Performance Analysis of Soft and Hard Single-Hop and Multi-Hop Circuit-Switched Interconnects for FPGAs

Jae Young Hur[1], Kees Goossens[1,2], and Lotfi Mhamdi[1]
[1] Computer Engineering Laboratory, Delft University of Technology, The Netherlands
[2] Research, NXP Semiconductors, Eindhoven, The Netherlands
Emails: {J.Y.Hur, K.Goossens, L.Mhamdi}@ewi.tudelft.nl

***Abstract*— This article presents a performance analysis of hard and soft on-chip networks for FPGAs. We applied the Jackson's queuing model to analyze the performance of a multiprocessor system on a chip (MPSoC). We further used the Jackson's model to analyze circuit-switched networks on chip (NoC). Our simulation results showed the same trend as that of the analytical model. Considering streaming media applications and the Æthereal NoC, an analysis is conducted to compare hard and soft NoCs. The analysis and simulation indicate that the hardwired networks perform significantly better than conventional soft NoCs. Finally, we propose to hardwire crossbars in FPGAs to improve the performance of the inter-processor communication. An MJPEG case study suggests that hardwired crossbars achieve significantly better throughput compared to soft crossbars.**

I. Introduction

Field-programmable gate array (FPGA) is increasingly used to implement modern systems on a chip. Compared to the ASIC counterpart, however, FPGAs are slow in speed [1]. When the communication functionality is mapped onto reconfigurable (or *soft*) resources such as bit-level interconnects and look-up tables (LUTs), performance is degraded because of the long wires and the delay variation. Additionally, inter-IP communication is mostly coarse grained. Nevertheless, the fine-grained reconfigurability is a valuable asset to implement any IP functionality with the desired granularity. Due to these different requirements, inter-IP and intra-IP interconnects should be differently designed as discussed in [2]. It is well known that a crossbar performs high for small (or intermediate) sized networks and provides non-blocking communication. However, when these crossbars are implemented with reconfigurable resources in FPGA, the area is the bottleneck due to all-to-all interconnects inside the crossbar. This can be solved by the custom crossbar [3] which establishes necessary soft interconnects for a given application. However, the custom crossbar still utilizes reconfigurable bit-level interconnects, which are slow and occupy certain on-chip logic resources in FPGAs. Another problem of the soft interconnects is the inefficiency of the partial reconfiguration. Though bus macros can be utilized to geographically split different modules [4], it is difficult to split the computation and communication IPs because the communication IP is by nature distributed over the chip. To solve these problems, the networks-on-chip (NoC) can be directly implemented in (*hard*) silicon [2][5][6]. Though regular hardwired NoC (HWNoC) is promising for future FPGAs, design methods utilizing existing on-chip resources are not well defined. To bridge the gap between the state-of-the-art soft interconnects and the future HWNoC, we propose to hardwire crossbars in FPGAs.

Queuing analysis is one of the widely used modeling methods in telecommunication networks and provides a reasonable fit to the reality with relatively simple formulation [7]. However, the analysis of NoC-based multiprocessor systems on a chip (MPSoC) is challenging because traffic patterns in the MPSoC and telecommunication networks have different implications. First, unlike Internet traffic, in MPSoC, 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 communication and the computation are highly inter-related in MPSoC. Subsequently, it is desirable to derive the (worst-case) performance from the system or application perspective. To do this, we derive an approximated service time and utilize Jackson's open queuing model [8] for the comparative analysis. Our analysis intends to guide a system designer to determine network parameters at an early stage, by deriving a relative performance, for example with a topology exploration, an application mapping and a network dimensioning. The main contributions of this work are:

- Considering the Æthereal NoC [9] as an example, we apply the Jackson's queuing model [8] and derive the relative network performance to analyze (virtual) circuit-switched NoCs.
- Additionally, we present the effectiveness of the hardwired NoC in FPGAs. The simulation results indicate that hardwired NoC provides 4.2$\times$ better network latency for the MJPEG task graph, when compared to soft NoCs.
- We propose that crossbars are built in FPGAs for the inter-IP communications. In our MJPEG case study, the hardwired crossbar is 5$\times$ better in network throughput and 40% better in system throughput, compared to the soft crossbar.

This paper is organized as follows. In Section II, related work is reviewed. In Section III, hardwired crossbars are discussed. We present our performance analysis for hard and soft on-chip networks with a case study in Section IV. Experimental results are presented in Section V. Finally, conclusions are drawn in Section VI.

II. Related Work

Little has been reported regarding the queuing analysis of on-chip networks. In [10], a router with virtual channels is modeled. We present an analysis of a system as well as networks on chip, whereas only a single router is considered in [10]. 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. While the performance analysis in terms of latency and throughput is not presented in [11], we present the performance analysis from the system and network perspective. In [3], Jackson's model is applied to analyze soft single-hop crossbar networks. In this work, we present an analysis of an entire system as well as hard and soft multi-hop circuit-switched networks.

In [5][6], general approaches on the hardwired NOCs are discussed, where no architectural or implementation details are presented. Architectures and implementations of soft, firm, and hard Æthereal NoC [9] instances are compared in [2]. Additionally, unifying configuration and functional network is proposed in [2]. Though interfaces of the HWNoC with existing resources are discussed in [9], the design method needs to be significantly changed. In this work, we present hardwired crossbars and an analysis from the system perspective.

III. Hardwiring Crossbars in FPGAs

A soft shared bus is still widely used for system platforms such as FPGAs. However, a hardwired bus has following advantages. First, the system designer can use existing design method and existing IPs with minor modifications. The bus component is only needed to be instantiated as a hard macro. Second, the available bandwidth in the hardwired bus significantly increases because of the increased clock frequency. Accordingly, the contention probability of a network is reduced, which means that 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 network because the dedicated interconnects are physically established. Data transactions inside a crossbar can be fully parallel. Though an area cost is a bottleneck, the area of the crossbar can be adequate for small-sized, for example up to 16 ports. In order for many IPs to communicate via multiple hardwired crossbars, these small-sized crossbars are interconnected or arbitrated using soft bridges. Similar to shared buses, the existing design methods and IPs can be used without modifications. As described in the next sections, the hardwired crossbar (HFBAR) performs significantly better than soft crossbars. The HFBAR does not occupy any reconfigurable logic such as look-up tables. Moreover, bus macros are not necessary for the partial reconfiguration.

As depicted in Figure 1(1), hardwired crossbars can be interconnected using soft bridges. In this work, the transaction protocol in [12] is considered as an example, as depicted in Figure 1(2). Figure 1(3) and (4) depict a possible physical hard & soft interface for the Xilinx FPGA layout. We aim to bridge the gap between the current soft interconnects [3][4] and the future NoC-based hardwired networks [2]. Since all-to-all hardwired interconnects are established inside a crossbar, the wire utilization can be low. However, it can be noted that the reconfigurable logic such as LUTs is usually a bottleneck in modern FPGAs. Moreover, as presented in Section V, the area of the HFBAR is small enough for the utilization problem to be mitigated. Traffic congestion in the soft bridge between the crossbars can be also a bottleneck. However, many practical applications exhibit a traffic locality property [3]. The hardwired crossbars combined with soft bridges are beneficial especially when the traffic pattern has such a locality property. This is due to the fact that the hardwired crossbar can accommodate these localized traffic with reduced congestion.

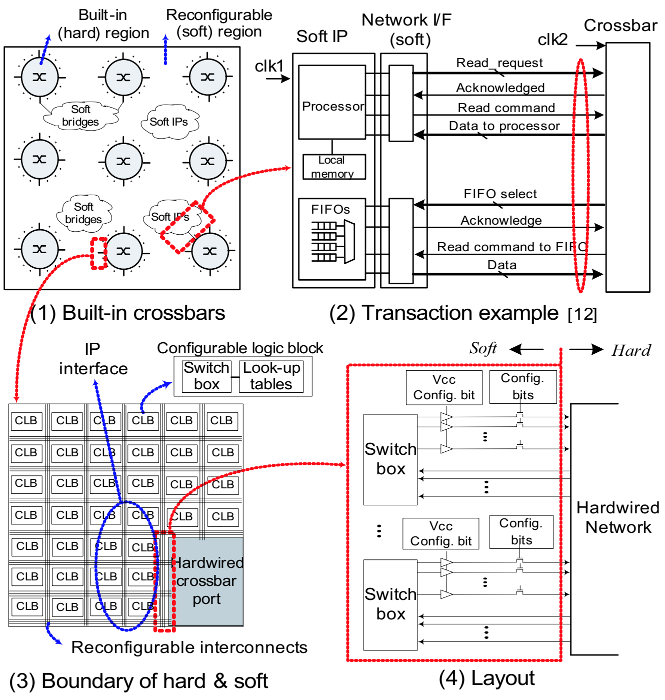


Fig. 1. Built-in crossbars and physical interface in FPGAs.

IV. Performance Analysis

In this section, we present the performance analysis of hard and soft on-chip networks with a case study using the Jackson's model. For Poisson-distributed incoming traffic, Jackson model can be generally used for a network of queues with arbitrary topologies [7]. For the analysis, we reuse specifications of network IPs in [3][9] and computation IPs in [4]. As a system model, we assume that the physical FIFOs are established for a logical connection to constitute a network of queues [3].

A. Jackson's 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 system 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:

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

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 i^{th} queuing system. μ_i is the service rate of the i^{th} queuing system. Jackson's model is also useful in that an average buffer size can be directly obtained from the formulation. $\frac{\lambda_i}{\mu_i-\lambda_i}$ corresponds to the buffer size of the i^{th} queuing system. Subsequently, $\sum_{i=1}^{N}\frac{\lambda_i}{\mu_i-\lambda_i}$ corresponds to the number of items in the entire system.

Figure 2 depicts our model for an MJPEG application. A task graph with 7 logical connections is depicted in Figure 2(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 Figure 2(1b). $N = 7$, i.e. there are 7 queuing systems (numbered 1-7). Figure 2(1c) depicts individual queuing systems for each logical connection. The queuing system consists of the waiting queue and the server. A server consists of the network component that provides transportation service and the computation component that provides a data processing service. To compute λ_i in Equation (1), we utilize the bandwidth information in Figure 2(1a). As an example, λ_1 is computed by $\frac{62}{62+0.6+1+0.6}\lambda = 0.97\lambda$. For a given λ, the system response time is determined from μ_i. Figure 2(2) and (3) depict the traffic mapping onto different networks such as crossbars or NoCs. Given the computation IPs, the service rate μ_i varies with different network components. As depicted in Figure 2(2), a connection consists of two logical channels, a *request* and a *response* channel.

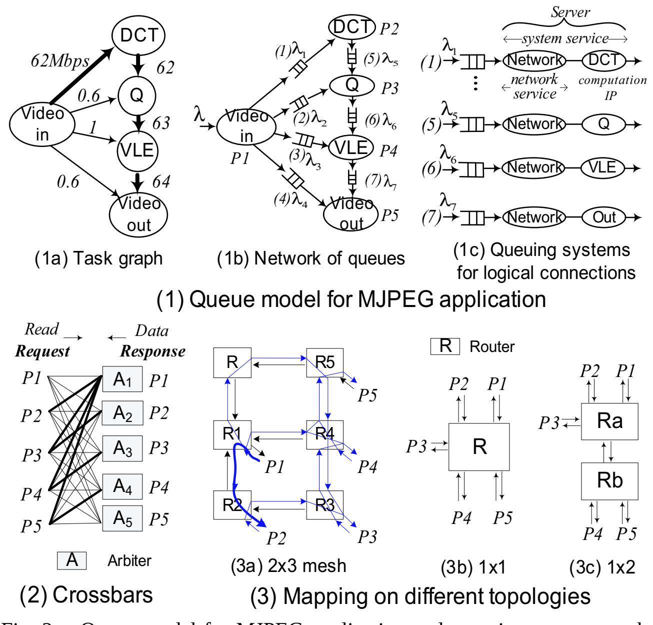


Fig. 2. Queue model for MJPEG application and mapping onto networks.

The system performance varies with specific computation and computation patterns. Our aim is to derive a relative performance and determine network parameters by a topology exploration and an application mapping. The network parameters (for example, buffer sizes) are usually dimensioned at design time for the worst case. Therefore, we consider the worst case scenario and assume that a computation IP is sequentially operated. As an example, a master IP requests data block to a slave IP and waits until an entire data block arrives. Then the master IP sequentially processes the data block. Accordingly, communication transactions are not pipelined. While the transactions in the entire system are concurrent, an individual network queuing server serves an entire transaction at a time. In this way, the response time is derived in a conservative manner.

B. System and network service rates

We formulate two types of service rates, namely the *system service rate* μ_{system} that includes the computation time and the *network service rate* μ_{token} that does not include the computation time (see Figure 2(1c)).

1) System service rate: The worst-case system service rate μ_{system} denotes the back-to-back service rate for logical channels and is defined as:

$$\mu_{system} = (T_{block} + T_{compute})^{-1} \tag{2}$$

where T_{block} denotes the network service time for the delivery of a data block. $T_{compute}$ denotes the computation time in the server for the data block.

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

$$\mu_{token} = T_{token}^{-1} = (T_{arbit} + T_{transmit})^{-1} \tag{3}$$

where T_{token} is the network service time[1] 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. T_{arbit} denotes the arbitration time. $T_{transmit}$ is the actual data transmission time.

3) Network and system performance: The network (or system) response time $T_{response}$ can be derived by substituting the μ_{token} (or μ_{system}) in Equation (1). It can be noted that the computation IP often requires block of data in order to operate. As an example, MJPEG application operates in image blocks with 8×8 pixels and we consider that the token size is 3 words in this work. Subsequently, T_{block} in Equation (2) can be a multiple of T_{token} in Equation (3). This means that μ_{system} is less than μ_{token}.

C. Crossbars

We consider the full crossbar (FBAR) with all-to-all interconnects and the custom crossbar (CBAR) with on-demand interconnects. The on-demand interconnects for the MJPEG application are represented in bold lines in Figure 2(2). For these single-hop crossbars, we use the formulation in [3] summarized as follows. The arbitration time for a full

[1]Generally, data transmission through the NoC is pipelined resulting in shorter delays. Therefore Equation (3) is the worst-case for the NoC.

crossbar T_{arbit_full} and a custom crossbar T_{arbit_custom} are approximated as:

$$T_{arbit\_full} \approx (\lfloor \frac{\text{\#ports}}{2} \rfloor + C_{hand})/(f_{net}) \tag{4a}$$

$$T_{arbit\_custom} \approx (\lfloor \frac{\text{\#links}}{2} \rfloor + C_{hand})/(f_{net}), \tag{4b}$$

where a request check latency is approximated as $\lfloor \frac{\text{\#ports}}{2} \rfloor$ or $\lfloor \frac{\text{\#links}}{2} \rfloor$ cycles. C_{hand} refers to the handshaking latency in number of cycles. The arbitration time T_{arbit} in our crossbar varies with number of ports #ports or logical channels #links. The transmission time $T_{transmit}$ in Equation (3) corresponds to the token size, as derived by following:

$$T_{transmit} = \frac{S_{token}}{f_{net}}, \tag{5}$$

where S_{token} denotes the token size or the number of words. f_{net} refers to the clock frequency of a network, which is equivalent to the word rate.

D. MJPEG case study for hardwired crossbars

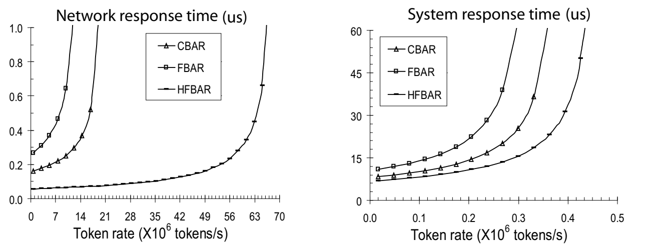
We derive the crossbar network and system performance for the MJPEG specification depicted in Figure 2(1a). We consider a hardwired full crossbar (HFBAR) for the token size S_{token}=3 words and the number of ports #ports=8 ports. The handshaking latency C_{hand} is 2 cycles and the clock frequency f_{net} is 446 MHz from the implementation (see Section V).

1) Network service rate: The network service rate μ_{token} in the HFBAR is derived as follows. Since C_{hand} = 2 cycles and f_{net} = 446MHz, T_{arbit_full} is derived by $(\lfloor \frac{8}{2} \rfloor + 2)/(446 \times 10^6)$ for Equation (4a). Since S_{token} = 3 words, the transmission time $T_{transmit}$ is derived by $\frac{3}{446 \times 10^6}$ seconds. The μ_{token} is derived by substituting T_{arbit_full} and $T_{transmit}$ in Equation (3). Subsequently, $\mu_{token} = \frac{446 \times 10^6}{\lfloor \frac{8}{2} \rfloor + 2 + 3} = 49 \times 10^6$ tokens/s. This means that the HFBAR provides a physical network bandwidth of 49×10^6 tokens/s for a logical connection.

2) Network performance: The network response time is derived by substituting the network service rate μ_{token} in Equation (1). Figure 3(1) depicts the network performance for soft and hard crossbars. The network performance for soft crossbars is derived in [3]. As a result, the throughput of HFBAR is 5× better than the soft full crossbar (FBAR) and 3× better than the custom crossbar (CBAR). This is mainly because of the higher clock frequency.

3) System service rate: We derive the system service rate μ_{system} that includes the computation time. In MJPEG, DCT is the most time-consuming task, in which 6 cycles are required for each pixel [4]. The Quantization (Q) task requires 74 cycles per image block [4]. The computation time of variable-length encoding (VLE) task is determined only in run time, which means that the service rate is not predictable at design time. In this work, we obtained the approximate computation time by profiling the application. Subsequently, the VLE task requires 40% of the computation time of DCT. The system service rate for each logical connection is derived by substituting these computation time in Equation (2).

4) System performance: The system response time is derived by substituting the system service rate μ_{system} in Equation (1). As a result, Figure 3(2) depicts the system response time for a single image block. The general trend is similar to Figure 3(1), while the performance gap is much reduced. This is because of the intensive computation load. As depicted in Figure 3(2), the throughput of the HFBAR is 40% better than the FBAR and 20% better than the CBAR.



(1) Network response time for single token (token size = 3 words)

(2) System response time for single image block (block size = 64 words)

Fig. 3. Crossbar network and system performance for MJPEG.

E. Applying Jackson's model to circuit-switched networks

In this section, we conduct a performance analysis for the circuit-switched networks (CSN). We consider GT (guaranteed throughput) Æthereal NoC [9] 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 single-hop crossbar. This means that the worst-case performance of the CSN can be analyzed 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 single-hop crossbar with a physically pipelined transmission. We assume connections are long-lived, and ignore the time associated with their set up and tear down [14].

1) Arbitration time: In Equation (3), the arbitration time is derived as follows. 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:

$$T_{arbit} = \frac{S_{slot} \times \lceil \frac{S_{tab}}{2 \times A_{slot}} \rceil}{f_{net}}, \tag{6}$$

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 3 words (in the worst case, 1 header and 2 payload words). Similar to Equation (4), 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 Figure 2(2), 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:

$$T_{transmit} = \begin{cases} \dfrac{\lceil \frac{S_{req}}{S_{slot-1}} \times \frac{S_{tab}}{A_{slot}} \rceil + \#\text{hop} \times C_{SW} + S_{req}}{f_{net}} & \text{for } \textit{request} \\ \dfrac{\lceil \frac{S_{resp}}{S_{slot-1}} \times \frac{S_{tab}}{A_{slot}} \rceil + \#\text{hop} \times C_{SW} + S_{resp}}{f_{net}} & \text{for } \textit{response}, \end{cases} \tag{7}$$

where $(S_{slot} - 1)$ refers to the slot size in number of payload words, while 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 $\lceil \frac{S_{req}}{S_{slot-1}} \times \frac{S_{tab}}{A_{slot-1}} \rceil$ refers to the number of cycles to send data. This is an approximation because it assumes slots are equally spaced. However, otherwise the first term needs to be split in a *div* term for the number of table revolutions, and a *mod* term for the delay in the last revolution. 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 S_{req} or S_{resp} refers to the length of message.

F. MJPEG case study for hardwired circuit-switched networks

We derive the performance of the hardwired circuit-switched network (HCSN). The MJPEG task graph is mapped onto different topologies as depicted in Figure 2(3). Consider the HCSN with 2×3 mesh topology depicted in Figure 2(3a). The slot size S_{slot} is 3 words. The clock frequency of the hardwired network $f_{net} = 500$ MHz from the implementation (see Section V). The switching latency per router hop C_{SW} is 3 cycles from the implementation. The token size for the request channel S_{req} and the token size for the response channels S_{resp} are 3 words. The slot table size S_{tab} and number of the reserved slots per channel A_{slot} are derived from the bandwidth distribution. In this work, we used the automated design flow in [15] to obtain S_{tab} and A_{slot} for an MJPEG task graph. The arbitration time and transmission time are derived as follows:

1) Arbitration time: Using the tools of [15], S_{tab} is 4 slots and A_{slot} is 1 slot per channel. The arbitration time is derived by substituting the S_{slot}=3 words, S_{tab}=4 slots, and A_{slot}=1 slot in Equation (6). Subsequently, the arbitration time T_{arbit} is derived by $\frac{3 \times \lceil \frac{4}{2\times 1} \rceil}{500 \times 10^6} = 12ns$ per channel.

2) Transmission time: Since S_{slot}=3 words and S_{req}=S_{resp}=3 words, the time to send data is $\frac{\lceil \frac{3}{3-1} \times \frac{4}{1} \rceil}{500 \times 10^6} = 12ns$. From the topology mapping and the routing strategy, the number of hops #hop is obtained for each channel. The routing paths are depicted in Figure 2(3a) for the response channels. As an example, #hop between $P1$ and $P2$ is 2 (see bold line in Figure 2(3a)). Since C_{SW}=3 cycles, $T_{transmit}$ is derived by $\frac{\lceil \frac{3}{3-1} \times \frac{4}{1} \rceil + 2\times 3 + 3}{500 \times 10^6} = 30ns$ for a channel between $P1$ and $P2$.

3) Network performance: The service rate can be derived by substituting T_{arbit} and $T_{transmit}$ in Equation (3). As an example, the service rate μ_1 for the connection $P1$ and $P2$ is derived by $\frac{1}{2\times(12+30)ns} = 12 \times 10^6$ tokens/s. Note that 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 Equation (1). Similarly, the performance of hard and soft networks with different topologies are derived, as depicted in Figure 4(1). The 11CSN in Figure 2(3b) performs relatively better than other topologies due to the single-hop communications. In addition, the hardwired NoC is significantly better in latency and throughput than the soft NoC.

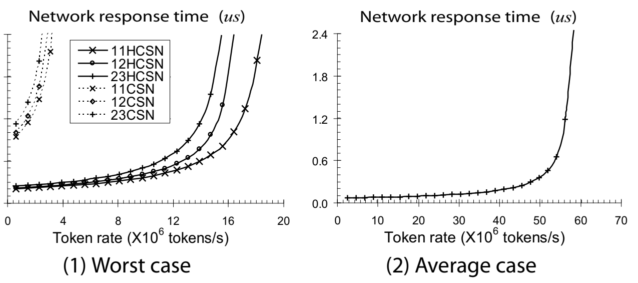


Fig. 4. The worst-case (1) and the average-case (2) network performance for MJPEG task graph in Figure 2. Token size is 3 words. 23(H)CSN denotes a soft (hardwired) circuit-switched network with 2×3 mesh topology.

G. Average case analysis for circuit-switched networks

The service rate in the previous section is derived for the worst case, 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. An average service rate for a logical channel in the HCSN μ_{HCSN} (tokens/s) is represented by:

$$\mu_{HCSN} = \frac{A_{slot}}{S_{tab}} \times \frac{f_{net}}{S_{token}}, \tag{8}$$

where A_{slot}, S_{tab}, S_{token} and f_{net} are defined in the previous section. First term $\frac{A_{slot}}{S_{tab}}$ denotes how much bandwidth is allocated for a channel. Second term $\frac{f_{net}}{S_{token}}$ denotes the maximum token rate. Similar to the previous section, we conduct an MJPEG case study, as depicted in Figure 4(2). Compared to the worst-case scenario, the network performs 3.3× better in throughput at the average case.

In our MJPEG case study, the performance of the HFBAR is comparable to the average case performance of the HCSN, from Figure 3(1) and Figure 4(2). In general, on-chip networks provide the maximum bandwidth of $\frac{f_{net}}{S_{token}}$ per link at the best case. In our case study, the maximum bandwidth of the HFBAR is therefore $\frac{446\times 10^6}{3} = 149 \times 10^6$ tokens/s and the maximum bandwidths of the HCSN is $\frac{500\times 10^6}{3} = 167 \times 10^6$ tokens/s per link. Note that we aim to explore the different topologies, we consider the worst case latency model for the analysis as well as the experiments in the next section.

V. Experiments and Results

In this section, we present the simulation and the hardware implementation results. We conduct three experiments. First, to verify the analysis, we experiment with cycle-accurate SystemC simulation for the Æthereal NoC [15] and compare it with our worst-case latency model. Figure 5(1) depicts an average of connection latencies for the MJPEG task graph (in Figure 2). 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 [15]. Max_Sim denotes the maximum experienced latency in the simulation. As depicted in Figure 5(1), our analysis provides the same trend as the simulation. Second, we compared hard and soft NoCs in the simulation. Figure 5(2) depicts 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.

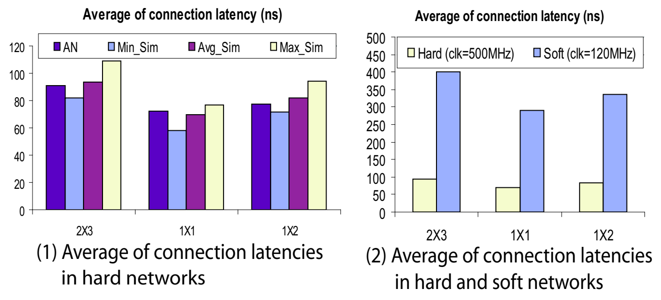


Fig. 5. Simulation results for MJPEG task graph in Figure 2.

To evaluate the area cost and the clock frequency, the networks are synthesized in FPGAs for soft networks and in ASICs for hardwired networks, as shown in Table I. For soft networks, we used Xilinx ISE tool to synthesize, place and route in Virtex-4 XC4VLX200 device with 90nm CMOS technology. For hardwired networks, we used Cadence Encounter tool to place and route in ASIC with 130nm CMOS technology. As a result, the clock frequency of hardwired networks is $4.7\times$ 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 HFBAR is $0.29mm^2$. Considering the large area (=$735mm^2$) in our target FPGA device[2], the area overhead of the hardwired network is small. For comparison, the soft full crossbar FBAR with the same design occupies more than 8% of the available logic slices in the targeted device. This means that the soft network occupies non-negligible logic resources.

VI. Conclusions

This article conducted a performance analysis of NoC-based systems, using the Jackson's queuing model. We also applied the Jackson's model to analyze circuit-switched networks. Our simulation results showed the same trend as that of the derived analytical model. Hardwired NoC is promising for future FPGAs. As our analysis and simulation results indicate, the hardwired network is significantly better in speed, throughput and area. To fill the gap between soft interconnects and hardwired NoCs, we also proposed to use crossbars as built-in network components in FPGAs. Our analysis and implementation results suggest that the hardwired crossbars significantly improve the performance compared to soft interconnects at an acceptable cost.

[2]Estimated die size from `http://www.fpga-faq.org/`

TABLE I

Hardware implementation results

Soft (90nm CMOS FPGA Virtex-4, XC4VLX200)			
Type	Size	Area (slices)	Clock Freq. (MHz)
FBAR	8 ports	2048	97
	12 ports	4182	75
CSN	2×3 mesh	3450	120
	3×4 mesh	9802	120
CBAR	MJPEG (Fig. 2(1a)	284	101
Hard (130 nm CMOS ASIC)			
Type	Size	Area (mm^2)	Clock Freq. (MHz)
HFBAR	8 ports	0.11	446
	12 ports	0.29	410
HCSN	2×3 mesh	0.51	500
	3×4 mesh	1.21	500

References

[1] I. Kuon and J. Rose. "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[2] K. Goossens, M. Bennebroek, J. Y. Hur, and M. A. Wahlah, "Hardwired Networks on Chip in FPGAs to Unify Functional and Configuration Interconnects," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'08)*, pp. 45–54, Apr. 2008.

[3] J. Y. Hur, T. Stefanov, S. Wong, and S. Vassiliadis, "Customizing Reconfigurable On-Chip Crossbar Scheduler," *IEEE Int'l Conference on Application-specific Systems, Architectures and Processors (ASAP'07)*, pp. 210–215, Jul. 2007.

[4] Xilinx, Inc., http://www.xilinx.com/.

[5] R. Gindin, I. Cidon, and I. Keidar. "NoC-Based FPGA: Architecture and Routing," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'07)*, pp 253–264, May 2007.

[6] R. Hecht, S. Kubisch, A. Herrholtz, and D. Timmermann. "Dynamic Reconfiguration with hardwired Networks-on-Chip on future FPGAs," *Int'l Conference on Field Programmable Logic and Applications (FPL'05)*, pp 527–530, Aug. 2005.

[7] R. O. Baldwin, N. J. David IV, S. F. Midkiff, and J.E. Kobza, "Queueing network analysis: concepts, terminology, and methods," *Journal of Systems and Software*, vol. 66, no. 2, pp. 99–117, May 2003.

[8] J. Jackson, "Networks of waiting lines," *Operations Research*, vol. 5, no. 4, pp. 518–521, Aug. 1957.

[9] K. Goossens, J. Dielissen, and A. Rădulescu. "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sept-Oct 2005.

[10] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. R. Das, "Modeling and Implementation of an Output-Queuing Router for Networks-on-Chips," *Symposium on Architecture for Networking and Communications Systems (ANCS'05)*, pp. 173–182, Oct. 2005.

[11] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Modeling and Implementation of an Output-Queuing Router for Networks-on-Chips," *Int'l Conference on Embedded Software and Systems (ICESS'07)*, pp. 241–248, May 2007.

[12] H. Nikolov, T. Stefanov, and E. Deprettere, "Multi-processor System Design with ESPAM," *Int'l Conference on HW/SW Codesign and System Synthesis (CODES-ISSS'06)*, pp. 211-216, Oct. 2006.

[13] M. Coenen, S. Murali, A. Rădulescu, K. Goossens, and G. De Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," *Int'l Conference on HW/SW Codesign and System Synthesis (CODES-ISSS'06)*, pp. 130–135, Oct. 2006.

[14] A. Hansson and K. Goossens, "Trade-offs in the configuration of a network on chip for multiple use-cases," *IEEE Int'l Symposium on Networks-on-Chip (NOCS'07)*, pp 233–242, May 2007.

[15] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Rădulescu, and E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," *Int'l Conference on Design, Automation and Test in Europe (DATE'05)*, pp. 1182–1187, Mar. 2005.