbitrary weights

Arbiters with more complex selection policies, such as backlog-aware policies, or first-come-first-served (FCFS) [7], can be designed following the same two-step approach used for simple round-robin. In these cases, each input is associated with a weight that denotes the input's priority relative to the rest inputs. Although the maximum value of the weights can be chosen arbitrarily, in most cases it suffices to be equal to the number of input ports. The operation of the arbiter is to find the requests with the largest weight and then to select among them the one that appears first in a fixed order. Consequently, the arbiter's implementation is similar to that presented for the round-robin policy, if the weights are thermometer coded (otherwise, an additional code-conversion step is required)..

The arbitration policies are differentiated only by the way the weight of each input is updated at each cycle. For example in FCFS, the priority of the currently granted client is set to the lowest priority, while the priorities of all the requesting clients not yet been granted, are increased by one. In every case, the implementation of the priority update logic should not be a problem since it runs in parallel to multiplexing.

III. Merged FPA and multiplexer

The proposed arbitration logic ends up in an FPA, whose grant signals drive a multiplexer. In order to speedup the overall implementation, we aim to merge these two final steps of FP arbitration and multiplexing in a new combined circuit. This merging can be achieved if we treat the request signals of the FPA as numbers with values 0 and 1, and the fixed priority arbitration as a sorting operation on these numbers. Practically, the selection of the rightmost 1 as dictated by the FPA, can be equivalently described as the selection of the maximum number that lies in the rightmost position.

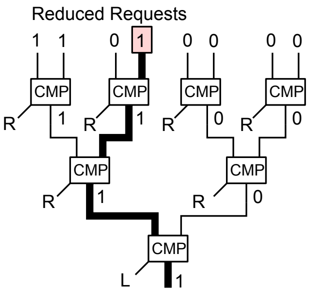


Fig. 5. Fixed-priority arbitration (priority encoding) as a sorting problem.

The proposed sorting-based FPA can be implemented as a binary tree with $N-1$ comparison nodes that all implement the same operation. Such a tree is shown in Fig. 5 for an 8-port FPA. Each node receives 2 single-bit numbers as input and computes the maximum of the two, along with a flag, that denotes the origin, left or right, of the maximum number. In case of a tie, when equal numbers are compared, the flag always points to the right according to the FPA policy. Note though that when both the numbers under comparison are equal to 0, the direction flag is actually a don't care value and does not need necessarily to point to the right.

In every case, the winning path that connects the winning input with the output is defined by the direction flags of the CMP nodes. Thus, if we use these flags to switch the data words that are associated with the input numbers (requests), we can route at the output the data word that is associated with the winning request. This combined operation can be implemented by adding a multiplexer next to each CMP node, as shown in Fig. 6.

Each CMP should identify the maximum of the two single-bit inputs, denoted as S_L and S_R, and declare via the direction flag F if the largest number comes from the left ($F=1$) or the right ($F=0$). The first output of the CMP node, that is the maximum, can be computed by the logical OR of S_L and S_R. The other output of the CMP node -flag F- is asserted when the maximum is the left number S_L. Therefore, F should be equal to 1 when $(S_L, S_R) = (1,0)$ that translates to $F = S_R \cdot \overline{S}_R$ in boolean algebra. However, we have already seen that we can also assert the flag when $(S_L, S_R) = (0,0)$ since they both represent inactive requests and their order is irrelevant. So, if

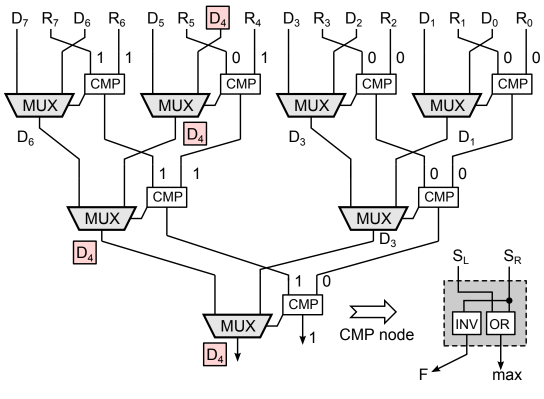


Fig. 6. The proposed merged arbiter multiplexer.

we embed the second case to the assertion of the direction flag, F becomes equal to $\overline{S}_R$ without changing the operation of the FPA. The OR gate and the inverter that implement the CMP node are shown in the right side of Fig. 6.

Finally, the AG signal that declares if any input was actually granted, is connected to the maximum output of the root node of the tree. If the maximum symbol produced is equal to 0, it means that no input request was actually granted.

A. Grant signal computation

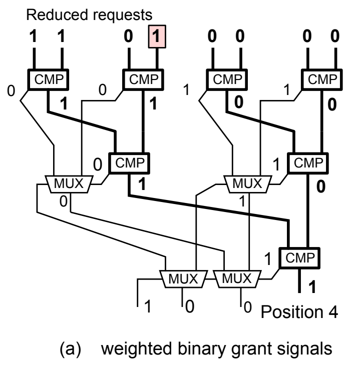
The merged arbiter multiplexer, besides transferring at the output the data word of the granted input, should also return in a useful format the position of the winning request (or equivalently the grant index). The proposed maximum-selection tree, shown in Fig. 5, that replaces the traditional FPA, can be enhanced to facilitate the simultaneous generation of the corresponding grant signals via the flag bits (F) of the CMP nodes. The generation of the grant signals in weighted binary representation, as well as in onehot code is shown in Fig. 7. Please notice that, contrary to Fig. 5, when the symbols under comparison are both equal to 0, the direction flag is equal to 1 following the optimized implementation of the CMP node that was shown in Fig. 6.

Observe that, if we replace the invert-AND gates of Fig. 7(b) with OR gates, the outcome would be a thermometer-coded grant vector instead of the onehot code. In this way, with minimum cost, we cover all possible useful grant encodings, thus alleviating the need for additional translation circuits in the pointer update logic.

IV. Experimental results

In this section we present the results gathered after performing various sets of experiments that helped us quantify the benefits of the proposed RTL macros. In all cases, the designs were implemented in a 65nm CMOS technology using a standard-cell based design flow.

In the first set of experiments, we compared the proposed merged round-robin arbiter-multiplexer against a standalone round-robin arbiter driving a multiplexer. The fastest approach for the latter pair involves the parallel-prefix arbiters of [8] along with the AND-OR implementation of the multiplexer. For both the compared designs, the whole switch allocator and



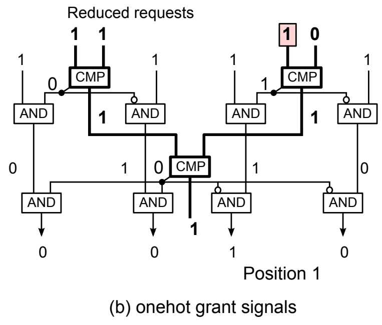


Fig. 7. The grant generation circuits that run in parallel to the CMP nodes.

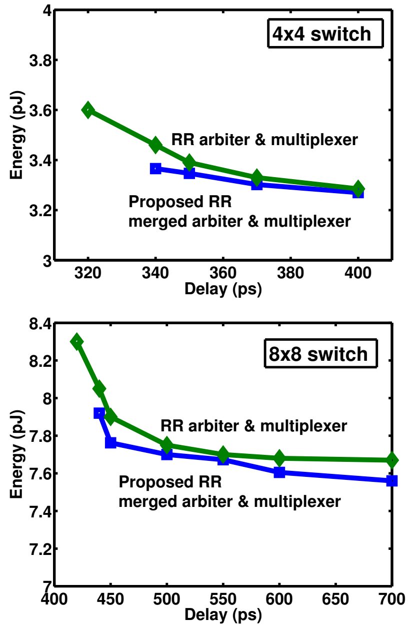


Fig. 8. The energy delay curves for the the case of round robin (RR) arbiters and multiplexers.

crossbar for a $N \times N$ switch was implemented, including also for each arbiter the corresponding priority state and priority update logic shown in Fig. 1. Both the input data and the input requests are registered. The same also holds for the data outputs of the switch. These output registers practically put switch and link traversal in different cycles.

From the derived energy-delay curves of Fig. 8 we observe that the two designs consume nearly the same amount of energy, with the proposed one being slightly better, while, in terms of speed, the parallel-prefix structure is slightly faster

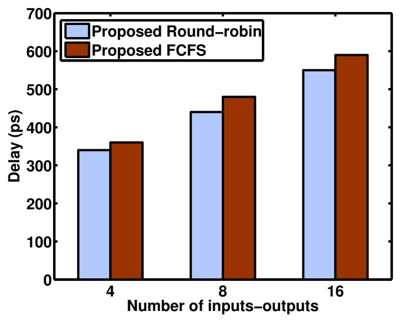


Fig. 9. The delay of the proposed switches implementing FCFS and standard round-robin arbitration policies.

(4% in average). These results lead us to the conclusion that, although the proposed design is based on a generic architecture that targets more complex weight-based selection policies and computes the grant signals in multiple formats, it offers, even for the simple round-robin policy, an implementation that is as good as the most efficient round-robin-only design. Note that such specific designs do not provide any obvious way of extending their functionality to other selection policies.

The last set of experiments measures the delay overhead imposed by a sophisticated weight-based policy, namely FCFS, relative to the best delay achieved by the simple round-robin implementation. From the bars of Fig. 9, it is obvious that the aforementioned delay overhead is negligible (6% on average) and is diminishing with increasing the size of the switch. This is probably the largest advantage of the proposed macros; they can be directly utilized for implementing more efficient selection policies that yield significant throughput benefits at the network level [7], while, at the same time, these performance benefits are not compromised by the clock frequency of the switches, since the latter are only slightly slower than those of the standard round-robin case.

V. Conclusions

On-chip network switches that can significantly benefit by the adoption of the introduced RTL soft macros. The proposed circuits adapt efficiently to simple and more complex arbitration policies under a unified architecture, and, at the same time, they offer area/energy/delay efficient implementations due to their merged arbitration and multiplexing structure.

References

[1] W. J. Dally and B. Towles, Route Packets, Not Wires: On-Chip Interconnection Networks, DAC, 2001.
[2] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction router: Yet another low latency on-chip router architecture," HPCA, 2009.
[3] Anh T. Tran and Bevan M. Baas, "RoShaQ: High-Performance On-Chip Router with Shared Queues," ICCD, 2011.
[4] J. Kim, "Low-cost router microarchitecture for on-chip networks", MICRO-42, 2009.
[5] S. S. Mukherjee, *et al.* "A comparative study of arbitration algorithms for the alpha 21364 pipelined router," ASPLOS-X, 2002.
[6] P. Gupta and N. McKeown, "Design and implementation of a fast crossbar scheduler," IEEE Micro, 1999.
[7] M. Pirvu, L. Bhuyan, and N. Ni, "The Impact of Link Arbitration on Switch Performance", HPCA, 1999.
[8] G. Dimitrakopoulos, N. Chrysos and K. Galanopoulos, "Fast arbiters for on-chip network switches", ICCD, 2008.