The Benefits of Using Clock Gating in the Design of Networks-on-Chip

Michele Petracca, Luca P. Carloni
Dept. of Computer Science, Columbia University, New York, NY 10027

***Abstract*—Networks-on-chip (NoC) are critical to the design of complex multi-core system-on-chip (SoC) architectures. Since SoCs are characterized by a combination of high performance requirements and stringent energy constraints, NoCs must be realized with low-power design techniques. Since the use of semi-custom design flow based on standard-cell technology libraries is essential to cope with the SoC design complexity challenges under tight time-to-market constraints, NoC must be implemented using logic synthesis. In this paper we analyze the major power reduction that clock gating can deliver when applied to the synthesis of a NoC in the context of a semi-custom automated design flow.**

I. Introduction

Multi-core systems-on-chip (SoC) are an emerging computing platform for several classes of embedded applications. These systems are realized by assembling heterogeneous components from different sources under tight time-to-market constraints and, typically, strict power dissipation requirements. The design of the entire chip through the composition of many heterogeneous cores creates portability, compatibility, and reusability issues. Cores can be designed by different design teams, but they still need to interact efficiently. Moreover, the design of a core is an expensive task, so its often desirable to reuse the same core into different contexts and products. Networks-on-chip [1], [2], [3] play a twofold critical role for multicore SoCs: at run time they provide essential inter-core communication and synchronization services while at design time they are key to coping with the challenges of component integration and the complexity of system-level design and validation.

As an important part of the design the NoC requires a certain amount of resources. The links that transfer packets from router to router are implemented with wires. The routers need logic to determine the path from source to destination and they also need storage units to store packets in case of congestion. The NoC takes a portion of the die area and, more importantly, consumes power to move data among cores. A good NoC should provide enough bandwidth and low latency to the communications to avoid becoming the system bottleneck, but, at the same time, should minimize its impact on the limited power budget of the chip. Obviously, the higher the portion of the power budget dedicated to the NoC for forwarding data, the lower is the portion that can be assigned to the cores for actual computation.

The NoC has then a key role in the design of multi-core systems: it should have *static* properties, such as flexibility and modularity, for easily interfacing the cores, and it should also have *dynamic* properties, such as performance and power efficiency. In the design of Systems-on-Chip (SoC) the two classes of properties are particularly important. A SoC is generally composed by assembling very heterogeneous cores, often in the form of Intellectual Property (IP), that the designers have to interface under very strict time-to-market constraints. Moreover, SoC are increasingly used in portable applications, where the power budget guaranteed by batteries is very limited [4], [5].

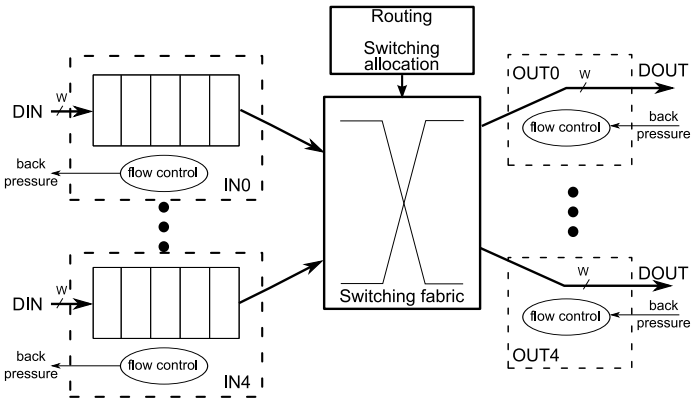

Fig. 1. Reference 5×5 input-queue wormhole NoC router.

Consider the case of an SoC device for next generation mobile phones. Such device can host very different applications. The traffic patterns are somehow unpredictable because they depend on the set of applications the final user decides to run. The traffic patterns can also depend on the features installed on the Operating System (OS), or on the version of the OS. Periods of high traffic activity can be followed by periods of very low traffic activity. Also data burstiness is unpredictable in the general case. An efficient NoC should adapt to the time-varying traffic patterns, without wasting power when the network is not forwarding any valid data. The NoC should adapt to the traffic also spatially, reducing power dissipation in the portion of network that are unused. On the other hand, the NoC should provide full performance when requested by a given application.

Contributions. In this paper we consider, analyze and quantify the effects of clock-gating on the semi-custom design of an NoC router. Clock-gating is a well-known low-power design technique. We analyze a typical NoC router architecture and we qualitatively assess that clock-gating delivers huge power saving with no overhead because:

- An NoC router has a high ratio of sequential elements over combinatorial elements;
- The router queues architecture is such that clock gating can be applied with virtually no overhead;
- The combination of flow-control over the NoC and the FIFO architecture strongly reduces the activity factor of the input data of all the sequential elements implementing a FIFO.

Then, by means of commercial design tools, we apply clock-gating without any modification to the router RTL design. We measure the power, under different injection rates and congestion probability. Finally, we perform a case study, where we measure the power gain deriving from the use of clock gating on the layout of a router employed in a 4×4 NoC, under uniform traffic.

Clock gating applied to the semi-custom design of NoC routers delivers a power reduction of about 70%.

II. Related Work

Clock gating is a design technique aimed at reducing power dissipation of the sequential elements composing a circuit. Clock gating is also often associated with stallability: the state of a circuit can be forced not to evolve by simply disabling the clock. Several works have used clock gating to introduce stallability in synchronous pipelines [6], [7]. Other works used it on pipelined communication channels [8]. Clock-gating has already been applied extensively in NoC design, e.g. in the Teraflops processor [9] and the Tile processor [10]. Still, to the best of our knowledge, the present work is the first complete study that analyses why clock gating can lead to dramatic power reduction in the semi-custom design of NoCs and that quantifies this impact using experimental results obtained with an industrial standard-cell semi-custom design flow.

III. Background on NoC Router Architecture

An NoC is an interconnection network that provides communication services among the cores of an SoC. An NoC is composed by routers and links, interconnected according to a desired topology. Data flow through links on a router-to-router basis, from source to destination. The routers determine the source-to-destination path according to the routing algorithm. Since a packet-switched NoC is a shared medium, then data from different sources can compete inside the routers as they request the same output link (em congestion).

The routers implement congestion management, targeting an high utilization of the resources. The router *queues* store data during congestion, until the link becomes available. However, queues have limited depth, thus they fill up if the congestion lasts too long. Deeper queues can successfully "absorb" longer periods of congestion, but in a NoC environment queues are short, because of the limited available area and power budget.

In order to avoid the loss of data during congestion, *flow control* is adopted. When the buffers are filling up, the router triggers a back-pressure along the uplink routers back to the source. The back-pressure temporarily stops the data generation, preventing the queues from overflowing.

Fig. 1 shows a typical architecture of input queue router for NoCs. Beside input queues and flow control, a router is equipped with: *(i)* a *switching fabric* to transfer data from inputs to outputs, *(ii)* a *routing* engine that determines the output port for each packet, and *(iii)* a *switch arbitration* engine that generates input-output matchings as function of the traffic. We take this architecture as the *reference design*. We consider a router with 5 I/O ports because this is used in common 2-D Mesh NoCs, but the results that we present hold for any number of ports.

A. Input buffers

Many router architectures have been proposed, with different organization of the data buffering. Buffers are generally organized in FIFO queues, placed at the outputs and/or at the inputs of the router. NoC routers commonly adopt input queues, because simpler to implement.

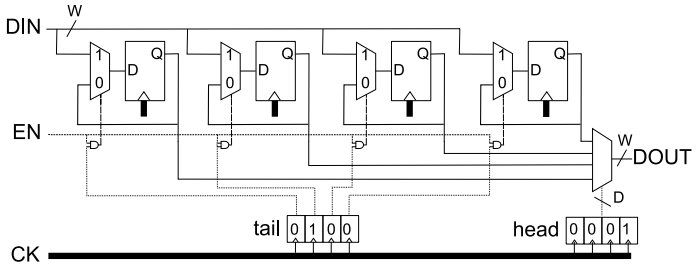

Fig. 2. FF-RF implementation of FIFO queue. The TAIL pointer lets only one location to be overwritten with the new incoming data DIN (if EN=1). The HEAD pointer assigns to DOUT the oldest stored data. The clock CK is directly connected to all FFs. HEAD and TAIL are two double-direction shift registers.

There are different ways to implement a FIFO queue of depth D and width W. One possibility is to use of a double-port $W \times D$ SRAM Register File (RF) (SRAM-RF) with head and tail pointer. The head pointer indicates the location of the oldest data token, while the tail indicates the first available location for storing a new data token.

An alternative implementation of a FIFO queue is based on a set of W shift-registers (FF-SHIFT) [11]. Such implementation does not require any head or tail pointer, but it simply slides all the data tokens towards the front of the queue every time a new data token is read. Each incoming flit is stored in the first free location, right on the back of the last flits already stored in the queue. A shift-register requires the use of flip-flops (FFs).

The SRAM is an efficient way of implementing only large RFs, while queues in NoC routers are generally short. Moreover, the SRAM is technology dependent and it introduces layout challenges in a semi-custom design flow. Instead, the use of FFs is highly technology independent, thus portable. However, FF-SHIFT is rather inefficient in terms of power because it shifts all the stored data at every read operation.

Based on the pros and cons of the two implementations above, we believe that an efficient and portable FIFO implementation relies on the use of FFs for realizing the register file (FF-RF), as shown in Fig. 2. Head and tail pointers are still used, but each of the D RF entries is an array of W FFs.

B. Flow Control

Flow-control protocols can be classified according to the channel bandwidth and buffer-allocation granularity [12]. The basic unit of bandwidth and storage allocation is a *flit (flow control digit)* and packets are decomposed in sequences of flits. Generally the first flit of a packet, called *header* carries all the routing information relative to the packet. The following flits instead do not carry any routing nor sequencing information. A flit-buffer flow control allocates both bandwidth and buffers in units of flits. This has three advantages: it *(i)* reduces the storage required for correct operation of a router, *(ii)* provides stiffer back-pressure from a point of congestion back to the source of a flit stream, and *(iii)* enables more efficient use of storage. Since these advantages match well the characteristics of on-chip communication, flit-buffer flow control methods are seen as a promising solution for NoC, where typically the size of a flit matches the parallelism of a channel.

The two main flit-buffer protocols are *wormhole* and *virtual-*

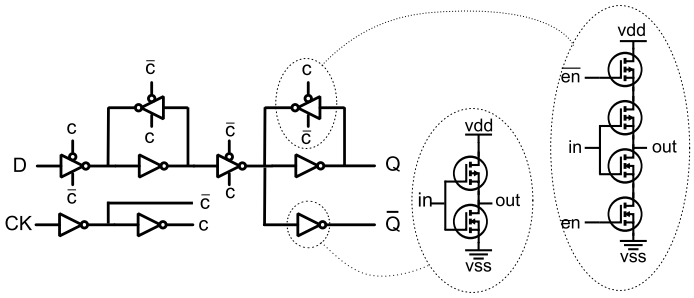

Fig. 3. Typical FF implementation

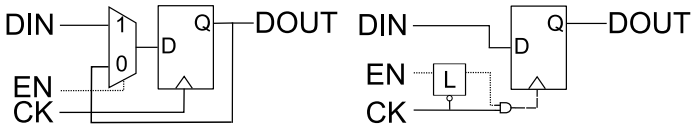

Fig. 4. Typical EFF implementations. EN=1 enables to overwrite DOUT with DIN. (Left) Without clock-gating. (Right) With clock gating - the latch L filters glitches on the gated clock

channel flow controls. These need to be supported by a buffer-management flow control that allows the downlink router to communicate the buffer availability to the uplink router and operate back-pressure when necessary. In NoC applications, one of the most used buffer-management flow control is *credit-based*. In our reference design we assume the use of *wormhole* and *credit-based*. In any case, our results mainly relate to the queues' architecture, and therefore they *can be applied to any flow control that adopts input queues and back-pressure in case of congestion.*

IV. Background on clock gating

Clock gating is a low-power design technique that masks the clock activity to those FFs whose stored data needs to be kept from one clock cycle to the following. Fig. 3 shows a basic transistor-level implementation of a FF. Such FF is composed by 26 transistors: 12 (46%) transistors are driven by signals that change accordingly to the clock (*CK*, *c*, $\bar{c}$), while the remaining 14 transistors are driven by a signal that changes according to the input data (*D*). Signal *CK* switches twice per clock cycle, while *D* switches at most once per clock cycle. Furthermore, if we assume a uniform distribution of the possible input values, on average *D* switches at most once every 2 clock cycles. In conclusion, *D* has a switching activity at least $4\times$ lower than *CK*. The clock is then responsible for roughly 80% of the power dissipated in a FF.

Fig. 4 shows two possible implementations of an FF with an enable signal (*EFF*): *(Left)* an EFF that *does not* adopt clock-gating and *(Right)* an EFF that *does* adopt clock-gating. An EFF is a FF that samples the new DIN when *EN*=1, but keeps the current data for the next clock cycle when *EN*=0. When *EN*=1, the clock rising edge schedules the sampling of DIN. On the contrary, when *EN*=0, DIN is ignored and DOUT does not change. In an implementation without clock gating the clock edge is still used to re-sample DOUT. In a clock-gated implementation, instead, the clock is masked, thus no sampling is performed.

In the implementation of an EFF *without* clock-gating, when *EN*=0 only the transistors connected to the clock signal switch

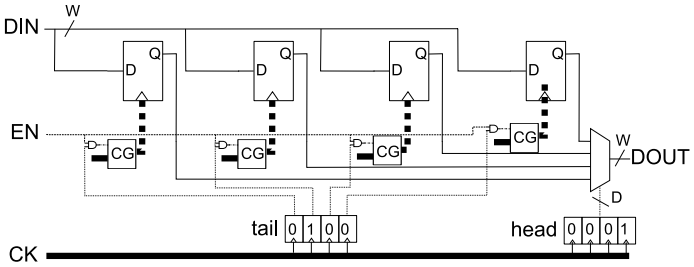

Fig. 5. Application of clock-gating to a FF-RF FIFO. CG is the block implementing clock-gating (Latch + AND Gate).

(46% of all FF's transistors), because there is no activity on the data. Note that if DIN changes in a clock cycle with *EN*=0, it is still responsible for changing only the state of the gates into the multiplexer, which we conservatively neglect in our model. On the other hand, in the implementation of an EFF *with* clock-gating, when *EN*=0 the transistors driven by the clock signal do not switch. Any change on DIN affects only the state of 4 transistors in the first two buffers in the FF (Fig. 3), which are only 15% of the total number of transistors. Summarizing, during a disabled cycle, clock gating reduces the switching power at least by a factor of $3\times$. During an enabled cycle, instead, data and clock wires switch without masks in both implementations, and therefore the power consumptions are comparable.

Generalizing, clock gating (i) has no impact on the combinatorial logic, (ii) has no impact on the static power dissipation due to current leakage, and (iii) introduces overhead in terms of area and clock tree if applied at the granularity of a single FF. Therefore, clock gating is a design technique with high potential only when (a) the number of EFFs in the circuit is high, and (b) their *enabling rate* ($Probability(EN = 1)$) is expected to be low.

V. The effect of clock-gating on an NoC router

The amount of the router logic that is required to implement routing and switch arbitration is negligible. The two main components in terms of number of gates are the switching fabric and the input queues. The switching fabric is essentially made of combinational logic. In a semi-custom flow, for each of the W bits of the flit width, the crossbar is generally implemented as a set of p ($p \times 1$)-multiplexers, where p is the number of router ports. Instead, the input queues are mainly composed by EFFs and multiplexers, as showed in Fig. 2.

As discussed in Section IV, the probability of an EFF to have *EN*=0 determines the power gain delivered by clock-gating. In some classes of circuits, such as highly pipelined systems with a throughput close to one, there is no need for enabling or disabling FFs because they sample new data at every clock cycle. For such circuits the application of clock gating to FFs is not necessary. Clock gating can however be adopted on the entire circuit for preserving a state or for temporarily turning-off the computation.

In NoC input queues, instead, we often have the need to disable a location. As discussed in Section III-A, an input queue has by construction a depth D and at most one flit per clock cycle can be received and stored. This flit is written in the location pointed by the tail and the value in every other location does not change. Therefore, the enabling rate

is f_{ck}/D, where f_{ck} is the clock frequency. Note that a read operation consists of the selection of one stored data by the output multiplexer, but no data in the location pointed by the head is overwritten: such location is overwritten only when it is pointed as tail and a new flit arrives.

We just analyzed how the enabling rate of each FF composing a FIFO queue is *at the most* f_{ck}/D. This value is derived assuming a new flit is received by the queue at every clock cycle, but in an NoC this is almost never the case. Links are often used below their maximum line-rate, due to low traffic activity. The same links can also be fully used in other periods of time, but this event generally causes congestion, that propagates over the network as back-pressure, with consequent reduction of the link utilization in other areas of the network. According to topology, network size, traffic patterns, and router architecture, the maximum average link utilization can be as low as 30% due to congestion [13]. In that case the FF enabling rate would be lower than $f_{ck}/12$.

In Fig. 5 we show the circuit resulting from the application of clock-gating to the FF-RS FIFO of Fig. 2. In the next section we show experimental results that quantify the power gain delivered by clock-gating.

In summary, NoC routers are a class of digital circuits that can highly benefit from the use of clock gating to minimize power consumption for two main reasons:

- The number of sequential elements in an NoC is very high and its micro architecture is dominated by the input queues.
- The disabling rate of the FFs in the input queues is very high.

VI. How big is the gain delivered by clock-gating?

In this section we study the impact of clock-gating on reducing the router power consumption. The analysis is based on gate-level simulations and power estimation.

A. Experimental setup

For the following experiments we used an industrial standard-cell library for 45nm CMOS technology. The experimental procedure consists of the following steps:

1. Technology mapping: using Synopsys Design Compiler, we synthesized the RTL description of FIFO queues and routers combining different queue depths $D \in \{2, 4, 8, 16\}$ and different flit width $W \in \{32, 64, 128\}$. The tool also provides area reports.

2. Clock gating: for each synthesized circuit, we performed clock-gating insertion with Synopsys Power Compiler, which automatically inserts the necessary circuitry around EFFs in a way that is transparent to the designer.

3. Gate-level simulation: we plugged each circuit obtained from the technology mapping and clock gating insertion steps into a suitable test-bench, that emulates traffic injection and ejection. Through Cadence NC-SIM, we simulated at gate-level all the test-benches, varying the traffic load. We set the clock frequency to $1GHz$.

4. Switching-activity back-annotation: during every simulation, we collected data on the switching activity for all the nets in the circuits.

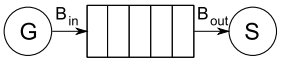

Fig. 6. Tesh-bench for FIFO simulation. The traffic source G generates flits with rate B_{in}. The sink S consumes data from the FIFO with rate B_{out}.

5. Power consumption estimation: we used Synopsys Primetime PX to estimate the power consumption of each synthesized circuit at a clock frequency of $1GHz$, according to the switching activity obtained from simulation. Primetime PX estimates the size of the clock tree based on the number of FFs in the design. We used a synthetic wire model available in the technology library. The tool also provides static timing analysis reports.

6. Comparison: we compared the circuits with clock gating and without clock gating in terms of power, area, and critical path.

B. Power gain on a single FIFO queue

We simulated different instances of FIFOs into the test-bench depicted in Fig. 6. Data traffic is generated from the source G with a rate B_{in} and it is enqueued in the FIFO. The sink S extracts data from the queue with rate B_{out}. When $B_{in} = 100\%$ the source generates one flit per clock cycle, exploiting the available linerate in full. Values of $B_{in} < 100\%$ emulate low traffic activity and linerate underutilization. When $B_{out} = 100\%$, the sink is able to consume one flit per clock cycle. Values of $B_{out} < 100\%$ emulate congestion.

Fig. 7 shows the power gain of using clock-gating to implement a FIFO as we vary the traffic load. We define: (a) *unused bandwidth* as the portion of linerate that the generator does not exploit due to lack of data, $100\% - B_{in}$ and (b) *congestion* as the percentage of clock cycles when the sink is not able to consume data, $100\% - B_{in}$. Fig. 7(a) is obtained by varying B_{in} and keeping $B_{out} = 100\%$. Fig. 7(b) is obtained by varying B_{out} and keeping $B_{in} = 100\%$. The power gain is defined as $1 - \frac{P_{CG}}{P_{\overline{CG}}}$, P_{CG} is the power dissipated by the clock-gated implementation of the FIFO, and $P_{\overline{CG}}$ is the power consumed by an instance of the same FIFO synthesized without clock gating.

Both low link utilization and high congestion increase the power gain delivered by clock-gating. In both cases the rate of data flowing through the FIFO is low, so the number of flits written in the FIFO per unit of time is also low. As discussed above, in a clock-gated FIFO writing is the only operation that dissipates power, so a low number of written flits per unit of time leads to low power consumption. On the other hand, a FIFO without clock gating dissipates the power due to the switching of the clock when flits are written as well as when there is no input activity.

Fig. 7(a) and (b) have exactly the same trends. In fact, congestion is simply another cause of low link utilization. The test-bench (Fig. 6) is by construction a balanced system that dynamically adapts its global data rate to $B = min(B_{in}, B_{out})$. The rate is low not only when the generator has no data to send, but also when the sink is not able to consume them, because in that case the FIFO quickly fills up, and the generator stops data generation to avoid overflow. Therefore, congestion is another possible cause of low writing activity in the FIFO.

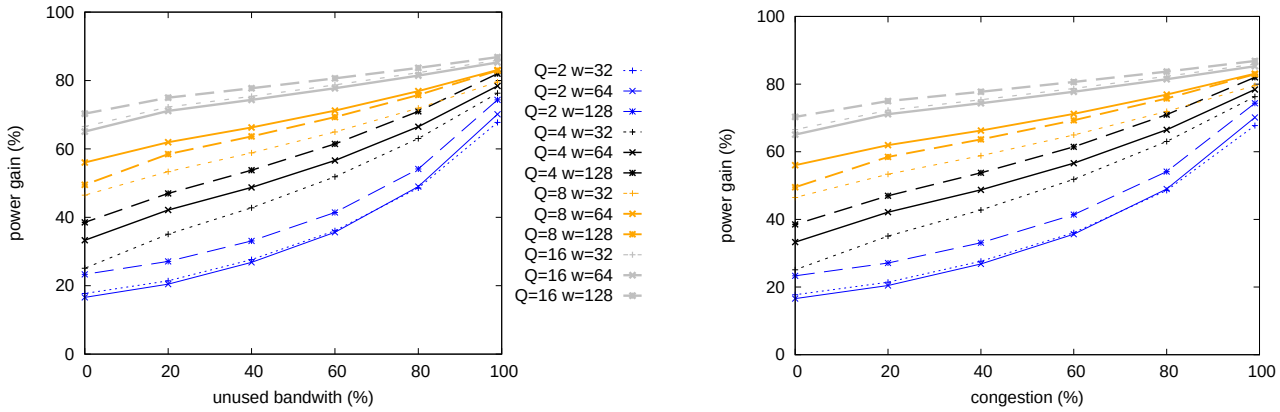

Fig. 7. Power gain in using a clock-gated implementation of a FIFO under (a) low traffic and (b) congestion.

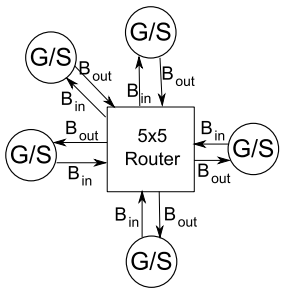

Fig. 8. Tesh-bench for router simulation. At each of the 5 I/O ports of the router, a generator/sink (G/S) module generates packets with rate B_{in} and consumes data from the router outputs with rate B_{out}.

Summarizing, *a clock-gated FIFO dynamically adapts its power consumption to the data rate flowing through it, both in case of low traffic as well as congestion.*

C. Power gain on a router

We simulated different instances of 5×5 routers into the test-bench depicted in Fig. 8. We used the same conventions as the test-bench for the FIFOs. Data traffic is injected (G) into each port with rate B_{in} and is ejected (S) from each output with rate B_{out}. The traffic injection has random generated destination port per each packet, i.e. group of consecutive flits. Values of $B_{in} < 100\%$ emulate underutilization of the link, while values of $B_{out} < 100\%$ emulate congestion.

Fig. 9 shows the power gain obtained through clock gating insertion in an NoC router, in case of (a) link underutilization and (b) congestion. The curves show similar trends as observed for the stand-alone FIFOs. This is not surprising, since the power dissipation of a router is dominated by the sequential elements. Moreover, in contrast with the case of the stand-alone FIFO, the router suffers internal congestion beside the congestion forced by the external sinks. When more packets are addressed to the same output, only one can be served at a given time: this further congestion source makes clock gating adoption even more beneficial. Note that the curves of power gain in Fig. 9 are less regular than the curves relative to the FIFO analysis (Fig. 7): we attribute this phenomenon to the effect of logic synthesis, because it is a non-linear procedure and routers have higher complexity that a stand-alone FIFO. Note also that we run experiments with different packet lengths and we found that packet length has minimal impact on the power gain due to clock gating.

D	W	Area gain (%)	Frequency gain (%)
2	32	9	-5
2	64	13	0
2	128	12	-13
4	32	17	-3
4	64	17	-1
4	128	20	-3
8	32	20	0
8	64	20	7
8	128	21	3
16	32	22	4
16	64	24	-11
16	128	25	15

TABLE I
AREA AND CLOCK FREQUENCY GAIN WHEN ADOPTING CLOCK GATING, FOR DIFFERENT INSTANCES OF NOC ROUTER VARYING QUEUE DEPTH D AND FLIT WIDTH W.

We showed again that *a clock-gated router dynamically adapts to the traffic, minimizing the power consumption during low traffic as well as congestion.*

D. Timing and Area trade-offs

In Table I we report the area and clock frequency gains deriving from the insertion of clock gating. Area gain is defined as $1 - \frac{A_{CG}}{A_{\overline{CG}}}$, where A_{CG} is the area of the router synthesised with clock gating, and $A_{\overline{CG}}$ is the area of the same router synthesized without clock gating insertion. In the same way, the clock frequency gain is defined as $1 - \frac{T_{pd,CG}}{T_{pd,\overline{CG}}}$. Values lower > 0 represent a gain in using clock gating, while values < 0 indicate an overhead.

Clock gating delivers a consistent area gain, up to 25% for big queues. Recalling Fig. 2 and 5, a FIFO without clock gating has one multiplexer per bit beside the FF, while a FIFO with clock gating does not have the additional multiplexer, but only one CG cell per FIFO location. Moreover, the bigger is the FIFO, the higher is the impact of the FIFO area on the overall router area, therefore more noticeable is the area gain.

In terms of timing we observe a remarkable non linearity. Sometimes the use of clock gain delivers advantages, sometimes it presents an overhead. However, both gain and overhead vary in a $\pm 15\%$ range.

In conclusion, we can assess that clock gating introduction has virtually no overhead on the design, but rather delivers some area savings.

VII. CASE STUDY

We consider one possible router implementation, with $D = 8$ and $W = 64$. The experimental flow is the same as the

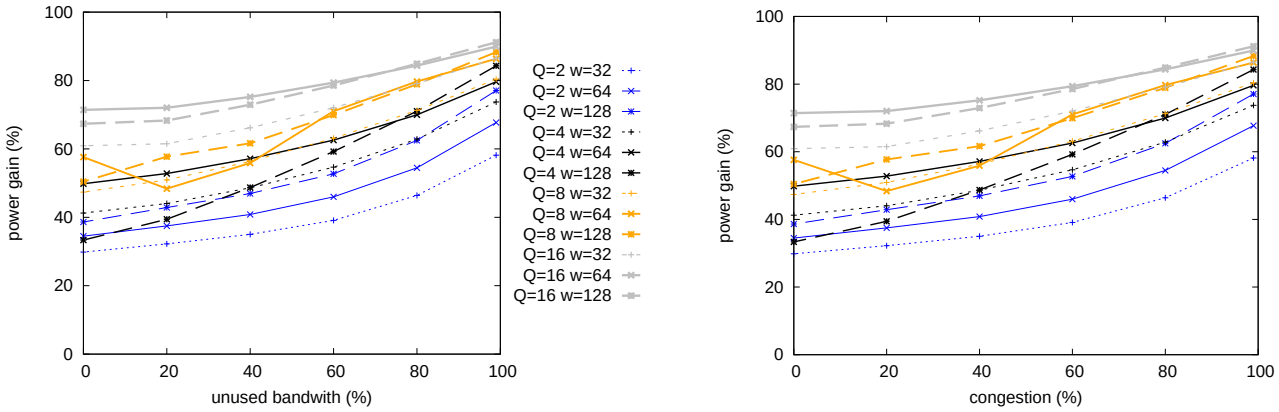

Fig. 9. Power gain in using a clock-gated implementation of a router under (a) low traffic and (b) congestion.

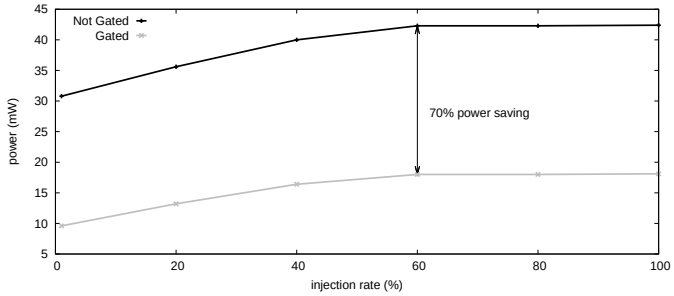

Fig. 10. Power consumption of a clock-gated and a not clock-gated $5\ times5$ NoC router, with in a 4×4 2D-Mesh, with XY-routing, under different injection rates of uniform random traffic.

one discussed in Section VI-A. Moreover, between the *Clock gating* and *Gate-level simulation* steps we performed *Circuit layout*, using Cadence SoC Encounter. During the layout step the tool performs real Clock Tree Synthesis (CTS). We also extracted RC parasitics and back-annotated such information during the *Power consumption estimation*.

We inserted the router in a 4×4 2D-Mesh NoC and simulated the entire NoC under uniform random traffic (URT) at different traffic injection rates. We set the ejection rate to 100%, emulating cores that are always able to correctly store the received data. This setting allows us to model only the network congestion deriving from colliding traffic, removing any network congestion possibly introduced by slow cores.

Fig. 10 compares the power dissipation of one router located in the middle of the 2D-Mesh, case clock-gating is adopted and in case of clock-gating is not adopted. In both cases increasing traffic leads to higher power consumption. Note that the power consumption saturates for traffic injection higher that 60%, because for a 2D-Mesh, with XY-dimension order routing, under URT, the saturation throughput is around 50%. A packet injection higher than 50% of the linerate results in a constant back-pressure, which forces the source to discard exceeding traffic before even being able to inject it into the network.

The main difference between the two curves is a constant power gain of 70%, which corresponds to the power dissipated due to clock activity during periods of low traffic at the router inputs. In summary, *clock-gated routers reduces power consumption of a factor of* 70% *under every injection rate*.

VIII. Conclusions

The adoption of standard-cell technology libraries and commercial CAD tools is required to cope with the increasing complexity and to tight time-to-market constraints. RTL design of NoC components and logic synthesis deliver the modularity and portability, responding to the need of interfacing heterogeneous IPs when building a multi-core system.

NoCs are highly demanding in terms of storage, in particular to handle traffic congestion. In semi-custom design the storage is generally implemented by using FFs. Then, the high number of FFs and the necessary clock distribution dominate the power dissipated by a NoC.

We observed that, *(i)* given the possible architecture of routers' input buffers and *(ii)* given the typical characterization of the NoC traffic due to congestion and flow control, the FFs in the NoC are disabled for the majority of the time, i.e., do not sample a new data on the majority of the clock cycles.

Hence, NoCs are particularly suitable to the application of clock gating as a design technique that can be fully integrated with standard semi-custom design flows to deliver dramatic improvements when FFs are disabled. We explained and quantified to which extent NoC routers can benefit from this technique. Based on our comprehensive analysis, clock gating applied to NoC *(i)* provides full adaptability of the routers to the traffic load by minimizing the power wasted during link inactivity and congestion, *(ii)* is completely transparent to the RTL designer and perfectly handled by CAD tools, *(iii)* introduces no area or timing overhead.

References

[1] L. Benini and G. D. Micheli, "Networks on chip: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. of the Design Automation Conference*, Jun. 2001, pp. 684–689.

[3] A. Hemani *et al.*, "Network on chip: An architecture for billion transistor era," in *18th IEEE NorChip Conference*, Nov. 2000.

[4] P. Kollig, C. Osborne, and T. Henriksson, "Heterogeneous multi-core platform for consumer multimedia applications," in *Conf. on Design, Automation and Test in Europe*, Apr. 2009, pp. 1–6.

[5] C. H. van Berkel, "Multi-core for mobile phones," in *Conf. on Design, Automation and Test in Europe*, Apr. 2009, pp. 1–6.

[6] J. Cortadella, M. Kishinevsky, and B. Grundmann, "Synthesis of synchronous elastic architectures," in *Proc. of the Design Automation Conference*, Jul. 2006, pp. 657–662.

[7] H. M. Jacobson, P. Kudva, P. Bose, P. W. Cook, and S. Schuster, "Synchronous interlocked pipelines," in *Proc. of Intl. Symp. on Asynchronous Circuits and Systems*, apr 2002, pp. 3–12.

[8] W. Liao and L. He, "Full-chip interconnect power estimation and simulation considering concurrent repeater and flip-flop insertion," in *Proceedings of the international conference on Computer-aided design ICCAD*, Nov. 2003, pp. 574–580.

[9] Y. Hoskote *et al.*, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sept.-Oct. 2007.

[10] D. Wentzlaff *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, 2007.

[11] A. Kahng, B. Li, L. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Design, Automation, and Test in Europe*, Apr. 2009, pp. 423–428.

[12] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[13] L. Bononi and N. Concer, "Simulation and analysis of network on chip architectures: ring, spidergon and 2D mesh," in *Conf. on Design, Automation and Test in Europe*, vol. 2, Mar. 2006, pp. 154–159.