Information

Extended Abstract submission for HPCN'97, Vienna, April 1997.

Title : Parallel Synthesis of Large Combinational Circuits for FPGAs

Author / Presenting author : Laurent LEMARCHAND

Contact :
- email : lemarch@univ-brest.fr
- Tel : (33) 2 98 01 62 17
- Fax : (33) 2 98 01 62 15
- postal : L. Lemarchand
  LIBr
  Universite de Bretagne Occidentale
  UFR Sciences et Techniques
  Departement d'Informatique
  BP 809
  29285 Brest Cedex
  FRANCE

Parallel Synthesis of Large Combinational Circuits for FPGAs

L. Lemarchand

Laboratoire d'Informatique de Brest,
Université de Bretagne Occidentale,
6 av. Le Gorgeu, Brest cedex 29287, France
e-mail: lemarch@univ-brest.fr

Abstract

High level synthesis tools are the main stream today for the rapid design of electronic circuits. At a lower level, logic synthesis systems like SIS [7] are in charge of the optimization of the combinational part of the circuit. These tools also realize the mapping of the design on programmable devices such as FPGAs. The logic synthesis is a computation intensive task. We propose in this paper to partition the graph representing the circuit to reduce the synthesis problem size. Splitting decreases runtime and allows the use of more performant algorithms. However, it leads to a lost of quality for the final circuit. To perform an efficient partitioning, we have adapted an up-to-date algorithm, Metis [9], for the partitioning of circuit graphs.

Since subcircuits are processed separatly using SIS, a distributed implementation based on PVM [6] has also been realized. Results are very encouraging for both the runtime and the solution quality of the synthesis on a network of up to 8 workstations.

Introduction Electronic circuit CAD tools are in charge of converting a description of a digital circuit into an interconnection of logic gates, namely a *gate-level net-list* (see fig 1).

Specifications can be seen as a hierarchy of behavioral, register-transfer-level (RTL) and/or gate-level componants descriptions. Compiling these descriptions for hardware involves several stages as high level optimization, scheduling and sharing of resources. At the RTL level, control and operators appear as an assembly of registers and combinational circuits : memory elements are fixed, and logic is represented as a set of boolean equations.

Logic synthesis is an optimization activity admitting RTL descriptions as an entry point. Its goal is to improve the circuit on criteria such as speed or area (*logic optimization*), and then to produce an equivalent specification suitable for a target technology such as ASIC or FPGA (*technology mapping*).

Generating a circuit involves others tasks depending on the technology. As an example, FPGAs need binding, placement, routing and configuration production (see fig 1).

FPGA synthesis As the FPGAs vendors are making much efforts to follow integration technologies, the reconfigurable circuits become much larger, reaching the 100.000 gates, and much faster with system clock in excess of 100 MHz. Reconfigurable computing will be an alternative for some computation intensive applications. However, to be accepted, very important progresses must be achieved in the programming methods : FPGA programming must be, at least, comparable in speed to DSP assembly programming.

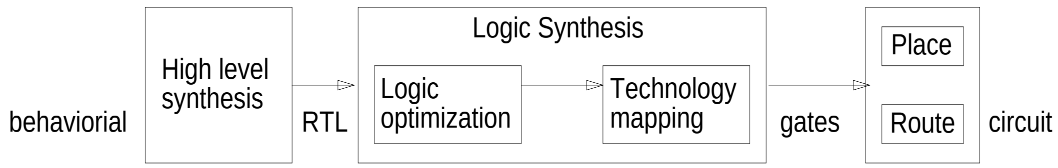

Figure 1: The CAD design process for FPGA target

Several bottlenecks limit the design process for FPGAs. To speed up the synthesis, attention must be paid to both the high level synthesis tools and the logic synthesis systems.

An Object-Oriented framework could provide a very interesting support on critical points such as CAD management and high level elementary circuits modeling. Our laboratory is developping an integrated environment for fast prototyping and co-design on FPGAs [10]. This environment is implemented using Smalltalk-80 on top of the Berkeley logic synthesis tools (SIS [7]). SIS is an open software for the synthesis of RTL circuits. It implements various algorithms, including logic minimization of boolean circuits (MisII), sequential circuits minimization, technology mapping for different targets such as FPGAs, testing, re-synthesis... SIS allows the use of different algorithms at each step of the synthesis according to the designer objectives. The synthesis process is described as a *script* corresponding to a sequence of synthesis tasks execution.

Parallel synthesis using boolean network partitioning Logic synthesis of large circuits involves important runtime and huge memory requirements, even if heuristics are used. The aim of the current work is to improve execution time of the logic synthesis process. We focus on the boolean part of RTL circuits. The set of equations describing the circuit is represented by a *boolean network*. This network is a directed acyclic graph (*dag*). A node of this DAG is associated to each boolean equation, and there is an arc between two nodes N_i and N_j if the output of N_i is an input to N_j.

The proposed way to reduce the execution time of logic synthesis consists in partitioning the initial boolean network. After partitioning, each subnetwork is synthesized separatly. The problem size reduction is a very important point because common algorithms are often NP-Hard. Dividing the circuit also allows the use of more efficient algorithms. Furthermore, since subnetworks are considered as independant, synthesis can be parallelized easily.

However, the synthesis is no more global over the full network, whereas logic optimization and technology mapping are based on boolean properties and dependances between nodes. Since optimization relies partially on node relationships, partitioning leads to a lost of quality.

Algorithm To take advantage of logic partitioning without degrading the circuit quality significantly, we use k-way partitioning techniques. k-way partitioning affects each node of a graph to a cluster, while minimizing the number of edges crossing part boundaries, and balancing the number of nodes in the k clusters. Applied to boolean networks, this technique minimizes the lost of dependance informations and allows to balance the amount of work per processor in a parallel implementation.

The distributed algorithm is the following :

1 The boolean network is partitioned using Metis [9], an up-to-date k-way multi-level algorithm from Karypis and Kumar. Metis has already been applied sucessfully in the field of circuit partitioning [1]. We have adapted it to the problem of boolean network partitioning and it has been integrated in SIS.

2 Each subnetwork is processed separatly in parallel. At this level, any SIS script can be applied, or a recursive partitioning can take place.

3 Optimized subnetworks are merged back together. The resulting network is functionally equivalent to the original circuit.

An implementation of this parallel algorithm and the results obtained for the synthesis of circuits on FPGAs are presented next.

Implementation A master/slave manager has been developped using PVM [6] for parallel synthesis based on SIS. The master process spawns SIS tasks corresponding either to partitioning or synthesis tasks. Partitioning is implemented as a SIS command that interfaces the Metis software. The command splits the network into subnetworks available as data files. Synthesis tasks are spawned with the accurate SIS script on these files. The master/slave paradigm also allows us to mix distributed phases and global optimization on a single processor. The software works on a network of workstations or on a Paragon parallel computer.

Results The partitioning algorithm has been tested and compared to previous circuit partitioning algorithms [4, 8, 2, 3]. The parallel synthesis tool has been compared to another parallel tool that also use logic partitioning in parallel [5]. These points will be detailled in the full paper. Table 1 shows results for 1, 4, and 8 partitions running on 8 Sun SS5 workstations with 32 Mb of memory. In each case, we present the number of elementary cells (CLB) for the circuit mapped on a Xilinx FPGA before placement and routing. Overall times are in seconds. Times include partitioning, parallel application of a standard synthesis script of SIS and merging of optimized subnetworks. The tested circuits are from ISCAS'89 public benchmark suite [11].

	1P		4P		8P	
Circuit	CLB	time	CLB	time	CLB	time
C2670	105	793	133	138	140	111
dalu	192	2257	255	228	305	225
i8	232	759	326	143	434	325
C3540	229	1251	241	256	233	105
i10	435	37272	448	379	476	257
C5315	273	953	292	314	306	167
k2	253	15838	385	379	454	248
C6288	463	992	537	302	551	231
C7552	656	1295	297	480	321	266
des	1070	3055	701	1451	693	873
Total	3908	64465	3615	4070	3913	2808

Table 1: Parallel synthesis. Comparison of runtime (seconds) and solution qualities (minimizing number of CLBs)

Speed up and solution qualities are very encouraging on a network of up to 8 workstations. The best results, as compared to sequential execution, are obtained with difficult circuits taking long delays on one processor, e.g. C3540, i10, k2, C7552 and des. Splitting avoids the degradation of performances observed for these circuits; in sequential, these bad runtimes are coming mainly from memory swapping involved by the circuit sizes.

Solution quality is surprisingly sometimes better in parallel. This is due to the automatic switching of some algorithms from heuristic to exact methods, according to the size of the circuits processed.

Software will be soon available for public use at
`http://ubolib.univ-brest.fr/~lemarch/PPart`.

References

[1] C.J. Alpert and A.B. Kahng. A hybrid multilevel/genetic approach for circuit partitioning. In *Fith Physical Design Workshop*, pages 100 – 105, April 1996.

[2] D. Cheng, C.-C. Lin, and M. Marek-Sadowska. Circuit partitioning with logic perturbation. In *Proc. ICCAD*, 1995.

[3] H. Cho, G. Hatchel, M. Nash, and L. Setiono. BEAT_NP: A tool for partitioning boolean networks. In *Proc. ICCAD*, pages 10–13, 1988.

[4] J. Cong, Z. Li, and R. Bagrodia. Acyclic multi-way partitioning of boolean networks. In *Proc. Design Automation Conf.*, pages 670 – 675, 1994.

[5] K. De and P. Banerjee. Parallel logic synthesis using partitioning. In *Proc. of ICPP*, 1994.

[6] A. Geist et al. *PVM 3 User's Guide and Reference Manual.* Oak Ridge National Lab., May 1994.

[7] E. Sentovich et al. SIS: a system for sequential circuit synthesis. memorandum UCB/ERL M92/41, University of California at Berkeley, mai 1992.

[8] S. Iman, M. Pedram, C. Fabian, and J. Cong. Finding uni-directional cuts based on physical design and logic restructuring. In *Proc. Physical Design Workshop*, pages 187 – 198, 1993.

[9] G. Karypis and V. Kumar. METIS: Unstructured graph partitioning and sparse matrix ordering system. Technical report, University of Minnesota, August 1995.

[10] B. Pottier. Resiviting smalltalk-80 blocks. In *Proceedings of NAPA'96*, 1996.

[11] S. Yang. Logic synthesis and optimization benchmarks user guide. Technical report, Stanford University, 1991.