DATA REUSE DRIVEN MEMORY AND NETWORK-ON-CHIP CO-SYNTHESIS*

Ilya Issenin and Nikil Dutt
University of California, Irvine, CA 92697

Abstract: NoCs present a possible communication infrastructure solution to deal with increased design complexity and shrinking time-to-market. The communication infrastructure is a significant source of energy consumption and many attempts at energy efficient NoC synthesis have been proposed. However, in addition to the communication subsystem, the memory subsystem is an important contributor to chip energy consumption. These two subsystems are not independent, and a design with the lowest memory power consumption may not have the lowest overall power consumption. In this paper we propose to exploit a data reuse analysis approach for co-synthesis of memory and NoC communication architectures. We present a co-synthesis heuristic targeting NoCs, such as Æthereal, with mesh topology. We show that our data reuse analysis based synthesis reduces communication energy alone by 31% on average as well as memory and communication energy by 44% on average in comparison with the similar approaches that do not employ data reuse analysis. We also show that our memory/NoC co-synthesis heuristic further reduces communication energy by up to 38% in comparison with a computationally less expensive traditional two-step synthesis approach, where the memory subsystem is synthesized before the synthesis of NoC. To the best of our knowledge, this is the first work to investigate the influence of memory subsystem synthesis on NoC power consumption and to present a memory and NoC co-synthesis heuristic.

Key words: Network-on-Chip synthesis, memory synthesis, data reuse analysis

1. INTRODUCTION

Multiprocessor Systems-on-Chips (MPSoCs) are becoming a popular solution to meet the growing processing demands of embedded applications.

* This work was partially supported by NSF grant CCR-0203813.

One important aspect of SoC design is the communication subsystem. There are several conflicting requirements for this subsystem: it should be low power, scale well with increasing SoC size and allow easy modifications of the chip when part of the chip has to be redesigned. Due to good scalability, Network-on-Chip (NoC) is one of the promising communication architectures that is well suited for larger SoCs.

Memory and communication subsystem are both important contributors to SoC energy consumption, which is a scarce resource in typical application of embedded SoCs. While there are a number of approaches for minimizing energy consumption in memories and NoCs, traditionally such approaches are applied independently, typically with the memory subsystem designed first, followed by NoC synthesis. This may result in a system that doesn't have the lowest possible overall power consumption. Our work focuses on co-synthesis of NoC together with a memory architecture. We obtain high customization of memory subsystem by employing our data reuse analysis technique DRDM [10]. To the best of our knowledge, this is the first work to explore the influence of memory subsystem configuration on NoC power consumption and to propose a co-synthesis heuristic aimed at minimizing total power consumption.

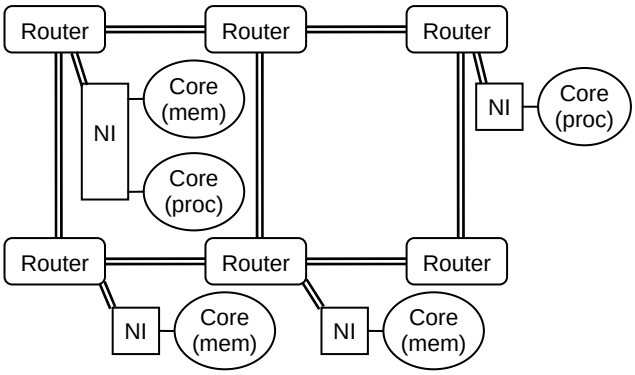

Figure 1. Example of 3x2 mesh-type NoC communication architecture

To illustrate our approach we target the Æthereal NoC [3][5], although our approach is applicable to other NoCs as well. Æthereal is a synchronous NoC that provides guaranteed throughput, fixed latency communication between cores. Figure 1 shows a typical NoC architecture with 3x2 mesh topology. The system consists of communicating cores and routers with network interfaces (NIs) connected by links. The routers provide packet routing according to a predetermined static schedule. Network interfaces provide conversion of packed-based NoC communication to protocols that cores use, as well as clock domains conversion [3]. The cores represent

processors, memories with or without DMA controllers, or I/O devices. For instance, the NoC depicted in Figure 1 has two cores that are processors and three cores that contain memory.

The problem of NoC co-synthesis is the task of customizing the memory configuration together with determining which network interface each core is connected to, and identifying the routing of the packets between communicating network interfaces.

Traditionally, communication requirements between cores (which are used as input information for NoC synthesis) are represented by communication graph, which consists of nodes representing communicating cores and edges that specify the direction and amount of communication between the cores. This is also called communication throughput graph in [17], core graph in [16] or application characterization graph in [9]. Such a representation implies that the number of cores (processors, memories, peripheral devices) and communication requirements are fixed before the NoC synthesis.

In contrast, we do not fix the memory architecture before the NoC synthesis. In our approach we use a *data reuse graph* that concisely captures memory reuse patterns that can be exploited to perform effective memory customizations. It can be also viewed as a set of different communication graphs, which differ in the number of additional buffers that exploit data reuse (temporal locality) in processor accesses. The *data reuse graph* can be obtained by performing data reuse analysis [10].

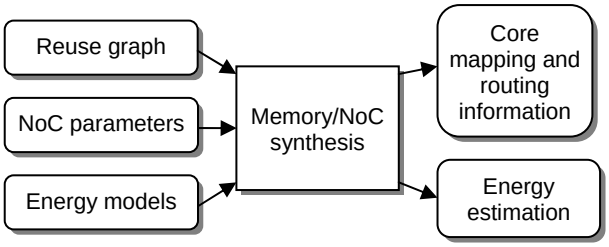

Figure 2. Synthesis framework

The design flow of our approach is depicted in Figure 2. It takes application reuse graph (obtained with our technique described in [10]), NoC parameters (such as mesh size) and energy models as input. After performing synthesis with one of the algorithms described later in Section 4, the framework provide synthesis information as well as estimation of the energy consumption of the synthesized configuration.

In this work we present a memory/NoC co-synthesis power-aware heuristic that uses the data reuse graph as input and compare it with a two-step heuristic, where the memory configuration is decided before the NoC

synthesis. We show that our proposed co-synthesis heuristic outperforms the two-step synthesis approach by achieving reductions of up to 38% in communication (NoC) energy and by up to 26% in combined communication and memory energy.

2. RELATED WORK

The potential of using NoCs in MPSoCs as a future on-chip interconnects has been recognized in [2][6][8]. Since then a large amount of research was done investigating different aspects of NoC, such as NoC architecture, routing, interconnect technology, development support, etc.

In the area of NoC synthesis, a number of researchers have proposed core mapping and routing techniques for NoC architectures. Murali et al. [16] present two techniques aimed at average communication delay reduction in the presence of bandwidth constraints. Srinivasan et al. [19] proposed technique that handles both bandwidth and latency constraints. Hu et al. [9] perform energy aware mapping with the bandwidth constraints. Manolache et al. [13] add redundancy in message transmission as a way to address the problem of transient link failures. Guz et al. [7] solve the problem of allocating link capacities in NoC with non-uniform link capacities. Srinivasan et al. [20] present layout aware mapping and routing approach.

In [19] it was shown that the NMAP algorithm from [16] provides solutions within 10% from the optimal ones obtained by MILP, and it is on par (or better on average) than the solutions obtained by algorithm from [19] if no latency constraints (only bandwidth constraints) are taken into account. We use a modified NMAP algorithm in our technique to perform NoC synthesis.

All of the abovementioned techniques work with a given communication graph and do not perform any memory customizations during the synthesis. The advantages of using memory/communication co-synthesis was shown in [11][17] for bus-based communication architectures. To the best of our knowledge, our approach is the first work that applies memory/communication co-synthesis to MPSoCs which use NoC interconnect architecture.

3. MEMORY MODEL AND DATA REUSE GRAPH

In our approach we employ *data reuse graphs* produced by a data reuse analysis technique [10]. They are built by analyzing the application's memory request patterns. Reuse graphs represent a hierarchy of buffers, with

each buffer holding the data that is used several times by the processors or by the buffers which are lower in the memory hierarchy. Each reuse graph node (or buffer) can be private, i.e., providing data to only one processor, or shared, holding data that are used by several processors. Reuse graph edges represent the data flow, i.e., shows the source or destination of the data that is kept in the buffer. An approach to modify the application code to include the necessary DMA transfers between the buffers and provide synchronization between updates of the shared buffers and processors is also included in [10].

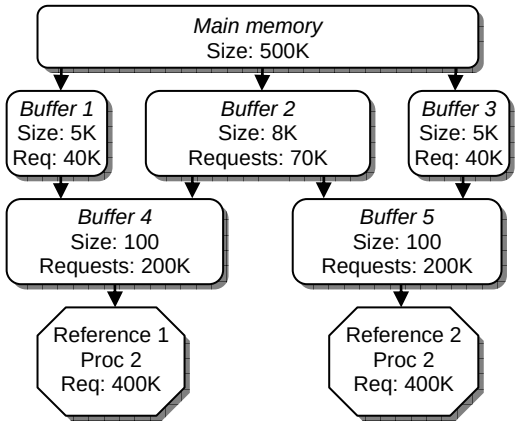

Figure 3. Example of a multiprocessor data reuse graph

An example of data reuse graph is shown in Figure 3. It has several private buffers (1, 3-5) and one shared buffer (2). The data reuse analysis tool provides the sizes of the buffers and the number of data requests to a memory of higher level in memory hierarchy.

Each reuse graph buffer may be mapped to a physical memory and implemented as a NoC core (possibly together with DMA engine). Implementing a buffer may reduce the overall energy consumption since it can be placed near data consuming core in a small and low energy consuming memory, thus eliminating repeated memory accesses to more energy expensive main memory. However, in other cases, it may lead to an increase in energy consumption due to the overheads of copying data from one memory to another and additional power consumption in network interfaces and routers. Our proposed algorithm performs selection of which buffers to implement and which network interfaces and routers the memories should be attached to minimize the energy consumption.

4. APPROACH

4.1 Architectural Model

The problem of memory and NoC synthesis is the task of mapping arrays (logical buffers) to physical memories, mapping these physical memories along with other cores to routers and determining static routes for the data between cores, while trying to minimize the total energy consumption.

Recollect that in our approach the scratch pad memories which hold circular buffers from the reuse graph are connected to network interfaces as cores (Figure 1). We also make an assumption that all processing cores have their own small private memories, which hold small arrays, local variables and are also used for allocating the stack. NoC (and data reuse buffers described above) are used only for transferring the data shared between several processors or for accessing large arrays of multimedia data (such as frame or image) that cannot fit into small local memories. Our analysis shows that for such separation of data 1-2 KB of local processor memory is enough for most of the typical multimedia benchmarks, such as those described in Section 5. The rest of the memory requests, not handled by the local processor memories, are analyzed by our data reuse analysis technique [10] to obtain a set of possible intermediate data reuse buffers, which can hold the data that are reused many times in additional small memories near the cores that consume that data, thus reducing access time and energy consumption. The use of memories for such buffers instead of caches, in addition to lower power consumption per access, creates a system with fully predictable timing and thus eliminated the need for overdesigning the communication architecture to meet the worst case constraints.

We also assume that the memories we are using are fast enough to provide required bandwidth.

To model off-chip main memory, we represent it as a core whose mapping is fixed to the middle router in the first row of routers in NoC mesh.

We modeled the Æthereal NoC in our experiments. This is a synchronous NoC, which means that all the links and routes are operating on the same NoC frequency. However, network interfaces provide clock conversion, which mean that cores are not required to operate on the same frequency as NoC. Thus, the NoC frequency can be adjusted independently of that of cores. During the NoC synthesis we select its frequency to be the lowest possible that still provides enough bandwidth to perform all the required NoC data transfers in the allotted time.

In the next subsections we present two heuristics, namely the traditional two-step and our co-synthesis heuristic. The first one is the traditional

approach to system design, when memory synthesis is followed by NoC synthesis. The second, our proposed co-synthesis heuristic, performs these two tasks simultaneously. We compare effectiveness of these approaches in Section 5.

Algorithm 1. Two-step synthesis heuristic
input: reuse graph *RG*;
set of nodes *B* that represent memory buffers in the reuse graph;
set of routers RT of a mesh-type NoC
output: synthesized NoC

1. Create min memory energy communication graph CG out of the reuse graph:
1.1. Create communication graph CG out of main memory and processor nodes of the reuse graph
1.2 do {
1.3. for all nodes $b \in B$ do {
1.4. Add b to communication graph CG
1.5. Estimate memory energy
1.6. Remove b from communication graph CG
1.7. }
1.8. Add node $b_{best} \in B$ that reduces energy the most to CG (if found)
1.9. } while such node b_{best} is found

2. Perform NoC synthesis using communication graph CG:
// Initial placement
2.1. N = a set of CG nodes
2.2. Find a node $b \in N$ with the max communication requirements
2.3. Map b to the router in the center of NoC mesh
2.4. while there are unmapped nodes do {
2.5. Find unmapped node $b \in N$ which communicates the most with the mapped nodes
2.6. Map b to a router so that communication cost is minimized
2.7. }
// Mapping improvement
2.8. Perform NoC routing and floorplanning
2.9. E_{best}=estimated energy
2.10. for all $r_1 \in RT$ do {
2.11. if no cores are mapped to r_1, continue
2.12. swap_type=0
2.13. for all $r_2 \in RT$ do {
2.14. perform swap type 1 of routers r_1 and r_2
2.15. perform NoC routing and floorplanning
2.16. E_1=estimated energy
2.17. undo the swap
2.18. if ($E_1 < E_{best}$)
2.19. $E_{best} = E_1$; swap_type=1; $r=r_2$
2.20. perform swap type 2 of routers r_1 and r_2
2.21. perform NoC routing and floorplanning
2.22. E_2=estimated energy
2.23. undo the swap
2.24. if ($E_2 < E_{best}$)
2.25. $E_{best} = E_2$; swap_type=2; $r=r_2$
2.26. }
2.27. if (swap_type = 1)
2.28. perform first type of swap on routers r_1 and r
2.29. if (swap_type = 2)
2.30. perform second type of swap on routers r_1 and r
2.31. }
2.32. perform NoC routing and floorplanning
2.33. E_{best}=estimate_energy()
2.34. return

Figure 4. Two-step synthesis heuristic

4.2 Two-Step Memory/NoC Synthesis Heuristic

The first heuristic consists of two parts. First, we determine which buffers from the reuse graph should be implemented in order to achieve minimum energy consumption in memories, regardless of communication energy. The selected set of buffers together with the traffic between them forms communication graph, which is then used to perform NoC synthesis with the goal of reducing communication energy.

The outline of the heuristic is shown in Figure 4.

In the first part of the heuristic (Step 1), we start from the communication graph that includes only processors and main memory out of the reuse graph nodes obtained by the DRDM technique [10] (Step 1.1). Then we find a buffer (or a set of symmetric private buffers, which have the same size and configuration but belongs to the different processors) that result in the maximum possible decrease of total memory energy when added to the communication graph (Steps 1.3 – 1.7). If such node (or set of nodes) is found, we add it to the communication graph (Step 1.8) and repeat this process again. The memory energy is estimated based on the number of accesses to each buffer (this information is extracted from the reuse graph) and on the memory energy per access.

The second part of the heuristic, the NoC synthesis (Step 2), is based on the NMAP algorithm for single minimum-path routing from [16]. We improved the algorithm to allow mapping of several memories (or processor and memories) to the same router. This approach better exploits the Æthereal architecture in which several cores can be connected to the same router. Our experiments show that this feature often leads to solutions with lower overall energy consumption.

The NoC synthesis consists of two parts. In the beginning, the initial mapping of communication graph nodes to routers is obtained (Steps 2.1 – 2.9). Only one node is allowed per router. Then, in Steps 2.10 – 2.34, nodes are remapped to achieve the lowest possible NoC energy consumption.

During the initial mapping phase, we start with the communication graph node that has the highest communication demands. We map it to the central router in the mesh-type NoC (Steps 2.2 – 2.3). Then, we select a node that communicates the most with already mapped node(s) (Step 2.5), and map it to such a router that the communication cost with the already mapped nodes is minimized (Step 2.6). Communication cost is calculated as a sum of products of bandwidth requirements of the communication graph edge and the minimum distance (in NoC links) between the routers to which the nodes connected with that edge were mapped. The loop (Steps 2.4 – 2.7) is repeated until all nodes are mapped.

In the mapping refinement phase (Steps 2.8 – 2.33), we start with initial estimate of memory and NoC energy consumption (Steps 2.8 – 2.9). For that, routing (mapping of communication graph edges to the NoC links) and floorplanning (needed to estimate NoC link lengths) is done.

Minimum-path routing (Step 2.8) is performed by executing Dijkstra's shortest path algorithm for every communication graph edge to be routed, in the decreasing order of edges bandwidth requirements. Dijkstra's algorithm is performed on a smallest rectangular subset of NoC mesh that includes the routers to which the ends of the edge are mapped to. The bandwidth requirements of the communication graph edges that have been already mapped to NoC links are used as distances for the Dijkstra's algorithm. To approximate NoC link lengths, we use area numbers from memory and processor models and assume that the cores are of square shape.

The total energy consumption is calculated by adding memories, NoC links, routers and NIs energy consumptions (Step 2.9). Routers and NIs have two energy components: one that is consumed every clock cycle, and one that is proportional to traffic. The total number of clock cycles is determined by the busiest NoC link (thus we assume that NoC is running as slow as possible to still meet the deadline).

Next, we try to improve the current mapping by swapping the cores connected to the routers (Steps 2.10 – 2.31). We use two types of swapping: first type is when we completely swap all the NIs and cores between the two routers (Steps 2.14 – 2.19), and the second type is when we try to move memory node (excluding main memory) from the first router to the second (Steps 2.20 – 2.25). We leave the swaps that reduces the total energy the most (Steps 2.27 – 2.30), and undo the rest.

The complexity of the heuristic is $B^3+BR^2+ER^3lgR$, where B and E – the number of nodes and edges in the reuse graph, and R – the number of routers in NoC. If the number of reuse graph nodes and the number of routers are of the same order of magnitude (which is a reasonable assumption), the complexity is ER^3lgR, the same as of the NMAP NoC synthesis algorithm [16].

In the next subsection we present a co-synthesis algorithm, where the memory subsystem is customized together with NoC synthesis.

4.3 Memory/NoC Co-Synthesis Heuristic

The second heuristic is a co-synthesis heuristic, i.e. the memory synthesis is performed simultaneously with NoC synthesis.

The outline of the heuristic is shown in Figure 5.

In the first part of the heuristic (Step 1) we set a goal of reducing overall power consumption by reducing the NoC clock frequency, since as it was

shown in [21], the biggest contributor of additional power consumption after replacing buses with Æthereal NoC in a TV companion chip was the NoC clock power (54%). Due to this reason our heuristic first tries to reduce the maximum bandwidth requirements among all links between routers and between routers and NIs by performing memory customization.

In the second part of the heuristic (Step 2), we try to add additional buffers from the reuse graph to see if they can further reduce memory/NoC power consumption.

Algorithm 2. Co-synthesis heuristic
input: reuse graph *RG*;
set of nodes *B* that represent memory buffers in the reuse graph RG;
set of links (between routers and between routers and NIs) L of the NoC
output: synthesized NoC

1. Try to add buffers so that NoC clock frequency is reduced:
1.1. Create communication graph CG using main memory and processor nodes from the reuse tree
1.2. Synthesize(CG) // synthesize NoC as in step 2 of Alg. 1
1.3. do {
1.4. Find a link l∈L with the highest bandwidth
1.5. for all comm. graph edges cgl mapped to link l, in the order of decreasing bandwidth, do {
1.6. M = set of reuse graph buffers that affect edge cgl
1.7. Among all buffers b∈M, find the one b_{best} that reduces total energy the most
1.8. If found, add b_{best} to CG; go to step 1.11
1.9. }
1.10. go to step 2.1
1.11.} while (true)

2. Try to add other memory buffers to further reduce total energy consumption:
2.1. E_{best}=estimated energy
2.2. do {
2.3. Find b∈B that have not been tried to be added to CG before and with the highest reduction of traffic
2.4. Add b to CG
2.5. Synthesize(CG)
2.6. Remove b from CG
2.7. E=estimated energy
2.8. if (E< E_{best})
2.9. E_{best} = E; Add b to CG
2.10. }
2.11. return

Figure 5. Co-synthesis heuristic

In the first part of the Algorithm 2, we create communication graph CG using main memory and processor nodes from the reuse graph (Step 1.1). In Step 1.2 we synthesize NoC using the communication graph CG as it was explained in Step 2 of the Algorithm 1. In Step 1.4 we find the NoC link *l* with the highest bandwidth requirements. In Step 1.5, we analyze the communication graph edges that were routed through NoC link *l* and iterate over edges in the order of decreasing contribution to the NoC link traffic. In Step 1.6 we obtain the set of reuse graph buffers *M*, adding which to the communication graph splits the current edge *cgl* into several edges (with the

added buffers in-between) with different (lower or equal) bandwidth requirements. Next, in Step 1.7, we try to figure out adding which of the buffers (or a set of symmetrical buffers) from *M* reduces the total energy consumption the most. This step also involves NoC synthesis, which is done using the algorithm from Step 2 of the Algorithm 1. If the buffer that reduces the energy is found, it is added to the current communication graph (Step 1.8) and the loop (Step 1.5) which analyzes and tries to reduce the traffic of the busiest NoC link is repeated. If there are no more buffers that can reduce the maximum NoC clock frequency, the execution proceeds to the second part of the Algorithm 2.

In the second part of the Algorithm 2, we try to find a reuse graph buffer that has not been tried to be added to the communication graph earlier and which reduces the traffic to the higher level of memory hierarchy the most (Step 2.3). We add such buffer to the communication graph and evaluate if this addition reduces total energy consumption (Steps 2.4 – 2.8). If it does, we leave the buffer in the communication graph (Step 2.9) and proceed with the evaluation of other buffers.

The complexity of the heuristic is $B^2R^2+EBR^3lgR$, where B and E – the number of nodes and edges in the reuse graph, R – the number of routers in NoC. If the number of reuse graph nodes and the number of routers are of the same order of magnitude, the complexity is ER^4lgR, which is higher than that of the two-step heuristic.

In the next section we compare and establish the efficacy of the results obtained by the co-synthesis heuristic over the two-step heuristic.

Table 1. Experiment configurations

Experiment number	Benchmark	Processors	Main memory	NoC config
1	QSDPCM	6	800K on-chip	mesh 5x5
2	QSDPCM	6	8MB off-chip	mesh 5x5
3	Laplace	4	800K on-chip	mesh 5x5
4	Laplace	4	8MB off-chip	mesh 5x5
5	Laplace	16	4M on-chip	mesh 6x6
6	Laplace	16	8MB off-chip	mesh 6x6
7	Susan	4	800K on-chip	mesh 4x4
8	Susan	4	8MB off-chip	mesh 4x4

5. EXPERIMENTS

5.1 Experimental Setup

In our experiments we used several typical streaming video and image processing applications: QSDPCM, a video encoder [22], Laplace, image

filtering algorithm, and Susan, image recognition application. All the benchmarks were parallelized and distributed over several processors (Table 1).

Our energy model consists of several components. For on-chip memory power consumption we used (and extrapolated in some cases) the data from MPARM simulator [15]. We used CACTI [18] for 130 nm memory area estimations. We estimated area of TI C64 processors using data from [1][23]. Off-chip memory consumption was calculated based on 8MB Micron low-power CellularRAM chip [14].

Our model for Æthereal NoC power estimation is based on measurements described in [4]. The models for different NoC components are shown in Table 2. Since the network interface energy was not reported, we assumed the energy per port to be similar to that of the router. The area numbers are taken from [3].

Table 2. NoC energy and area

NoC Component	Energy, pJ	Area, mm^2
Router (guaranteed throughput)	E=(16.1+40.3α)*B + 32*P*C α=0.5-activity factor B-number of flits transferred P-number of router ports C-number of NoC clock cycles	0.17 (for 130 nm technology)
Network Interface	same as router	0.13
Link wires	E=(0.27+0.58*L)*N L-length of wire, mm N=32-number of wires	-

5.2 Experimental Results

The list of the benchmarks and parameters we used to synthesize NoC for each of our experiments is shown in Table 1. For each of the benchmarks we synthesized NoC for two cases: assuming that main memory is located on-chip and off-chip. For the latter case, we modified the synthesis algorithms to fix the position of the router to which the main memory was connected to a router on the edge of the chip, as described earlier.

For each benchmark configuration in Table 1, we performed three experiments. First, we synthesized NoC without performing data reuse analysis, i.e., we pruned the reuse graphs to include only main memory and processors. Note that this is the case when both of our heuristics return the same results as NMAP algorithm [16] since no memory synthesis could be performed (the memory subsystem is fixed) and processors and main memory are all mapped to different routers (due to restrictions in our modified NMAP synthesis algorithm) as it happens in the original NMAP

algorithm. Second, we synthesized NoC using sequential two-step synthesis heuristic (Algorithm 1 in Section 4). Finally, we performed memory/NoC co-synthesis using Algorithm 2 in Section 4.

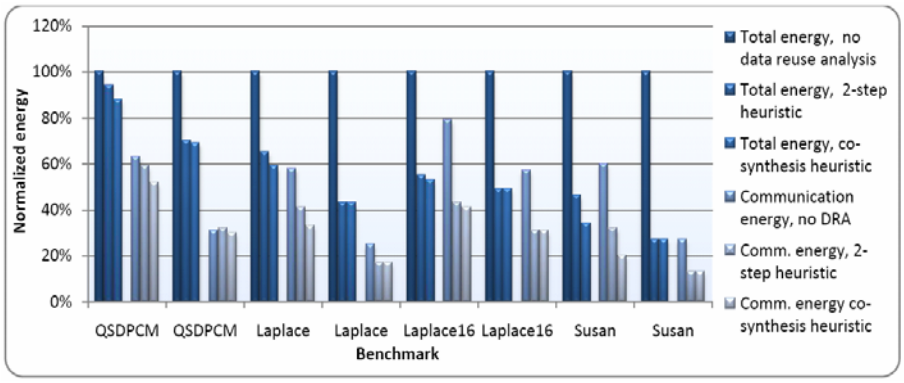

Figure 6. Experimental results

The results are shown in Figure 6. The graph shows energy consumption of the memory and communication subsystems normalized to the first case, when data reuse analysis is not used. For each benchmark, the first three graphs show the combined energy consumption of memory and NoC, and the last three graphs show the energy consumption of NoC alone. Results shows that customizing memory subsystem by employing data reuse analysis allows to reduce communication energy by 31% on average, and total energy by 44% on average. Furthermore, the use of memory/NoC co-synthesis heuristic instead of two-step synthesis allows to further reduce the communication energy by up to 38% (10% on average) and total (memory and communication) energy by up to 26% (6% on average).

6. CONCLUSION

In this work we investigated the influence of memory subsystem on power consumption of NoC communication architecture. We showed that customizing memory subsystem by employing data reuse analysis allows reducing communication energy by 31% on average. We also proposed a data reuse based memory/NoC co-synthesis heuristic, and showed that it outperforms traditional two-step synthesis approach by reducing the communication energy by up to 38% and total energy by up to 26%. Although our approach was experimented on the Æthereal NoC architecture, we believe this co-synthesis strategy will be useful for a wide range of other NoC architectures. Future work will investigate these issues.

REFERENCES

[1] S. Agarwala et al. A 600-MHz VLIW DSP. IEEE Journal of Solid-State Circuits, Nov. 2002.

[2] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In Proc. of DAC, June 2001.

[3] J. Dielissen et al. Concepts and Implementation of the Philips Network-on-Chip. IP-Based SOC Design, Grenoble, France, Nov. 2003.

[4] J. Dielissen et al. Power Measurements and Analysis of a Network on Chip. Philips Research, Technical Note TN-2005-0282, Apr. 2005.

[5] K. Goossens et al. Æthereal Network on Chip: Concepts, Architectures, and Implementations. IEEE Design and Test of Computers, 2005.

[6] P. Guerrier and A. Greiner. A generic architecture for on-chip packet switched interconnections. In Proc. of DATE, March. 2000.

[7] Zvika Guz et al. Efficient Link Capacity and QoS Design for Network-on-Chip. In Proc. of DATE, March 2006.

[8] R. Ho, K. Mai, and M. Horowitz. The future of wires. In Proc. of the IEEE, April. 2001.

[9] Hu et al. Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints. In Proc. of ASP-DAC, 2003.

[10] I. Issenin, E. Brockmeyer, B. Durinck, N. Dutt. Multiprocessor System-on-Chip Data Reuse Analysis for Exploring Customized Memory Hierarchies. In Proc. of DAC, 2006.

[11] I. Issenin, N. Dutt. Data Reuse Driven Energy-Aware MPSoC Co-Synthesis of Memory and Communication Architecture for Streaming Applications. In Proc. of CODES+ISSS, 2006.

[12] S. Kim et al. Efficient Exploration of On-Chip Bus Architectures and Memory Allocation. In Proc. of CODES+ISSS, 2004.

[13] Manolache et al. Fault and Energy-Aware Communication Mapping with Guaranteed Latency for Applications Implemented on NoC. In Proc. of DAC, 2005.

[14] Micron Async/Page/Burst CellularRAM Memory MT45W4MW16BCGB, http://download.micron.com/pdf/datasheets/psram/64Mb_burst_cr1_5_p25z.pdf.

[15] MPARM project, http://www-micrel.deis.unibo.it/sitonew/ research/mparm.html.

[16] S. Murali, G. De Micheli. Bandwidth-Constrained Mapping of Cores onto NoC Architectures. In Proc. of DATE, 2004.

[17] S. Pasricha, N. Dutt. COSMECA: Application Specific Co-Synthesis of Memory and Communication Architectures for MPSoC. In Proc. of DATE, 2006.

[18] P. Shivakumar, N. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. WRL Technical Report 2001/2, Aug. 2001.

[19] K. Srinivasan et al. A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures. In Proc. of ISLPED, 2005.

[20] K. Srinivasan et al. Layout Aware Design of Mesh based NoC Architectures. In Proc. of CODES+ISSS, 2006.

[21] F. Steenhof et al. Networks on chips for high-end consumer-electronics TV system architectures. In Proc. of DATE, 2006.

[22] P. Stobach. A new technique in scene adaptive coding. In Proc. of EUSIPCO, Grenoble, 1988.

[23] Z. Yu et al. An Asynchronous Array of Simple Processors for DSP Applications. In Proc. of ISSCC, 2006.