Merged Switch Allocation and Traversal in Network-on-Chip Switches

Giorgos Dimitrakopoulos, Emmanouil Kalligeros, *Member*, *IEEE*, and Kostas Galanopoulos, *Student Member*, *IEEE*

Abstract—Large systems-on-chip (SoCs) and chip multiprocessors (CMPs), incorporating tens to hundreds of cores, create a significant integration challenge. Interconnecting a huge amount of architectural modules in an efficient manner, calls for scalable solutions that would offer both high throughput and low-latency communication. The switches are the basic building blocks of such interconnection networks and their design critically affects the performance of the whole system. So far, innovation in switch design relied mostly to architecture-level solutions that took for granted the characteristics of the main building blocks of the switch, such as the buffers, the routing logic, the arbiters, the crossbar's multiplexers, and without any further modifications, tried to reorganize them in a more efficient way. Although such pure high-level design has produced highly efficient switches, the question of how much 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 soft macros that can handle concurrently arbitration and multiplexing and can be parameterized with the number of inputs, the data width, and the priority selection policy. With the proposed macros, switch allocation, which employs either standard round robin or more sophisticated arbitration policies with significant network-throughput benefits, and switch traversal, can be performed simultaneously in the same cycle, while still offering energy-delay efficient implementations.

Index Terms—Switch allocation, arbiters, crossbar, interconnection networks, and logic design

1 Introduction

Interconnection networks lie at the kernel of any complex SoC and provide a modular infrastructure that parallelizes the communication between system's modules by utilizing a network of switches connected with multiple point-to-point links [1]. A critical factor to overall network's efficiency, both in terms of performance and power, is the selection of the appropriate network topology that would reduce the communication distance and utilize efficiently the abundant on-chip wiring resources [2], [3], [4]. At the same time, the microarchitecture of the switches detemines the latency per node and the throughput that the network can actually sustain [5]. The first on-chip interconnection network designs have mimicked the designs that were architected for large, high-performance multiprocessors. However, as interconnects migrate to the on-chip environment, constraints and tradeoffs shift, making the effects of switch latency and power more pronounced.

The switches follow roughly the architectures depicted in Fig. 1 [6]. Fig. 1a shows a typical Wormhole (WH) switch with three pipeline stages. Routing computation (RC) logic unwraps incoming packets' header and determines their output destination. Such decoding and RC can be prepared in the previous switch and used in the current one. This optimization is called lookahead routing (LRC) [7], [8] and allows RC to be performed in parallel with the rest tasks. At the same time, the packet's header competes for the selected output port, because the rest input queues may have a request for the same output port. If it wins this stage, called switch allocation (SA), it will traverse the crossbar (ST—switch traversal) in next cycle, and, one cycle later, it will pass the output link (LT—link traversal) toward next switch. In WH switches, the SA stage is constructed using a single arbiter per output of the switch that decides independently which input to serve. The grant signals of each arbiter drive the corresponding output multiplexer and they are given back to the inputs to acknowledge the achieved connection. Both LRC and SA are performed only for the head flit of each packet. The remaining body and tail flits will follow the same route that has already been reserved by the head flit. Therefore, in a WH router, if a packet at the head of a queue is blocked, either because it loses SA or because the downstream buffer is full, all packets behind it also stall.

This head-of-line blocking problem can be solved by a virtual-channel (VC) switch [9], as shown in Fig. 1b. In this case, each input buffer is separated into multiple parallel queues. Each queue is called a VC and allows packets from different queues to bypass each other and advance to the crossbar instead of being blocked by a packet at the head queue. Because now an input port has multiple VC queues, each packet has to choose a VC at the input port of the next router (output VC) before SA. Matching input VCs to output VCs is a task performed by the VC allocator (VA). Since VA is performed in parallel with LRC, passing a switch requires four cycles, as shown in Fig. 1b. The VCs of each input share a common input port of the crossbar via a

- *G. Dimitrakopoulos is with the Electrical and Computer Engineering Department, Democritus University of Thrace, Xanthi, Greece. E-mail: dimitrak@ee.duth.gr.*
- *E. Kalligeros is with the Information and Communication Systems Engineering Department, University of the Aegean, Samos, Greece. E-mail: kalliger@aegean.gr.*
- *K. Galanopoulos is with the Electrical and Computer Engineering Department, National Technical University of Athens (NTUA), Athens, Greece. E-mail: galanopu@elab.ntua.gr.*

Manuscript received 16 Nov. 2011; revised 29 Feb. 2012; accepted 28 Apr. 2012; published online 30 May 2012.
Recommended for acceptance by R. Marculescu.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2011-11-0897.
Digital Object Identifier no. 10.1109/TC.2012.116.

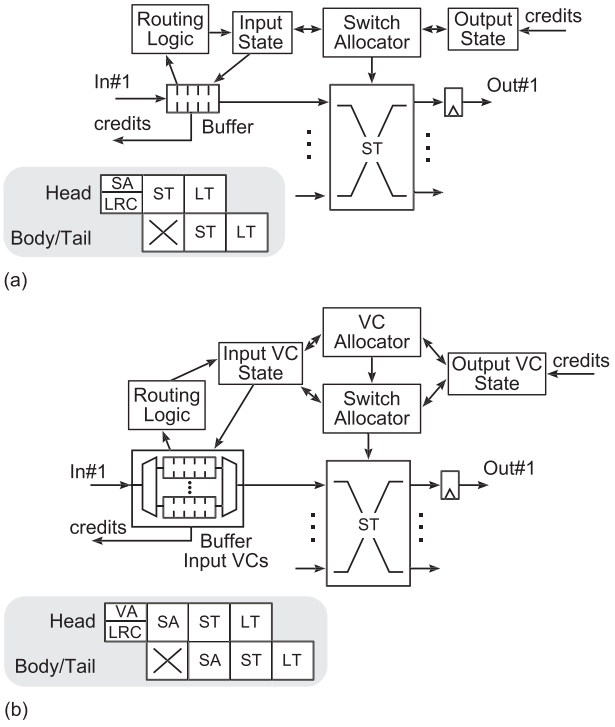


Fig. 1. Typical switch architectures and their pipelines: (a) 3-stage WH and (b) 4-stage VC switch.

local multiplexer (see Fig. 1b). This is enough for on-chip networks, because the rate of incoming traffic from each VC does not justify the allocation of a separate port of the crossbar to each one of them [10]. However, this feature complicates significantly the design of the SA stage relative to that of a WH switch. Specifically, in case of a switch with VCs, SA is organized in two phases, because both per-input and per-output arbitration is needed [11].[1] Even though the per-input and per-output arbiters operate independently, their eventual outcomes in SA are very much dependent, each one affecting the aggregate matching quality of the switch [14], [15].

With the goal to minimize the number of pipeline stages per switch, RC is performed in parallel to the rest operations via lookahead, and VA operates in parallel to SA using speculation [16], [17]. However, SA and ST remain closely interrelated with SA always preceding and guiding ST. Such dependency can be only removed by predicting the decisions of SA and paying the cost of a wrong prediction [18].

The kernel of SA and ST involves arbiter and multiplexer pairs that need to be carefully cooptimized to achieve an overall efficient implementation. For example, the encoding selected for the grant signals directly affects the design of the multiplexers. In the first case, shown in Fig. 2a, the grant decision is encoded in onehot form, where only a single bit is set, and the multiplexer is implemented using an AND-OR structure. On the contrary, in Fig. 2b, the multiplexer is implemented as a tree of smaller multiplexers. In this case, the select lines that configure the paths of each level of the tree are encoded using weighted binary representation.

1. An alternative to separable allocation is a centralized allocator like the wavefront allocator [12]. The delay of wavefront allocator grows linearly with the number of requests, while the cyclic combinational paths that are inherent to its structure prohibit static timing analysis. The latter constraint can be removed only after doubling the already aggravated delay [13].

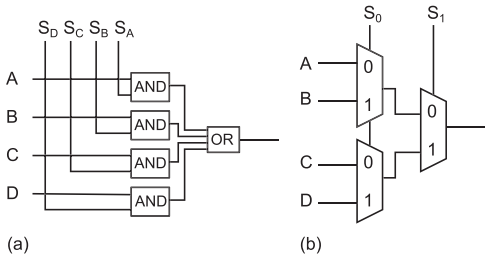


Fig. 2. Design options for a 4-input multiplexer.

Even if the design choices for the multiplexer, at least for a standard-cell-based design flow, are practically limited to the alternatives shown in Fig. 2, the design space for the arbiter is larger [19]. The arbiter, apart from resolving any conflicting requests for the same resource, it should guarantee that this resource is allocated fairly to the contenders, granting first the input with the highest priority [5]. Therefore, for a fair allocation of the resources, we should be able to change dynamically the priority of the inputs. A generic Dynamic Priority Arbiter (DPA), as shown in Fig. 3, consists of two parts; the arbitration logic that decides which request to grant based on the current state of the priorities, and the priority update logic that decides, according to the current grant vector, which inputs to promote. The priority state associated with each input may be one or more bits depending on the complexity of the priority selection policy. For example, a single priority bit per input suffices for round-robin policy [20], while for more complex weight-based policies [21], such as first-come-first-served (FCFS) [22] or age-based allocation [23], multibit priority quantities are needed.

In this paper, we target the optimization of the combined operation of arbitration and multiplexing, irrespective of the complexity of the priority selection policy, aiming at the design of efficient high-radix and low-latency switches. To achieve this goal, we utilize a new structure called *Merged ARbiter and multipleXer* (MARX) that merges efficiently the functionality of the arbiter and the multiplexer shown in Fig. 3, in a new circuit that performs the two steps in parallel. Both simple round robin as well as more complex weight-based selection policies that offer much better throughput can be implemented following the proposed design methodology. The transition from simple round robin to much more efficient weight-based policies, such as FCFS, queue-backlog-aware or shortest packet first [22], is achieved with insignificant cycle time overhead. The

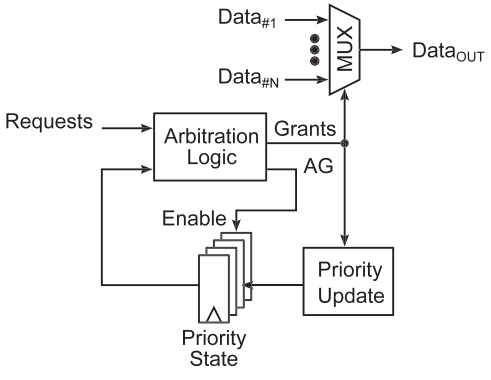


Fig. 3. The block diagram of a generic DPA that controls a multiplexer.

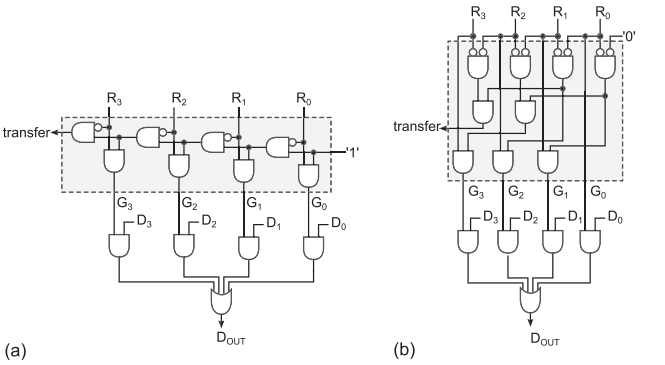


Fig. 4. (a) Ripple-carry and (b) Fast parallel prefix FPAs driving an AND-OR multiplexer.

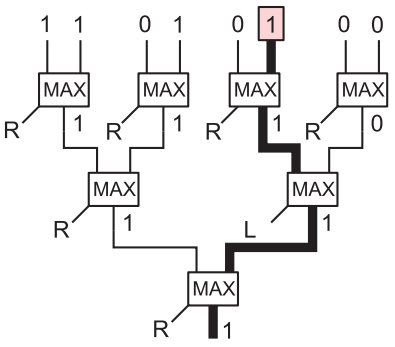


Fig. 5. Fixed-priority arbitration (priority encoding) as a sorting problem.

proposed circuits, besides other enhanced features, preserve all the characteristics of a DPA soft macro [24], such as the availability of the grant signals in multiple formats, and can be implemented in two ways, thus allowing the designer to fully explore the energy-area-delay design space. The first approach provides delay efficient solutions and leads to the fastest arbiter and multiplexer pairs, while the second one targets area/energy efficiency. In every case, the proposed designs are better than the state-of-the-art implementation of a separate arbiter and multiplexer.

We will present the new design methodology in a step-by-step fashion beginning in Section 2 from the design of a fixed priority MARX unit. Then, in Section 3, we will describe how the basic architecture can evolve to a more sophisticated MARX that implements both standard round robin and more complex weight-based policies. In the following, the area/energy efficient variant of the proposed architecture will be presented in Section 4, while the design of MARX-based switches will be discussed in Section 5. Experimental results that prove the benefits of the proposed circuits are reported in Section 6. Finally, conclusions are drawn in Section 7.

2 MARX with Fixed Priorities

The simplest form of switch allocators is built using Fixed Priority Arbiters (FPAs), also known as priority encoders. In this case, the priorities of the inputs are statically allocated and only the relative order of the inputs' connections determines the outcome of the arbiter. In any FPA, the request of position 0 (rightmost) has the highest priority and the request of position $N-1$ the lowest. For example, when an 8-port FPA receives the request vector $(R_7 \dots R_0) = 01100100$, it would grant input 2 because it has the rightmost active request. When at least one request is granted, an additional flag AG (Any Grant (AG)) is asserted. Two examples for the implementation of an FPA driving a multiplexer [25] are shown in Fig. 4.

Although fixed priority is not an efficient policy, and hence it is not used in practice, we deal with it in detail because: 1) it helps us derive the baseline MARX implementation cost; 2) it allows us to explain more easily the design of MARX units for more complex selection policies that are described in Section 3; and 3) it is used as a subcircuit in the area/energy efficient design variant presented in Section 4.

The merging of fixed priority arbitration and multiplexing 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.

Selecting the maximum of a set of numbers can be performed either by a tree or a linear comparison structure. Such structures compare recursively the elements of the set in pairs and the maximum of each pair is propagated closer to the output. Similarly, the proposed sorting-based FPA can be implemented as a binary tree with $N-1$ comparison nodes. Such a tree, for an 8-port FPA, is shown in Fig. 5. Each node receives two 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 numbers under comparison are equal to 0 (i.e., between the two compared requests, none is active), the direction flag is actually a don't care value and does not need necessarily to point to the right (this will be exploited later for optimizing the MARX structure).

In every case, the path that connects the winning input with the output is defined by the direction flags of the MAX nodes. Thus, if we use these flags to switch the data words that are associated with the input numbers (i.e., the 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 2-to-1 multiplexer next to each MAX node and connecting the direction flag to the select line of the multiplexer. The organization of this merged FPA and multiplexer is shown in Fig. 6.

2.1 Logic-Level Optimization

Each MAX node should identify the maximum of two single-bit input requests, denoted as R_L and R_R, and declare via the direction flag F if the request with the greatest value comes from the left ($F=1$) or the right ($F=0$). The first output of the MAX node, that is the maximum, can be computed by the logical OR of R_L and R_R. The other output of the MAX node, flag F, is asserted when the left request R_L is the maximum. Therefore, F should be equal to 1 when $(R_L, R_R) = (1,0)$ that translates to $F = R_L \cdot \overline{R_R}$ in boolean algebra. However, as noted earlier, we can also assert the flag when $(R_L, R_R) = (0,0)$ because they both represent inactive requests and their

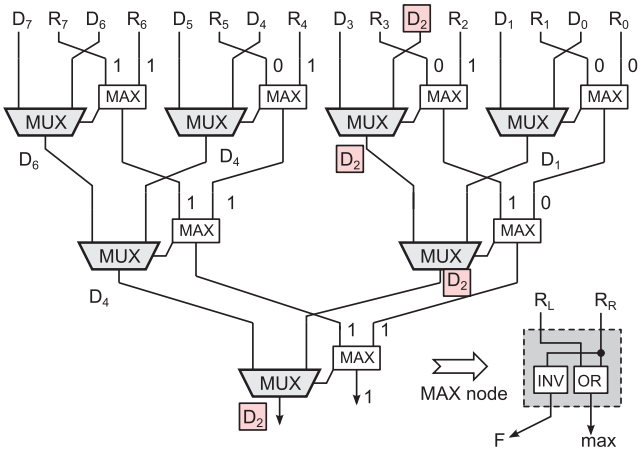


Fig. 6. MARX supporting fixed priority arbitration.

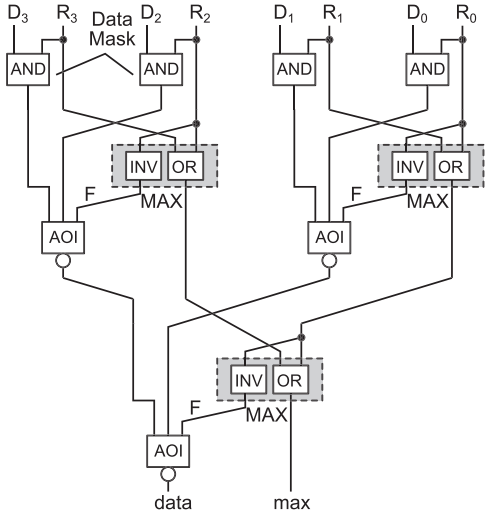


Fig. 7. Optimized fixed priority MARX.

order is irrelevant. So, if we embed the second case to the assertion of the direction flag, F becomes equal to $\overline{R_R}$ without changing the operation of the FPA. The OR gate and the inverter that implement the MAX node are shown in the right side of Fig. 6.

Additionally, the multiplexer at each node of the tree can be further simplified to a simpler AND-OR gate after making the following observation: If we mask to $00\ldots0$ all the data words that are associated with an inactive request, then when the right request R_R is equal to 0, we already know that the data from the right (D_R) will be also equal to $00\ldots0$. Therefore, the Boolean relation $D_{SEL} = D_L \cdot F + D_R \cdot \overline{F} = D_L \cdot \overline{R_R} + D_R \cdot R_R$ that corresponds to the output of each multiplexer, can be simplified to $D_{SEL} = D_L \cdot \overline{R_R} + D_R = D_L \cdot F + D_R$, where D_L and D_R correspond to the data words already masked with the input requests. In this way, the N-bit 2-to-1 multiplexer that stands next to each MAX node is transformed to a set of N AND-OR gates that are all driven by the F flag in one of their ports and exhibit better area/energy/delay characteristics than the multiplexer. The complete and optimized logic-level design of a 4-port MARX with fixed priorities is depicted in Fig. 7. Please note the clear reduction in logic levels offered by the proposed circuit compared to the fast variant of an FPA and multiplexer in Fig. 4b.

2.2 Grant Signal Computation

The MARX, 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 MAX nodes.

At first, we deal with the case that the grants are encoded in weighted binary representation. In this case, we can observe that, by construction, the weighted-binary encoding of the winning request is formed by putting together the flag bits of the MAX nodes that lie in the path from the winning input to the root of the tree (see Fig. 5, where the values L and R of the F flag correspond to 1 and 0, respectively). Consequently, the generation of the grant signals in weighted-binary representation is done by combining at each level of tree, the winning flag bits from the previous levels with the flags of the current level. This is achieved by means of some additional multiplexers, as shown in Fig. 8a. Please notice that, contrary to Fig. 5, when the requests under comparison are both equal to 0, the direction flag is equal to 1 following the optimized implementation of the MAX node that is shown in Figs. 6 and 7.

For the onehot encoding, we need a different implementation. Initially, i.e., at the inputs of the onehot-grant generation circuit, we assume that every position has a grant signal equal to 1. At the following levels, some of these grant signals are transformed to 0s if their associated requests are not the winning ones at the corresponding MAX nodes. Thus, at the outputs, only a single 1 will remain and the rest would be nullified. The circuit that generates the corresponding grant signals in onehot form, for four input requests, is shown in Fig. 8b. Keeping and nullifying the grant signals is performed by the AND gates that mask at each level of the tree, the intermediate grant vector of the previous level with the associated direction flags.[2]

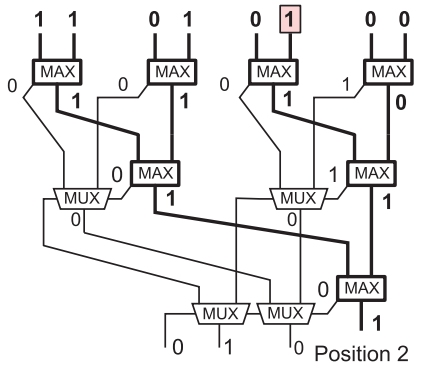
Observe that, if we replace the invert-AND gates of Fig. 8b with OR gates, the outcome would be a thermometer-coded grant vector instead of the onehot encoded. The resulting circuit is shown in Fig. 8c. In this way, with minimum cost, we are able to fully cover all possible useful grant encodings, thus alleviating the need for additional translation circuits.

Finally, the AG signal that declares if any input was actually granted is connected to the max output of the root node of the tree. If the maximum among all requests, which appears at this output, is equal to 0, it means that no input request was actually granted because all requests were inactive.

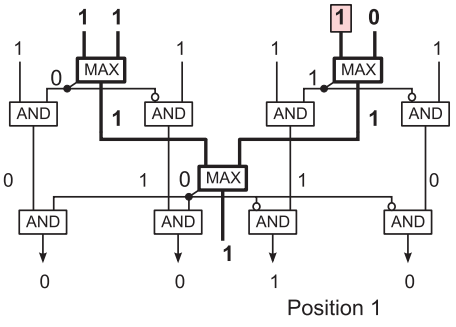
3 MARX with Dynamic Priorities

The basic idea of merging arbitration and multiplexing in a new combined sorting-based operation can be naturally extended to round robin and more complex weight-based selection policies.

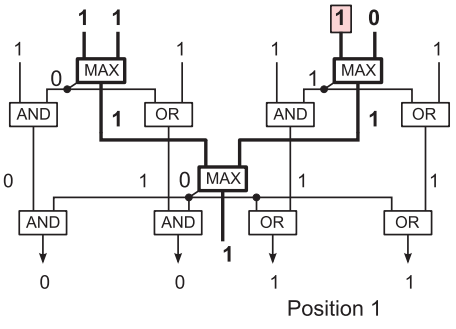
2. The design of Fig. 8b corresponds to a completely new modular organization for the design of a priority encoder that is based on sorting, instead of the traditional carry-propagate-like structures [25] shown in Fig. 4.



(a) weighted binary grant signals



(b) onehot grant signals



(c) thermometer grant signals

Fig. 8. The grant generation circuits that run in parallel to the MAX nodes.

3.1 MARX for Round-Robin Arbitration

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, which follow the thermometer code. For example, in the case of an 8-port round-robin arbiter, 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. If after scanning all inputs the circuit does not find an active request, it deasserts the AG signal.

As shown in the example of Fig. 9, the priority vector splits the input requests in two segments. The high-priority (HP) segment consists of the requests that belong to HP 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

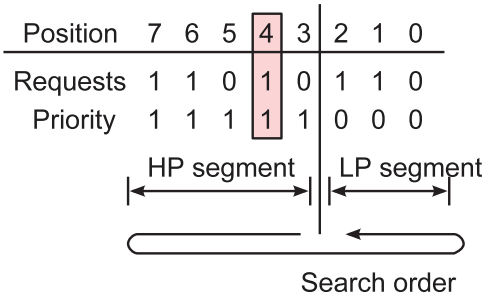


Fig. 9. The priority vector separates the input requests to an HP and an LP segment.

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 the input requests and the priority vector in a completely different way. Either at the HP or the LP segment, the pairs of bits (R_i, P_i) can assume any value. We are interested in giving an arithmetic meaning to these pairs. Therefore, we assume that the bits R_iP_i constitute a 2-bit unsigned quantity with a 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 randomly selected request and priority vector are shown in Fig. 10. From the four possible arithmetic symbols, 3, 2, 1, 0, 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. Since we do not care about the position of inactive requests, we can reduce the number of symbols by mapping symbol 1 to 0.

According to round-robin policy and the example priority vector of Fig. 10, 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. This general principle also holds for the case that the HP segment does not contain any active request. Then, all arithmetic symbols of the HP segment would be 0 and any active request on the LP segment would be mapped to 2. Therefore, the introduction of the arithmetic symbols

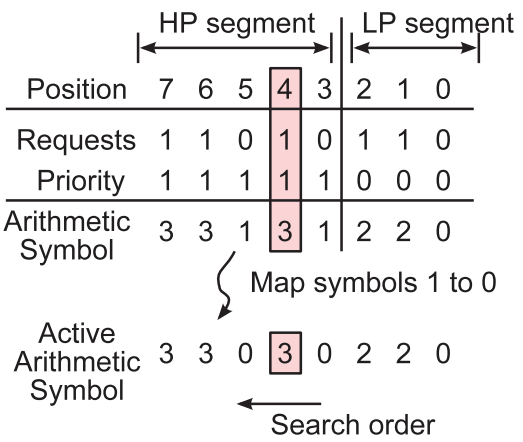


Fig. 10. The translation of the request and the priority vector to arithmetic symbols removes the cyclic search.

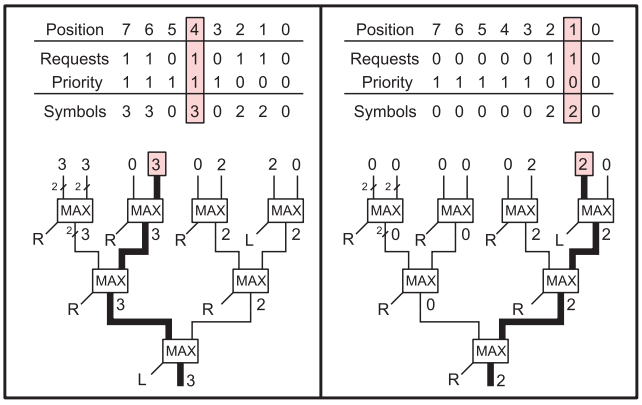


Fig. 11. The round-robin arbiter selects the rightmost maximum symbol. When there are no active requests in the HP segment, without any cyclic-priority transfer, the first request of the LP segment is granted.

R_i	P_i	Symbol	Thermometer
0	0	0	0 0
0	1	0	0 0
1	0	2	0 1
1	1	3	1 1

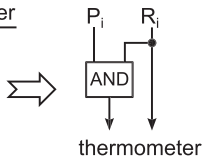


Fig. 12. Input arithmetic symbols recoding to 2-bit thermometer codewords.

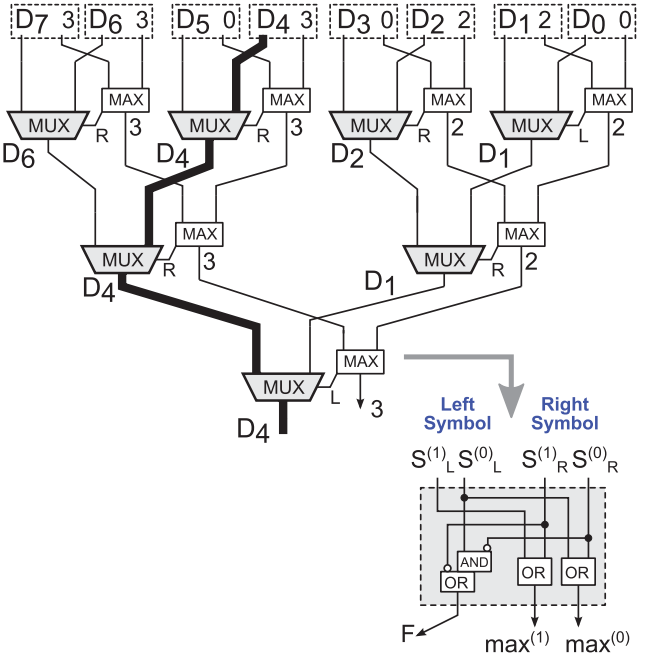


Fig. 13. The structure of the delay optimized MARX implementing round-robin policy.

practically transforms the cyclic round-robin arbitration to an acyclic sorting operation, as in the case of a FPA discussed in Section 2.

Thus, similar to the proposed FPA shown in Fig. 5, a binary tree can be utilized for selecting the maximum symbol. Two 8-input examples of the operation of the proposed round-robin arbiter are shown in Fig. 11. In the first case, the first request of the HP segment wins, while in the second case, there are no requests in the HP segment and the first request of the LP segment, in a round-robin order, wins. At each node of the tree, the L and R flags denote if the winning symbol belongs to the left or the right subtree, respectively. From both examples, it is evident that even if the comparison is performed directly on the 2-bit symbols, the result obeys the rules dictated by the round-robin policy. Similar to Fig. 6 the direction flags can be used to switch directly the corresponding data words. The only thing that differentiates this approach to the fixed priority case shown in Fig. 6 is the implementation of the new MAX node.

From the implementation viewpoint, the bottleneck operation is the selection of the maximum between the 2-bit arithmetic symbols. To significantly speedup this operation, we choose to recode the arithmetic symbols to thermometer codewords as shown in Fig. 12. This recoding of the symbols $\{0, 2, 3\} \rightarrow \{00, 01, 11\}$ is useful because now the maximum symbol can be easily identified as the one with the largest number of consecutive 1 s, which is equivalently reduced to a bitwise OR operation of the symbols under comparison (this is a property of thermometer coding that can be easily verified).

With this in mind, the MAX nodes used for the case of round-robin policy receive two thermometer codewords of 2 bits each. Identifying their maximum is easy because it involves two OR gates that run in parallel and compute the maximum thermometer codeword of the two. This is shown in the lower right corner of Fig. 13. The direction flag points to the left ($F = 1$) when the pairs of symbols (2, 0), (3, 0), (3, 2) arrive at the inputs of the MAX. To this set, we can also add two don't care conditions: The pair of symbols (0, 0) that represents two inactive requests, and, thus, their relative order does not play any role, and the pair (2, 3) that never happens because an active request in the LP segment (symbol 2) cannot be in a more-significant position compared to an active request of the HP segment (symbol 3). The logic level implementation of the direction flag that embeds all the aforementioned conditions is also shown in Fig. 13. Observing the derived circuit, we see that the proposed methodology not only merges round-robin arbitration with multiplexing, which is the first step for significant delay improvement, but also achieves this goal with a very compact circuit that consists only of an AOI gate for the direction flag and two OR gates for the computation of the maximum symbol.[3] Hence, the circuit variant of Fig. 13 constitutes the delay optimized version of the proposed architecture.

The grant signals computation circuits presented in Fig. 8 can be directly employed without any modification, because their functionality depends only on the direction flag, and thus, they are not affected by the granularity of the calculations performed by the MAX nodes. Also, in this case, the AG flag is produced via a OR gate at the root of tree that detects if the maximum symbol is non-zero.

3.2 MARX with Weight-Based Arbitration

Arbiters implementing more complex selection policies, such as backlog-aware policies [21], or FCFS [22], can be

3. The more complex function that flag F implements in this case does not allow us to simplify the multiplexers of the merged round-robin arbiter multiplexer to simpler AOI gates, as done in the case of the merged FPA multiplexer (Fig. 7).

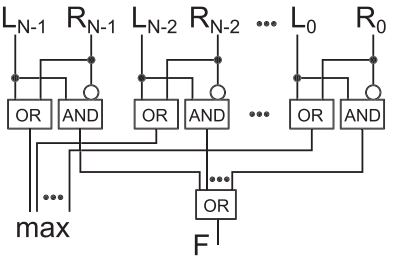


Fig. 14. The implementation of the MAX node in the case of weight-based priority selection policies.

designed in the same way as 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 equal the number of input ports [24]. 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.

Specifically, assume that in this case, each MAX node receives two N-bit thermometer coded weights $L_{N-1}L_{N-2}\ldots L_0$ and $R_{N-1}R_{N-2}\ldots R_0$ that come from the left and right side, respectively. Since the weights are thermometer coded, their maximum is again computed by the bitwise OR of the two input vectors. Thermometer encoding helps also in the derivation of the direction flag. Flag F is asserted ($F = 1$) when the left symbol is larger than the right one. This happens only if the L vector has more 1 s than the R vector, which means that there is at least one position i of the two vectors where $(L_i, R_i) = (1, 0)$. Based on this observation, we can compute flag F as follows:

$$F = L_{N-1} \cdot \overline{R_{N-1}} + L_{N-2} \cdot \overline{R_{N-2}} + \cdots + L_0 \cdot \overline{R_0}.$$

The logic-level implementation of this generic MAX node is shown in Fig. 14. As it will be shown in the experimental results section, the delay overhead compared to the simple round-robin case is negligible. Also, similar to standard round robin, in the case of weight-based arbitration, the weights entering the arbitration logic and correspond to inactive requests should be set equal to zero. Thus, only the active weights, i.e., the ones associated with active requests, participate in the comparisons. As for the AG flag, it is simply computed via an N-input OR gate that checks if the maximum symbol presented at the output of the root node equals a non-zero vector.

If the weights are stored in weighted binary representation, then a fast binary-to-thermometer decoder with few logic stages (e.g., [20]) should be employed before the arbitration logic.

The various arbitration policies are differentiated only by the way the weight of each input is updated in 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. If the weights are stored as thermometer codewords, their increment is just a simple shift by one that does not require more hardware than 2-to-1 multiplexers and appropriate wire rearrangement. In every case, the delay of the priority update logic should not be a problem because its operation overlaps in time with that of multiplexing. Also, the fact that the proposed circuits can provide, in parallel to arbitration and multiplexing, the position of the winning input (grant index) in multiple formats (see Section 2.2) removes the need for extra translation circuits in the priority update logic that are found in other fast arbiter implementations [20], [26].

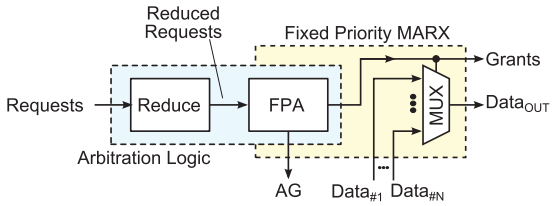


Fig. 15. Two-step arbtration logic; FPA is merged with the output multiplexer.

4 Area/Energy Efficient MARX with Dynamic Priorities

The already presented MARX units follow the same generic architecture and implement the corresponding priority selection policy, either fixed, round robin, or weight based, utilizing a sorting-based structure. In this way, arbitration and multiplexing are fully parallelized and delay efficient circuits can be constructed.

Nevertheless, we would like to take advantage of the simplified hardware structure presented in Fig. 7 for the case of MARX with fixed priorities, and design area/energy efficient structures for the rest selection polices too, namely round robin and weight based. This is performed by following a new two-step algorithm that is graphically depicted in Fig. 15. The first step transforms the requests and the corresponding input priority state to a new reduced request vector that involves only equal priority requests. Then, in the second step, a fixed-priority MARX unit operates on the reduced request vector and grants the active request with the highest priority, transferring also to the output the corresponding data.

At first, let us see how we can derive a reduced request vector for the case of round-robin selection policy. In the example shown in Fig. 16, 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. We have seen that this operation is equivalent to granting the first maximum symbol found when searching from right to left. Practically, the request on position 4 needs to fight for a grant only with the requests with equal maximum symbols. Therefore, our 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. 16 would be equal to 11010000, having a 1 only in the positions that correspond to a symbol 3, which is the largest. Then, using an FPA driven by the reduced request

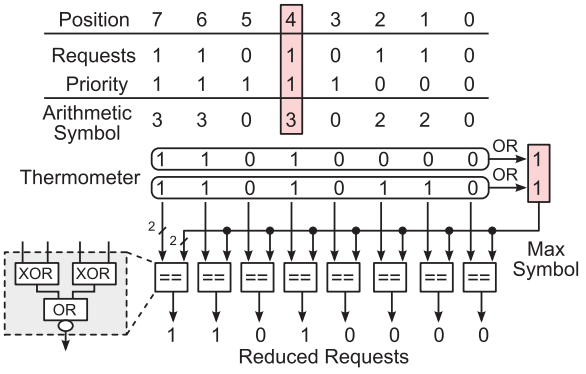


Fig. 16. The computation of the reduced request vector that contains only equal-priority requests.

vector, as shown in Fig. 15, suffices to identify the rightmost active request with the highest priority, which lies on position 4 in this example.

The transformation of the input requests and the priority vector to a reduced request vector involves finding first the maximum symbol and then marking the positions that have a symbol equal to the maximum. Since the symbols are thermometer coded, computing their maximum can be performed by a bitwise OR of all the elements of the set. Thus, in our case, we need two N-input OR gates to compute the maximum 2-bit arithmetic symbol. Then, as shown in Fig. 16, we can easily identify the positions that have the same priority as that of the maximum symbol, by employing a 2-bit equality comparator at each position. The output of each 2-bit equality comparator drives the corresponding bit of the reduced request vector, which is then passed to the fixed-priority MARX.

Extending the derivation of the reduced request vector for the case of weight-based selection is trivial. The maximum N-bit thermometer coded weight is derived in the same way as in the round-robin case, while a N-bit comparator at each bit position checks if this position has a weight equal to the maximum. Those that do, will have their corresponding bit asserted at the reduced request vector.

5 MARX-Based Switches

The design of WH switches that employ the proposed dynamic priority MARX units is straightforward. Fig. 17 depicts how a 2-input x 2-output WH switch that uses separate arbiters and multiplexers can evolve to a MARX-based switch, with RC either preceding (left subfigure), or performed in parallel to MARX (by employing LRC—right subfigure). The arbiter-multiplexer pairs at each output are directly replaced by the corresponding dynamic priority MARX units. In this way, the tasks of SA and ST of the original WH switch (see Fig. 1) are merged to a new stage called switch allocation and traversal (SAT), which is directly implemented by the new MARX units and allows for efficient low-latency WH switch implementation.

In the case of switches with VCs, the design is more complicated due to the per-input and per-output stages of arbitration and multiplexing. An input-first allocator allows each input to send to the outputs the request of a single VC. To decide which request to send, each input arbitrates locally among the requests of each input VC. On the contrary, in the case of output-first allocation, all VCs are

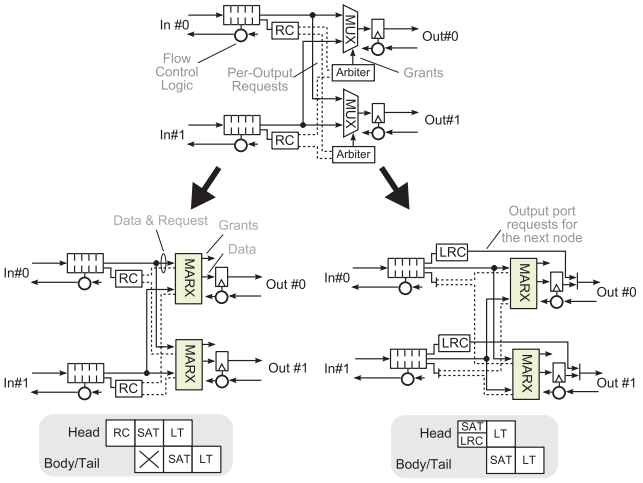


Fig. 17. WH switches using MARX units.

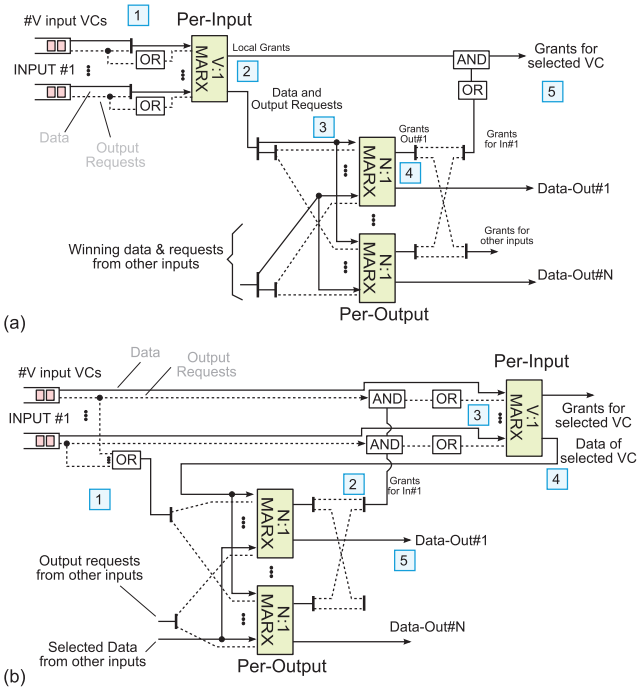


Fig. 18. VC switches with MARX units: (a) input first, (b) output first.

free to forward their requests to the output arbiters and then a local arbitration takes place again to select one among the possibly many output grants that the input received. Input-first allocation allows some time overlap between local input multiplexing and per-output arbitration. Such an overlap is not possible in output-first allocation, where both stages of arbitration should be first completed before driving the multiplexers.

The rough organization of VC-based switches with N input/output ports and V VCs per input that use MARX is shown in Fig. 18. The presented alternatives follow exactly the input and output-first allocators presented in [6], [10], where the identified arbiter and multiplexer pairs, including those of the crossbar, have been replaced by MARX units. In each figure, the order of signal propagation has been depicted with time steps so as to clarify the order of events. In both cases, we assume that each input VC has the output requests ready because LRC is employed.

Step 1 of Fig. 18a first concatenates the per-output requests of each input VC along with the corresponding data. This is done because the requests of the winning VC for each input, should reach the per-output arbiters along with the corresponding data, to participate in the output arbitration stage. If an input VC requests an output, then it will participate in the local (per-input) arbitration. If one active request exists in an input VC, this is computed by the local per-input VC OR gates. As long as the local arbitration finishes, i.e., at step 2, the data and the output requests are concurrently provided at the output of the $V:1$ MARX unit. At step 3, the winning data and the corresponding output requests are distributed to the N output MARX units that perform both global (per-output) arbitration and data movement. This task is finished at step 4, where the correct data have reached the outputs of the switch and start LT. Concurrently, on step 5, the grants of the outputs are gathered per input to inform the winning input VC (if an input's VC has won global arbitration, then the OR operation of the corresponding grant signals of the per-output MARX units will be equal to 1, and, as a result, the input's local grants will be enabled via the bitwise AND operation shown in Fig. 18a).

On the contrary, in the output-first case shown in Fig. 18b, all the output requests of the input VCs are initially gathered per output (bitwise OR operation of the VCs' request vectors) and distributed to the output MARX units. At this step (step 1), only the arbitration part of MARX works, because the data selected per input will arrive at the output MARX units on step 4. On step 2, arbitration is finished and the grants for each input are gathered together. The bitwise AND operation promotes only the requests that have been granted for every input, which are then merged together per VC via an OR operation, to form a single-bit request (per VC). On step 3, local arbitration is performed concurrently with the movement of the data of the winning VCs (those already granted by the outputs), by the per-input MARX units. The data are then distributed to the output MARX units (step 4), which switch them to the correct output on step 5. Observe that during this step, no per-output arbitration takes place, because the paths between inputs and outputs have already been set (on step 1).

We have to note that output-first allocators will always be slower than input-first allocators, either if they are implemented with separate components or with MARX macros. The reason is a fundamental property of output-first allocation (the inability of overlapping in time input multiplexing and output arbitration) and not a property of the circuits used for its implementation. The purpose of MARX is to reveal the concurrency of arbitration and multiplexing locally (at the inputs or at the outputs) and to translate it to either delay reductions or energy savings.

The matching quality of the allocator, besides its structural properties, input or output first, is decisively determined by the followed priority update policy. Sophisticated weight-based policies, such as FCFS, or age-based allocation that provably yield significant throughput benefits at the network level, can be designed more efficiently with MARX, without compromising the clock frequency of the switch (see Section 6).

In VC-based switches, it is possible all input VCs to be accommodated on a shared memory space that is implemented using an SRAM block, while the manipulation of the VCs is done via a linked-list-like structure. Since the input VCs are held inside an SRAM block, there is no multiplexer that connects the input VCs to the input port of the crossbar (practically this multiplexer implicitly exists inside the SRAM, implemented virtually by the bitlines of the SRAM). For this reason, in the first stage of allocation, only an arbiter is used and there is no need for a MARX module. MARX is employed only in the second stage, merging in one circuit the per-output arbiters and the multiplexers of the crossbar. In this case, MARX allows the overlap of buffer read and per-output arbitration tasks as well, because the arbiter inside MARX, implemented by the MAX nodes, does not require the data to be ready at the inputs of the corresponding multiplexers.

Finally, MARX can be also used for the implementation of adaptive switches [8], [27], where each input (for WH) or input VC (for VC-based switches) can request more than one output ports. The output port that will be actually selected is a matter of an additional selection unit (not shown in Figs. 17 and 18) that either picks randomly a destination or decides after sensing the state of the network [27]. Apart from the selection unit, the organization of the rest of the switch will be exactly the same as that shown in Fig. 17 or Fig. 18. Note though that the selection unit in a MARX-based adaptive switch will not be a MARX module, because MARX takes care of arbitration and data switching (multiplexing). Its utilization as a selection unit would introduce unnecessary hardware overhead without offering any benefit. Moreover, a selection unit takes into account other criteria that try to load balance traffic throughout the network and it does not perform arbitration like MARX.

6 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 MARX units. In all cases, the designs were implemented in a 65-nm CMOS technology using a standard-cell-based design flow. The parameterized form of the circuits was described in VHDL, while synthesis, placement and routing were performed using Synopsys Design Compiler and Cadence SOC encounter, respectively. Timing analysis and power measurements were performed after back annotating all parasitic information extracted from the layout.

In the first set of experiments, we want to quantify the benefits of the proposed fixed priority MARX design of Fig. 7 (denoted as MARX-FP) versus the baseline combination of a separate FPA driving a multiplexer. For both the circuits under comparison, we designed a complete switch allocator and a crossbar for an $N \times N$ switch that consists of a separate arbiter and a multiplexer per output. The combined operation of SAT determines the critical path of single-cycle switch implementations, assuming that RC [28] is performed in parallel by utilizing the routing lookahead technique [7]. 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 LT in different cycles and they are employed in almost any switch implementation. The registers at each output of the switch are loaded with a 0.2pF capacitance, which corresponds roughly to the capacitance of a 1-mm metal-5 wire in our technology.

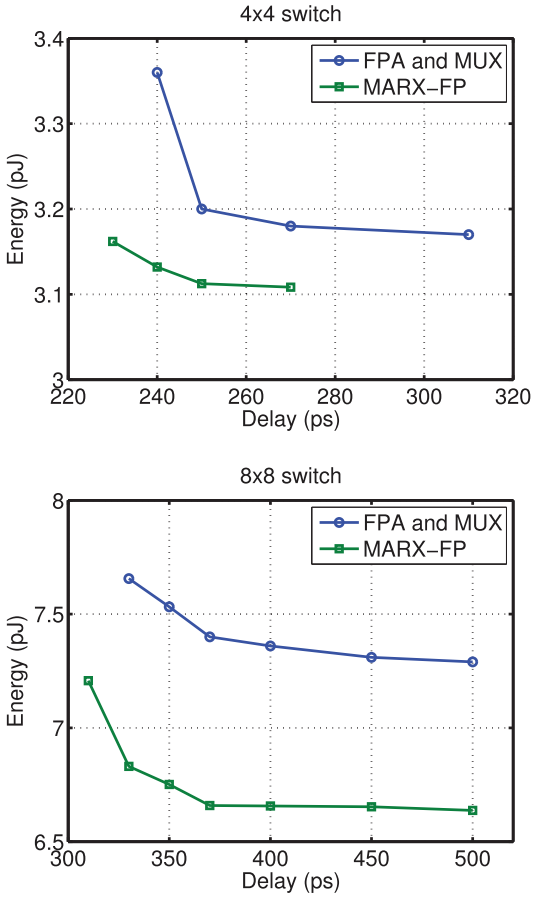


Fig. 19. The energy-delay curves for the baseline case of separate FPA and multiplexer versus the proposed MARX with fixed priority.

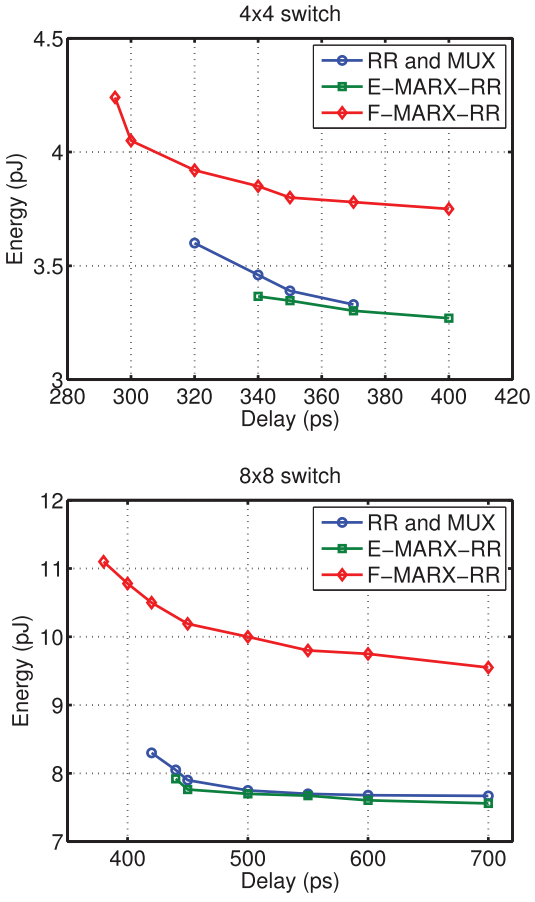


Fig. 20. The energy-delay curves for the case of round-robin arbiters and multiplexers.

The energy-delay curves derived for a 4×4 and an 8×8 switch, with a data width of 32 bits, for both circuits under comparison are shown in Fig. 19. In every case, MARX-FP is both faster and less energy consuming than the fastest version of a separate FPA and multiplexer in the literature (Fig. 4b). The same results hold even for a wider data path of 64 and 128 bits. The reported savings stem from the optimized sorting-based structure presented in Fig. 7 that parallelizes, with low complexity, request arbitration and data movement.

Although this concurrent movement of requests and data seems to increase the switching activity relative to the baseline design, this is done only slightly, assuming that both designs are driven by register-based input queues. Such concurrent movement of data and requests is chosen even in the case of SRAM-based input queues without worrying about the extra switching activity. For example, a recent design from Intel chooses to speculatively read out of the memories the requesting flits and send them to the crossbar, even if they do not actually win SA [29]. This is performed to reduce delay by overlapping in time, as much as possible, the buffer read and SA. The switching activity caused by a flit leaving the switch is the same for all methods. The only decisive parameter is the structure of the crossbar's multiplexers (Fig. 2) and the overall throughput of the switch. At high throughput, with data leaving the switch from all output ports almost on every cycle, high switching activity is inevitable.

In the following, we compared the proposed delay-optimized round-robin MARX design (denoted as F-MARX-RR, i.e., Fast-MARX-Round Robin) against a separate round-robin arbiter driving a multiplexer (denoted simply as RR). The fastest approach for the latter pair involves the parallel-prefix arbiters of [30] along with the AND-OR implementation of the multiplexer (Fig. 2a). As in the previous example, for all designs under comparison, we implemented a complete switch allocator and crossbar, including also for each arbiter the corresponding priority state and priority update logic shown in Fig. 3.

The derived energy-delay curves are shown in Fig. 20. In both cases, the proposed F-MARX-RR configuration, shown in Fig. 13, offers the minimum delay. The delay savings are more than 8 percent compared to the high-speed separate RR arbiter-multiplexer implementation and they increase above 15 percent for switch radices of 16 ports or more. The separate arbiter multiplexer and the area/energy efficient implementation of MARX (denoted as E-MARX-RR, where "E" stands for Energy efficient) consume nearly the same amount of energy, with the proposed one being slightly better, while, in terms of speed, the separate FPA and multiplexer is slightly faster than E-MARX-RR (4 percent in average). From Fig. 20, we can observe that the E-MARX-RR is on average 20 percent more energy efficient than F-MARX-RR, when compared under equal delay. Note that the area-delay behavior of all the designs follows roughly the presented energy-delay behavior, with only 2-3 percent

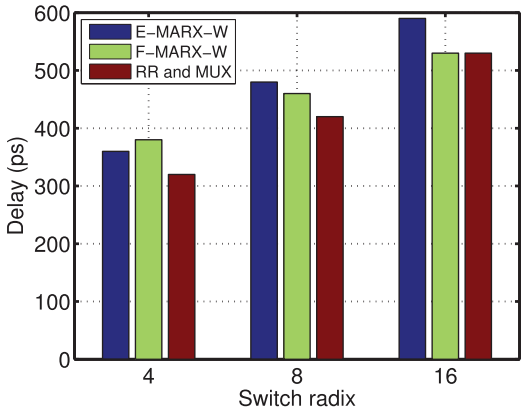


Fig. 21. The delay of the proposed switches implementing FCFS policy versus the fastest separate arbiter and multiplexer implementing round-robin policy.

difference in the area overheads of F-MARX and savings of E-MARX relative to the separate arbiter multiplexer case.

These results lead us to the conclusion that although the proposed designs are based on a generic architecture that can also handle more complex weight-based selection policies and computes the grant signals in multiple formats, it offers, even for the simple round-robin policy, two circuit alternatives that can be either faster or more energy efficient than the most efficient round-robin-only design. We also verified experimentally that the same trend is followed in the case of switches with VCs that employ the round-robin selection policy both in the per-input and the per-output allocation step, assuming, for the proposed modules, the organization shown in Fig. 18.

To measure the significance of this result, we should take also into account that previous state-of-the-art designs do not provide any obvious way of extending their functionality to other selection policies. The same holds for other ad hoc techniques that aim at simultaneously performing arbitration and multiplexing [31], [32]. Besides their tricky circuit architectures, they are designed to follow only a specific form of arbitration policy that has serious limitations and leads to network-wide performance losses.

Also, it should be made clear that the energy of the switch allocator and the crossbar is just 20-30 percent of the power consumption of the switch [33], [29]. The majority of the switch's power is dissipated by the buffers and on the network links. Therefore, even the 20-30 percent energy overhead of F-MARX-RR does not translate to more than 4-9 percent of energy overhead in the switch as a whole. For example, the capacitance of four 32-bit output links of 2 mm is equal to 51.2 pF, assuming that the wire capacitance is roughly equal to 0.2 pF/mm. If the links operate at 1 V, as in the case of our library, with 0.05 to 0.1 switching activity factor, then their energy will be between 2.56 and 5.12 pJ per operation, which is either equal or greater than the measured energy of the MARX-based switches.

The effectiveness of the generality of the proposed architecture is tested in the last set of experiments that measure the delay overhead imposed by a sophisticated weight-based policy, namely FCFS, relative to the best delay achieved by the most efficient stand-alone round-robin arbiter implementation and multiplexer (RR and MUX) that was also used in the previous experiments. As shown in Fig. 21, the delay reported from the proposed modules involves both the E-MARX and the F-MARX implementations (their weighted versions are denoted as E-MARX-W and F-MARX-W, respectively). The delay optimized version shows its full potential at high radices, while for small switches of four ports, the area optimized version has smaller delay too. Additionally, from the bars of Fig. 21, it is obvious that the delay overhead imposed by the adoption of a weight-based selection policy implemented with MARX units, relative to simple round robin with separate modules (the RR and MUX bars) is negligible and diminishes as the radix of the switch increases. In fact, at 16 ports, F-MARX-W has a delay equal to the delay of a separate round-robin arbiter and multiplexer. This is probably one of the largest advantages of the proposed macros; they can be directly utilized for implementing more efficient selection policies that yield significant throughput benefits at the network level [22], while, at the same time, these performance benefits are not compromised by the clock frequency of the switches, because the latter are only slightly slower or equally fast to the standard round-robin case.

7 Conclusions

In any engineering discipline, the quality of the building blocks that are available to the designer, crucially determines the quality of the final product. The same principle holds also in the case of on-chip network switches that can significantly benefit by the adoption of the proposed MARX units. The proposed circuits adapt efficiently to simple and more complex arbitration policies under a generic architecture, and, at the same time, offer area/energy/delay efficient implementations due to their merged arbitration and multiplexing structure and the new sorting-based arbitration algorithms. In any case, their application is orthogonal to all other architectural techniques and can help the designer invent switch organizations that are currently unexplored.

References

[1] W.J. Dally and B. Towles, "Route Packets, Not Wires: on-Chip Interconnection Networks," *Proc. 38th Design Automation Conf. (DAC)*, June 2001.

[2] A. Golander, N. Levison, O. Heymann, A. Briskman, M.J. Wolski, and E.F. Robinson, "A Cost-Efficient L1-L2 Multicore Interconnect: Performance, Power, and Area Considerations," *IEEE Trans. Circuits and Systems-I: Regural Papers*, vol. 58, no. 3, pp. 529-538, Mar. 2011.

[3] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary, "Exploring Concetration and Channel Slicing in on-Chip Network Router," *Proc. Int'l Symp. High-Performance Computer Architecture (HPCA)*, Feb. 2009.

[4] B. Grot, J. Hestness, S.W. Kekler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," *Proc. 15th Int'l Symp. High-Performance Computer Architecture (HPCA)*, 2008.

[5] W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[6] G. Dimitrakopoulos and D. Bertozzi, "Switch Architecture," *Designing Network on-Chip Architectures in the Nanoscale Era*, Jose Flich and Davide Bertozzi, eds., CRC Press, 2010.

[7] M. Galles, "Spider: A High-Speed Network Interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34-39, Jan./Feb. 1997.

[8] A.S. Vaidya, A. Sivasubramaniam, and C.R. Das, "Lapses: A Recipe for High Performance Adaptive Router Design," *Proc. Fifth Int'l Symp. High Performance Computer Architecture (HPCA '99)*, pp. 236-243, 1999.

[9] W.J. Dally, "Virtual-Channel Flow Control," *Proc. 17th Ann. Int'l Symp. Computer Architecture (ISCA)*, pp. 60-68, May 1990.

[10] D.U. Becker and W.J. Dally, "Allocator Implementations for Network-on-Chip Routers," *Proc. ACM/IEEE Int'l Supercomputing Conf.*, 2009.
[11] S.S. Mukherjee, F. Silla, P. Bannon, J.S. Emer, S. Lang, and D. Webb, "A Comparative Study of Arbitration Algorithms for the Alpha 21364 Pipelined Router," *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, 2002.
[12] Y. Tamir and H.-C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication Switches," *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13-27, Jan. 1993.
[13] J. Hurt, A. May, X. Zhu, and B. Lin, "Design and Implementation of High-Speed Symmetric Crossbar Schedulers," *Proc. IEEE Int'l Conf. Comm. (ICC)*, pp. 253-258, June 1999.
[14] A. Kumar, P. Kundu, A. Singh, L.-S. Peh, and N.K. Jha, "A 4.6 Tbits/s 3.6 GHz Single-Cycle Noc Router with a Novel Switch Allocator in 65Nm CMOS," *Proc. IEEE Int'l Conf. Computer Design (ICCD)*, 2007.
[15] M. Azimi, D. Dai, A. Mejia, D. Park, R. Saharoy, and A.S. Vaidya, "Flexible and Adaptive on-Chip Interconnect for Tera-Scale Architectures," *Intel Technology J.*, vol. 13, no. 4, pp. 62-77, 2009.
[16] L.-S. Peh and W.J. Dally, "A Delay Model and Speculative Architecture for Pipelined Routers," *Proc. Seventh Int'l Symp. High-Performance Computer Architecture (HPCA-7)*, 2001.
[17] R.D. Mullins, A.F. West, and S.W. Moore, "Low-Latency Virtual-Channel Routers for on-Chip Networks," *Proc. Int'l Symp. Computer Architecture (ISCA)*, pp. 188-197, 2004.
[18] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction Router: Yet Another Low Latency on-Chip Router Architecture," *Proc. IEEE Symp. High-Performance Computer Architecture (HPCA)*, pp. 367-378, Feb. 2009.
[19] G. Dimitrakopoulos, "Logic-Level Implementation of Basic Switch Components," *Designing Network on-Chip Architectures in the Nanoscale Era*, Jose Flich and Davide Bertozzi, eds., CRC Press, 2010.
[20] P. Gupta and N. McKeown, "Design and Implementation of a Fast Crossbar Scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20-28, Jan./Feb. 1999.
[21] N. Chrysos and G. Dimitrakopoulos, "Practical High-Throughput Crossbar Scheduling," *IEEE Micro*, vol. 29, no. 4, pp. 22-35, July/Aug. 2009.
[22] M. Pirvu, L. Bhuyan, and N. Ni, "The Impact of Link Arbitration on Switch Performance," *Proc. Fifth Int'l Symp. High-Performance Computer Architecture (HPCA)*, 1999.
[23] D. Abts and D. Weisser, "Age-Based Packet Arbitration In Large K-Ary N-Cubes," *Proc. ACM/IEEE Conf. Supercomputing (SC)*, 2007.
[24] Synopsys, "Arbiter with Dynamic Priority Scheme," *DesignWare Building Block IP*, www.synopsys.com, June 2009.
[25] N. Weste and D. Harris, *CMOS VLSI Design a Circuits and Systems Perspective*, third ed. Addison Wesley, 2010.
[26] C. Savin, T. McSmyrthus, and J. Czilli, "Binary Tree Search Architecture for Efficient Implementation of Round Robin Arbiters," *Proc. IEEE Int'l Conf. Acoustics, Speech, Signal Processing (ICASSP)*, 2004.
[27] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and Analysis of a New Selection Strategy For Adaptive Routing in Networks-on-Chip," *IEEE Trans. Computers*, vol. 57, no. 6, pp. 809-820, June 2008.
[28] J. Flich and J. Duato, "LBDR: Logic-Based Distributed Routing for NoCs," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 13-16, Jan. 2008.
[29] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. Vangal, G. Ruhl, P. Kundu, and N. Borkar, "A 2Tb/s 6x4 Mesh Network with DVFS and 2.3Tb/s/W Router in 45nm CMOS," *Proc. Symp. VLSI Circuits*, 2010.
[30] G. Dimitrakopoulos, N. Chrysos, and C. Galanopoulos, "Fast Arbiters for on-Chip Network Switches," *Proc. IEEE Int'l Conf. Computer Design (ICCD)*, pp. 664-670, 2008,
[31] K. Lee, S.-J. Lee, and H.-J. Yoo, "A Distributed On-Chip Crossbar Switch Scheduler for on-Chip Network," *Proc. Custom Integrated Circuits Conf. (CICC)*, Sept. 2003.
[32] A.O. Balkan, G. Qu, and U. Vishkin, "Arbitrate and Move Primitives for High-Throughput on-Chip Interconnect," *Proc. Int'l Symp. Circuits and Systems (ISCAS)*, 2004.
[33] J. Balfour and W.J. Dally, "Design Tradeoffs for Tiled CMP on-Chip Networks," *Proc. 20th ACM Int'l Conf. Supercomputing (ICS)*, June 2006.

Giorgos Dimitrakopoulos received the Dipl Ing in computer engineering from Computer Engineering and Informatics Department of the University of Patras in 2001, the MSc degree in "integrated hardware-software systems" in 2003, and the PhD degree from the same department in 2007. Between 2008 and 2010, he worked as a postdoctoral fellow at the Computer architecture and VLSI Systems Lab of the Institute of Computer Science (ICS) of the Foundation for Research and Technology Hellas (FORTH) and at the University of Crete. Later on, he was appointed as a lecturer to the Informatics and Communication Engineering Department, of the University of West Macedonia, Kozani, Greece. Since January 2012, he has been a lecturer of Digital Integrated Circuits in the Electrical and Computer Engineering Department of the Democritus University of Thrace (DUTH), Xanthi, Greece. His research interests lie in the broad areas of digital integrated circuits and computer architecture, and more specifically, he is interested in the design of on-chip interconnection networks for both ASIC and FPGA fabrics, in ultra low-power digital design, as well as in the definition of new architectures for high-performance graphics accelerators. He regularly serves as a reviewer for various IEEE journals and conferences, and as a member of the technical program committee in recent FPL and DATE conferences as well as INA-OCMC and NoCArch workshops.

Emmanouil Kalligeros received the Diploma in computer engineering and informatics in 1999, the MSc degree in computer science and technology in 2001, and the PhD degree in embedded testing in 2005, all from the Computer Engineering and Informatics Department, University of Patras, Greece. Between 2006 and 2008, he served as an adjunct professor at University of Patras, University of Peloponnese and University of the Aegean, teaching courses related to electronics and digital circuits design. Since 2008, he has been a faculty member at the Information and Communication Systems Engineering Department, University of the Aegean, Samos Island, Greece, where he currently holds an assistant professor position. His main research interests include VLSI design and test, design for testability, CAD methodologies for VLSI testing and test-data compression architectures. In these areas, he has published more than 30 papers in prestigious international journals and conferences. He also serves as an invited reviewer in various well-known conferences and most accredited journals. He is a member of the Technical Chamber of Greece and the IEEE.

Kostas Galanopoulos received the Diploma degree in computer engineering and informatics from the University of Patras, Greece in 2009, and is currently working toward the PhD degree in electrical and computer engineering at the National Technical University of Athens, Greece. He has coauthored six technical papers in IEEE journals and conferences. His research interests include design and optimization of mixed-signal, digital and microprocessor data path circuits, low-power optimization, and all-digital frequency synthesis techniques. He regularly serves as a reviewer for IEEE transactions and conferences. He is a student member of the IEEE.