PPart : Parallel Synthesis of Large Circuits for FPGA
User Manual
Version 2.0

L. Lemarchand

January 25, 1999

1 Introduction

This user manual provides information on the use of the PPart package for the parallel synthesis of large combinatorial and sequential circuits. We are particularly interested in the logic synthesis of circuits for the Xc4000 and Xc6000 FPGAs from Xilinx [4].

Logic synthesis of large circuits involves important runtime and huge memory requirements, even if heuristics are used. The aim of PPart is to improve execution time of the logic synthesis process.

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. Hierarchical circuits provide a natural way of partitioning which can also be used parallelize the synthesis. However, if the circuit is partitioned, 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.

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.

```
# this is a comment
# pscript file
# command 'pvm'
#  list of available processors, with
#  their relative speeds (default: 100)
pvm mach1 mach2 120 mach3 mach4 130 mach5 110

# command 'part'
# 8-partitioning of circuit stored in toto file
# partitioning results are stored in titi
```

```
part 8 toto titi

# command 'source'
# parallel titi optimization with SIS script commands stored vite file
# optimized circuits are stored in tutu
# (both files are in BLIF format)
8 source vite titi tutu

# command 'merge'
# merge optimized circuits in file final
merge 8 tutu final
```

This pscript is invoked by running `ppart -f pscript`. It will process in parallel `toto` on `mach1`, `mach2`, `mach3`, `mach4` and `mach5` processors. Synthesis results will be stored in `final` BLIFfile.

2 Partitioning boolean networks within SIS

The PPart package provides an interface with the Metis and HMetis standalone library. Two commands are available :

The main command is based on an hypergraph partitioning tool, HMetis [2]. It is implemented in SIS by `part` command. `part` is applied on current circuit within sis (e.g. obtained from a file with the `read_blif`). The syntax is the following :
`sis> part nb out`
with `nb` corresponding to the number of partitions, and `out` the filename for the output files storing resulting sub-circuits.

Partitioning is realized while trying to balance th number of internal nodes per partition. By default all nodes have unit weight, but it is possible to affect different weights to nodes, according to the SOP complexity of the logic associated to each of them.

The complete syntax of the command is the following :
`part [options] [-f out]`

options are :

- `-p np` : `np` is the number of desired blocks (default 4).
- `-f out` : `out` is the filename for writing output parts in BLIF format (default `partition`).
- `-u` : unit weight model for nodes (default).
- `-l` : weights for nodes are defined as the number of literals in the SOP representation of the logic equation associated to the node.
- `-h` : makes use of HMetis for partitioning (default : use Metis).
- `-b` : uses a *Bin-Packing* algorithm instead of Metis or HMetis. Les CPUconsuming, but doesn't minimizes the cut.
- `-n` : modifies option `-h` cut counting metric.

Merging partitioned files is implemented within SISby the `merge` command, which syntax is :

`merge [-f in] [-p nb]`

`in` are the `nb` input file names to be merged (in BLIFformat) (default nb is 4, filename `partition`).

3 Distributed synthesis

The previously described commands can be used independently, or called from the `ppart` program in the case of a distributed synthesis.

`ppart` has been developped using PVM [1] for parallel synthesis based on SIS. The master process spawns SIS tasks corresponding either to partitioning or synthesis tasks. The partitioning 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.

Distributed synthesis is specified as a *pscript* corresponding to a sequence of phases, involving one or more processors. The master/slave paradigm used allows large flexibility in the definition of the sequential/parallel execution of the synthesis. However, because of communication costs over NFS, the system is intended to manage coarse-grain tasks.

PPart program and pscript commands The `ppart` program has the following syntax : `ppart -f <pscript>` . A pscript file contains a sequence of actions to run. Empty lines and lines beginning with `#` are ignored.

Pscript commands are divided into 3 classes :

- partitioning and merging (see above),
- SIS scripts execution in parallel
- micellenous commands for starting the PVM machine, printing stats, testing the network ...

3.1 SIS scripts execution in parallel

3 commands are available for appliyng a script onto a set of subnetworks :

- `pvm machine [speed] [machine [speed]]` starts a PVM machine. Default speed is 100. The machine running the ppart program must be included in the machine list. The speed information is relevant only for schedulers 0 and 1. This command must be called before any command implying parallel tasks spawning.
- `speed file script` determines machine speeds online. The speed of a machine is calculated as the execution time of the SIS script `script` onto the file `file` on this machine.
- `verbose` activates the SIS tasks outputs collect on the master tty. It also allows the tracing of the tasks executed, and the printing of statistics concerning the different processors used during parallel synthesis.

- `silent` deactivates the SIS tasks outputs collect on the master tty. This is the default behavior. Notice that `verbose` and `silent` commands have no effect after the starting of the pvm machine (see `pvm` command above concerning the slave outputs.

- `time <counter>` prints on the master tty the total time in seconds since the beginning of the program execution, the ellapsed time since the last call to `time`, and accumulates this ellapsed into counter `<counter>`. accumulated in timer `counter` since the starting of the `ppart` Timers are numbered starting from `0`. Timer `0` is started automatically when starting `ppart`.

- `[np] ps in` executes the SIS `print_stats` command for the boolean network(s) stored in `in` file(s). The `verbose` mode must be set to collect the resulting data on the master output.

- `[np] part parts in out` partitions file(s) `in` of into `parts` files. `parts` must be ≥ 1. Output file(s) `out` format is BLIF.

- `[np] merge parts in out` merges file(s) `in.part-0`, `in.part-1`, ..., `in.part-<parts - 1>` into `out` file. `in` must be in BLIF format. `out` is in `tout` format. Recall that if `np` is > 1, `in` and `out` are first expanded in `in.part-x` and `out.part-x`. `parts` must be ≥ 1.

- `[np] source sis_com in out` executes the SIS commands of SIS script `sis_com`. The input is (are) the file(s) `in`. The resulting boolean network(s) is (are) stored in file(s) `fout`.

Notice that the file format is always BLIF. For the 4 last commands, `np` default is 1.

pscript example Files corresponding to this example can be found into the `examples/STD` directory of the PPart distribution. The synthesized circuit is `cc.blif` from the ISCAS'89 benchmark suite [5].

• The pscript `pscript` defines the different phases of the synthesis. We are going to synthezise a circuit onto 4 processors, using 8 processes. We want to realize the logic optimization and the technology mapping in parallel suitable for the Xc4000 series of Xilinx.

```
#########################
# Pscript file STD/pscript : #
#########################

verbose
# starting the virtual machine, with different speeds (default 100)
pvm machine1 machine2 200 machine3 500  machine4 200

# Phase 1 : partioning cc.blif
part 8 cc.blif blif cc

# Phase 2 : optimizing each partition in parallel.
# script is the standard script of MISII.
8 source script cc cc_misii

# Phase 3 : technology mapping in parallel.
```

```
# script.xilinx_4k produces a 4-bounded network
8 source script.xilinx_4k cc_misii cc_xc4k

# Phase 4 : merging resulting files into a single boolean network.
merge 8 cc_xc4k cc_xc4k
```

• The files `script` and `script.xilinx_4k` are the SIS scripts executed in parallel for respectively the logic optimization and the technology mapping of the algorithm. These scripts are called from `pscript`.

• `cc.blif` is the initial BLIF file.
The synthesis could be started by the following command :

```
% ppart -f pscript
```

4 Full example

Files corresponding to this example can be found into the `examples/PLA` directory of the PPart distribution. This file has been generated automatically by a high level tool [3]. It is a large circuit described in PLA format (82 Kb). The goal is to map it onto Xc4000 series. We want to simplify the circuit and minimize the size of the resulting layout. The circuit is a 6-bit adder with a special arithmetic. The circuit is available as a 12 inputs / 6-outputs PLA. We consider that 8 processors are available in the PVM machine.

The processing steps are the following :

- The PLA representation corresponds to a 6-nodes boolean network. Each node is associated to an output of the circuit. The corresponding boolean equations are very complex (fully specified). The first step consists in simplifying each equation, and splitting corresponding node into a set of smaller nodes. This is usefull for the subsequent steps of parallel synthesis since logic will be well distributed over the boolean network. To realize this first step, we split the network into 6 parts corresponding to the 6 nodes of the PLA representation, simplify each node in parallel and then collapse the resulting subnetworks.

```
#########################
# Pscript file PLA/pscript :
#########################

pvm machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8
verbose
time 0

# the circuit is a 6 outputs PLA --> 6 nodes
# in the BLIF representation. Each node is first optimized
# separatly using the SIS script described online.
```

```
part 6 num6add.blif num6add
6 source online
simplify
tech_decomp -a 4 -o 4
end num6add num6add_d
merge 6 num6add_2 num6add_d

# standard synthesis (1P) is applied on 16 small partitions with
# 8 processors, at a first optimization step.
time 0

part 16 num6add_d num6add_d
16 source 1P num6add_d num6add_16
merge 16 num6add_16 num6add_16

# the circuit is re-synthesized with 8 partitions only, to
# improve quality.
time 0
part 8 num6add_16 num6add_8
8 source 1P num6add_8 num6add_opt
merge 8 num6add_opt num6add_opt
time 0
```

- The resulting network is very large. We are going to optimize it with an efficient but computation-intensive SIS script (1P). Thus we partition it into a lot of subnetworks (16) to reduce the runtime.
- The network is optimized, but lots of global optimizations have not been realized, since the number of partitions is large. Thus we re-apply the optimization and techonology mapping script with a smaller number of parts (8) to improve the final circuit quality.

These commands are stored in the `pscript` pscript. The synthesis could be started by the following command :
`% ppart -f pscript`

Environment configuration

• PVM environment must be set properly (see [1]).

• The synthesis manager `ppart` and the `sis` executable must be user-executable. `sis` (or a link named `sis`) must be available in the `$HOME/bin/$PVM_ARCH` onto each machine intended to be included into the parallel machine during the synthesis.

• The working directory (NFS accessible over the network !) is the current directory where the `ppart` program is started.

Processing

- go to working directory

  `% cd /user/ppart/examples/PLA`

- process the synthesis

  `% ppart -f pscript`

References

[1] G. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine - A Users Guide and Tutorial for Network Parallel Computing.* MIT Press, 1994.

[2] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning : Application in VLSI domain. In *Proc. ACM/IEEE Design Automation Conf.*, June 1997.

[3] B. Pottier. Resiviting smalltalk-80 blocks. In *IEEE Workshop on* FPGA*s for Custom Computing Machines*, Napa, CA, 1996.

[4] Xilinx. *The Programmable Gate Array Data Book.* Xilinx, San Jose (USA), 1994.

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