Published in IET Computers & Digital Techniques
Received on 15th December 2011
Revised on 14th July 2012
doi: 10.1049/iet-cdt.2011.0169

ISSN 1751-8601

Comparative analysis of soft and hard on-chip interconnects for field-programmable gate arrays

J.Y. Hur[1] *K. Goossens*[2] *L. Mhamdi*[3] *M.A. Wahlah*[1]

[1]*Computer Engineering Laboratory, TU Delft, The Netherlands*
[2]*Faculty of Electrical Engineering, TU Eindhoven, The Netherlands*
[3]*School of Electronic and Electrical Engineering, University of Leeds, West Yorkshire, Leeds LS2 9JT, UK*
E-mail: JaeYoung.Hur@gmail.com

Abstract: It is well-known that any logical functionality can be implemented using the reconfigurability in field-programmable gate arrays (FPGAs). However, the reconfigurability is traded with the reduced functional performance, increased cost and increased configuration overheads. Hardwiring the interconnect fabric is gaining notice as an alternative solution to tackle the mentioned problems. In this article, first, the authors present that hardwired built-in crossbars that can improve the performance of the inter-processor communication. The authors conduct an analysis of functional performance, cost and configuration cost for soft and hard crossbar (SBAR and HBAR) interconnects. The queuing model is applied to compare soft and hard interconnects. A motion JPEG (MJPEG) case study suggests that HBAR achieve significantly better throughput and less cost compared to SBAR. Second, the authors present the effectiveness of the hardwired network-on-chip (NoC) in FPGAs. Considering the Æthereal NoC, an analysis is conducted to compare hard and soft NoCs. Consequently, the analysis, implementation and simulation indicate that the hardwired networks perform significantly better than soft networks.

1 Introduction

Field-programmable gate array (FPGA), as a key component in a modern reconfigurable platform, is increasingly used to implement modern systems on a chip. FPGA contains the configurable logic blocks (CLBs) and reconfigurable interconnects. The flexibility in FPGAs is realised by the configuration circuitry. Any logical functionality can be mapped by exploiting the reconfigurability of the device. We call an IP block 'soft' when it is mapped on the reconfigurable resources. Modern FPGAs are dominated by millions of regularly structured bit-level wire segments. The fine-grained reconfigurability is a valuable asset to implement any IP functionality with the desired granularity. However, the interconnect fabric itself has the following limitations:

(1) *Functional performance and cost:* Compared to the application specific integration circuit (ASIC) counterpart, FPGAs are slow in speed [1]. FPGA implementations are reported to occupy 35× more area, operate 3.5× slower, and use 14× more energy [1], when compared to ASIC implementation. This is mainly because of the bit-level reconfigurable interconnects. Interconnects account for more than 60% of the delay, 75% of area and 80% of power consumption [2]. Moreover, inter-IP communication functionality requires (usually significant) on-chip logic resources.
(2) *Granularity:* An inter-IP communication is mostly required to be coarse-grained (e.g. words, flits, packets, messages or transactions). However, an underlying fabric is still bit-level. The fine-grained interconnects must be used to implement any intra-IP functionality with the desired granularity. Owing to these different requirements, inter-IP and intra-IP interconnects should be differently designed. It can be noted that the interconnect fabric in modern FPGAs does not distinguish the intra-IP and inter-IP interconnects.
(3) *Wire delay and variation:* The (bit-level) wire delay in the soft networks are highly unpredictable before placement and routing step. Subsequently, it may be difficult to meet the timing requirements at design-time because of the unpredictable net delay skew.
(4) *Configuration performance and cost:* The configuration circuit contains a controller and a datapath including the configuration memory cells. Each element in FPGAs is configured by writing bitstreams onto associated configuration memory cells. Typically, the internal configuration space of the FPGA is partitioned into primitive segments, namely 'frame', which is the smallest load unit [3]. When a system is dynamically reconfigured, the functional interconnect must be (partially) reconfigured. The computation IPs are typically rectangle-shaped and they can be efficiently configured by the frame (or module) based configuration circuit. Typically, bus macros are required to be geographically spread out between different modules [3]. The partial reconfiguration of the soft networks is then not efficient because the interconnect is spread over large surface area. In this case, the network IPs occupy significant reconfigurable resources. Additionally, a significant portion of the configuration memory is

unnecessarily allocated for the partial reconfiguration of the inter-IP networks, leading to increased configuration cost and increased configuration time.

These problems can be solved by implementing the networks directly in ('hard') silicon [4–6] rather than configurable elements of the FPGA. First, the performance of the hardwired network is better and occupies less area than the soft networks. Second, the granularity problem can be solved by implementing the hardwired network at a coarse-grain. Third, the hardwired network is a pre-verified IP and provides highly predictable timing information. Fourth, the partial reconfiguration is highly efficient because the hardwired fabric and reconfigurable fabric are by nature decoupled. Regular hardwired NoC is promising for future FPGAs. To bridge the gap between the state-of-the-art soft interconnects and the future hardwired NoC, we also propose to hardwire crossbars (HBAR) in FPGAs. We conduct a comparative analysis of functional performance of soft and hard interconnects using queuing theory. Furthermore, we evaluate the functional cost, configuration performance and configuration cost. The main contributions of this work are:

- We conducted wire fabric and bitstream analysis. Considering Virtex-II Pro as a targeted device, we derive Rent's exponent from the number of wires per logic block. Our study indicates that wires become increasingly critical resources.
- We propose that HBAR are built in FPGAs for the inter-IP communications. In our MJPEG case study, the HBAR is 5× better in network throughput, compared to the soft crossbar (SBAR).
- We present the effectiveness of the hardwired circuit-switched NoCs (HCSN) in FPGAs. Considering the Æthereal NoC [7] as an example, we apply the Jackson's queuing model [8] to compare with the soft circuit-switched NoCs (SCSN). The simulation and analysis results indicate that HCSN provides 4.2× better network latency for the MJPEG task graph, when compared to SCSN.
- The functional cost is obtained from the implementation. Additionally, configuration performance and cost are derived. We present that these overheads can be reduced using the hardwired interconnects.

This paper is organised as follows. In Section 2, the related work is reviewed. In Section 3, wire fabric is analysed and HBAR are discussed. We present our performance analysis for hard and soft on-chip networks with a case study in Sections 4 and 5. Implementation and experimental results are presented in Section 6. Finally, conclusions are drawn in Section 7.

2 Related work

In [5, 6], general approaches on the hardwired packet-switched NoCs are discussed, where architectures, implementations, and analysis details are not presented. Architectures and implementations of soft, firm, and hard Æthereal NoC [7] instances are compared in [4]. Additionally, the HCSN in [7] is reprogrammable in that network parameters (such as routing paths or slot allocations) can be dynamically 'programmed' by means of memory-mapped I/O (MMIO) registers rather than 'configured'. Furthermore, unification of configuration and functional network is proposed in [4]. In [4], usage examples to use the hardwired NoC is described. To our knowledge, there is no prior work related to HBAR interconnects in FPGAs. In [9], we presented the HBAR and an analysis. In this paper, we additionally present an analysis of the wire fabric, bitstreams, configuration performance and cost.

Little has been reported regarding the queuing analysis of on-chip networks. In [10], a single router with virtual channels is modelled. In this work, we present an analysis of a network on chip. In [11], a queuing analysis for a single output-queuing router is conducted to determine the buffer size and reduce packet loss probability. An M/D/1/B model with deterministic service rate is used. In practice, packet loss should be avoided. In [12], Jackson's model is applied to analyse SBAR interconnects. In this work, we also present an analysis of hardwired circuit-switched networks (CSN).

3 HBAR in FPGAs

In this section, we first describe the relationship between the number of wires and logic blocks. Second, we describe the portion for bitstream wiring resources. Third, the HBAR interconnect fabric is discussed.

3.1 Wire fabric analysis

The relationship between the average number of terminals and blocks in a partitioned design can be described by the Rent's rule [13]. This is represented by $T = tB^p$, where T is the total number of terminals, t is the average number of terminals per logic block, B is the number of logic blocks, and p is the Rent exponent. The t and the p values vary depending on the design. An experimental study in ASIC technology indicates that the typical value of p is between 0.5 and 0.75 [13]. To derive the p value in modern FPGAs, we used FPGA editor [3] tool. Fig. 1 depicts a primitive CLB tile of the Virtex-II Pro device consisting of logic slices and wiring resources. The FPGA device consists of regularly structured CLB tiles and interconnects. A logic slice cell contains look-up tables (LUTs), flip-flops, and associated logic gate resources. The rest are wiring resources that include the switch-box and various types of wires. We counted the number of wires (in a handcrafted

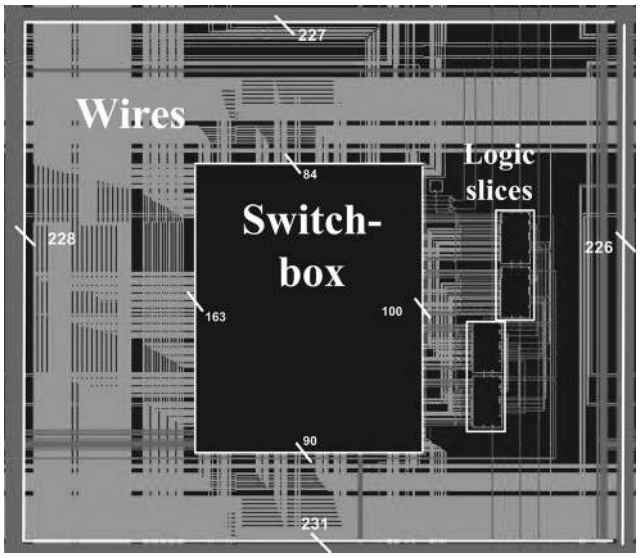

Fig. 1 *Tile in Virtex-II Pro xc2vp30 device*

doi: 10.1049/iet-cdt.2011.0169

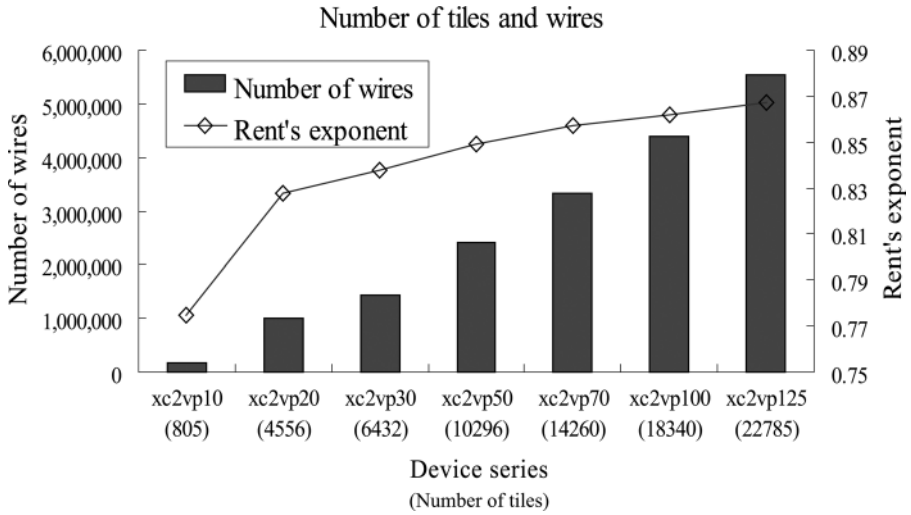

Fig. 2 *Total number of wires in Virtex-II Pro device series*

way) around the CLB tile in the Virtex-II Pro device. As a result, we obtained the value t of 936. Fig. 2 depicts the relationship between the number of wires and the number of CLB tiles. The number of wires is taken from [14, 15]. The number of CLB tiles are also depicted in the parenthesis of the x-axis in Fig. 2. Fig. 2 can be interpreted in the following ways. The value of Rent exponent p is in the range of [0.78, 0.87], which is greater than the typical range [0.5, 0.75]. This indicates that the routing wires in FPGAs are more abundant than typical ASIC implementations. However, as the number of tiles increases, the number of wires increases in a similar 'linear' manner. This is due to the fact that CLB tiles and wires are regularly structured in an island style. Obviously, the 'number of wires' should grow more than linear manner to maintain the point-to-point wirability between (especially long-distance) logic tiles. This indicates that the routing wires become increasingly critical and decreasingly scalable resources.

We also study how the wires are utilised in the bitstream. This can be quantified by Number of frames for wires/Total number of frames because the frame is the atomic configuration unit. Fig. 3 depicts the portion that on-chip resources occupy in the bitstream for the Virtex-II Pro xc2vp30 device. The xc2vp30 device contains 1756 frames in total [16] and wires occupy 1064 frames [17]. Therefore the routing wires occupy 1064/1756 or 60% of the entire bitstream, whereas the logic slices occupy only 7%. This means that the majority of the bitstream (or the configuration memory) is occupied by the wire resources. It can be noted that the configuration time is proportional to the bitstream size. Accordingly, the routing wires contribute to the configuration time of the entire device by 60%. Our analysis indicates that the bit-level interconnects are not only the critical component for the functional performance and cost but also for the configuration performance and cost. We

Fig. 3 *Percentile occupation of a bitstream in Virtex-II Pro xc2vp30 device*

targeted a Xilinx Virtex-II Pro device particularly because a partial reconfiguration is supported, whereas any reconfigurable hardware can be a targeted device as well. The state-of-the-art FPGA devices have reduced configuration frame size and increased logic density. However, all of these devices stem from the same origin and the architectural difference is minor.

3.2 HBAR interconnects

A soft shared bus is widely used for FPGA platforms. Disadvantages of soft interconnects and advantages of hardwired interconnects are described in Section 1. We propose to HBAR in FPGAs to bridge the gap between the soft interconnects [3] and the future hardwired NoCs [4]. When the bus fabric is hardwired, the available bandwidth in the hardwired bus significantly increases because of the increased clock frequency. The bus component is only needed to be instantiated as a hard macro. Accordingly, the contention probability of a network is reduced, which means that the hardwired bus performs better than the soft bus. However, because many buses are sequential, they suffer from traffic congestion before concurrent interconnects do. Therefore we propose to hardwire the crossbars as built-in components in FPGAs. The main advantage of a crossbar is that minimum traffic congestion occurs inside the crossbar because the dedicated interconnects are physically established. Data transactions inside a crossbar can be fully parallel. Although an area cost is a bottleneck, the area of the crossbar can be adequate for small or intermediate sized interconnects [18]. The cost of crossbars quadratically increases as the number of ports increase. As described in Section 6, our experiment indicates that a radix of 8 or 12 can be the feasible option. However, the optimal radix can vary with different target systems and technologies.

Fig. 4*a* depicts the HBAR as built-in components in FPGAs. Fig. 4*b* depicts the transaction protocol example in [19]. The transaction traffic on the master–slave architecture can operate with the local-write and remote-read scheme. The master processor locally writes data to the local slave first in first out (FIFOs). The master processor remotely requests data to the remote slave FIFOs. If the remote FIFOs are not empty, the master processor remotely reads the FIFOs. The interconnects forward these requests

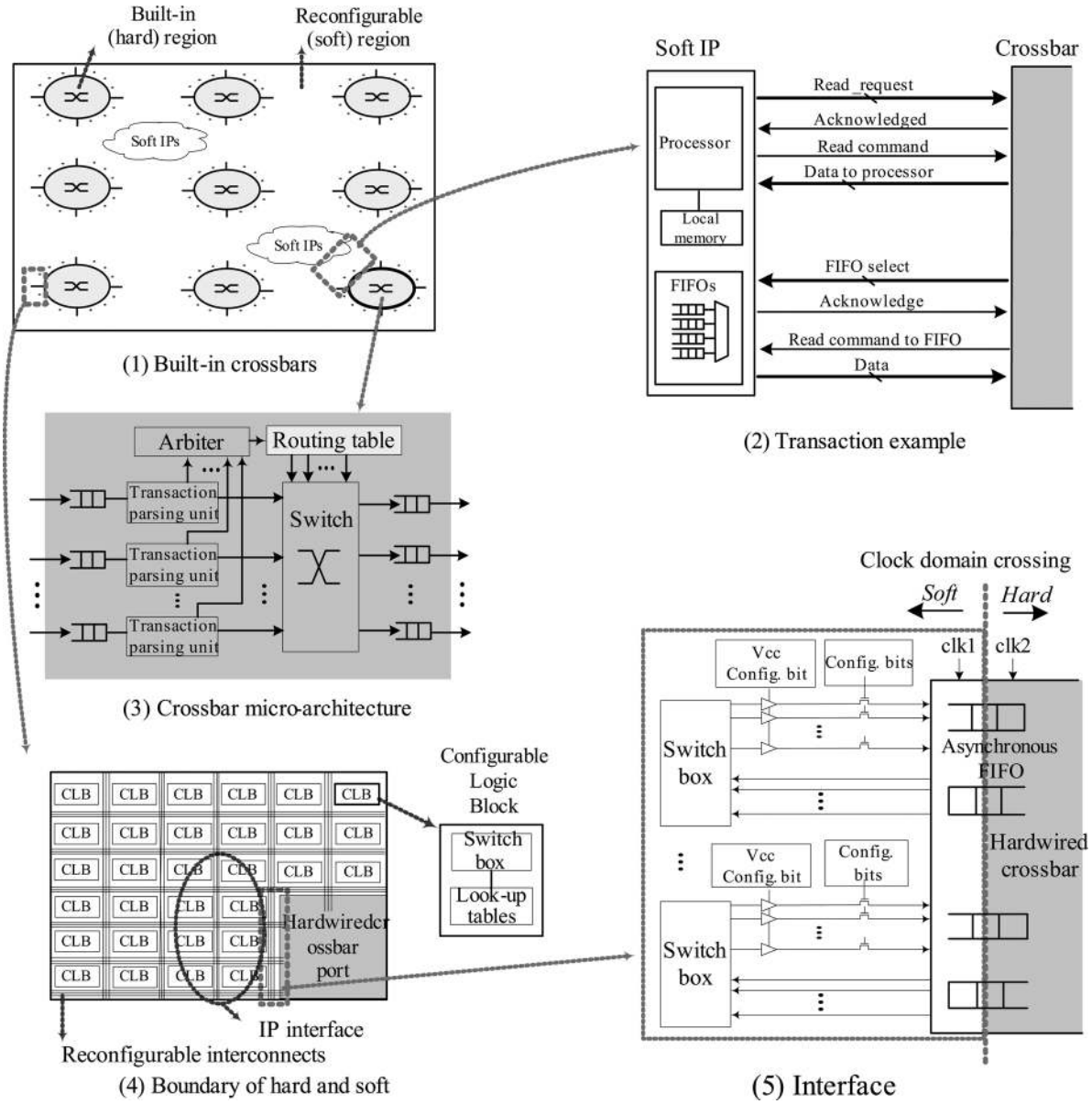

Fig. 4 *Built-in crossbars and physical interface in Xilinx FPGAs*

(from the master processor) and data (from the slave FIFOs). It can be noted that the transactions can be based on any protocol, such as AXI [20] and OCP [21]. In practice, significant overheads for reading words from the FIFOs in the soft domain can be a bottleneck in the system. This may render the speedup offered by the crossbar insignificant. In this work, we focus on the comparative interconnect performance analysis. It can be noted that the hardwired interconnects have better functional interconnect performance per area, given that the area of hard interconnects is significantly less than the soft interconnects. Furthermore, by accommodating the hard interconnects, the bitstream size, required configuration memory, and the configuration time can be significantly reduced in FPGAs, because the hard domain does not need to be configured as described in the next sections. In the presented scheme, a synchronisation needs to take place between the soft and the hard clock domain. Fig. 4*c* depicts a possible micro-architecture of a crossbar including asynchronous FIFOs (for the clock domain crossing), an arbiter, a routing table and a switch. Asynchronous FIFOs are instantiated at each crossbar port. Although these FIFOs typically add a few cycles of latency, with proper dimensioning they do not reduce throughput. It can be noted that clock synchronisers can be required at each router port in the general NoC-based GALS architectures [22]. Fig. 4*d* depicts a physical layout for the Xilinx FPGA. The soft portion consists of reconfigurable logic elements (such as LUTs) interconnected by reconfigurable interconnects (such as switch box and wires). Fig. 4*e* depicts a possible hard and soft interface circuit, in which the interconnection is controlled by the configuration bits.

The data width should be coarse-grained. In this work, we considered the data-width of 32-bit. It is possible that a monolithic or multiple crossbars can be utilised for different use-cases with different connectivity options. Multiple crossbars can be connected across the soft domain with different topologies as far as traffic does not incur deadlock. The deadlock avoidance conditions vary with communication protocols. When AXI protocol (e.g. in Xilinx Virtex-5) is considered, a deadlock can occur when transactions with the same one-dimensional (ID) have different routing paths [23]. Since the crossbar is hardwired and the topology can be configurable, the traffic initiator

doi: 10.1049/iet-cdt.2011.0169

can maintain the deadlock free routing by deterministic source routing [7]. In the next section, the presented hardwired interconnects are evaluated in the following perspectives:

- *Functional performance and cost*: Based on our implementation of hard and soft interconnects, we analytically derived comparative functional crossbar performance in Section 4 and NoC performance in Section 5. Functional cost is evaluated in Section 6.
- *Configuration performance and cost*: We derived configuration time and bitstream size for soft interconnects in Section 6.

4 Performance analysis of HBAR

To conduct functional performance analysis in FPGAs, we conduct queuing analysis based on the following considerations. First, we exploit the fact that traffic information can be extracted from the application specification. This means that a-priori logical information such as topology and bandwidth can be exploited for the analysis and design. Second, we usually reuse pre-verified IP components and their specifications. This means that the physical information such as the area, the clock speed, and/or latency of IPs are available at design time. Third, the network performance varies with specific communication patterns. Our analysis intends to compare hard and soft interconnects by deriving relative network performance. The network parameters (e.g. buffer sizes) are usually conservatively dimensioned at design time [24]. Subsequently, it is desirable to derive the performance conservatively. To do this, we derive an approximated service time and utilise the open queuing model [8] for the comparative analysis. The functional performance is derived using our simplifying assumptions, for example, the clock speed and the area for hard and soft interconnects were independently measured. Additionally, we assumed that traffic has Poisson-distributed patterns and independent inter-arrival time, as these assumptions are often utilised in the literature [8, 25, 26]. Although the traffic may be non-Poisson distributed and inter-related, we were able to efficiently derive the comparison (network throughput and latency) between hard and soft interconnects, which is sufficiently accurate for relative comparisons. For Poisson-distributed incoming traffic, Jackson model can be used for a network of queues [26]. For the analysis, we reuse specifications of network IPs in [7, 12]. As a system model, we assume that the physical FIFOs are established for a logical connection to constitute a network of queues.

4.1 Queuing model

Jackson's model states that the number of items in the system is the summation of the number of items in the individual queuing systems. Then the response time is derived by dividing the number of items (in the system) by the arrival rate of the incoming traffic. Accordingly, the response time is formulated as

$$T_{\text{response}} = \frac{1}{\lambda}\sum_{i=1}^{N}\frac{\lambda_i}{\mu_i - \lambda_i} \tag{1}$$

where N is the number of individual queuing systems. λ is the incoming arrival rate to the entire system. λ_i is the incoming arrival rate to the ith queuing system. μ_i is the service rate of the ith queuing system. Subsequently, $\sum_{i=1}^{N}\lambda_i/(\mu_i - \lambda_i)$ corresponds to the number of items in the entire system.

Fig. 5 depicts our model for an MJPEG application. A task graph with seven logical connections is depicted in Fig. 5(1a), where numbers on the edge indicates the minimum bandwidth requirement of an application. The bold line represents streaming data path for an application. The corresponding network of queues is depicted in Fig. 5(1b). N is 7, which means that there are seven queuing systems (numbered 1–7). Fig. 5(1c) depicts individual queuing systems for each logical connection. The queuing system consists of the waiting queue and the server. A server means the network component that provides transportation service. To compute λ_i in (1), we utilise the bandwidth information in Fig. 5(1a). As an example, λ_1 is computed by $(62/(62 + 0.6 + 1 + 0.6))\lambda = 0.97\lambda$. For a given λ, the response time is determined from μ_i. Figs. 5*b* and *c* depict the traffic mapping onto different interconnects such as crossbars or NoCs. The service rate μ_i varies with different network components. As depicted in Fig. 5*b*, a connection consists of two logical channels, a 'request' and a 'response' channel.

4.2 Network performance

(1) *Network service rate:* The network service rate μ_{token} for a single token can be derived by

$$\mu_{\text{token}} = T_{\text{token}}^{-1} = (T_{\text{arbit}} + T_{\text{transmit}})^{-1} \tag{2}$$

where T_{token} is the network service time for a single token, from the first word (in the departure queue) to the last word (absorbed by the destination IP). A 'token' is the set of consecutive words and refers to the primitive communication unit. The service time T_{token} is composed of the arbitration time T_{arbit} and the actual data transmission time T_{transmit}.

(2) *Network performance:* The network response time T_{response} can be derived by substituting the μ_{token} in (1).

4.3 Crossbar analysis

We consider the SBAR or HBAR as depicted in Fig. 5*b*. For these crossbars, we use the formulation in [12] summarised as follows. The arbitration time T_{arbit} for a full crossbar is approximated as

$$T_{arbit} \simeq \frac{\left\lfloor \frac{\#ports}{2} \right\rfloor + C_{hand}}{f_{net}} \tag{3}$$

where a request check latency is approximated as $\lfloor \frac{\#\text{ports}}{2} \rfloor$ cycles. C_{hand} refers to the handshaking latency in number of cycles. The arbitration time T_{arbit} in our crossbar varies with the number of ports #ports. The transmission time T_{transmit} in (2) corresponds to the token size as derived below

$$T_{\text{transmit}} = \frac{S_{\text{token}}}{f_{\text{net}}} \tag{4}$$

where S_{token} denotes the token size or the number of words. f_{net} refers to the clock frequency of a network.

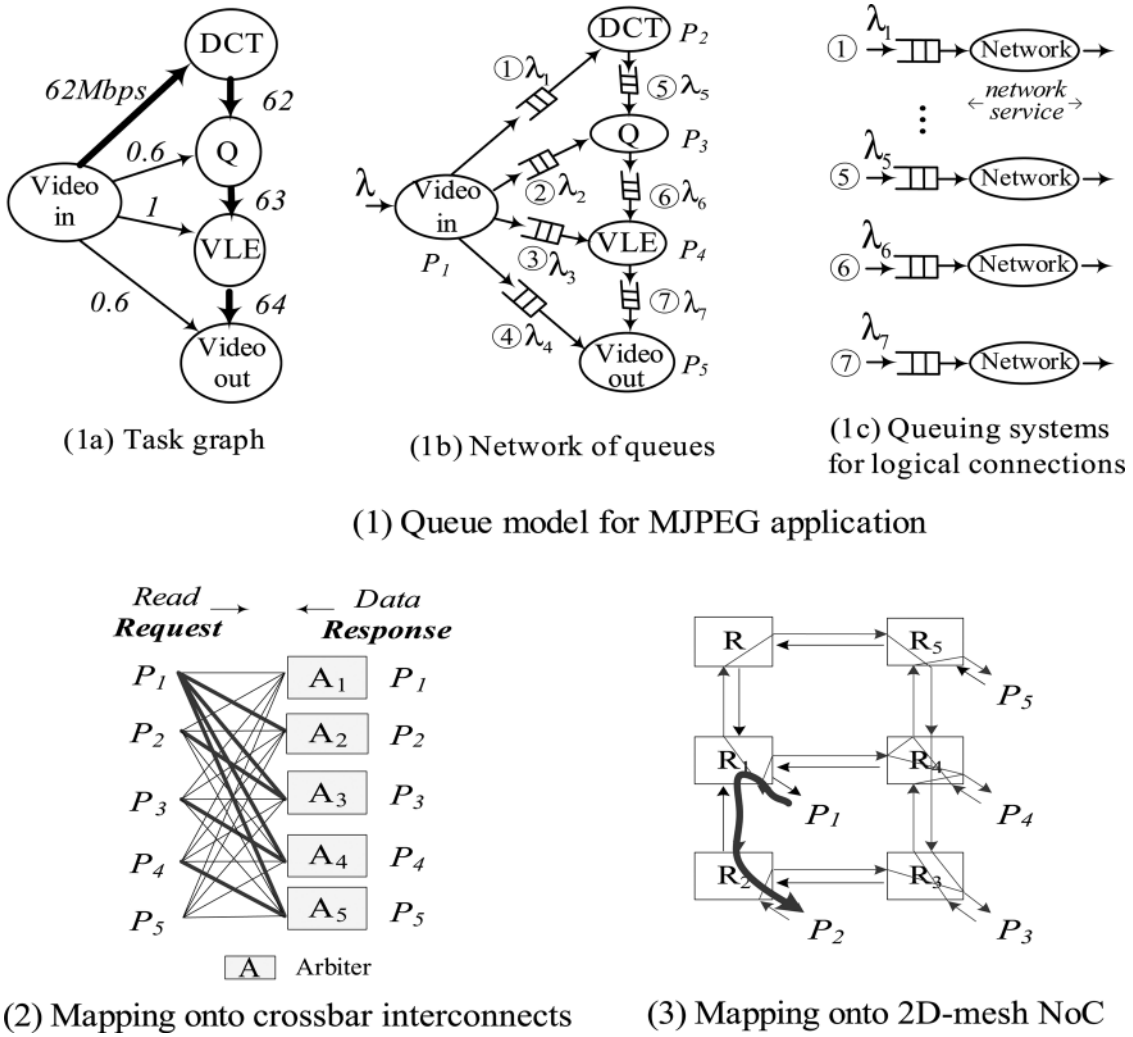

Fig. 5 *Queue model for MJPEG application and mapping onto networks*

4.4 MJPEG case study for HBAR

We derive the functional performance of SBAR and HBAR for the MJPEG specification depicted in Fig. 5(1a). We consider the token size $S_{\text{token}} = 3$ words and the number of ports #ports = 8 ports. The handshaking latency C_{hand} is 2 cycles and the clock frequency f_{net} is 446 MHz from the implementation (see Table 1 in Section 6). The actual arbitration delay depends on the arbitration (e.g. priority, sequential) and how it is implemented. Here we use the sequential arbiters of [12] that have a variable arbitration time. The worst-case arbitration delay is shown in (3). To roughly compare the HBAR and SBAR, (3) is a conservative approximation based on our implementation.

Table 1 Hardware implementation of crossbars

Type	Size	Area, slices	Clock freq., MHz
Soft (90 nm CMOS FPGA Virtex-II Pro)			
SBAR	8 ports	2052	95
	12 ports	4188	73
Hard (130 nm CMOS ASIC)			
		Area, mm^2	
HBAR	8 ports	0.11	446
	12 ports	0.29	410

The handshake time is added since the handshaking can require a latency (request 1 cycle and response 1 cycle).

1. *Network service rate:* The network service rate μ_{token} in the HBAR is derived as follows. Since $C_{\text{hand}} = 2$ cycles and $f_{\text{net}} = 446$ MHz, T_{arbit} is derived by $\lfloor \frac{8}{2} \rfloor + 2/446 \times 10^6$ s for (3). Since $S_{\text{token}} = 3$ words, the transmission time T_{transmit} is derived by $3/446 \times 10^6$ s. The μ_{token} is derived by substituting T_{arbit} and T_{transmit} in (2). Subsequently, $\mu_{\text{token}} = 446 \times 10^6 / \lfloor \frac{8}{2} \rfloor + 2 + 3 = 49 \times 10^6$ tokens/s. This means that the HBAR provides a physical network bandwidth of 49×10^6 tokens/s for a logical connection. Since word rate is derived by (token rate) × (token size in words), this is equivalent to 147×10^6 words/s for a connection.
2. *Network performance:* The network response time is derived by substituting the network service rate μ_{token} in (1). The network performance for SBAR can be derived similarly. Fig. 6 depicts the network performance for SBAR and HBAR. As a result, HBAR performs significantly better and provides 5× better throughput than SBAR. This is mainly because of the higher clock frequency.

5 Performance analysis of CSN

In this section, we extend our performance analysis for the CSN. We consider guaranteed throughput (GT) Æthereal

doi: 10.1049/iet-cdt.2011.0169

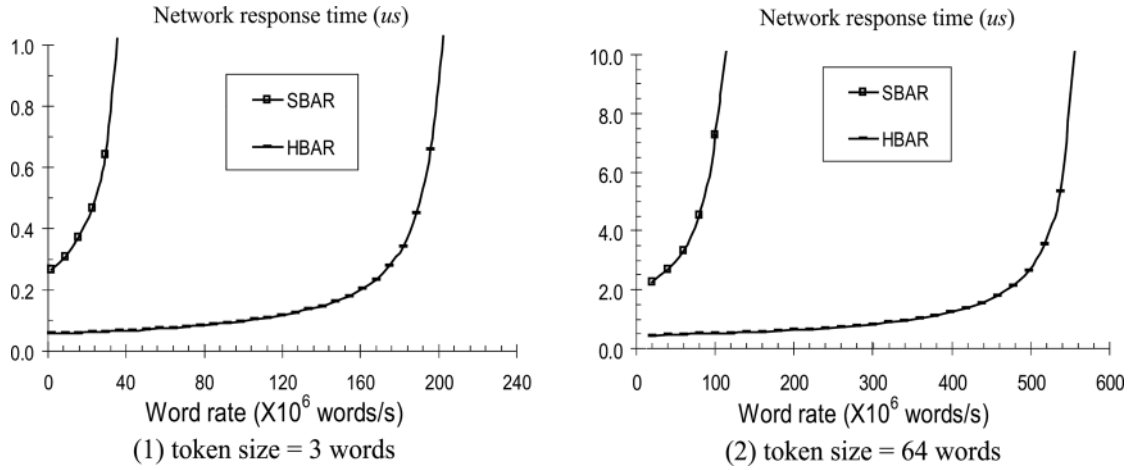

Fig. 6 *Crossbar performance for a single token for MJPEG task graph in Fig. 5*

NoC [7] as an example. The required bandwidth for each logical connection is reserved by allocating time-division-multiplexed slots. The global scheduler in the network interface multiplexes (or arbitrates) channels based on the allocated slot table and the remote buffer space. When the channel is arbitrated, the worst-case transmission time in the router network is derived similarly to a crossbar. This means that the worst-case performance of the CSN can be analysed similarly to the crossbar. The main differences are the arbitration time and the number of hops that the packet (or token) traverses in the multi-hop router network. This also means that the GT-mode Æthereal NoC operates as a virtual crossbar with a physically pipelined transmission. We assume that connections are long-lived, and ignore the time associated with their set up and tear down [27]. Similar to crossbar analysis in the previous section, the network performance can be derived by substituting the service rate μ_i in (2) to (1). The service time T_{token} in (2) is composed of the arbitration time T_{arbit} and the actual data transmission time T_{transmit} as derived by the following latency model.

5.1 Latency model

(1) *Arbitration time:* The arbitration time is determined by the slot size, the slot table size, and the number of allocated slots for a channel. The arbitration time T_{arbit} for a token is approximated by

$$T_{\text{arbit}} = S_{\text{slot}} \lceil \frac{S_{\text{tab}}}{2A_{\text{slot}}} \rceil / f_{\text{net}} \tag{5}$$

where S_{slot} denotes the slot size in number of words. S_{tab} denotes the slot table size in number of slots. A_{slot} denotes the number of slots that is reserved for a channel in the slot table. In this work, a 'slot' contains three words (in the worst case, 1 header and 2 payload words). Similar to (3), we divide by 2 for the circular round-robin to derive an approximate average arbitration time. Note that T_{arbit} is an approximation because it assumes slots are equally distributed and does not consider the (small) messagisation overhead.

(2) *Transmission time:* Similar to the crossbar in Fig. 5*b*, the transaction consists of 'read request' and 'data response' channels. The transmission time consists of the time to send data, the pipeline delay, and the length of a message, which can be approximated by

$$T_{\text{transmit}} = \begin{cases} \dfrac{\lceil \dfrac{S_{\text{req}}}{S_{\text{slot}-1}} \dfrac{S_{\text{tab}}}{A_{\text{slot}}} \rceil + \#\text{hop}C_{\text{SW}} + C_{\text{misc}}}{f_{\text{net}}} & \text{for request} \\ \dfrac{\lceil \dfrac{S_{\text{resp}}}{S_{\text{slot}-1}} \dfrac{S_{\text{tab}}}{A_{\text{slot}}} \rceil + \#\text{hop}C_{\text{SW}} + C_{\text{misc}}}{f_{\text{net}}} & \text{for response} \end{cases} \tag{6}$$

where $(S_{\text{slot}} - 1)$ refers to the slot size in number of payload words, whereas a slot contains 1-word of header. S_{req} and S_{resp} denote the token size in the number of words for the request and response channel, respectively. The first term $\lceil \frac{S_{\text{req}}}{S_{\text{slot}-1}} \frac{S_{\text{tab}}}{A_{\text{slot}}} \rceil$ refers to the number of cycles to send data. This is an approximation because it assumes slots are equally spaced. The second term refers to the pipeline delay. #hop refers to the number of intermediate routers in the routing path. C_{SW} denotes the number of cycles for the switching per router hop. The third term C_{misc} denotes the number of cycles spent in the network interfaces for packetisation and de-packetisation.

5.2 MJPEG case study for hardwired circuit-switched networks (HCSN)

We derive the performance of the HCSN. The MJPEG task graph is mapped to 2×3 two-dimensional (2D)-mesh topology as depicted in Fig. 5*c*. Fig. 7*a* depicts the connection between P_1 and P_2, where P_1 sends data to P_2 through the response channel. Design parameters of the Æthereal NoC are the following. The slot size S_{slot} is three words. The clock frequency of the hardwired network $f_{\text{net}} = 500$ MHz from the implementation. The switching latency per router hop C_{SW} is 3 cycles from the implementation. The slot table size S_{tab} and number of the reserved slots per channel A_{slot} are derived from the bandwidth distribution. The miscellaneous cycles C_{misc} is 3 cycles, since the request, packetisation, and de-packetisation requires 1 cycle each. We used the automated design flow of [28] to obtain S_{tab} and A_{slot} for an MJPEG task graph. As a result, S_{tab} is 4 slots and A_{slot} is 1 slot per channel.

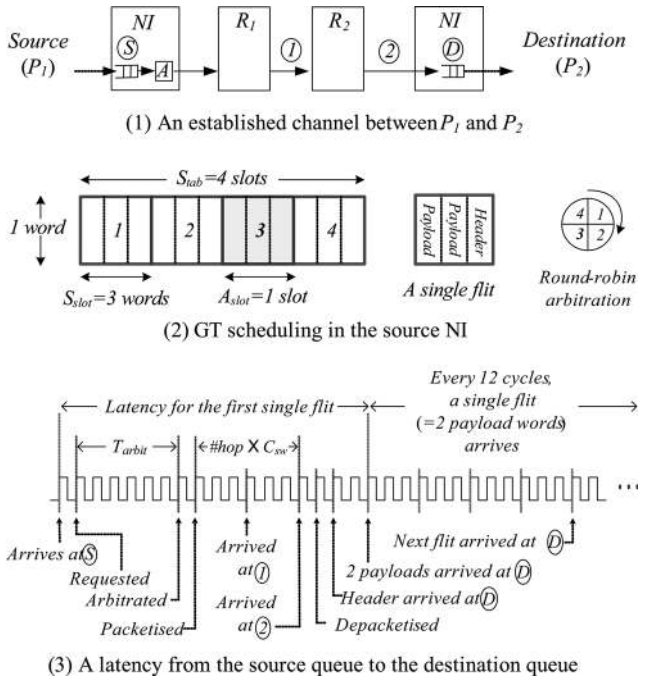

Fig. 7 *An example of the delay model of CSN*

The arbitration time and transmission time are derived as follows:

(1) *Arbitration time:* The arbitration time is derived by substituting the $S_{slot} = 3$ words, $S_{tab} = 4$ slots, and $A_{slot} = 1$ slot in (5). Fig. 7*b* depicts an example. The round-robin pointer points to the 3rd slot and the 1st, 2nd, 4th slots are occupied by other channels. Since each slot contains three words, the arbitration requires approximately $\left(3\lceil\frac{4}{2\times 1}\rceil\right) = 6$ cycles. Subsequently, the arbitration time T_{arbit} is derived by $3\lceil\frac{4}{2\times 1}\rceil/500\times 10^6 = 12$ ns per channel. In Fig. 7, we clarify the definitions of word, flit, packet and token. Flit is the minimum flow control unit. Packet can be composed of multiple flits. In the examples in the paper, a token is the primitive communication unit, which is equivalent to a packet. A packet consists of one or more flits, each of which contains three words (or phits) of 32 bits. Each packet contains one header phit, and zero or more data phits, depending on the slot assignment and data availability.

(2) *Transmission time:* Fig. 7*c* depicts a latency from the source queue to the destination queue. Since $S_{slot} = 3$ words and $S_{req} = S_{resp} = 3$ words, the time to send data is $\lceil\frac{3}{3-1}\frac{4}{1}\rceil/500\times 10^6 = 12$ ns. This means that a single flit (2 payload words) can be transmitted per every revolution of 12 (=3 words × 4 slots) cycles. From the topology mapping and the routing strategy, the number of hops #hop is obtained for each channel. The routing paths are depicted in Fig. 5*c* for the response channels. As an example, #hop between P_1 and P_2 is 2 (see bold line in Fig. 5*c*). Since $C_{SW} = 3$ cycles and $C_{misc} = 3$ cycles, $T_{transmit}$ is derived by $\lceil\frac{3}{3-1}\frac{4}{1}\rceil + 2\times 3 + 3/500\times 10^6 = 30$ ns for a channel between P_1 and P_2.

(3) *Network performance:* The network service rate μ_{token} can be derived by $1/(T_{arbit} + T_{transmit})$. As an example, the service rate μ_1 for the connection P_1 and P_2 is derived by $1/(2(12 + 30)\text{ns}) = 12\times 10^6$ tokens/s. We multiply by 2 because a connection consists of two channels, the 'request' and the 'response' channel. The total network response time is derived by substituting individual service rates in (1). Similarly, when a packet contains a single flit (or 3 words), the performance of hard and soft networks are derived, as depicted in Fig. 8*a*. Fig. 8*b* depicts the network response time when a packet contains 20 flits (or 60 words). As a result, the hardwired NoC is significantly better in latency and throughput than the soft NoC.

The service rate in the previous section is conservatively derived, by considering the sequentially operated computation IPs. Typically, the multi-hop latencies and the arbitration latencies can be hidden, since multiple logical channels are pipelined in the shared physical links. As described earlier, we consider the worst case latency model for the analysis to roughly compare hard and soft interconnects.

6 Implementation and experimental results

In the previous sections, functional performance is presented. In this section, to evaluate functional cost and configuration overheads, we conduct three experiments based on

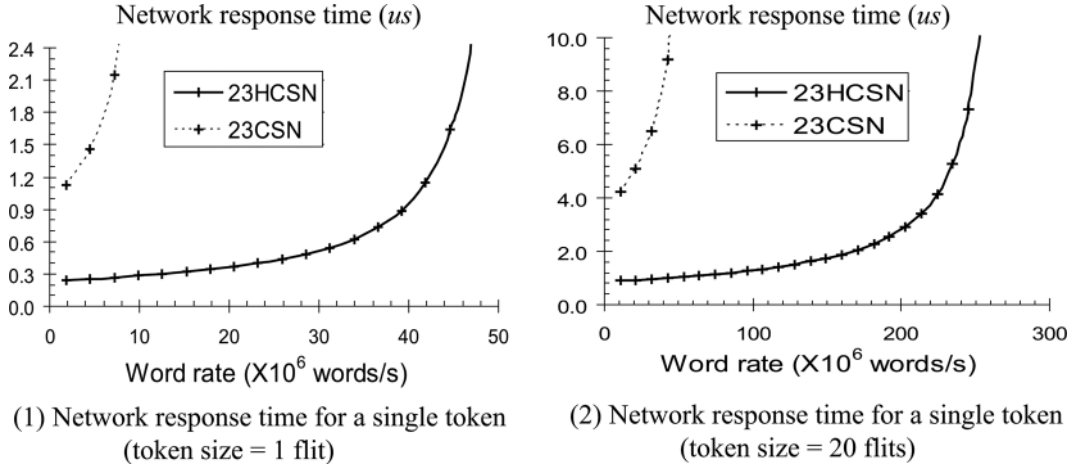

Fig. 8 *NoC performance for MJPEG task graph in Fig. 5*
23(H)CSN denotes a soft (hardwired) CSN with 2 × 3 2D-mesh topology

doi: 10.1049/iet-cdt.2011.0169

hardware implementation. First, we present area cost of the soft and hard functional interconnects. Second, configuration performance and cost overheads are derived from the implementation. Third, we conduct the simulation to verify the analytical latency model of CSNs.

(1) *Area cost*: To measure the area cost, the networks are synthesised in FPGAs for soft networks and in ASICs for hardwired networks, as shown in Tables 1 and 2. To obtain the area and the frequency of the hardwired interconnects, we used generic ASIC design flows (netlist synthesis, floor-planning, instantiation of power grids, place standard cells, route wires between standard cells, and design rule checks) and obtain the graphic database Format (GDS) to be integrated in FPGA implementation. We conducted ASIC placement and routing, using TSMC standard cell 130 nm library and Cadence first encounter to layout the hard interconnects. For soft networks, we used Xilinx ISE tool to synthesise, place and route in Virtex-II Pro device with 90 nm CMOS technology. As a result, the clock frequency of hardwired networks is 4.7× higher than soft networks. Though we experiment with different technologies, the implementation results clearly indicate that the clock frequencies of hardwired networks are significantly better than soft networks. This means that the hardwired networks provide much higher bandwidth. The area overhead of hardwired networks is also significantly smaller, compared to soft networks. As an example, the area of the 12-port HBAR is 0.29 mm^2. Considering the large area ($\simeq$ 397 mm^2 for our targeted xc2vp30 device, estimated die size in [29], the area overhead of the hardwired network is small. For comparison, the soft full crossbar SBAR with the same design occupies more than 30% of the available logic slices in the targeted xc2vp30 device. This means that the soft network occupies significant logic resources.

(2) *Configuration performance and cost*: Soft networks by nature entail configuration overheads in terms of configuration time, configuration memory and bitstream. To analyse the configuration overhead of the soft networks, we derive a lower bound configuration time based on the utilised area. The configuration time is determined by the required number of frames. The required number of frames varies with placement and routing policies. However, the 'lower bound' of the configuration time (or the number of frames) can be derived from utilised logic slices. Assuming that the utilised logic slices are maximally packed into each frame, the lower bound of the number of frames can be derived as

$$\left\lceil \frac{\text{number of utilised slices}}{\text{number of slices per CLB} \times \text{number of CLBs per column}} \right\rceil \times (\text{number of frames per column})$$

The ceiling operator is used because of to the fact that the CLB column is the basic coherent unit for the configuration. As an example, the 5-port SBAR occupies 852 slices. There are four slices per CLB, 80 CLBs per column, and 22 frames per column. This means that 'at least' $\lceil \frac{852}{4 \times 80} \rceil 22 = 66$ frames are required. Since a single frame requires

$$(16.5\ \mu s = (206\ \text{words} \times 32\ \text{bits}/8\text{- bit interface} \times 50\ \text{MHz}))$$

the configuration time is derived by $66 \times 16.5 = 1089\ \mu s$. The required bitstream size is derived by (number of frames) × (number of bits per frame). A single frame in Virtex-II Pro contains 6592 bits (206 words and each word is 32 bits wide [3]). Therefore the required bitstream size is at least 435 072 bits (=66 frames × 6592 bits). Fig. 9 depicts the lower bound of bitstream sizes required by the SBAR. As depicted in Fig. 9, the soft interconnects have significant configuration overheads in terms of the bitstream size and the configuration time. Moreover, these configuration overheads quadratically increases as the number of ports (linearly) increases. It is important to note that these overheads do not occur when these networks are 'hard'. By hardwiring networks, the reconfigurable resources can be fully utilised for the intra-IP functionality. Consequently, the on-chip resources are better utilised from

Table 2 Hardware implementation of NoCs

Type	Size	Area, slices	Clock freq., MHz
Soft (90 nm CMOS FPGA Virtex-II Pro)			
SCSN	2 × 3 mesh	3450	120
	3 × 4 mesh	9802	120
Hard (130 nm CMOS ASIC)			
		Area, mm^2	
HCSN	2 × 3 mesh	0.51	500
	3 × 4 mesh	1.21	500

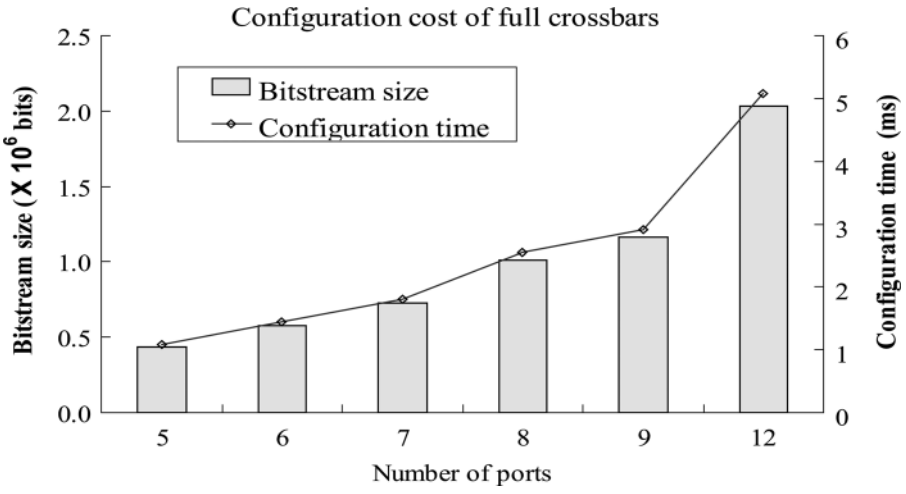

Fig. 9 *Lower bound of bitstream size and configuration time overheads of SBAR*

Table 3 Connection latencies of 2 × 3 2D-mesh NoCs

Type	Hard/soft	Min/Avg/Max	Latency, ns
HCSN	hard	AN	90.9
		Min_Sim	81.7
		Avg_Sim	93.7
		Max_Sim	109.2
CSN	soft	Avg_Sim	399.9

the functional perspective and configuration perspective, when the inter-IP networks are hard.

(3) *NoC simulation*: To verify the latency analysis in Section 5.1, we experiment with cycle-accurate SystemC simulation for the Æthereal NoC [28] and compare it with our latency model. Table 3 shows an average of connection latencies for the MJPEG task graph (in Fig. 5). The average of connection latency in our analysis is represented by *AN*. The minimum/average/maximum simulated connection latencies are obtained from the design flow [28]. 'Max_Sim' denotes the maximum experienced latency in the simulation. As shown in Table 3, our analysis provides the same trend as the simulation. Second, we compared hard and soft NoCs in the simulation. Table 3 also shows an average of connection latencies of hard and soft networks, by changing the clock frequency in the simulation. As a result, on average 4.2× of the latency is reduced in the hardwired network.

7 Conclusions

This article conducted a comparative analysis of hard and soft interconnects in FPGAs. We showed that routing wires in the reconfigurable fabric become increasingly critical resources, as the chip density increases. To compare the performance of hard and soft interconnects, the Jackson's queuing model has been utilised. Our simulation result showed that hardwired NoC provides significantly better network latency than the soft NoC. Hardwired NoC is promising for future FPGAs. To fill the gap between soft interconnects and hardwired NoCs, we also proposed to use crossbars as built-in network components. Consequently, our analysis and implementation results suggest that the hardwired interconnects significantly improve the functional performance, area cost, configuration time and configuration cost, compared to soft interconnects at an acceptable cost.

8 References

1 Kuon, I., Rose, J.: 'Measuring the gap between FPGAs and ASICs', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2007, **26**, (2), pp. 203–215

2 DeHon, A.: 'Reconfigurable architectures for general-purpose computing'. PhD thesis, Massachusetts Institute of Technology, September 1996

3 Xilinx, Inc., http://www.xilinx.com/

4 Goossens, K., Bennebroek, M., Hur, J.Y., Wahlah, M.A.: 'Hardwired networks on chip in FPGAs to unify functional and configuration interconnects'. IEEE Int. Symp. Networks-on-Chip (NOCS'08), April 2008, pp. 45–54

5 Gindin, R., Cidon, I., Keidar, I.: 'NoC-based FPGA: architecture and routing'. IEEE Int. Symp. Networks-on-Chip (NOCS'07), May 2007, pp. 253–264

6 Hecht, R., Kubisch, S., Herrholtz, A., Timmermann, D.: 'Dynamic reconfiguration with hardwired networks-on-chip on future FPGAs'. Int. Conf. Field Programmable Logic and Applications (FPL'05), August 2005, pp. 527–530

7 Goossens, K., Dielissen, J., Rădulescu, A.: 'The Æthereal network on chip: concepts, architectures, and implementations', *IEEE Des. Test Comput.*, 2005, **22**, (5), pp. 414–421

8 Jackson, J.: 'Networks of waiting lines', *Oper. Res.*, 1957, **5**, (4), pp. 518–521

9 Hur, J.Y., Goossens, K.G.W., Mhamdi, L.: 'Performance analysis of soft and hard single-hop and multi-hop circuit-switched interconnects for FPGAs'. IFIP/IEEE Int. Conf. Very Large Scale Integration (VLSI-SOC'08), October 2008, pp. 224–229

10 Kim, J., Park, D., Nicopoulos, C., Vijaykrishnan, N., Das, C.R.: 'Modeling and implementation of an output-queuing router for networks-on-chips'. Symp. Architecture for Networking and Communications Systems (ANCS'05), October 2005, pp. 173–182

11 Elmiligi, H., El-Kharashi, M.W., Gebali, F.: 'Modeling and implementation of an output-queuing router for networks-on-chips'. Int. Conf. Embedded Software and Systems (ICESS'07), May 2007, pp. 241–248

12 Hur, J.Y., Stefanov, T., Wong, S., Vassiliadis, S.: 'Customizing reconfigurable on-chip crossbar scheduler'. IEEE Int. Conf. Application-specific Systems, Architectures and Processors (ASAP'07), July 2007, pp. 210–215

13 Landman, B.S., Russo, R.L.: 'On a pin versus block relationship for partitions of logic graphs', *IEEE Trans. Comput.*, 1971, **C-20**, (12), pp. 1469–1479

14 Steiner, N., Athanas, P.: 'An alternate wire database for Xilinx FPGAs'. Proc. Field-Programmable Custom Computing Machines (FCCM'04), April 2004, pp. 336–337

15 Steiner, N.: 'A standalone wire database for routing and tracing in Xilinx Virtex, Virtex-E, and Virtex-II FPGAs'. Master thesis, Virginia Polytechnic Institute and State University, August 2002

16 Virtex-II Pro Handbook, Xilinx, Inc., 2002, http://www.xilinx.com

17 Raaijmakers, S., Wong, S.: 'Run-time partial reconfiguration for removal, placement and routing on the Virtex-II-Pro'. Int. Conf. Field Programmable Logic and Applications (FPL'07), August 2007, pp. 679–683

18 Passas, G., Katevenis, M., Pnevmatikatos, D.: 'A 128 × 128 × 20Gb/S crossbar, interconnecting 128 tiles in a single hop, and occupying less than 5% of their area'. ACM/IEEE Int. Symp. Networks-on-Chip (NOCS'10), May 2010

19 Nikolov, H., Stefanov, T., Deprettere, E.: 'Systematic and automated multi-processor system design, programming, and implementation', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2008, **27**, (3), pp. 542–555

20 AMBA AXI 3 Protocol Specification, ARM Ltd. http://www.arm.com/

21 Open Core Protocol (OCP) Specification http://www.ocpip.org/

22 Panades, I.M., Greiner, A.: 'Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures'. IEEE Int. Symp. Networks-on-Chip (NOCS'07), May 2007, pp. 83–94

23 Harris, A.J., Mathewson, B.J., Wrigley, C.E. (US 'Bus deadlock avoidance'. US Patent 7,219,178, ARM Ltd., 2007

24 Coenen, M., Murali, S., Rădulescu, A., Goossens, K., De Micheli, G.: 'A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control'. Int. Conf. HW/SW codesign and System Synthesis (CODES-ISSS'06), October 2006, pp. 130–135

25 Frost, V.S., Melamed, B.: 'Traffic modeling for telecommunications network', *IEEE Commun. Mag.*, 1994, **32**, (5), pp. 70–81

26 Baldwin, R.O., David, N.J., Midkiff, S.F., Kobza, J.E.: 'Queueing network analysis: concepts, terminology, and methods', *J. Syst. Softw.*, 2003, **66**, (2), pp. 99–117

27 Hansson, A., Goossens, K.: 'Trade-offs in the configuration of a network on chip for multiple use-cases'. IEEE Int. Symp. Networks-on-Chip (NOCS'07), pp. 233–242

28 Goossens, K., Dielissen, J., Gangwal, O.P., Pestana, S.G., Rădulescu, A., Rijpkema, E.: 'A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification'. Int. Conf. Design, Automation and Test in Europe (DATE'05), March 2005, pp. 1182–1187

29 Public repository for Frequently Asked Questions (FAQs) for designers of systems using FPGAs, http://www.fpga-faq.org

doi: 10.1049/iet-cdt.2011.0169