Mapping Streaming Applications on Multiprocessors with Statically Configured NoC

Usman Mazhar Mirza Flavius Gruian Krzysztof Kuchcinski
{Usman_Mazhar.Mirza, Flavius.Gruian, Krzysztof.Kuchcinski}@cs.lth.se
Department of Computer Science, Lund University
221 00 Lund, Sweden

ABSTRACT

This paper addresses design space exploration for streaming applications (such as MPEG) running on multi-processor platforms with guaranteed service interconnects. In particular, we solve mapping, path selection and router configuration problems. Given the complexity of these problems, state of the art approaches in this area largely rely on greedy heuristics, which do not guarantee optimality. Our approach is based on a constraint programming formulation that combines a number of steps, sequential in classical approaches. Thus, our method has the potential of finding optimal solutions with respect to resource usage under processing and bandwidth constraints. The experimental evaluation shows that our approach is capable of exploring a range of solutions while giving the designer the opportunity to emphasize the importance of various design metrics.

1. INTRODUCTION

Modern computing systems are becoming predominantly multi-core based and this trend is even more apparent in embedded systems, required to maintain certain throughput in video and audio processing, networking and various other signal processing applications. In order to meet the computational demands of these sort of streaming applications, multiprocessor systems-on-chip (MPSoC) are often employed. A large variety of MPSoCs is currently available or under development [1, 2, 6], comprising different interconnect structures and computing nodes.

Streaming applications are usually modelled using dataflow graphs, with their synchronous (SDF, [13]), cyclo-static (CSDF, [3]) and dynamic (DDF, [5]) flavours. This way of modelling may exhibit the parallelism inherent in such applications, composed of modules (actors) that can run in parallel, only interacting through FIFO buffers (channels). The difficulty of design space exploration today rests in mapping these sort of applications on the hardware platform, while keeping the throughput, latency, power and/or energy constrains. Depending on the type of processing nodes running the actors, and the type of hardware interconnect implementing the channels, the performance of an application may widely vary with the mapping. Usually the assignment of actors to nodes and of channels to hardware interconnect is carried out gradually, either via automatic tools or by experienced designers. However, it has been noted that a more unified automatic process is better suited to find solutions [9]. In this paper, we introduce a highly unified approach, that may provide optimal solutions, in contrast to the traditional greedy heuristics, while at the same time allowing for fine grain designer interaction. Furthermore, we show how the choice of cost function supports a more thorough exploration of the design space, targeting the metrics that matter for that specific design.

2. RELATED WORK

Design space exploration for streaming application on MPSoCs usually involves several decisions, such as architecture selection, mapping, and data routing. A method for mapping tasks (actors) onto network-on-chip (NoC) is proposed in [15] that reduces bandwidth requirements by splitting the traffic across multiple paths. Other mapping methods consider energy and performance [10], delay [23] or bandwidth [22], for example. Time-Division Multiplexing (TDM) based NoCs such as Æthereal NoC [8] and dAElite [17] provide contention free routing with guaranteed service. The mapping problem for architectures based on TDM NoC has been studied, for instance, in [9] and [19].

The approach presented in this paper employs dataflow as the computational model for the applications, constraint programming (CP) as modelling and optimization environment, and generic multi-processor platforms, accommodating homogeneous or heterogeneous processing units (PU), with NoC based on TDM [8, 17] as hardware support. We assume that readers have sufficient background knowledge of (C)SDF, DDF and its profiling [4], CAL [7] actors language, constraint programming, JaCoP CP environment [11, 12], network flow [18] and bin packing constraints [16] and TDM based circuit switched NoCs with slot tables.

3. OUR APPROACH

Given the target architecture and a streaming application described as a graph of actors, we are interested in finding (1) a mapping of actors to processing nodes, such that the processor loads remain under certain given limits, (2) a mapping of channels to NoC paths (path selection), such that the allocated bandwidth remains under a given maximum, (3) an exact TDM slot allocation table for each router in the network. Furthermore, the intention is to explore the design space in terms of number of processors, communication/processing cost, and slot table size. Selecting the appropriate importance for each of these metrics, which translates in tuning a cost function, is the method used by the designer for exploring the design space.

The novelty of our approach rests in unifying the mapping the actors to processors with path selection followed by slot allocation. Furthermore, the designer has more possibilities to influence the results by formulating additional constraints. In our approach network flow model formulated with constraints is used for problem specification as illustrated in Fig.1. During *Mapping and Routing* step, we can achieve several objectives (explained in section 4.1). In *NoC Configuration* step, we are minimizing slot table size.

The inputs to our network flow model are target architecture parameters, application parameters, application profile data, and performance constraints. The target architecture parameters required by our model include the available com-

putational load for each processor (clock cycles per second), network topology, bandwidth for each link in the network. Our model supports all SDF, CSDF and DDF dataflows and as input it expects a dataflow graph (DFG) of the application. Finally, performance constraints (e.g. throughput) are the requirements that should be reflected in the solution, the application running on the target architecture.

The output of our design flow consists of mapping of actors to the processors, the paths assigned to each channel, along with the exact slot allocation, as well as the slot table configuration for each router in the network.

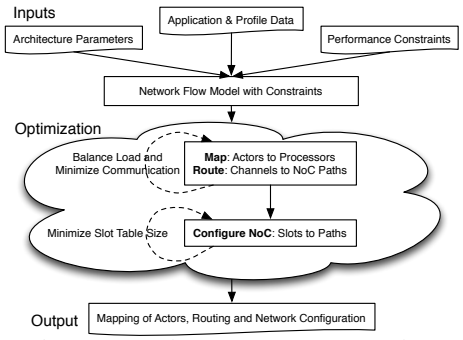

Figure 1: Unified approach for solving the problem of mapping, routing and slot allocation

4. MODELLING IN DETAIL

Application characteristics, processing and bandwidth requirements, are arrived at slightly differently for (C)SDF or DDF models. The available information from DFGs of (C)SDF and DDF is the total number of clock cycles, cc_a, required by each actor to execute on PU, the size of tokens, sz_c, in bits for each channel, the repetition vector, rv_a, for each actor (only for (C)SDF), and number of tokens, $nt_{a,c}$, produced by each actor on their respective channels for a given number of iterations of DFG (only for DDF). Available numbers for (C)SDF are only for one iteration of DFG.

Using the information above and performance constraints such as throughput (frames-per-second (FPS) for video applications), we can derive the computing requirements, rc_a, for each actor and bandwidth requirements, $\overline{bw_c}$, for each channel, for both static and dynamic DFGs using the following formulation. If a is the source actor for channel c,

$$\text{SDF}: \quad rc_a = cc_a \cdot rv_a \cdot FPS,\ \overline{bw_c} = sz_c \cdot rv_a \cdot FPS \quad (1)$$

$$\text{DDF}: \quad rc_a = cc_a \cdot FPS,\ \overline{bw_c} = sz_c \cdot nt_{a,c} \cdot FPS \quad (2)$$

4.1 Mapping and Path Selection

We formulate mapping of actors to processors and path selection for communication between source actor a_i and destination actor a_j as a network flow problem. One channel c in DFG between two actors is represented by one network flow as shown in Fig.2. The terms channels/flows are used interchangeably in this paper. Let N is the number of processing nodes in the target architecture ($N = 9$ here).

The processing nodes (denoted as boxes) are connected using solid directed arcs to form a mesh interconnect but our approach is not limited to mesh. The additional N arcs from source to processors $p_i \in P$ and from processors to destination model the assignment of these actors to any of the processors in P. These arcs have a "zero" cost, meaning that they do not contribute to the total cost of network flow. The many dotted arcs and one solid arc between the actor and processors in Fig.2 show that the actor can be assigned to exactly one processor.

In CP problem formulation of mapping to processors and path selection, each channel of DFG results in one network flow constraint. Each network flow constraint computes actors mapping to processors and a communication cost, $cost_c$, for each channel. The constraint $\sum_c cost_c$ computes the total communication cost, $comm\_cost$, for all the channels.

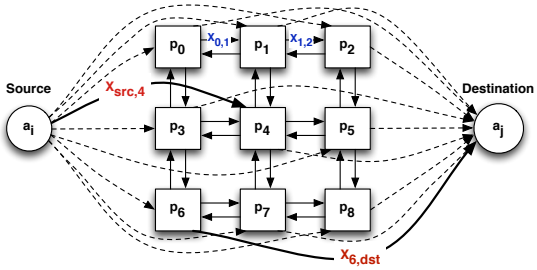

Figure 2: A network flow formulation of processor mapping and path selection for a communication between actors a_i and a_j. For simplicity, routers are not depicted.

Each network flow constraint has separate finite-domain-variables (FDVs) x^c_{ij}, $\forall(i,j) \in E$, with domain $0..\kappa_{bw}$, where E is set of directed arcs [18], i and j represent the source and destination nodes of arcs respectively (see Fig.2), c is the flow identifier and κ_{bw} is the link capacity for each link in NoC. These variables, for each flow, define bandwidth assigned to link from processor i to processor j. The variables for source to processor arcs define the bandwidth requirement ($\overline{bw_c}$) of flows. The distinct feature of our network flow formulation is the possibility of splitting of a single flow through several links. Additional constraints are imposed, to model the mapping of an actor to just one processor, to avoid splitting the flow from processors p_i to actors.

Each actor in the DFG has some computing requirements in terms of clock cycles per second (cc/s) as discussed earlier. When we try to map some actors to a particular processor, the cumulative computing requirements of those actors should not exceed the computational capacity, also in cc/s, of that processor. This is modelled using the bin packing constraint [16]. In our case, the number of bins are equal to number of available processors N.

While allocating the bandwidths to the flows on NoC, we need to make sure that accumulated allocated bandwidth does not exceed the capacity (κ_{bw}) of the links. We assume that all the links have the same capacity. This condition is enforced using the constraint defined below

$$\forall(i,\ j) \in E : \sum_c x^c_{ij} \leq \kappa_{bw} \quad (3)$$

$x^c_{ij} < \overline{bw_c}$ when the flow is splitted and $x^c_{ij} = \overline{bw_c}$ when the flow is not splitted while allocating the bandwidth.

We defined our multi-objective cost function for our unified approach using the constraint in (4). The first term in (4) is the number of used processors, the second represents the maximum load on a processor combined with the outward communication flow generated by that processor, while the third term is the total communication cost.

$$cost = c_0 \cdot no\_processors + c_1 \cdot \max_{p_i \in P}(w_0 \cdot load_{p_i} + w_1 \cdot out\_flow_{p_i}) + c_2 \cdot comm\_cost \quad (4)$$

The intuition behind our cost function is to achieve different

objectives by adjusting the weights for design space exploration. It is possible, for instance, to minimize the number of used processors, better allocate the actors to processors by considering the outwards communication as well as balance the load on processors, minimize the total communication cost and different combinations of previous three.

4.2 Slot Allocation

NoCs such as Æthereal [8] and dAElite [17] provide contention free routing by regulating the injection of flow-control-digits (flits) using a slot based TDM table, such that no two flits arrive at a link at the same time (See example in Fig.3). Each slot in a slot table corresponds to a fixed size flit, assumed to be 32 or 96 bits, in this paper. See aforementioned TDM based NoCs for more understanding of the problem.

To model the slot allocation problem we make several assumption similar to those in [14]. We assume, as before, that each link has a defined link capacity κ_{bw} and each flow c has a minimum bandwidth requirement ($\overline{bw_c}$).In addition we introduce an actual implementation bandwidth (bw_c). A necessary condition that implementation bandwidth cannot be lower than the required bandwidth is ensured by a constraint. Link capacity constraint is the same as in (3). We are skipping mathematical formulations of the constraints here due to the lack of space. The variable S defines the

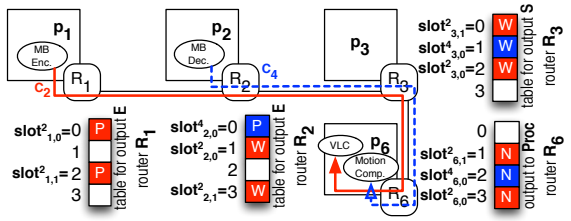

Figure 3: An example for Contention-free Routing.

size (or period) of the slot table and has the domain $1..M$. We also define a slot variable for each flow c on each link l for slot s, called $slot^c_{l,s}$, as in Fig.3. A constraint is enforced to fulfill the condition that slot variables must have different values of assigned slots for different flows sharing the same link. Another constraint also ensure the routing principle that flits (in a slot table) are forwarded to the immediately following slot by each router. The number of assigned slots for each flow must also follow the bandwidth requirement, means assigned slots to flows must be according to $\overline{bw_c}$.

A solution to the above constraints provides a valid slot allocation. Moreover, minimizing S generates a valid solution with minimum slot table size for a given bandwidth requirements and mapped traffic into the network.

5. EXPERIMENTAL EVALUATION

We carried out experiments for both SDF and DDF models on a system with 2GHz Intel Core i7 processor, with 8GB RAM, running Mac OS X 10.7. For CP problem formulation we used Java 6 and JaCoP-3.2 constraint solver [12]. The solutions were found in all cases in less than one minute. In most of the cases these were the optimal solutions. In rest of the cases we abandoned the search after five minutes.

Here we assumed a low performance architecture with 100MHz PU and NoC frequency, 3x3 mesh NoC and 32 bits flit size and a high performance architecture with 500MHz PU and NoC frequency, 3x3 mesh NoC and 96 bits flit size.

5.1 DDF: MPEG4SP

The first set of experiments were carried out using the reference RVC-CAL description for MPEG4 Simple Profile decoder [21]. This is a DDF model, with 13 actors and 29 channels, working on a QCIF picture format. We assumed low performance target architecture, described earlier.

The workload for actors and the required bandwidth for channels is based on profile information (rather than static analysis, used for (C)SDF). We used simulation and causation trace [4] to get average number of clock cycles for each actor and bandwidth requirements for channels.

Table 1: MPEG4 QCIF mapped on a 3x3 mesh NoC

Cost Func.	CPU	Load(%)		Link	Util.(%)		Throu.	Slot
$c_0c_1w_0w_1c_2$	#	μ	σ	#	μ	σ	(FPS)	#
01111	6	13.03	4.64	9	0.13	0.09	81.48	9
00xx1	1	78.16	0.00	0	–	–	65.33	–
10xx0	1				”			
10xx1	1				”			
01100	8	9.77	2.93	18	0.27	0.23	100.06	8
01110	9	8.68	1.78	20	0.11	0.10	108.29	7
01101	4[a]	19.54	8.56	6	0.10	0.11	79.22	5
11100	2	39.08	0.0035	7	0.51	0.19	57.53	15

[a] Timed out.

Table 1 reflects the exploratory capability of our approach, by varying the parameters of the cost function (the first column) as defined in equation (4). The next six columns list the number of processors, average load and its standard deviation of the solution, and similarly for the mesh links. The throughput in FPS and the size of the slot table for the TDM network are listed in the last two columns.

Note how choosing different parameters for the cost function can lead to different solutions, depending on the optimization criteria relevant for each specific design. For instance, the first row ignores the number of processors, but balances processor and communication load, while minimizing the communication as well. Compare this to the last row, where the number of processors is minimized, but the communication cost ignored, resulting in higher link utilization, but fewer processors. Notably, rows two through four show that minimizing only the number of processors and/or the communication results in a single processor solution, which is still feasible, due to the small picture size used for this experiment. However, for a four times larger CIF, the single processor solution may not achieve the requirement of 25FPS, pointing to a solution that achieves at least 100FPS for QCIF (rows five and six).

5.2 SDF: some SDF^3 examples

The second set of experiments were carried out for SDF models of H263Decoder, H263Encoder and MP3Decoder taken from [20]. For SDF models, the workload for actors and the required bandwidth for channels is obtained using static analysis as described earlier in section 4. The results of these experiments are listed in Table 2. We assumed low performance architecture, described earlier, and required throughput of 15CIF/QCIF FPS for H263Encoder/Decoder and 25FPS for MP3Decoder.

All the actors, for H263Decoder, were able to map on a single processor and resulted in zero communication cost, when cost function with parameters 00xx1 was used to minimize the communication cost. Cost function with parameters 01100 do not consider communication at all and results in balancing the load on processor and higher link utilization, see experiment for H263Encoder. For H263Encoder,

Table 2: Results for different streaming applications

		Cost Func.	CPU	Load(%)				Link	Util.	Throu.	Slot
		$c_0c_1w_0w_1c_2$	#	P_0	P_1	P_2	*Avg.*	#	(%)	(FPS)	#
H263Decoder	**QCIF**	01111	2	4.89	4.98	–	4.93	2	0.07	257.39	1
	CIF	01111	2	19.54	19.92	–	19.73	2	0.29	64.35	1
	4 Actors	00xx1	1	39.47	–	–	39.47	0	0.00	38.01	–
	6 Channels	01100	2	19.54	19.92	–	19.73	2	0.29	64.35	1
		01110	2	19.54	19.92	–	19.73	2	0.29	64.35	1
		01101	2	19.54	19.92	–	19.73	2	0.29	64.35	1
		10xx0	1	39.47	–	–	39.47	0	0.00	38.01	–
		10xx1	1	39.47	–	–	39.47	0	0.00	38.01	–
		11100	2	19.54	19.92	–	19.73	2	0.29	64.35	1
H263Encoder	**QCIF**	01111	2	15.21	12.88	–	14.05	2	0.07	78.98	1
	CIF	01111	2	60.84	51.51	–	56.18	2	0.29	19.74	1
	5 Actors	00xx1	2	60.84	51.51	–	56.18	2	0.29	19.74	1
	7 Channels	01100	3	23.63	38.77	49.95	37.45	4	0.43	19.45	2
		01110	3	23.63	37.21	51.51	37.45	4	0.29	19.60	1
		01101	3	23.63	37.21	51.51	37.45	4	0.29	19.60	1
		10xx0	2	75.14	37.21	–	56.18	2	0.57	19.67	1
		10xx1	2	60.84	51.51	–	56.18	2	0.29	19.74	1
		11100	2	52.19	60.16	–	56.18	2	0.86	19.38	2
MP3	14 Actors	01111	3	18.50	93.25	93.25	68.33	2	0.007	12.21	1
	17 Channels	11100	3	18.50	93.25	93.25	68.33	2	0.01	12.21	2

cost function with parameters 10xx0 resulted in minimum number of used processors without balanced load. We can balance the load on processors as well as minimize the number of processors using parameters 11100 (Table 2 row 18).

All experiments for MP3Decoder ended up with minimum three processors and same distribution of actors on the processors. This is because two of the actors have loads such that each of them occupies almost 94% of the processor. Whereas, remaining 12 actors can fill just 19% of the processor. Notably, the lower throughput of the application is because of the way the application is designed.

We also mapped and routed a group of applications (H263-Encoder, H263Decoder, MP3Decoder) at the same time (see Table 3). We assumed high performance architecture, described earlier, and required throughput of 15QCIF FPS for H263Encoder/Decoder and 100FPS for MP3Decoder. To estimate throughput, we divided the processor time proportionally to the load generated by actors assigned to these processors and then used the related time slots to compute throughput of the applications. The estimated throughput, in last row, for H263Decoder is lower than the rest of the experiments, because for load balance, we ended up with a mapping where proportion of the processor's time assigned to H263Decoder was less than 3%. We can see from the

Table 3: Results for the combined streaming applications

Cost Func.	CPU	Load(%)				Link	Util.	Throu. (FPS)			Slot
$c_0c_1w_0w_1c_2$	#	P_0	P_1	P_2	*Avg.*	#	(%)	enc.	dec.	mp3	#
01111	3	22.39	74.60	74.60	57.20	2	0.0019	66.99	67.20	58.38	1
00xx1	2[a]	96.99	74.60	–	85.79	1	0.0019	15.46	15.51	58.31	1
01100	3	22.39	74.60	74.60	57.20	2	0.0029	66.99	67.20	58.38	2
01110	3	22.39	74.60	74.60	57.20	2	0.0029	66.99	67.20	58.38	2
01101	3[a]	22.39	74.60	74.60	57.20	2	0.0019	66.99	67.20	58.38	1
10xx0	2	96.99	74.60	–	85.79	1	0.0019	15.46	15.51	58.31	1
10xx1	2[a]	96.99	74.60	–	85.79	1	0.0019	15.46	15.51	58.31	1
11100	2[a]	85.79	85.79	–	85.79	2	0.0105	17.49	9.67	58.37	5

[a]Timed out.

results presented that selecting different parameters for our cost function gives designers an opportunity to achieve and explore different objectives.

6. CONCLUSIONS

In this paper, we have presented a unified approach based on constraint programming for design space exploration of streaming applications running on multiprocessors with guaranteed service interconnects. The flexibility of our formulation with CP makes it also possible to easily modify the model and the related constraints. This enables the exploration of more design parameters by allowing the designer to directly affect mapping and routing decisions. The experimental evaluation shows that our approach can find optimal solutions, for a number of cost function configurations, therefore making it worth to investigate further.

As a continuation of this work, we plan to expand our method to include memory constraints, heterogeneous processing units, and different network topologies in the target architecture. We also plan to investigate ways of adding throughput constraints in our optimization loop in order to guarantee the throughput requirements of the applications.

7. REFERENCES

[1] Adapteva Inc., `http://www.adapteva.com/`. *Epiphany Architecture Reference (G3)*, revision 3.12.12.18 edition.

[2] S. Bell et al. Tile64 - Processor: A 64-Core SoC with Mesh Interconnect. In *Solid-State Circuits Conference*, 2008.

[3] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete. Cyclo-static data flow. In *International Conference on Acoustics, Speech, and Signal Processing*, 1995.

[4] S. C. Brunet, M. Mattavelli, and J. W. Janneck. Profiling of dataflow programs using post mortem causation traces. In *IEEE Workshop on Signal Processing Systems*, 2012.

[5] J. Buck and E. Lee. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1993.

[6] M. Butts. Synchronization through communication in a massively parallel processor array. *Micro, IEEE*, 27(5):32–40, 2007.

[7] J. Eker and J. W. Janneck. CAL language report specification of the CAL actor language. Technical report, University of California, Berkeley, 2003.

[8] K. Goossens, J. Dielissen, and A. Rădulescu. Æthereal network on chip: concepts, architectures, and implementations. *Design Test of Computers, IEEE*, 22(5):414 – 421, 2005.

[9] A. Hansson, K. Goossens, and A. Rădulescu. A unified approach to mapping and routing on a network-on-chip for both best-effort and guaranteed service traffic. *VLSI Design*, 2007.

[10] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Asia and South Pacific Design Automation Conference*, 2003.

[11] K. Kuchcinski. Constraints-driven scheduling and resource assignment. *ACM Transactions on Design Automation of Electronic Systems*, 8(3):355–383, 2003.

[12] K. Kuchcinski and R. Szymanek. JaCoP Library. User's Guide. `http://www.jacop.eu`, 2013.

[13] E. Lee and D. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, pages 1235 – 1245, 1987.

[14] Z. Lu and A. Jantsch. Slot allocation using logical networks for TDM virtual-circuit configuration for network-on-chip. In G. G. E. Gielen, editor, *ICCAD*, pages 18–25, 2007.

[15] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In *Proc. of the conference on Design, automation and test in Europe*, 2004.

[16] P. Shaw. A constraint for bin packing. In M. Wallace, editor, *Principles and Practice of Constraint Programming*, pages 648–662. Springer Berlin, 2004.

[17] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens. A TDM NoC supporting QoS, multicast, and fast connection set-up. In *Proceedings of the Design, Automation and Test in Europe Conference*, 2012.

[18] R. Steiger. Network flow constraint. Semester Project, Laboratoire d'Intelligence Artificielle École Politechniqué Fédérale de Lausanne, 2010.

[19] S. Stuijk, T. Basten, M. Geilen, and H. Corporaal. Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs. In *Design Automation Conference*, 2007.

[20] S. Stuijk, M. Geilen, and T. Basten. SDF^3: SDF for free. In *Sixth International Conference on Application of Concurrency to System Design*, 2006.

[21] `http://orcc.sourceforge.net`. Open RVC-CAL compiler.

[22] J. Wang, Y. Li, S. Chai, and Q. Peng. Bandwidth-aware application mapping for NoC-based MPSoCs. *Journal of Computational Information Systems*, 7(1):152–159, 2011.

[23] W. Zhou, Y. Zhang, and Z. Mao. An application specific NoC mapping for optimized delay. In *International Conference on Design and Test of Integrated Systems in Nanoscale Technology*, 2006.