International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-2 Number-4 Issue-6 December-2012

Adaptive mapping for reducing power consumption on NoC-Based MPSoC

Lalit Tembhare[1], Yogeshver khandagre[2], Alok Dubey[3]
Department of Electronic & Telecommunication, Trinity College of Engineering & Technology
Bhopal, India

Abstract

Intricacy of the S/W increased considerably nowadays. It needs smart application specific architectures as NoC based MPSoCs to achieve the high data communication with computation requirements. The design of these systems needs efficient functionalities accumulating fragmenting and mapping. In this manner, this paper makes partitioning and mapping influence on energy consumption of homogeneous NoC-Based MPSoC. In addition it compares two strategies to get efficient dynamic mappings. First one is, map tasks directly on the processors. Second is, applies a previous static task-partitioning and then use this information to choose the dynamic task mapping.

Keywords

Partitioning, Mapping, MPSoC, NoC.

1. Introduction

Nowadays large quantity of applications, demanding enormous computational power & large memory sizes with reduced energy consumption and efficient communication. All this needs boost the research and development of specific architectures, like a NoC-based MPSoC. In which the complete system functionality is implemented into a single chip and supports the heavy communication needs of multiple processors with efficient energy consumption. By the processing point of view, homogeneous MPSoCs are composed by multiple processors of the same type and heterogeneous MPSoCs are composed by multiple processors with different architectures. Heterogeneous MPSoCs can support a wide variety of applications, since each processor has specific computation and communication features, otherwise homogeneous MPSoCs are easier to program & increase the mapping and fragmenting possibilities, and enable global load balancing through application-task migration. Furthermore, the homogeneity may minimize the global energy consumption and area occupation for some set of applications [1]. This work employs homogeneous NoC-based MPSoC as target architecture, and presents a partial design flow containing the application task partitioning into groups of tasks, where each group is mapped onto tiles of the target architecture. Here, tile is an area of limited target architecture which is having processor, router and local memory with auxiliary circuits. Several works relate to task mapping onto NoC-based architectures and some ones describe the tasks partitioning into groups ([2 - 13]), but none evaluate the effect of using static partitioning as a previous step of the dynamic mapping. Here, we propose comparison of two approaches: (i) the traditional one that during the run time map tasks onto processors and (ii) the other, which performs a previous analysis of tasks affinity by a partitioning step and uses this information to choose fast and efficient mappings. Moreover, several works uses the same name "mapping" to define both mapping and partitioning, while the name "partitioning" is used only to explore hardware/software division.

2. Methodology

The traditional flow associates tasks directly to tiles, which is called here as task mapping. On the other hand, our flow considers tasks affinity to generate groups. The grouping of all application tasks, which is the task partitioning activity, generates a partition. The next step is to perform the selection of the best place that each group of tasks will be associated, which is the task-group mapping onto tiles activity. To better understand the concepts of partitioning and mapping of the proposed flow, Figure 1 exemplifies the partitioning of a hypothetical application composed by 22 tasks into 6 groups and the corresponding mapping of these task groups onto tiles of 2D-mesh NoC architecture. The application is composed by a set of parallel communicating tasks $T = \{t1, t2 \ldots\ldots t22\}$. The tasks partitioning, which is represented by continuous arrows, generates groups $G = \{g1, g2 \ldots\ldots g6\}$ i.e. a set of task-groups. Finally, task-groups mapping onto tiles is represented by the dotted arrows. These one associates each element of G to an element of the set of NoC tiles $\Gamma = \{ \tau 1, \tau 2, \ldots, \tau 6\}$. In addition, each tile contains a single element of the set of processors $P = \{p1, p2, \ldots, p6\}$.

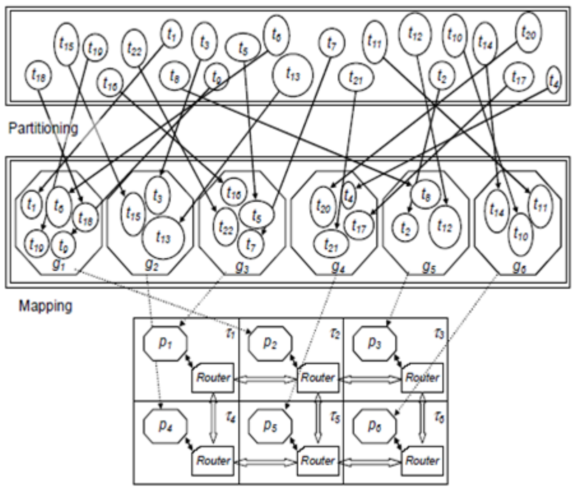

Figure 1: Partitioning and mapping understanding.

The task mapping onto tiles of the target architecture adds the complexities of partitioning tasks across groups and the mapping of these groups onto tiles. In this case, the complexity is much higher because the number of solutions to be explored with mapping tasks onto tiles is much higher. Thus, even applying good algorithms, the results obtained with this activity tend to be worse, when compared to those obtained with the flow proposed here, mainly in the cases where it is done at run time, since the mapping has a short time to be accomplished, requiring fast but sometimes inefficient algorithms.

The partitioning and mapping have three main data structures that are set out below:
Definition 1: A Task Communication Graph (TCG) is a directed graph <T, V>. The set of vertices T = {t1, t2……..m} represents the set of m tasks in one parallel application. Assuming Vab is the bits amount of all packets sent from a task ta to a task tb, then the set of edges V is {(ta, tb) | ta, tb ∀ T and Vab ≠ 0}, and each edge is labelled with the value Vab. V represents all communications between the application tasks.
Definition 2: A Communication Weighted Graph (CWG) is a directed graph <P, W>, similar to the TCG. However, the set of vertices P = {p1, p2...… pn} represents the set of processors in one application. The quantity of processors n is equal to the total quantity of tiles, since each tile has a single processor. Furthermore, w_{ab} is the total quantity of bits sent from a processor p_a to a processor p_b. Then the set of edges W is $\{(p_a, p_b) \mid p_a, p_b$ € P and $w_{ab} \neq 0\}$, and each edge is labelled with the value W_{ab}. W represents all communications between the MPSoC processors, while CWG reveals information of application's relative communication volume. The mapping is performed regarding to a 2D mesh NoC using wormhole and deterministic XY routing algorithm. The communication resource graph stated below captures the NoC topology.

Definition 3: A Communication Resource Graph (CRG) is a directed graph <Γ, L>, where the vertex set is the set of tiles Γ = { τ1, τ2, …, τn} and the edge set L = {(τi, τj), ∀ τi, τj € Γ} gives the set of paths from τi to τj. The value n is again the total quantity of tiles and is equal to the product of NoC lines and columns. CRG edges and vertices represent physical links and routers of the target architecture, respectively. Both, processors (with the whole memory hierarchy) and communication architecture originate energy consumption. The sum of the energy consumed by the execution of all tasks grouped on a processor enable estimating its energy consumption. This value is used, together with the communication volume between tasks, to choose good partitions. On the other hand, the amount of bits transmitted between tasks grouped and mapped onto different processors contributes to estimate the energy consumption used to choose good mappings. The approach used here to model the NoC's energy consumption is similar to those shown in [13] and [14]. Dynamic energy consumption is proportional to switching activity, arising from packets moving across the NoC, consuming energy on the links and inside of each router. The concept of bit energy *EBit* is used to estimate the dynamic energy consumption of each bit, when this flips its polarity from a previous value. *EBit* is split into three components: (i) bit dynamic energy consumed by the router (wires, buffers and logic gates) (*ERbit*); (ii) bit dynamic energy consumed on horizontal (*ELHbit*) and vertical (*ELVbit*) links between tiles; and (iii) bit dynamic energy consumed on the links between the router and the local processor (*ECbit*). Equation (1) expresses the relationship between these quantities, which computes the dynamic energy consumption of a bit passing through a router, a vertical or horizontal link and a local link.

$$EBit = ERbit + (ELHbit \text{ or } ELVbit) + ECbit \qquad (1)$$

ERbit depends on the buffer structure and technology to estimate how many bit-flips occur to write, to read and to preserve the information. *ELbit* is directly proportional to the tile dimension. For regular 2D-mesh NoCs with square tiles, it is reasonable to consider that *ELHbit* and *ELVbit* have the same value. Therefore, the next equation uses *ELbit* as a simplified representation of *ELHbit* and *ELVbit.*
Equation (2) computes the dynamic energy consumed by a single bit traversing a NoC, from tile τi to tile τj, where η corresponds to the number of routers through where this bit passes.

$$EBit_{ij} = \eta \times ERbit + (\eta - 1) \times ELbit + 2 \times ECbit \qquad (2)$$

Let τi and τj be the tiles to which p_a and p_b are respectively mapped. Then, the dynamic energy consumed by a $p_a \rightarrow p_b$ communication is given by $EBit_{ab} = w_{ab} \times EBit_{ij}$. Equation (3) gives the total amount of NoC's dynamic energy consumption (*ENoC*) that is computed for all bits of all communications between processors (|W|) [10].

$$ENoC = \sum_{i=1}^{|W|} EBit_{ab}(i), \ \forall p_a, p_b \in P \qquad (3)$$

3. Proposed methodology

Figure 2 illustrates the design flow description, which is implemented inside CAFES [2], a framework for MPSoC design. The task mapping flow is represented by dashed arrows, while the proposed one is symbolized by continuous arrows. In addition, dotted arrows represent input information for mapping and / or partitioning. Task partitioning into groups has as entries: (i) application description, which has all tasks and their communications; (ii) Processors list, which has the name and quantity of processors enabling to compute the quantity of task groups; (iii) Processor use that is a constraint to limit the quantity of task grouped into the same processor; and (iv) NoC energy parameters that is used to compute the energy consumption of a given partition.

The partitioning cost function takes into account the minimization of the overall communication volume. The algorithm tries to achieve a minimum cost, which implies to cluster into the same processor high communicating tasks. In addition, the algorithm tries to balance the processor use through fair distribution of tasks over the available processors, respecting processor use constraint. In other words, tasks that communicate most are grouped as far as they do not compromise more than the maximum processor use for each processor a parameter set according to the application requirements. The processor use constraint is neglected only in cases where there are no other available processors, i.e. the task association to every processor always implies more than the maximum processor use. The partitioning tool generates a CWG description (Section 2) that contains all processor tasks associations.

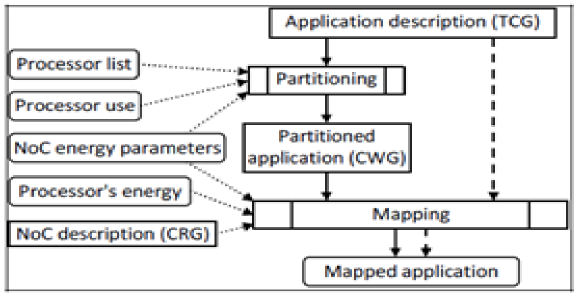

Figure 2: Design flow showing the partitioning and mapping

The SA algorithm [13] achieves good results for static partitioning problem, since the designer has much time to perform it. On the other hand, dynamic mapping requires fast decisions to not postpone the application execution implying a simpler but efficient algorithm.
This work implements two mapping algorithms: (i) one that has as input the TCG description, which maps the most communicating tasks onto the same processor, while the maximum processor use is not reached. When it happens a new neighbour tile is searched using a mapping cost function; and (ii) the second one that has as input the CWG description. This algorithm searches in the set of task groups for the tile where at least one other task of the same group had been previously mapped. If no previous task was mapped, the algorithm uses the same mapping cost function of the previous algorithm to search the new target tile.

The mapping cost function takes into account the communication volume between processors and the NoC energy parameters to compute the energy consumed on a given mapping. Considering a given pair of communicating processors, together with CRG and NoC parameters, the energy consumption is computed through the energy model described on Section 2. The energy consumption achieved by task running on processor is used only to compute the total energy consumption, but does not affect the mapping choice. For both flows, the mapping generates a file containing all processor tile associations that implies a minimum energy consumption of all evaluated maps.Partitioning and mapping cost functions use the same NoC energy parameters stated by Equation (2). However, while mapping specifies the exact processor place into the NoC, the partitioning only explores the communication needs, but the number of hops there is between two communicating processors is unknown. In these sense, partitioning cost function uses the concept of average of hops that enables to compute the average energy consumption of all possible paths.

4. Conclusion

The mapping on the tiles of the specific architecture is an NP-complete design activity, and then accomplished at runtime may not get good results, due to the sparse time and to explore the large number of solutions. This work proposes application of fragmentation of tasks into fragments before the mapping. Once fragmented, the search space of the mapping is reduced, that allows achieving efficient dynamic mapping algorithm that may performs an ideal mapping in a small period of time. Consequently multiple applications requirement can be fulfilled splendidly.

References

[1] Jalier C etal, "Heterogeneous vs. homogeneous MPSoC approaches for a Mobile LTE modem", DATE, pp.184-189, 2010.
[2] Le Beux, S etal, "Combining mapping and partitioning exploration for NoC-based embedded systems", JSA, Vol.56(7), pp.223-232, 2010.
[3] Sahu P etal, "A new application mapping algorithm for mesh based Network-on-Chip design", INDICON, pp.1-4, 2010.
[4] Bo Yang etal, "Multi-application multi-step mapping method for many-core Network-on-Chips", NORCHIP, pp.1-6, 2010.
[5] Carvalho, E.; Calazans, N.; Moraes, F., "Dynamic Task Mapping for MPSoCs", IEEE Design & Test, v.27(5), pp. 26-35, 2010.
[6] Leupers, R., Castrillon, J., "MPSoC programming using the MAPS compiler", ASP-DAC, pp.897-902, 2010.
[7] Guang S. etal, "Energy-aware run-time mapping for homogeneous NoC-SoC", pp.8-11, 2010.
[8] Nedjah N., Silva M., Mourelle L., "Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization", JSA, v.57(1), pp.79-94, 2011.
[9] Tsai K. etal, "Design of low latency on-chip communication based on hybrid NoC architecture" NEWCAS, pp.257-260, 2010.
[10] Youness H. etal,"A high performance algorithm for scheduling and hardware-software partitioning on MPSoCs" DTIS, pp.71-76, 2009.
[11] Go_hringer, D. etal, "A Design Methodology for Application Partitioning and Architecture Development of Reconfigurable Multiprocessor Systems-on-Chip", FCCM, pp.259-262, 2010.
[12] Marcon, C. etal, "CAFES: A framework for intrachip application modeling and communication architecture design", JPDC, v.71(5), pp.714-728, 2011.
[13] Bononi, L. etal, "NoC Topologies Exploration based on Mapping and Simulation Models", DSD, pp.543-546, 2007.
[14] Chen-Ling C., Marculescu R., "Contention-aware application mapping for Network-on-Chip communication architectures", ICCD, pp.164-169, 2008.

I am **Lalit N Tembhare** from Nagpur Maharashtra, studding in Trinity College of Engineering & Technology, Bhopal (M.P.). I am 25 Years old pursing M. Tech. in VLSI.