Mixed Integer Linear Programming-based Optimal Topology Synthesis of Cascaded Crossbar Switches

Minje Jun
Yonsei University,
Seoul, Korea
mjun@dtl.yonsei.ac.kr

Sungjoo Yoo
Samsung Electronics,
Yongin, Korea
sungjoo.yoo@samsung.com

Eui-Young Chung
Yonsei University,
Seoul, Korea
eychung@yonsei.ac.kr

Abstract - We present a topology synthesis method for high performance System-on-Chip (SoC) design. Our method provides an optimal topology of on-chip communication network for the given bandwidth, latency, frequency and/or area constraints. The optimal topology consists of multiple crossbar switches and some of them can be connected in a cascaded fashion for higher clock frequency and/or area efficiency. Compared to previous works, the major contribution of our work is the exactness of the solution from two aspects. First, the solving method of our work is exact by employing the mixed integer linear programming (MILP) method. Second, we generalize the crossbar switch representation in MILP in order that the optimal topology can include any arbitrary sizes of crossbar switches together. The experimental results show that the topologies optimized for the clock frequency (area) give up to 37.3% (12.7%) improvements compared to the conventional single large crossbar switch networks for two industrial strength SoC designs.

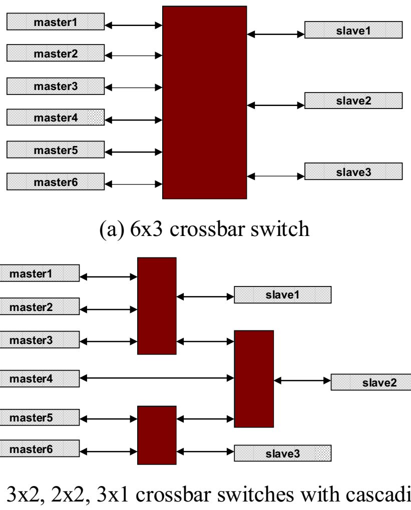
I. Introduction

As the semiconductor process technology advances rapidly, billion transistors or hundred cores are integrated on a single chip. Also, many embedded systems are designed for data-intensive applications in which frequent data transfers among the computing cores and memories are unavoidable. Therefore, the interconnection method of those components has become a more challenging problem as the performance requirement is increasingly demanding.

To keep pace with such performance requirement, several works proposed the enhancement of the traditional on-chip buses. Efficient arbitration schemes are proposed in [4][5] to increase the performance of a shared bus. In [7][14], new shared bus architectures with physical segmentation are proposed. In spite of their effectiveness, the nature of shared buses highly limits the performance. More precisely, the performance of shared bus is not scalable with respect to the number of components attached to the bus, since it is temporally shared by those components.

Network-on-Chip (NoC) is a viable solution to cope with the performance and the scalability issues, and actively studied in academia [3][8][16][17]. Also, there is a consensus that NoC will be one of the preferred interconnection methods for high performance systems in the near future. It is commonly observed that NoC is targeting larger scale systems than crossbar switches. Crossbar switches have been used as backbone interconnection especially in CMP's [19]. Recently, the industry has focused on crossbar switch solutions for SoC. ARM Inc. productized the crossbar switch named as PL301 and Sonics Inc. also commercialized the crossbar switch called Sonics-SMX [1][15]. Several research groups in academia also worked on the crossbar switch (bus matrix) [9][10][11][19]. Thus, our study focuses on the crossbar switch-based design of SoC backbone bus.

In this paper, we will focus on the specialization of crossbar-based interconnection topology to the given applications. It is well known that a fully connected crossbar switch outperforms shared busses [6][13]. However, the single fully connected crossbar suffers from the increase of its area and the decrease of its operating clock frequency as the number of input / output ports (masters / slaves) increases [12]. For this reason, it is often better to use multiple small crossbar switches rather than a single large crossbar switch for a higher operating clock frequency with small area overhead. Fig. 1 illustrates this idea. Fig. 1 (a) gives an example of the conventional single large crossbar and Fig. 1 (b) is an equivalent cascaded crossbar network. As shown in Fig. 1 (b), the cascaded crossbar switches consist of smaller (i.e. faster) crossbars than the conventional large (i.e. slow) one. Thus, the cascaded crossbar switches can give higher operating frequency. The application-specific crossbar network can also increase area efficiency while maintaining the



(a) 6x3 crossbar switch

(b) 3x2, 2x2, 3x1 crossbar switches with cascading

Fig. 1 Concept of cascaded crossbar switch topology

higher frequency.

The aim of our method is to optimize the topology with crossbar switches for the given design constraints, but it improves the optimization quality by considering two factors concurrently. First, the solving method of our work is exact by employing the mixed integer linear programming (MILP). At the same time, our method can include multiple arbitrary sizes of crossbar switches in an optimal topology. To the best of our knowledge, none of the previous works considers these two factors in a single method.

We will introduce the related works and address our topology synthesis method in Section 2 and Section 3, respectively. We will give the experimental results in Section 4 and conclude this paper in Section 5.

II. Related Works

The works related to the crossbar switches can be classified into two categories. The works in the first category focused on the improvement of the single crossbar switch, since a fully-connected crossbar switch is wasting the channel resources when there is no need of communications between all pairs of masters and slaves [8][9]. The works in the second category addressed optimal topology synthesis method with multiple switches [10] [16][17][19].

Murali *et al.* proposed a technique for partially-connected bus matrix generation and showed that the area and the clock frequency of the switch could be improved with their method [9]. Pasricha *et al.* extended this concept by considering other design parameters such as arbitration scheme and buffer size for further improvement [10]. Murali *et al.* also attempted to minimize the size of a crossbar switch by clustering masters and slaves, respectively. Each cluster is connected to a central fully-connected crossbar switch [8]. They also showed significant reduction of the power consumption and the area. However, their method restricts the cluster into a shared bus, which can be thought of as an $n \times 1$ $(1 \times n)$ crossbar, and therefore can be considered as an extreme case of our work in terms of topology.

Yoo *et al.* proposed a topology synthesis method using crossbar switches in a cascaded fashion based on simulated annealing [19]. A bus pipeline stage called register slice can be inserted in-between each pair of cascaded switches for the timing isolation of the two switches. Their method is more general (hence, a huge design space) than other works in the sense that the optimal topology can include any arbitrary sizes of crossbar switches with register slice, asynchronous bridge, and downsizer and expander. However, their solving method is not exact. On the other hand, Srininvasan *et al.* proposed an exact method for solving NoC topology problem based on MILP [17]. Even though their solving method is exact, it is not as general as the work in [19], since only a fixed size of crossbar switch can be used in a single optimal topology. Such a limitation also affects the quality of the solution due to the partial search of the target design space.

The novelty of our work is to consider the positive features of both approaches in a single solving method - an exact topology synthesis method with arbitrary sizes of crossbar switches.

III. Optimal Topology Synthesis

A. Assumptions

Our method aims at synthesizing an optimal crossbar network topology for a specific application. The cost of the topology can be the crossbar area, the maximum network frequency, or a mixture of them. Thus, we assume that the communication characteristics of the target application and the area/frequency information of the crossbars are given.

On-chip communication network consists of IP's and bus components including crossbars and bridges (mostly for data width and frequency conversion). In our work, it is assumed that required bridges are already inserted into a part of IP's (e.g. IP interface) so that we focus on the central part of the on-chip network.

We also assume that the routing of the crossbar is based on physical address (e.g. ARM PL301 whose routing is based on AXI address and Sonics SMX based on OCP address). In terms of master-to-slave communication, we assume that there is a single communication path from master to slave as in the commercial solutions (PL301, SMX). Thus, multi-path communication is not considered at the master/slave level. However, a pair of IPs can communicate with each other through more than one path using multiple master/slave ports, and in this case each port is considered as an individual master/slave. We have another single connection assumption between any pair of two crossbars.

The main design target of our method is the backbone bus of high performance SoC. Most of backbone buses are characterized by a single high frequency. Thus, in our problem, we assume that the cascaded crossbar switches have a single operating frequency.

B. Problem Definition

The crossbar network consists of three types of components: master (*M*), slave (*S*), and crossbar (*X*). Also, its topology contains three types of connections: master to crossbar (*MX*), slave to crossbar (*SX*), and crossbar to crossbar (*XX*).

Basic topology decision variables

- $MX_{m,x}$ $(v_m \in V_M, x \in X)$: a decision variable matrix which indicates whether master v_m is connected to crossbar x or not. $MX_{m,x}$ is one only if master v_m is connected to crossbar x, otherwise zero.
- $SX_{s,x}(v_s \in V_S, x \in X)$: a decision variable matrix which indicates whether slave v_s is connected to crossbar x. $SX_{s,x}$ is one only if slave s is connected to crossbar x, otherwise zero.
- $XX_{x,x'}(x \in X, x' \in X)$: a decision variable matrix which indicates whether crossbar x is connected to crossbar x'. We will refer to the connections between any two crossbars as *links*. $XX_{x,x'}$ is one only if crossbar x is connected as a master

to x', otherwise zero. For the reduction in the number of decision variables, we give indexes to crossbars such that lower-indexed crossbar is always on the master side when there exists any crossbar to crossbar connection. Thus, we do not consider $XX_{i,j}$ s.t. $i \geq j$ in the MILP formulation.

Our topology synthesis problem can be defined as follows.

Given

- An undirected *communication trace graph (CTG)* $G(V_M, V_S, E)$ is the communication characteristics of the target application, where $v_m \in V_M$ denotes a master node, $v_s \in V_S$ a slave node, and $e_{m,s} \in E$ an undirected edge between v_m and v_s
- $w(e_{m,s})$ is the bandwidth requirement of $e_{m,s}$, and $d(e_{m,s})$ is the latency constraint of $e_{m,s}$.
- $A_{m,s}$ and $F_{m,s}$ are the area and the frequency of the crossbar with m master ports and s slave ports, respectively.

This problem is to **find** topology $T(MX', SX', XX')$ which optimizes the design objective (e.g., area and frequency).

s. t. the communication requirements $w(e_{m,s})$ and $d(e_{m,s})$, and other design constraints, such as area and frequency, are satisfied.

We set $|X|$ as the maximum number of crossbar to be used in the topology synthesis, and N as the maximum number of possible cascading stages. Note that they are design parameters to be set by the designer, but the optimality of the solution is not influenced if they are set large enough.

In the following subsections, we present our MILP formulation of the problem. The formulation consists of four types of constraint (for topology, communication path, latency and bandwidth) and two design cost factors (total network area and the network frequency).

C. Constraints

i. Topology Feasibility Constraint

An arbitrary selection of crossbars and their connection may not always give valid topologies. We define the valid topology of cascaded crossbar switches with the following two constraints.

Constraint 1 – A master or a slave must be connected to one and only one crossbar.

$$\forall v_m \in V_M,\ \textstyle\sum_{x \in X} MX_{m,x} = 1 \qquad (1)$$

$$\forall v_s \in V_S,\ \textstyle\sum_{x \in X} SX_{s,x} = 1 \qquad (2)$$

Constraint 2 – The degree of crossbar switch must be zero or greater than two.

$$PM_x = PS_x = 0 \qquad (3) \text{ or}$$

$$PM_x > 0 \,\&\, PS_x > 0 \,\&\, PM_x + PS_x > 2 \qquad (4)$$

where PM_x and PS_x are the number of master-side ports (shortly master port) and slave-side ports (shortly slave port) of crossbar x, respectively. The number of master ports of a crossbar depends on how many masters are connected to it and how many other crossbars cascade to it. On the other hand, the number of slave ports depends on the number of slaves connected to it and that of other crossbars it cascades to. They can be calculated using the basic decision variables as follows.

$$PM_x = \textstyle\sum_{v_m \in V_M} MX_{m,x} + \sum_{x' \in X, x' < x} XX_{x',x} \qquad (5)$$

$$PS_x = \textstyle\sum_{v_s \in V_S} SX_{s,x} + \sum_{x' \in X, x' > x} XX_{x,x'} \qquad (6)$$

As Eqn. (3) shows, it is possible for crossbar x to have both PM_x and PS_x of zeros, which means that crossbar x is not used in the synthesized topology. If crossbar x is used in the topology, Inequality (4) needs to hold because $0 \times n$ and $n \times 0$ crossbars are useless when n is non-zero as well as the 1×1 crossbar.

ii. Single Communication Path Constraint

If there is a bandwidth requirement between master v_m and slave v_s, i.e., $w(e_{m,s}) > 0$, then according to the assumption of single communication path in Section III-A, there must exist only one path on the topology from master v_m to slave v_s. Thus, to represent the constraint of single communication path, the following constraint is needed.

Constraint 3 – For any pair of master v_m and slave v_s whose $w(e_{m,s}) > 0$, there must exist only a single communication path.

$$\textstyle\sum_{n=1}^{N}\left(\sum_{x_1 \in X} \cdots \sum_{x_n \in X} D^n_{m,x_1,x_2,\ldots,x_n,s}\right) = 1 \qquad (7)$$

where variable $D^n_{m,x_1,x_2,\ldots,x_n,s}$ represents whether there is a communication path from master v_m to slave v_s through crossbars x_1, x_2,..., and x_n. If there exists such a path, the variable is one, otherwise zero. We define *depth* as the number of crossbars on the path through which master v_m and slave v_s are connected. We call the path with n intervening crossbars *depth-n path.* $D^n_{m,x_1,x_2,\ldots,x_n,s}$ is defined as follows.

$$D^n_{m,x_1,x_2,\ldots,x_n,s} = MX_{m,x_1} \times \textstyle\prod_{k=1}^{n-1} XX_{x_k,x_{k+1}} \times SX_{s,x_n} \qquad (8)$$

Linearization of Eqn. (8)

Eqn. (8) is not a linear expression. Thus, we apply a linearization to the equation and use the following inequalities instead of Eqn. (8).

$$D^n_{m,x_1,x_2,\ldots,x_n,s} \geq MX_{m,x_1} + \textstyle\sum_{k=1}^{n-1} XX_{x_k,x_{k+1}} + SX_{s,x_n} - n \qquad (9)$$

$$D^n_{m,x_1,x_2,\ldots,x_n,s} \leq 1/n \times \left[MX_{m,x_1} + \textstyle\sum_{k=1}^{n-1} XX_{x_k,x_{k+1}} + SX_{s,x_n}\right] \qquad (10)$$

Inequality (9) forces $D^n_{m,x_1,x_2,\ldots,x_n,s}$ to be one when master v_m is connected to slave v_s through x_1 to x_n. Inequality (10) forces $D^n_{m,x_1,x_2,\ldots,x_n,s}$ to be zero if there is any zero term in the summation on its right-hand side. Note that, as explained in Section III-A, since we assume $XX_{x,x'}$ is defined only where $x<x'$, $D^n_{m,x_1,x_2,\ldots,x_n,s}$ is also defined only for $x_1 < x_2 < \ldots < x_{n-1} < x_n$. Thus, we can effectively reduce the number of decision variables.

iii. Latency Constraint

The latency between components varies with the arbitration scheme, the buffer size, and their traffic characteristics as well as the hop count. This is the reason why the final solution is tested for latency constraint by overall simulation in [10]. In our method we abstract the latency constraint.

Constraint 4 – The latency constraint (e.g., $d(e_{m,s}) = k$) that master v_m has a latency constraint to slave v_s with requiring a hop count not exceeding k is represented as follows.

$$\forall (v_m, v_s)\ with\ d(e_{m,s}) = k,$$
$$\sum_{n=1}^{N}\left(\sum_{x_1\in X} \cdots \sum_{x_n \in X} D^n_{m,x_1,x_2,\ldots,x_n,s}\right) \geq 1 \qquad (11)$$

Inequality (11) represents that at least one D^n matrices with $n \leq k$ must be one.

iv. Bandwidth Constraint

In order to check to see if the given bandwidth requirement is met, we check the total communication bandwidth that each link in the topology carries. If more bandwidth is loaded on a link than its capability (i.e. peak bandwidth) the corresponding topology cannot meet the given bandwidth requirement.

Constraint 5 – The total bandwidth flowing on each edge must not exceed the peak bandwidth.

$$\forall x_1 < x_2 \in X,\ E_{x_1,x_2} \leq CostFreq \times channelwidth \quad (12)$$

where E_{x_1,x_2} is the total bandwidth flowing from crossbar x_1 to x_2, *CostFreq* is a continuous decision variable indicating the maximum operating clock frequency of the topology, and *channel_width* is the channel bit width. In the remaining of this subsection, we explain how to calculate variable E_{x_1,x_2} and *CostFreq*.

Linear expressions E_{x_1,x_2} and $\varepsilon^n_{x_1,x_2,m,s}$

First, we define $\varepsilon^n_{x_1,x_2,m,s}$ as the bandwidth induced by master v_m and slave v_s, flowing on the link between crossbar x_1 and x_2 in a depth-n path. Fig. 2 illustrates how to calculate $\varepsilon^n_{x_1,x_2,m,s}$. In Fig. 2, the link between x_2 and x_5 carries only the traffic between m_2 and s_2 through depth-3 connection ($\varepsilon^3_{x_2,x_5,m_2,s_2}$). On the other hand, the link between crossbar x_1 and x_2 delivers both the traffic induced by m_1 and s_1 which is in a depth-2 connection ($\varepsilon^2_{x_1,x_2,m_1,s_1}$), and the traffic induced by m_2 and s_2 which is in a depth-3 connection ($\varepsilon^3_{x_1,x_2,m_2,s_2}$). Therefore the load on the link from crossbar x_1 to x_2 can be calculated as follows.

$$E_{x_1,x_2} = \varepsilon^2_{x_1,x_2,m_1,s_1} + \varepsilon^3_{x_1,x_2,m_2,s_2} \qquad (13)$$

while $\varepsilon^n_{x_1,x_2,m,s}$ can be generally formulated as follows.

$$\varepsilon^n_{x_1,x_2,m,s} = w(e_{m,s}) \times \left[\sum_{k=n}^{N}\left(\sum_{x_3\in X} \cdots \sum_{x_k\in X} D^k_{m,x_1,x_2,\ldots,x_n,s}\right)\right] \quad (14)$$

Note that $\varepsilon^n_{x_1,x_2,m,s}$ does not need to be a decision variable matrix since it is a simple summation of the decision variable matrices. After calculating $\varepsilon^n_{x_1,x_2,m,s}$ for each depth of links, we can finally find the total bandwidth flowing on a link, E_{x_1,x_2}, by summing the bandwidths induced by all master and slave pairs from all connection depths as follows.

$$\forall x_1 < x_2 \in X,\ E_{x_1,x_2} = \sum_{k=2}^{N} \sum_{v_m\in V_M} \sum_{v_s \in V_S} \varepsilon^n_{x_1,x_2,m,s} \quad (15)$$

D. Desgin Cost Factors (Network Area and Frequency)

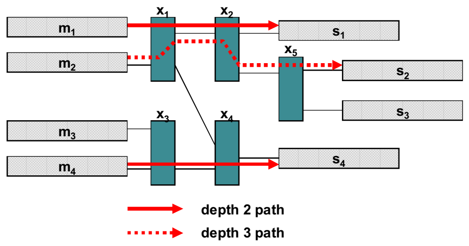


Fig. 2 Bandwidth accumulation in cascaded path

Decision variables CostFreq and I_x

The slowest crossbar determines the maximum clock frequency of entire cascaded crossbar switches. Thus, we need to find the minimum frequency among $|X|$ crossbars. However, finding the minimum value in a set is not a linear function. To implement this function, we define a decision variable matrix I_x which is one only if *CostFreq* equals to $Freq_x$, where $Freq_x$ denotes the maximum frequency of crossbar x. Then *CostFreq* can be calculated with linear expressions as follows.

$$\forall x \in X,$$
$$I_x \leq 1 - \delta(CostFreq - Freq_x) \qquad (16)$$
$$I_x \leq 1 + \delta(CostFreq - Freq_x) \qquad (17)$$
$$\sum_{x\in X} I_x \geq 1 \qquad (18)$$
$$CostFreq \leq Freq_x \qquad (19)$$

where δ is a positive fractional constant which is small enough to make $\delta(CostFreq\text{-}Freq_x)$ always less than or equal to one. In addition, it is recommended to set δ as large as possible until the aforementioned property holds for the branch- and-bound tightness. Inequality (16) forces I_x to be zero when *CostFreq* is greater than $Freq_x$ while Inequality (17) forces I_x to be zero when *CostFreq* is smaller than $Freq_x$. Inequality (18) forces *CostFreq* to be equal to at least one of $Freq_x$'s. Finally, I_x is allowed to be one only if *CostFreq* equals to the minimum of $Freq_x$'s by Inequality (19).

Decision variable matrix $K_{x,m,s}$ for table referencing

$Freq_x$ can be obtained as F_{PM_x,PS_x}. However, table referencing with decision variables is not a linear expression. Hence we define a new decision variable matrix $K_{x,m,s}$ for table referencing, with $m=0,1,\ldots,|V_M|$. $s=0,1,\ldots,|V_S|$, and $x \in X$. $K_{x,m,s}$ is one only when $PM_x=m$ and $PS_x=s$. Then, this matrix can be calculated as follows.

$$\forall 0 \leq m \leq |V_M|, 0 \leq s \leq |V_S|, and\ x \in X$$
$$K_{x,m,s} \leq 1 - \beta\{m - PM_x + \alpha(s - PS_x)\} \qquad (20)$$
$$K_{x,m,s} \leq 1 + \beta\{m - PM_x + \alpha(s - PS_x)\} \qquad (21)$$
$$\sum_{m=0}^{|V_M|}\sum_{s=0}^{|V_S|} K_{x,m,s} = 1 \qquad (22)$$

where α and β are positive fractional constants which are small enough to make the terms in the parenthesis and brace on the right-hand sides of Inequality (20) and (21) less than one, respectively.

Then F_{PM_x,PS_x} is calculated as the multiplication of constant matrix $F_{m,s}$ and variable $K_{x,m,s}$. Moreover, the *Constraint 2* can be applied by setting constraints to $K_{x,m,s}$ as follows.

TABLE I. Application workload profile

		Application I												Application II								
		M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	vu	au	cpu	rast	idct	risc	bab	upsp	dsp
BW (MB/s)	S1	5	0	0	0	50	60	240	120	180	40	5	0	190	0.5	60	600	0	0	32	910	0.5
	S2	0	5	600	0	0	0	0	0	0	300	0	5	0	0	600	40	0	0	0	0	0
	S3	0	0	0	0	690	0	540	0	0	120	0	90	0	0	0	0	250	500	193	670	0
	S4	0	0	60	720	330	0	0	0	0	500	90	0	N/A								
Latency Constraint		(M1, S1) - minimum latency required. (M2, S2) - minimum latency required. (M11, S1) – at most depth two connection allowed (M12, S2) – at most depth two connection allowed												(vu, mem1) - minimum latency required. (au, mem1) - minimum latency required.								

For Application II, S1, S2, and S3 are mem1, mem2, and mem3 in the CTG in [16], respectively

$$\forall x \in X,\ K_{x,1,1} = 0 \tag{23}$$

$$\forall x \in X, m \in \{1,2,\ldots,|V_M|\}, K_{x,m,0} = 0 \tag{24}$$

$$\forall x \in X, s \in \{1,2,\ldots,|V_S|\}, K_{x,0,s} = 0 \tag{25}$$

Linear expression CostArea

The total network area, $CostArea$, can be calculated as the sum of all crossbar areas found in the topology and inter-crossbar pipeline stages on all crossbar-to-crossbar links as follows.

$$CostArea = \sum_{x \in X} \sum_{m=0}^{|V_M|} \sum_{s=0}^{|V_S|} \left(A_{m,s} \times K_{x,m,s}\right) + \gamma \sum_{x_1 \in X} \sum_{x_2 \in X} XX_{x_1,x_2} \tag{26}$$

where γ is the area of a pipeline stage. The first term in the right-hand side of the equation is a linearized form of table referencing operation using $K_{x,m,s}$ matrix, which is exactly the same as $\sum_{x \in X} A_{PM_x,PS_x}$.

IV. Experiment

We applied the MILP formulation to two industrial strength SoC designs. Table I gives the summary of two applications including the bandwidth and latency requirements. Application I is +20Mgate design and has more than 50 masters/slaves. Many of masters/slaves are connected to low bandwidth local AXI/AHB/APB buses. The design of those buses is not as challenging as that of high performance backbone bus. Thus, we applied the MILP-based bus design method to the design of backbone bus that has 12 masters and 4 slaves. Masters and slaves include video codecs, graphics IPs, ARM11, DDR and NAND Flash memory[1]. Application II is an MPEG-4 decoder with 9 masters and 3 memories [16]. In Application II, we added latency constraints to two masters of the CTG in [16].

For each application, we synthesized the topologies for four different objectives; (Objective 1) to maximize the operating frequency without area penalty upper bound, (Objective 2) to maximize the operating frequency with the area penalty upper bound, (Objective 3) to minimize the total crossbar area without clock frequency lower bound, and (Objective 4) to minimize the crossbar area with clock frequency lower bound.

For area/delay characterization of crossbar, we use AMBA Designer [1] to generate the RTL code of crossbar. The synthesis result is obtained from a proprietary 90nm process technology. For the crossbar information table, $A_{m,s}$ is in mm^2, and $F_{m,s}$ in MHz.

[1] More details of masters and slaves are hidden for confidentiality reason.

We set the channel bit width to 32bit (4byte). For Application I, we set the maximum possible cascading stages, N, to 3 and the maximum number of available crossbars, $|X|$, to 5. For Application II, we used 2 and 5 for N and $|X|$, respectively. For Objective 2 we set the area penalty upper bound to 30% with respect to the area of one fully connected crossbar. For Objective 4, we used 380MHz as the required frequency lower bound. We solve our MILP formulation using XPRESS-MP [2] in Pentium4 based machine.

Fig. 3 shows the synthesized topologies of Application I and II for each of the four objectives. Each inter-crossbar link is also marked with the total bandwidth. Since the latency is proportional to the cascading depth, some paths with low latency constraints are established through only one crossbar. As shown in Fig. 3, (M1, S1) and (M2, S2) in Application I and (vu, mem1) and (au, mem1) in Application II are connected through only one crossbar for all cases since they require the minimum latencies. It is also shown that (M11, S1) and (M12, S2) in Application I are connected through no more than two crossbars, as required in the CTG. Regarding the bandwidth on the links, it is shown to be within the peak bandwidth capability of the links. For instance, the most loaded link in Fig. 3 (b) delivers 1590MB/s of traffic but it is still within the ideal bandwidth of the link which is 1612MB/s (4byte×406.5Mhz).

Table II gives the summary of experimental results. The values in the parenthesis are the improvement (+) or the overhead (-) with respect to the corresponding single crossbar design. The frequency improvement for Objective 1 is dramatic while the area overhead is also very large. For Objective 2, considerable increase in clock frequency is achieved within the area overhead upper bound of 30%. We could improve both area and frequency at the same time for Application II with Objective 3 and 4, while only Objective 4 achieved gain for Application I. Note that success in the area minimization accompanies with the frequency gain since we assume register-slicing on the cascaded paths. As shown in the table, the proposed method gives much better results in terms of frequency and/or area than single crossbar designs for the two industrial strength designs.

V . Conclusion

We proposed a method of finding the optimal topology of cascaded crossbar switches based on an MILP formulation. With the database of pre-characterized area/frequency information of crossbars, we applied the MILP formulation to

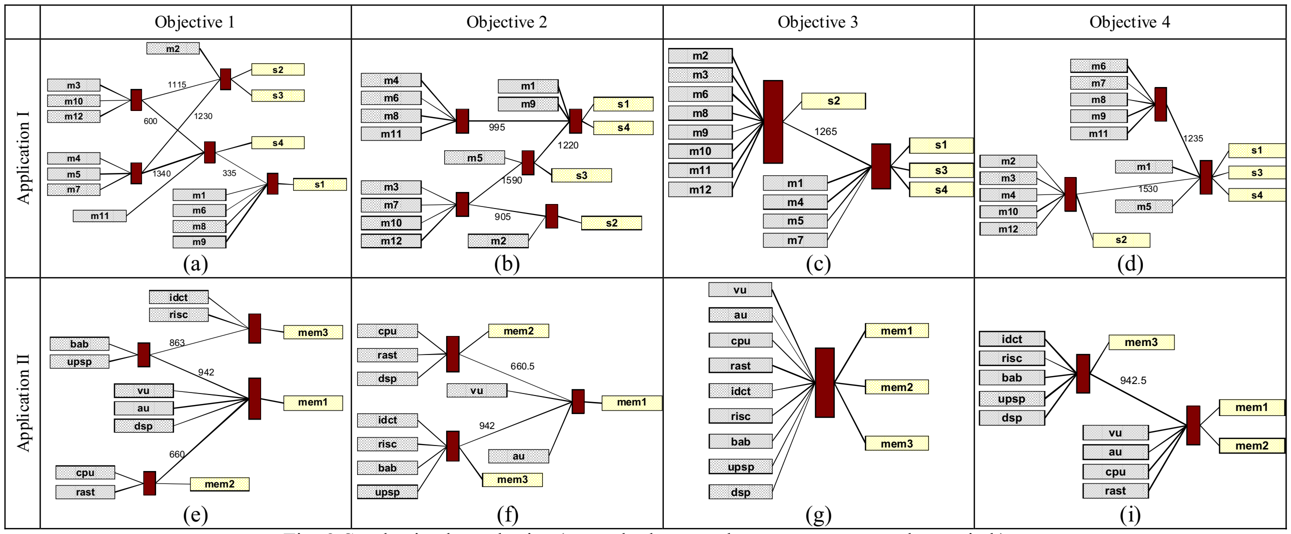


Fig. 3 Synthesized topologies (unmarked rectangle represents a crossbar switch)

Table II. Synthesis results

		Metric	Objective 1	Objective 2	Objective 3	Objective 4
Application I	12×4	Freq.	304.8			
		Area	0.629			
	Synthesized	Freq.	418.4(+37.3%)	406.5(+33.4%)	371.7(+21.9%)	384.6(+26.2%)
		Area	0.903(-43.6%)	0.813(-29.3%)	0.549(+12.7%)	0.613(+2.54%)
		Time	78687	118038	3710	26220
Application II	9×3	Freq.	344.8			
		Area	0.392			
	Synthesized	Freq.	423.7(+22.9%)	406.5(+17.9%)	344.8(0%)	384.6(+11.5%)
		Area	0.554(-68.4%)	0.509(-29.8%)	0.392(0%)	0.407(-4%)
		Time	4627	470	56	39

Freq. in Mhz, Area in mm^2, Time in seconds

two applications. The results show that our method gives significant improvements in most of the design cases; frequency improvement by up to 37.3% and area reduction by up to 12.7% together with 21.9% increase of clock frequency. As our future work, we will integrate power consumption into the MILP formulation.

Acknowledgement

This work was partially supported by grant No. R01-2006-000-10156-0 from the Basic Research Program of the Korea Science & Engineering Foundation, by IT R&D Project funded by Korean Ministry of Information and Communications, and by IDEC(IC Design Education Center).

References

[1] ARM, www.arm.com

[2] Dash Optimization, www.dashoptimization.com

[3] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, “xpipes: a Latency Insensitive Parameterized Network- on-chip Architecture For Multi-Processor SoCs”, *International Conference on Computer Design 2003*, pp.536-539

[4] M. Jun, K. Bang, H. J. Lee, N. Chang, and E. Y. Chung, "Slack based Bus Arbitration Scheme for Soft Real-time Constrained Embedded Systems", *ASPDAC 2007*, pp. 159-164

[5] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "LOTTERY-BUS: A New HighPerformance Communication Architecture for System-on-Chip Designs", *DAC 2001*, pp. 15-20

[6] M. Loghi, F. Angiolini, D. Bertozzi, and L. Benini, "Analyzing On-Chip Communication in a MPSoC Environment", *DATE 2004*, vol. 2, pp. 752-757

[7] R. Lu and C. K. Koh, "SAMBA-BUS: A High Performance Bus Architecture for System-on-Chips", *ICCAD 2003*, pp. 8-12

[8] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, “The Nostrum backbone - a communication protocol stack for networks on chip”, *Proceedings of the VLSI Design Conference*, Jan. 2004, pp. 693-696

[9] S. Murali, L. Benini, and G. De Micheli, "An Application- Specific Design Methodology for On-Chip Crossbar Generation", Trans. VLSI (to appear), available at http://infoscience.epfl.ch/

[10] S. Murali and G. De Micheli, "An Application-Specific Design Methodology for STbus Crossbar Generation", *DATE 2005*, pp. 1176-1181

[11] S. Pasricha, N. Dutt, M. Ben-Romdhane, "Constraint-Driven Bus Matrix Synthesis for MPSoC", *ASPDAC 2006*, pp. 30-35

[12] L. Peh and W. J. Dally, "A Delay Model for Router Microarchitectures", *HPCA 2001*, pp. 255-266

[13] K. K. Ryu, E. Shin, and V. J. Mooney, "A Comparison of Five Different Multiprocessor SoC Bus Architectures", *The Euromicro Symposium on Digital Systems Design 2001*, pp. 202-209

[14] K. Sekar, K. Lahiri, A. Raghunathan, and S. Dey "FLEXBUS: A High-Performance System-on-Chip Communication Architecture with a Dynamically Configurable Topology", *DAC 2005*, pp. 571-574

[15] Sonics Inc. www.sonicsinc.com

[16] K. Srinivasan, K. S. Chatha, “A low complexity heuristic for design of custom network-on-chip architectures”, DATE 2006, pp.130-135

[17] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear Programming Based Technique for Synthesis of Network-on- Chip Architectures", *IEEE TVLSI,* April 2006*,* vol. 14, pp. 407-420

[18] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers", *DAC 2002*, pp. 524-529

[19] J. Yoo, S. Yoo, and K. Choi, "Communication Architecture Synthesis of Cascaded Bus Matrix", *ASPDAC 2007*, pp. 171-177

[20] Niagara 2 Opens the Floodgates, Microprocessor Report, Nov. 2006.