Layout Conscious Bus Architecture Synthesis for Deep Submicron Systems on Chip

Nattawut Thepayasuwan, Alex Doboli
Department of Electrical and Computer Engineering
State University of New York at Stony Brook, Stony Brook, NY, 11794-2350
{nattawut, adoboli}@ece.sunysb.edu

Abstract

System-level design has a disadvantage in not knowing important aspects about the final layout. This is critical for SoC, where uncertainties in communication delay by very deep submicron effects cannot be neglected. This paper presents a layout-aware bus architecture (BA) synthesis algorithm for designing the communication sub-system of an SoC. BA synthesis includes finding bus topology and routing individual buses, so that constraints like area, bus speed and length, are tackled at the physical level. The paper presents the BA automatically synthesized for a network processor and a JPEG SoC.

1. INTRODUCTION

Systems on chip (SoC) include multiple IP cores connected through complex data, address and control buses. Over the next 4-7 years, it is foreseen that the number of SoC cores will steadily increase, while clock frequencies will range around 10-15 GHz. For very deep submicron designs (VDSM), physical attributes such as interconnect parasitics, substrate coupling, and substrate noise significantly influence system performance, e.g., bus communication speed, system latency, power consumption, and signal integrity [12]. Hence, the usual design paradigm of abstracting physical details during system-level design is of limited use for SoC implemented in a VDSM process. To successfully address the myriad of emerging challenges, research must focus on incorporating relevant layout elements into system-level synthesis.

The motivating example illustrates the impact of layout on bus architecture design and communication speed allocation. Figure 1 shows a set of three Power PC 405GP cores, which exchange data as part of their functionality. The physical dimensions of cores are presented in the figure. Without considering layout, the system design step decides to allocate a single system bus of speed 266MHz for all core communications. However, given the physical dimensions of the cores, it is difficult to implement a bus of this speed. The same latency performance can be obtained with three buses of lower speed, as shown in the figure. The bus speeds of 133MHz, 133MHz and 33 MHz were found based on the physical locations of cores, and the RLC effects of the routed buses [12]. This example argues that the communication sub-system of an SoC, including the bus architecture and bus speed allocation, needs to be designed while contemplating layout feasibility criteria, e.g., achievable bus speeds in the presence of parasitic.

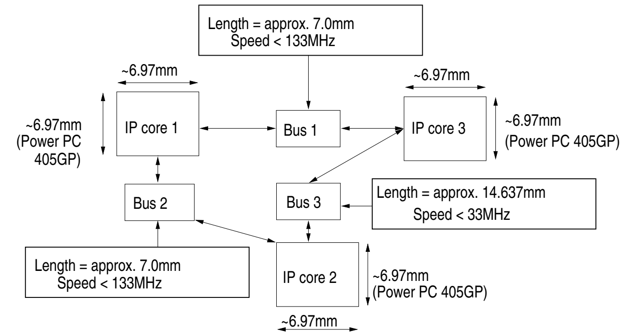


Figure 1: Impact of layout on BA and bus speed

This paper presents a layout parasitic aware bus architecture (BA) synthesis algorithm for designing the communication sub-system of an SoC implemented using a VDSM process. This is important, because it is difficult to postulate a unique BA as optimal for various applications and performance requirements. Instead, BA must be customized depending on the application specifics and design needs. BA synthesis includes finding the bus topology, and routing the individual buses. The paper presents BA for flat architectures, with and without redundant communication links. The BA synthesis algorithm first identifies the set of possible building blocks, and then assembles them into a topology using simulated annealing (SA) as an exploration engine. The cost function models bus length, bus topology complexity, potential for communication conflicts over time, and amount of unnecessary core connectivities in a topology (unnecessary connectivity needlessly increases bus length and decrease bus speed). We propose a special table structure (called BA synthesis table) and select-eliminate method to prune poor solutions. For example, buses with complex and redundant connectivity are early removed. This effectively reduces the exponential number of possible BA. The paper offers results of BA synthesis for a network processor using CoreConnect bus architecture [2], and for a JPEG SoC design.

Bus design and routing are critical problems for SoC design. Early work on bus and communication synthesis [4] [7] [10] [13] focuses on multiprocessor embedded systems. Research addresses interface design [4] [10], communication packeting, mapping and scheduling [13]. However, this work does not tackle details of hardware implementation and layout design. In many approaches, bus topology is assumed as given [7] [8]. Finding a very good bus architecture is a tedious task for SoC with large number of cores, and involves extensive designer knowledge. Dick *et al* [5] discuss bus synthesis as a part of the core-based synthesis tool MOCSYN. Floorplanning and placement are iteratively conducted inside a feedback loop to obtain point-to-point delay estimation. The delay information is used for recalculating link priorities of a core graph. Our method differs in that it does not require the number of buses as an input constraint, since it can be only determined after the optimized bus structure is available. More recently, Drinic *et al* [6] [9] present a method for SoC bus network design to maximize overall processing throughput. The design flow includes two steps: one produces the communication topology, and the other finds the core floorplanning. Our work partially resembles the methodology of Drinic *et al* [6]. However, it differs in considering more layout knowledge for BA synthesis, and suggesting a new pruning method to remove poor quality BA. This is important, because the BA with the highest speed critically depends on IP core placement. Also, the solution space is difficult to traverse without good pruning.

Section 2 discusses modeling for BA synthesis. Section 3 presents non-redundant flat BA synthesis, and Section 4 introduces redundant flat BA synthesis. Experimental results are given in Section 5. Finally, we offer conclusions.

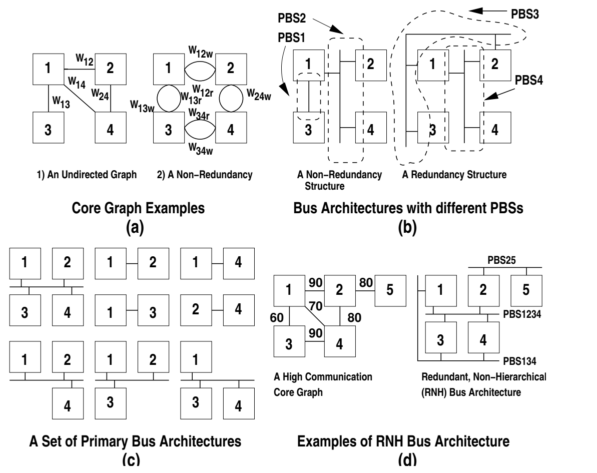
2. MODELING FOR BUS ARCHITECTURE SYNTHESIS

Definition: A *core graph* (CG) is a graph (V,E), where $v_i \in$ V is the i^{th} core in the architecture, and $e_{ij} \in$ E is the communication link between core i and core j. The weight w_{ij} is the communication load between core i and core j. The communication load (Cl) is the amount of data exchanged between two cores. The core size h_i x w_i is described along with node v_i.

The CG representation of a system architecture is used for BA synthesis. Similar concepts to CG are defined in [8] and [5]. CG has been illustrated in Figure 2(a.1). For simplicity of modeling, CG do not distinguish between unidirectional and bi-directional dataflow. Communication direction depends on whether an operation is a read or a write, and it isn't specified directly in a CG. However, the core graph can be modified in order to address the direction of data. Figure 2(a.2) presents the core graph for bi-directional communications.

Industrial bus standards can be expressed using the CG formalism. The superposition principle can be applied, if an industrial bus standard is used at the transaction level (post-topology synthesis). This can be done by classifying links according to their bandwidth (low, medium, and high), and generating CG for each category. This is sound because most industry standards for SoC, i.e., AMBA [1] and IBM CoreConnect [2], have different buses for data communication at different bandwidth. Bus architectures associated with different bandwidths are combined together to get the final solution.

Definition: Primary bus structure (PBS) is defined as a *potential* cluster of connected cores. PBS are the building blocks for bus architecture synthesis. A PBS is *valid*, if its node connectivities exist in the original CG. Otherwise, it is *invalid*.



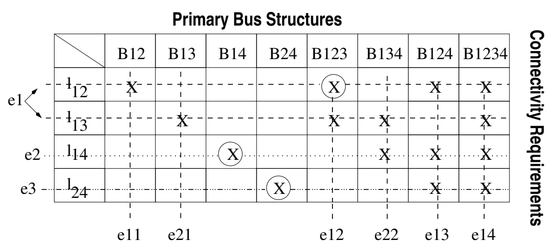
1
2
3
4
W12
W14
W13
W24
W12w
W12r
W13w
W13r
W24w
W34r
W34w
1) An Undirected Graph
2) A Non-Redundancy
Core Graph Examples
(a)
PBS1
PBS2
PBS3
PBS4
A Non-Redundancy Structure
A Redundancy Structure
Bus Architectures with different PBSs
(b)
A Set of Primary Bus Architectures
(c)
90
80
70
60
5
A High Communication Core Graph
PBS25
PBS1234
PBS134
Redundant, Non-Hierarchical (RNH) Bus Architecture
Examples of RNH Bus Architecture
(d)

Figure 2: Examples of Core Graph, PBS and Bus Architecture

Figures 2(b) and 2(c) show eight PBS for the CG in Figure 2(a.1). PBS in Figure 2(b) are valid. PBS are characterized by following physical and topological properties:

- *PBS utilization percentage*: Utilization is defined as the communication spread in a structure. For example, a PBS corresponds to two links in the CG, i.e., l_{12} and l_{13}. This PBS can also contain l_{23}, the connection between core 2 and core 3. There might, however, be no communication between these cores. Therefore the PBS under-uses its structure. We can also consider the unused element l_{23} as a redundant link of the PBS. The PBS utilization percentage, P_u, is defined as $P_u = \frac{2N_b}{n(n-1)} \times 100\%$, where N_b is number of links in a PBS, and n is number of associated cores in a PBS. The maximum PBS utilization occurs when all associated cores communicate between each other, and the PBS corresponds to a clique in the CG.
- *Communication conflict:* A PBS is implemented as a shared bus. For a static time scheduling of tasks, it is important to evaluate if there is a communication conflict in a PBS. Communication conflict of a PBS, $C_{conflict}$, is the amount of time overlap between communications mapped to the same link.
- *PBS bus length*: PBS bus length is a vital attribute for evaluating the bus speed in the presence of VDSM effects. Longer buses require more silicon area and additional circuitry like bus drivers [12]. Also, larger cross coupling and parasitic capacitances of longer buses increase interconnect delay [12]. Larger power dissipation for interconnect and drivers can be caused by longer buses. It is, however, difficult at the architecture level to estimate PBS bus length with high accuracy without contemplating the SoC layout. As explained in Section 3, hierarchical cluster growth placement is used for placing IP cores, and estimating PBS bus lengths.

Definition: *Bus architecture synthesis table* (BAST) presents the relationship between a set of PBS and the connectivity requirements in a CG. The number of rows is equal to the number of links in the CG. The number of columns is the


Primary Bus Structures
Connectivity Requirements
e1
e2
e3
e11
e21
e12
e22
e13
e14

	B12	B13	B14	B24	B123	B134	B124	B1234
l_{12}	X				X		X	X
l_{13}		X			X	X		X
l_{14}			X			X	X	X
l_{24}				X			X	X

```
L = number of communication link elements;
i = 1;
do
    select a PBS p to satisfy communication link i;
    eliminate all PBS which cover link i;
   for all links k satisfied by PBS p do
      eliminate all PBS which cover link k;
   end for;
   i = i + 1;
until   i > L
```

Figure 3: Select-eliminate algorithm

dimension of the PBS set. An entry in the table has value "X", if the PBS for the column includes the link element specific to the row. Figure 3 presents a bus architecture synthesis table.

3. NON-REDUNDANT FLAT BUS ARCHITECTURES

Depending on their structure, bus architectures (BA) can be either *non-redundant* or *redundant* structures, and *non-hierarchical* (flat) or *hierarchical*. The core connectivity offered by a non-redundant bus tightly matches the nature of the communication links in the CG of the application. Also, there is a single connection through a bus for any CG link. There are no core connections, which do not correspond to a link. Non-redundant structures have the benefits of using minimal resources for offering the needed core connectivity. They require no additional control circuitry (like for segmented buses), because a single channel is used to communicate between any two IP cores. The structure is simple, and thus simplifies the bus routing step. In contrast, redundant structures have superior concurrency, and thus decrease communication conflicts. However, expensive control logic is required to intelligently drive the shared bus. In flat BA there are no bus-to-bus communications, as buses link only cores. Hierarchical BA include bus-to-bus communications through bridge circuits [2]. Examples of non-redundant, non-hierarchical (NRNH) and redundant, non-hierarchical (RNH) BA are given in Figure 2(b). The left NRNH BA in Figure 2(b) uses the shared bus B_{124} to serve communications links l_{12},l_{14},l_{24}, and the point-to-point bus B_{13} for the link l_{13}. The right part of Figure 2(b) shows a RNH BA, in which two buses are used to implement link l_{12}.

We propose the select-eliminate (SE) algorithm to generate NRNH BA based on the satisfaction of the core connectivity requirements. SE algorithm is shown in Figure 3. To illustrate the algorithm, we use the simple BA synthesis table in Figure 3. For example, to satisfy the l_{12} connectivity, one of the four PBS $\{B_{12}, B_{123}, B_{124}, B_{1234}\}$ has to be chosen. Suppose PBS B_{123} is chosen, the rest of the candidates must be eliminated, so that there is no redundancy in the final structure. The horizontal dash line S_1 represents eliminated structures. Once a structure is eliminated, it automatically voids the whole column. Vertical dash lines e_{11},e_{12},e_{13},e_{14} show the eliminated column. PBS B_{123} does not satisfy only the l_{12} and l_{13} connectivity. Therefore, another horizontal line S_2 is created with vertical lines e_{21} and e_{22}. Connectivity l_{14} is considered next. There is a candidate left, namely, PBS B_{14}. Once PBS B_{14} is chosen, we have only one candidate, PBS B_{24}, left to satisfy l_{24} connectivity. The generated NRNH BA is composed of PBS B_{123}, PBS B_{14}, and PBS B_{24}. Circled structures in Figure 3 show the final BA.

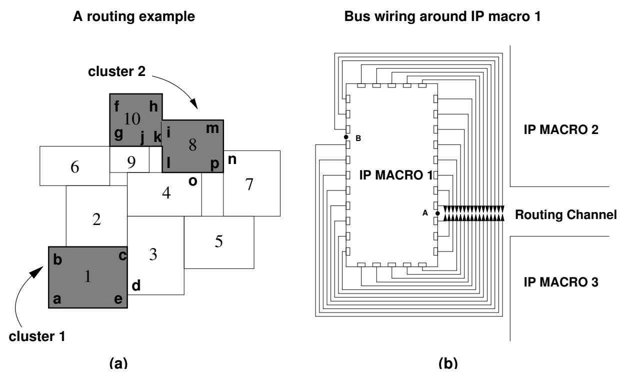


Figure 4: Routing example

The size of a synthesis table grows depending on the number of cores, and the amount of inter-core communication. If number of cores and interconnects between them is small, the SE algorithm contemplates all possible coverings of the CG links using the available PBS structures. However, if a system consists of more than 20 cores intensively tied up together, the exhaustive SE algorithm becomes infeasible. To allow SE algorithm to explore the PBS candidate space efficiently, we employed a simulated annealing algorithm to search different candidate PBSs while satisfying connectivity requirements. The algorithm randomly chooses a PBS from each requirement row, and combines it into a bus architecture. The total cost function that guides simulated annealing is given by the formula

$$\textit{Total cost} = w_l L_t + w_n N_b + w_c C_c + w_{ml} M_l - w_u C_u,$$

where L_t is the total bus length, N_b is the number of shared buses, C_c is the communication conflict, M_l is the maximum loss, and C_u is the total bus utilization percentage. w_l, w_n, w_c, w_{ml}, w_u are weight factors. The cost function parameters are defined in Section 2. Maximum loss reflects the maximum data loss in a bus architecture, if there is a conflict in a particular PBS. The algorithm objective is to minimize this cost function.

Buses are routed based on the IP core placement. IP cores are placed using a variant of the cluster growth placement algorithm [11]. The layout is described as an intersection graph [11] to diminish the size of the routing space. Depending on the IP vendor, a number of metal layers is available. To easy the analysis of VDSM effects, inter-macro communication uses only routing channels along the macro borders. To calculate the bus length of a PBS, inter-core routing path and bus wiring on a macro are taken into account. Inter-core routing algorithm starts by finding the shortest path of the link with the lowest communication load. This strategy is motivated by the placement method always placing two highly communicating cores close together. This implies that the longest path between two cores is at the link with the lowest communication load. Therefore, this link influences the total bus length of a PBS. We thus give it the highest priority to find the shortest path. Pin assignment along a core is considered for routing.

The shortest path between two cores is defined as the shortest path between any pair of intersection nodes around the border of two clusters. An intersection graph is illustrated in Figure 4(a). Cluster 1 contains core 1, with the interconnect set {a,b,c,d,e}, and cluster 2 contains cores 8 and 10 with the interconnect set {f,g,h,i,j,k,l,m,n,o,p}. The shortest path between clusters 1 and 2 is the shortest path from any two nodes of the two interconnect sets. If an intersection point of a core is chosen, all the wires which connect to every pin have to route around the core from the pin to a chosen intersection point. As seen in the bus routing diagram in Figure 4(b), the bus length is approximately half of the core perimeter. Therefore, if there is also another intersection point at point B, which is the longest distance from point A, all the wires have to route against the original direction. That is the total bus length is approximately a perimeter. This is considered to be the worst case, however, it is a good approximation if a macro has several intersection points required for inter-core routing. We use this worse case approximation in our bus length calculation. Lets l_{PBSi} be the bus length of the i^th PBS, l_{inter} be the inter-core bus length of the i^{th} PBS, and l_{macros} be the length of the bus-wiring around the cores associated with i_{th} PBS, the PBS bus length is defined as $l_{PBSi} = l_{inter} + l_{macros}$.

4. REDUNDANT FLAT BUS ARCHITECTURES

A redundant, non-hierarchical (RNH) BA allows more than one communication channel per connectivity, so that communication load can be shared across buses. This is to distribute communication load, and reduce the total amount of communication conflicts. Figure 2(d) represents a CG with high communication loads, and its possible RNH BA. The communication between core 3 and core 4 is satisfied either by PBS_{1234} or by PBS_{134}. Since RNH BA can have more than one channel to initiate communication, the synthesis algorithms must calculate the exact amount of communication occurred at each channel. For example, communication load of link l_{34} is 90. The synthesis algorithm may assign the load 63 (70%) to PBS_{134}, and 27 (30%) to PBS_{1234}. This is because PBS_{134} has more available bandwidth than PBS_{1234}.

RNH BA can be synthesized by extending the proposed BA synthesis table (BAST) concept. The synthesis algorithm is described in Figure 5. Using the method presented in Section 3, BA synthesis (using Select-Eliminate algorithm and SA) proceeds in line 2 after generating the initial BAST. Then, for the resulted BA (*PBS_set* is the collection of PBS in a BA), the links with maximum communication load in each conflicted PBS are propagated to a new BAST. These links are the connectivity requirements for the new BAST. Once the second set of PBSs is generated using the table, communication load is divided across the PBS, and the communication conflict for each PBS is re-calculated. If there are still PBS with communication conflicts, maximum communication load links for the PBS will be propagated to generate a new BAST. The algorithm stops when there is no communication conflicts in all PBS. The link, which has more than one PBS, equally distributes the communication load to the associated PBS. Such a load sharing is suitable for low communication CG or for CG with small number of cores. Then the total communication load per PBS is relatively low, and thus the instantaneous power consumption or switching noise is low. In contrast, a high communication CG, or a CG with high number of cores will potentially increase the number and size of the newly produced BAST. To limit the growth of the new BAST, any conflicted PBS in the previous table has to distribute a maximum load to the PBS in the current table, while not violating the conflict constraints. The BA synthesis method is efficient because the maximum communication loads aggressively decrease in every iteration. Therefore, the number and size of new BAST gradually decreases. Furthermore, load balancing can be applied to this iterative method as an additional constraint, so that communication loads of basis-elements are balanced.

```
1.  table = generate_initial_BAST(core_graph);
2.  PBS_set = synthesis(table);
3.  while  conflict(PBS_set) = true   do
4.  begin
5.    conflicted_PBS_set = conflict_extract(PBS_set);
6.    propagated_links = max_load(conflicted_PBS_set,threshold);
7.    new_table = build_new_BAST(propagated_link);
8.    new_PBS_set = synthesis(new_table);
9.    PBS_set = PBS_set + new_PBS_set;
10.   load_sharing(PBS_set);
11. end
```

Figure 5: RNH BA synthesis algorithm

Example: Figure 6(a) shows the initial BAST for the CG in Figure 2(d). Lets assume that the algorithm chooses PBS_{12}, PBS_{134}, and PBS_{245} as the first BA. The communication conflict of each PBS is calculated by summation of associated communication links in the PBS. PBS_{134} and PBS_{245} have communication conflicts. The algorithms therefore propagates the maximum load links (L_{24}, L_{25}, L_{34}) of these PBS to generate a new BAST, Table 6 (b). The selected PBS are excluded, as no replicated PBS are in the final BA. Once PBS_{2345} is selected, communication load of conflicted PBS are shared to the chosen PBS. In this case, L_{34} of PBS_{134} and PBS_{2345} and L_{24},L_{25} of PBS_{245} share communication load of 40, 40, and 45 each to the PBS_{2345}, respectively. Total communication loads and conflicts are then updated. Table 6(c) is generated, and this results in conflict free communication across the BA.

5. EXPERIMENTAL RESULTS

The first experiment presents the BA synthesis results for a network processor [3]. The processor receives Internet packets, reroutes them, and sends them out. A packet arrives through an Ethernet media access controller interface core (EMAC), and is sent to the multi-channel memory access layer core (MCMAL), a specialized DMA controller. MCMAL stores the packet in a buffer, and then transmits the buffer descriptor to the processor. The processor calculates the new destination address for the packet. MCMAL

	b12	b13	b14	b24	b25	b34	b123	b124	b134	b125	b234	b245	b1234	b1245	b2345	b1235	b12345
L12	X						X	X		X			X	X		X	X
L13		X					X		X				X			X	X
L14			X					X	X				X	X			X
L24				X				X			X	X	X	X	X		X
L25					X					X		X		X	X	X	X
L34						X			X		X		X		X		X

PBS	extracted links	total comm.load	conflict
b12	[L12]		
comm. load	90	90	0
b134	[L34,L14,L13]		
comm. load	90 70 60	220	1
b245	[L24,L25]		
comm. load	80 80	160	1

a) An Initial Synthesis Table with A Selected Bus Architecture

	b13	b14	b24	b25	b34	b123	b124	b125	b234	b1234	b1245	b2345	b1235	b12345
L24			X				X		X	X	X	X		X
L25				X				X			X	X	X	X
L34					X				X	X		X		X

PBS	extracted links	total comm.load	conflict
b12	[L12]		
comm. load	90	90	0
b134	[L34,L14,L13]		
comm. load	45 70 60	175	1
b245	[L24,L25]		
comm. load	40 40	80	0
b2345	[L24,L25,L34]		
comm. load	40 40 45	125	1

b) A Synthesis Table with propagated links

	b13	b14	b24	b25	b34	b123	b124	b125	b234	b1234	b1245	b1235	b12345
L14		X					X			X	X		X
L34					X				X	X			X

PBS	extracted links	total comm.load	conflict
b12	[L12]		
comm. load	90	90	0
b134	[L34,L14,L13]		
comm. load	15 35 30	80	0
b245	[L24,L25]		
comm. load	40 40	80	0
b2345	[L24,L25,L34]		
comm. load	20 40 15	75	0
b1234	[L12,L13,L14,L24,L34]		
comm. load	0 30 35 20 15	100	0

c) The Final Synthesis Table with propagated links

Figure 6: Example for RHN BA synthesis

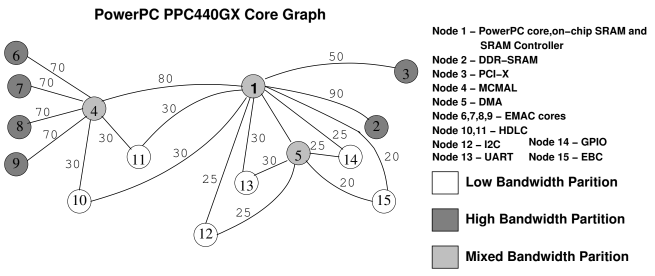


Figure 7: Core graph for the network processor

and one of the EMAC will send the packet out. Figure 7 shows the network processor CG.

For NRNH BA synthesis, the CG was divided into high speed and low speed parts according to their bandwidth requirements. Each part was separately synthesized. The cost function used following weights: $w_c = w_{ml} = 0.1$, and $w_l = w_u = w_n = 1.0$. These weights encouraged the using of shared PBS rather than point-to-point structures. The final architecture is shown at the top of Figure 8. There are three PBS in each of the high speed and low speed parts. The only conflicted PBS (with 360% conflict rate) is the one which connects three EMAC cores and PCI-X core to MCMAL core. Using this BA, the RNH BA was synthesized, and the result is shown at the bottom of Figure 8. Five maximum load links are propagated to a new BAST, and two additional PBS are added as a result. The maximum load is shared with the two new PBS. The amount of conflict for the original PBS is reduced to 161%. Then, four maximum

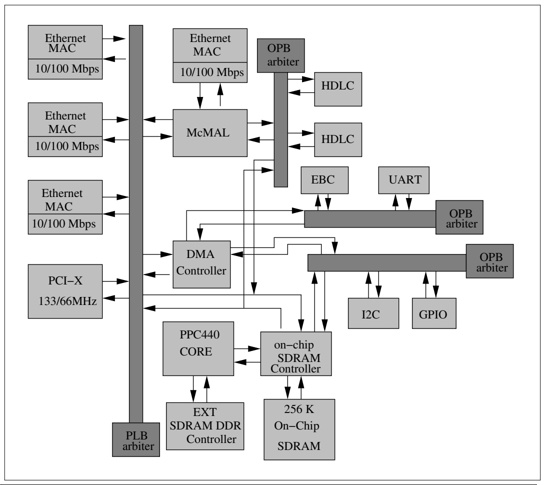
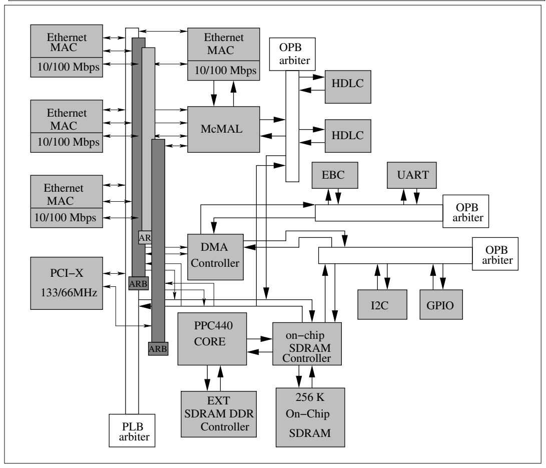



Figure 8: Bus architecture for network processor

load links are propagated to the third BAST, and another PBS is added. No conflicts result for this final BA. The RNH BA offers a higher communication concurrency at the transaction level. PCI-X core can send data packets to PPC core via the on-chip SRAM controller, while MCMAL core is receiving packets from EMACS core. However, RNH BA increases the total bus length, and leads to larger routing areas. Hence, NRNH BA can be used in resource constraint designs (i.e. smart sensors), while RNH BA are for high-performance applications.

BA synthesis was used to automatically produce optimized BA for the SoC of an JPEG image compression encoder. The task graph for the JPEG encoder included three identical and parallel sequences of tasks. Each sequence processes a different color of an image (RGB), and includes five consecutive tasks: preprocessing, FDCT, quantization, zigzag, and RLE & Huffman coding. The used architecture included three processor cores (a distinct core for each parallel sequence), an ASIC for the FDCT tasks, and memory modules for data communication. Each processor has its own local memory. Processors and ASIC communicate through shared memory. To decrease memory access times, the architecture used the interleaved memory blocks M4-M9. Figure 10 shows the resulting CG. The considered processing technology was 0.18μ TSMC. Microprocessor cores are

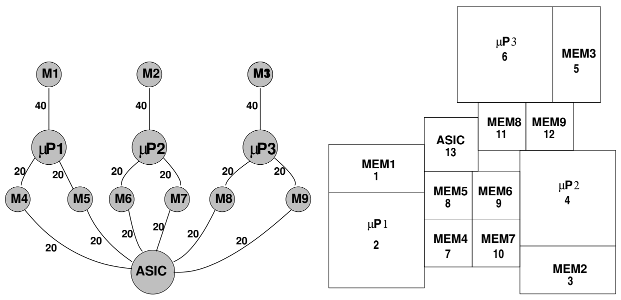


Figure 9: Core Graph for JPEG

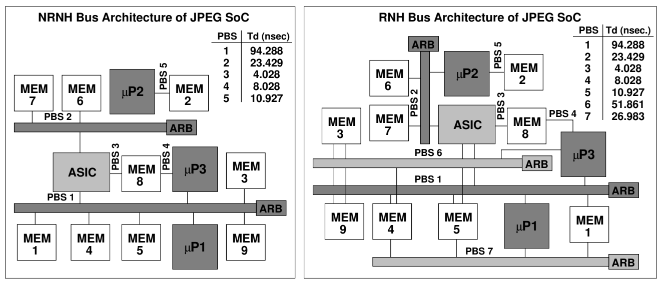
about $5 \times 5\ mm^2$, memory cores are about 25% and ASIC about 30% of the processor area.

Figure 10 shows BA synthesis results for the CG in Figure 9. The hierarchical cluster growth algorithm generated the core placement shown in Figure 9. The BA synthesis goal was to generate a fast architecture. BA complexity was not a major concern, because the number of IP cores is reasonably high. Also, for comparing NRNH BA and RNH BA, the communication conflict got a low priority ($w_c = w_{ml} = 0.1$), since RNH synthesis algorithm guarantees conflict free buses. Thus, the goal of BA synthesis was to minimize the total bus length ($w_l = 1.0$), while disregarding the number of buses and redundant structures in a BA ($w_n = w_u = 0.1$). After bus architecture generation, each of the buses was routed, and the resulting delays are indicated in Figure 10. Please note that the bus delay of $PBS5$ (10.927 nsec) is larger than that of $PBS3$ (4.028 nsec), but $PBS3$ carries a lower communication load (20). Even though it is against common sense, this result is correct because the bus length depends on the distances between cores and core perimeters (see Section 3). In this case, the core size dominated the total bus length and speed. For the RNH BA in Figure 10, two new PBS ($PBS5$ and $PBS6$) were introduced to share the communication load from the conflicted $PBS1$. As a result, the communication load of $PBS1$ decreased by 50%. The two additional PBS have less time delay, and thus enhance performance. However, the RNH BA increases routing area by 60%.

Note that the best NRNH BA is not perfectly regular, even though the CG is regular. Processor P1, and memory modules M1, M4 and M5 are linked through a shared bus, similar to processor P3 and memory blocks M3 and M9. This happens because the placements of these blocks is similar and close to each other. However, processor P2 and memories M2, M6 and M7 are linked through a different structure, which improves the bus speed for these blocks. This explains that optimized BA do not depend only on architectural elements (like the amount of exchanged data between cores), but on layout aspects, also.

6. CONCLUSIONS

This paper presents a BA synthesis method for designing the communication part of SoC fabricated in a VDSM process. BA synthesis creates customized BA depending on the application specifics and performance requirements. Synthesis includes finding the bus topology, allocating bus speeds, and routing individual buses. The algorithm identifies the



PBS	Td (nsec)
1	94.288
2	23.429
3	4.028
4	8.028
5	10.927

PBS	Td (nsec.)
1	94.288
2	23.429
3	4.028
4	8.028
5	10.927
6	51.861
7	26.983

Figure 10: Bus architecture for JPEG SoC

set of possible BA building blocks, and assembles them using simulated annealing as an exploration engine. The paper discusses generation of flat BA with and without redundant connections. The method employs a BA synthesis table and select-eliminate algorithm to prune poor solutions. This reduces the exponential number of potential solutions. The synthesis method is capable of exploring in short time the large solution space for SoC with many cores. Its layout awareness resulted in BA of higher speed.

7. ACKNOWLEDGEMENTS

This work was supported by a DAC graduate scholarship awarded to Nattawut Thepayasuwan, and IBM Faculty Partnership Award received by Dr. A. Doboli in 2001. The authors want to thank Dr. N. Dhanwada for his comments and suggestions.

8. REFERENCES

[1] "DesignWare AMBA On-Chip Bus Solution", *www.convergencepromotions.com/ARM/catalog/156.html.*

[2] IBM CoreConnect Bus Architecture White Paper, *//http:www-3.ibm.com/chips/products/coreconnect/index.html.*

[3] J. Darringer, R. Bergamaschi, S. Battacharyya, D. Brand, A. Herkersdorf, J. Morell, I. Nair, P. Sagmeister, Y. Shin, "Early Analysis Tools for System-on-a-Chip Design", *IBM J. of Research & Development*, Vol. 46, No. 6, 2002, pp. 691-707.

[4] J. M. Daveau, G. F. Marchioro, T. Ben Ismail, A. A. Jerraya, "Protocol Selection and Interface Generation for HW-SW Codesign", *IEEE Trans. on VLSI Systems*, Vol. 5, No. 1, pp. 136-144, March 1997.

[5] R. P. Dick, N. K. Jha, "MOCSYN: Multiobjective core-based single-chip system synthesis," *Design Automation and Test in Europe Conf.*, pp.263-270, 1999.

[6] M. Drinic, D. Kirovski, S. Meguerdichian, M. Potkonjak, "Latency-Guided On-Chip Bus Network Design", *Proc. of the ICCAD*, 2000, pp. 420-423.

[7] M. Gasteier, M. Glesner, "Bus-Based Communication Synthesis on System Level", *ACM Transactions on DAES*, Vol. 4, No. 1, January 1999, pp. 1-11.

[8] K. Lahiri, A. Raghunathan, S. Dey, "Efficient Exploration of the SoC Communication Architecture Design Space", *Proc. of the ICCAD*, 2000, pp. 424-430.

[9] S. Meguerdician, M. Drinic, D. Kirovski, "Latency-Driven Design of Multi-Purpose Systems-on-Chip", *Proc. of the DAC*, 2001.

[10] R. Ortega, G. Boriello, "Communication Synthesis for Distributed Embedded Systems", *Proc. of the ICCAD*, 1998, pp. 437-444.

[11] N. Sherwani, "Algorithms for VLSI Physical Design Automation", *Kluwer*, 1999.

[12] D. Sylvester, K. Keutzer, "A Global Wiring Paradigm for Deep Submicron Design", *IEEE Trans. on CADICS*, Vol. 19, No. 2, pp. 242-252, February 2000.

[13] T. Y. Yen, W. Wolf, "Communication Synthesis for Distributed Systems", *Proc. of the ICCAD*, 1995.