A User's Guide to PinHat Synthesis Engine

Version 0.1

For questions, email us at amoc@cs.uni-potsdam.de
http://www.cs.uni-potsdam.de/techinf/projects/amoc/

July 17, 2009

Chapter 1

Introduction

The tool-chain for the synthesis of flexible Chip Multiprocessor Systems from parallel program consists of three component groups. The first is PinHat, the GUI front end for configurations and for driving other components. The second is the synthesis engine, which in turn consists of the command line tool ILPF and external ILP and ASP solvers lp_solve and clasp. The last component group consists of FPGA synthesis tools. This guide describes ILPF from the user's perspective. The synthesis flow is shown in figure 1.1. The second component group is responsible for architectural high-level synthesis. The tool ILPF is used for schedulability analysis, ILP/ASP problem formulation, synthesis using heuristics, as well as for driving ILP and ASP solvers.

The most important aspect of this guide is the information you need to supply to the synthesis engine. As depicted in figure 1.1, these information contains precedence relationships between tasks, data traffic pattern between tasks, realtime requirements of individual tasks, cost models of processors and networks, and finally FPGA platform constraints. This information can be specified in text files, or they can be entered in PinHat's synthesis configuration wizard. In either case, it is important to understand the content or meaning of various parameters, as well as the relationship between various units.

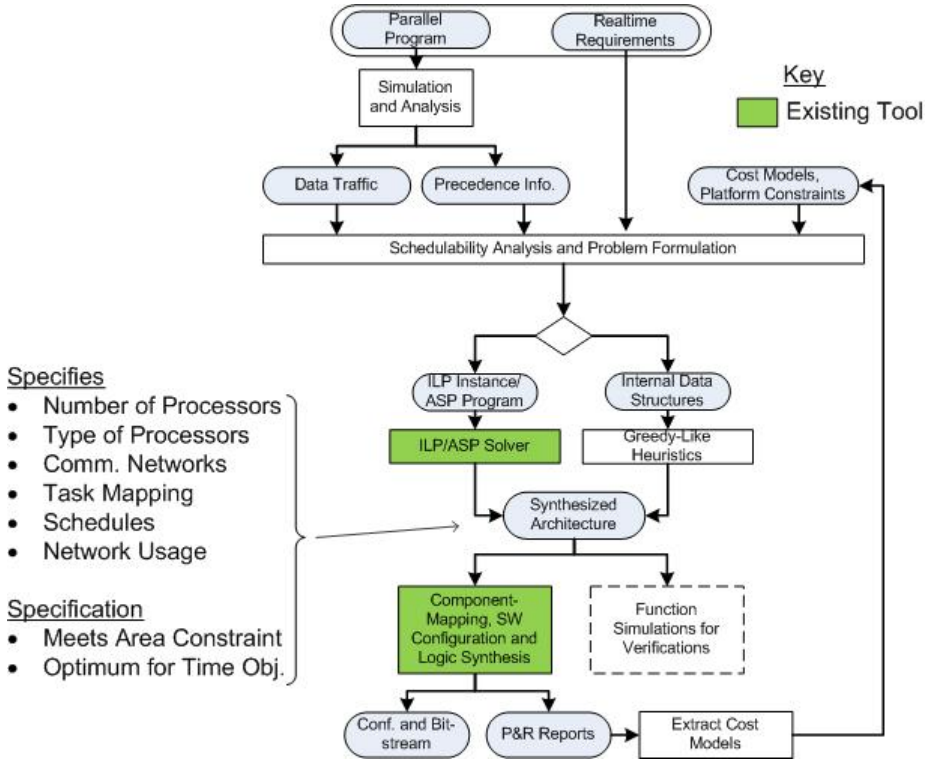

Figure 1.1: Architecture synthesis flow

Chapter 2

Command Line Options

ILPF can be started in four modes: mode 1 is ILP-based synthesis which generates model data and runs the solver lp_solve by way of the library `lpsolve55`, mode 2 accepts previously generated model data and runs the same solver, mode 3 is ASP-based synthesis which generates ASP programs and externally invoke the solver clasp, and mode 4 uses one of the three synthesis heuristics. The usage for mode 1, 3 and 4 is:

```
ilpf [-h] [-d data] [-l log] [-t timeout] -c clog -m model
-p processors -n networks -S sizes -C cycles [-P powerSetFile]
-I taskInfoFile [-M] [-A | -G | -G+r | -G+g | -G+p]
```

where [] are optional arguments.

-h: Prints help contents.

-d: When running in Integer Linear Programming (ILP) mode, ILPF invokes lp_solve and passes to it two files. The first is a model file in GNU Mathprog format, the second is the ILP data file (see lp_solve manual for more on this). This means that GNU Mathprog format allows an ILP model to be split into a generic model file, and a problem instance specific data file. Both files are read by the solver using an API defined in `xli_MathProg` library. Therefore, ILPF requires both dynamically linked libraries `lpsolve55` and `xli_MathProg` for solving and for reading the model respectively.

The option [-d] specifies the name of the GNU Mathprog ILP data file for output. This file is generated after the formulation, and can be passed as argument in mode 2. If the file is not specified, a default file `data.ilp` is generated instead.

When synthesizing in makespan mode, ILPF can go through several iterations to optimize the longest path in the graph described the parallel program. In each pass, a different data file is passed to lp_solve. Therefore, the name of the data file (regardless if passed or default) is post fixed with pass number if in makespan mode (e.g. `data.ilp_pass_3`).

-l: Specifies the name of the log file for the ILP solver. If the file is not specified, solver output will be printed to `stdout` only. A separate log file `synthesis.rpt` is generated for printouts originating from ILPF.

-t: Specifies the solver time out in seconds. The solver will abort after this time. If the parameter is not specified, a default value of 18000 seconds is used. Specifying this parameter influences the tool in this way:

- If an optimum solution is found, the solution is written out to a file `data.lp_solution`, where `data` is the output data file.
- If a user specifies the parameter and a feasible solution is found, then the solution is written out to a file `data.lp_solution_0`.
- If a default timeout is used, and a feasible solution is found, then the tool attempts to find an optimum solution by changing solver settings and restarting the solver. A maximum of three further attempts are done. Suboptimum solutions are written out to a file `data.lp_solution_n`, where n is the number of an attempt (0,1,2 or 3). Solver settings in further attempts are:

  Attempt 1:

```
set_bb_rule(lp, NODE_PSEUDONONINTSELECT +
                NODE_GREEDYMODE +
                NODE_DYNAMICMODE +
                NODE_RCOSTFIXING);
set_bb_floorfirst(lp, BRANCH_CEILING);
```

  Attempt 2:

```
set_bb_floorfirst(lp, BRANCH_FLOOR);
```

  Attempt 3:

```
set_bb_floorfirst(lp, BRANCH_AUTOMATIC);
set_mip_gap(lp, TRUE, 1.0e-8);
```

  For more on these solver settings, please refer to lp_solve manual.

-c: Specifies the name of the clog text file dump. CLOG contains logging information stemming from MPI simulations. ILPF extract traffic characteristics and task precedence information from this log file. Figure

  The log information generated from MPI-simulations is in a binary format called CLOG2. When invoking ILPF directly, you need to convert this log file first into a textual format using MPE clog2_extract utility by invoking

```
mpiexec -log -n <# tasks> -localonly <appl.> [appl. opt.]
```

to generate the binary clog2 file dump from MPI simulations, and then

```
java -jar "clog2_extract.jar" <application>.clog2
```

to convert clog2 into a text format. For more on this, please consult MPICH2 and MPE manuals.

-m: Specifies the name of the GNU Mathprog ILP model containing equations describing the synthesis problem. The currently supported version of the model is v10a. This file is located in the installation tree, and is not meant to be modified by the user.

-p: Specifies the name of the file containing processor parameters. This is a text file in which each line specifies parameters of a single instance of a processor as shown in Listing 1. The interval of the clock interrupt handler is in clock cycles. See

```
// This is a comment line.
P1,450,160e+6,64,112,3,40 // 1st instance of processor P1
P1,450,160e+6,64,112,3,40 // 2nd instance of processor P2
//| |     |   |   |  | |_ interval of clock-int. handler.
//| |     |   |   |  |____ID code for kernel model.
//| |     |   |   |_______Task switching overhead.
//| |     |   |___________Size of program memory.
//| |     |_______________Clock frequency.
//| |_____________________Processor area on FPGA.
//|_______________________The name of the instance.
```

Listing 1: Format for processor cost

remarks below for the units in the paragraph for clock frequency.

The ID code for the kernel model specifies the type of the kernel used on each processor instance. The model influences how task switching overhead is computed in the formulator, and whether there is a feasible schedule for a group of tasks. Mixing realtime and non-realtime models is not yet supported. Currently supported models are shown in table 2.1.

The task switching overhead is the actual cost of switching context in clock cycles (the parameter t_j). See remarks below for units in the paragraph for clock frequency.

The size of the program memory is used to determine during optimizations if a certain task, or group of tasks, can fit into this processor's instruction memory. Even though not enforced by the application, the units given here must be the same as those given in task size file (see below).

Table 2.1: Supported Kernel IDs

ID	Kernel	Remarks
0	Cyclic	This is a basic cyclic executive. There is no scheduling, so all group of tasks can be mapped on the processor, and there is no switching cost.
1	Timed	This model assumes that each task will be activated by interrupts. i.e. if an interrupt occurs, one or more tasks in a task group will execute. The group has a feasible schedule if the sum of all task durations $\sum T_{ij}$ is less than the interrupt interval. The switching cost is zero because there is no preemption. In this notation, i and j are task and processor indices respectively.
2	Fixed-Priority	Switching cost and schedule feasibility is computed using Rate Monotonic Analysis.
3	Round Robin	This is a round robin preemptive model. Any group has a feasible schedule. Switching cost is computed as $\lceil \sum T_{ij} / \text{clock interrupt interval} \rceil$.

The clock frequency does not go directly into the ILP model, but is used to derive other time units. For example, if the supplied parameter is in MHz, then other time units (e.g. clock interrupt handler interval) will be converted into μS. While not enforced by the tool, the units must be consistent for all processor instances.

The processor area on FPGA is used for area constraint. Any appropriate metric and units can be given as these are not further interpreted. However, units and metrics for all processors must be consistent, and they also must be consistent to those given under network parameters.

The name of instances are not interpreted in the formulator: their use is to help the designer differentiate between different instances in generated solutions. There is no restriction other than that a name must be representable as a C++ string.

Note that currently the operating system is assumed to have a negligible overhead with respect to processor cycles.

-n: Specify the name of the file containing network parameters. This is a text file in which each line specifies parameters of a single instance of a network as shown in Listing 2.

The type of the network is for readability only. It is not used in the formulator, but must be specified.

The probability bound is a positive number that specifies the bound on network arbitration time. The user should estimate this parameter for each network based on knowledge of the protocol of the network in question: i.e. what is the worst case wait time that a data transfer can incur, if several tasks or processors contend to use the same network? Note that this parameter is related to Quality of Service (QoS) guarantees for complex networks. See the paragraph below on units.

```
// This is a comment line.
fsl_1,2,451,6.25e-6,0,0,fsl // 1st instance of type FSL
fsl_2,2,451,6.25e-6,0,0,fsl // 2nd instance of type FSL
// |  |  |    |     | |  |____ network type.
// |  |  |    |     | |_______ probability bound.
// |  |  |    |     |_________ arbitration probability.
// |  |  |    |_______________ network latency.
// |  |  |____________________ network area.
// |  |_______________________ network capacity.
// |______________________________ instance name.
```

Listing 2: Format for network file

The arbitration probability is the probability that arbitration will occur due to contention of network resources. This is a number between 0 and 1. Invalid values will cause the formulator to abort with an error. The user should estimate this parameter in conjunction with arbitration bound.

The transfer latency for a word for the network is the time needed to completely transfer a word from source to destination. For complex networks, the user should estimate this parameter from QoS guarantees.

The network area on FPGA is used for area constraint. Any appropriate metric and units can be given as these are not further interpreted. However, units and metrics for all networks must be consistent, and they also must be consistent to those given under processor parameters.

The capacity specifies the maximum number of processors that can be attached to the network as discussed.

The name of instances are not interpreted in the formulator: their use is to help the designer differentiate between different instances. There is no restriction other than that a name must be representable as a C++ string.

Multi-hop networks are specified using the same format. Parameters given should reflect QoS guarantees.

All time units must be the same as those that will be derived from the clock frequency in the processor file. The tool performs no checking. If for instance the time unit for the parameters given here are in seconds, but the frequency is in MHz, then results will be falsely biased against network usage.

-S: Specifies the name of the file containing task sizes on processors. This information is used to determine which tasks can fit in the program memory of any of the processors as specified by the option [-p]. Sizes given in this file must be consistent with the size parameter in the processor file. The tool performs no checks. The user can estimate these parameters by compiling the tasks for each

```
/* specify the area constraints */
param A_pe  := 31584;
param A_net := 31584;
/* specify the table with task size on
each processor */
param taskSize : J0 J1 J2 J3 J4 J5 :=
            I0 5 5 5 5 5 5
            I1 5 5 5 5 5 5
            I2 5 5 5 5 5 5 ;
```

Listing 3: Format for task sizes on processors

of the individual processor. The format is as shown in Listing 3. Note that the format is essentially a table, where each row specifies the size of a given task on a given processor. Processor indices are J0, J1, etc, whereas task indices are I0,I1, etc. Also note that this file specifies two additional parameters: FPGA size constraints for processing elements A_{PE} and for networks A_{net}. These two constraints must be consistent with area parameters specified for processors and networks in corresponding files. The tool performs no checks.

-C: Specifies the name of the file containing task durations on processors (parameters T_{ij}) as shown in Listing 4.

```
/* specify the table with task durations on
each processor */
    param procTime : J0 J1 J2 J3 J4 J5 :=
     I0 6032e-9 6032e-9 1034e-9 1034e-9 11e-9 11e-9
     I1 3504e-9 3504e-9 601e-9 601e-9 601e-9 601e-9
     I2 909e-9 909e-9 909e-9 156e-9 156e-9 156e-9 ;
```

Listing 4: Format for task cycles on processors

Note that the format is essentially a table, where each row specifies the duration of a task on a processor. While not enforced, time units must be consistent with those given or implied in network and processor file. The information in this file is also used for computing task priorities and in schedulability analysis.

The user can estimate these parameters in several ways: using macro-modeling, from cycle accurate simulations, or less accurately, by extrapolating cycles based on processor MIPS.

-P: Specifies the name of the file containing the power set of the task set. The power set is simply a listing of all possible grouping of tasks. The information is used

for schedulability analysis. Listing 5 shows the format (only the last 5 lines for 9 tasks are shown).

This file is automatically generated by PinHat, and is only required for ILP and ASP synthesis modes.

```
I0 I1 I2 I4 I5 I6 I7 I8 I9
I0 I1 I3 I4 I5 I6 I7 I8 I9
I0 I2 I3 I4 I5 I6 I7 I8 I9
I1 I2 I3 I4 I5 I6 I7 I8 I9
I0 I1 I2 I3 I4 I5 I6 I7 I8 I9
```

Listing 5: Format for power set

-I: Specifies the name of the file containing task information (period, rate and deadline) as outlined in Listing 6. The file is generated by PinHat based on user input.

```
 I0 65 1 65 10240
 I1 4160 1 4160 159
//|   |  |   |   |__number of times the task ran
//|   |  |   |______deadline
//|   |  |__________rate
//|   |_____________period
//|_________________task ID
```

Listing 6: Format for task information

The number of times a task has ran allows the user to influence if the parameter the execution time for each task on each processor should be interpreted as being the time for one period of task execution, or for the total duration of the application. ILPF always treats the execution time as is this were a parameter for one period. To obtain the total duration, the execution time is multiplied by the number of times the task has ran. Setting the number of runs to 1 means that the optimization result is based on one period for that task. Specifying a number less than 1 will cause the formulator to abort with an error.

The deadline is used in schedulability analysis and in assigning priorities. This parameter is ignored if the kernel model does not consider deadlines. The time unit must be consistent with those given in network and processor file.

The rate is a reserved parameter currently not in use.

The period is also used in schedulability analysis and in assigning priorities. The time units must be consistent with those given in network and processor file.

The user should specify these parameters based on realtime requirements of the tasks.

-M: Toggles the Makespan mode. In this mode, ILPF attempts to optimize the most critical path in the parallel program, in the process considering execution times, data transfer times, as well as scheduling and network overheads.

In this mode, the parallel program must be representable as a Directed Acyclic Graph (DAG), and it must be connected. Otherwise, the makespan is undefined, and the formulator aborts with an error.

If this mode is not given, the overall execution time of the parallel program is optimized. In this later case, there is no restriction as to the structure of the application graph.

Note: The application graph is not directly specified, but is constructed from the CLOG file specified by the option [-c].

-A : Toggles synthesis in Answer Set Programming (ASP) mode. In this mode, ILPF invokes the ASP solver clasp, and passes to it an ASP program describing the synthesis problem. This mode requires special attention because the ASP-solver accepts integral values only. Therefore, scaling of time parameters is required to make sure that the integer range is not exceeded during the computation of ASP stable models (i.e. the solution). If that is not the case, and overflows occur, the result cannot be trusted. This mode should not be used for large problem instances.

The sizes should be scaled such that integral numbers can be specified when synthesizing in ASP mode because the tool will round up the figures entered in order to meet solver's integrality restrictions.

In this mode, the synthesis tool generates ASP programs and invokes Clingo for grounding and resolution. No options are used. ASP programs are written out to `<data file base name>.lp_pass_n`, where the base name of the data file is optionally specified by [-d], and n is the pass number when synthesizing in makespan mode. Clingo enumerates found stable models, which are piped to `<data file name>_pass_n.asp_solution`. The last model in the file is the best one. Additionally, a report file `synthesis.rpt` is generated.

-G Toggles greedy heuristic mode. All options apply as in -A mode, but in this case, the powerset file can be omitted since scheduling analysis is computed as needed, rather than completely in advance. This mode produces the report file `synthesis.rpt` only, which in this case also contains the solution.

Greedy mode can optionally be supplemented by "plus" options which further specify one of the three heuristics.

-G+r This is the default mode, equivalent to -G. The default mode uses the "replace and search" heuristic.

-G+g This uses the "group growing" heuristic.

-G+p This uses the "priority" heuristic.

The usage for mode 2 is:

```
ilpf -s -d data -m model
```

where switches -d and -m have the same meaning as discussed above. The switch -s toggles mode 2.

Chapter 3

Usage guide for different modes

Synthesis in ILP mode is the slowest, but also leads to accurate results. You should use this mode for relatively small problem instances with less than 10 processor instances, 4 network instances, and less than 12 tasks.

Synthesis in ASP mode is significantly faster by up to three orders of magnitude. For problems which are tough to solve in ILP mode, you should use ASP mode provided that the problem size and the scaling used is such that no overflows can occur.

The three heuristics are the fastest, really fast even for huge problems. You should use this mode if synthesis in ASP mode cannot be conducted for reasons mentioned. In that case, you should conduct synthesis using all three algorithms sequentially, and select the best result. We recommend this because it has been observed during experiments that at least one of the heuristics yield architectures which are close to those obtained using ILP mode.