XML-based Parametrization of SystemC Models

Dr. Andreas Döring
a d o @zurich.ibm.com
IBM Research Laboratory GmbH
Rüschlikon/Zürich

Content

- SystemC projects at ZRL
- Configuring SystemC models from XML
- Multiple Instances of the same thread

Network Processor Model

- Patricia Sagmeister, Frederic Worm, Srihari Narasimhan, Fabien Pouget
- Busses, Components (MAC, Memories,CPUs) for NPUs
- Design-space exploration
  - Optimization of bus arbitration (3 Busses: OPB, 2xPLB)
  - Parametrization of On-chip buffer sizes
  - Memory access policy (DRAM: page opening, R/W,...)
  - CPU cache models
- Three master thesis‘ (EURECOM & Stuttgart)
- 60K Lines of code
- Configuration uses constants in .h files

Switch Simulation Framework

- Mark Verhappen, Claudio Favi
- Use of XML for Configuration
- Evaluation of architectures for backbone switches
- 2500 Lines of Code

Timer Coprocessor

- Silvio Dragone, Philipp Roß (Lübeck)
- 5500 LoC
- Evaluation of an architecture for a scalable Timeout coprocessor with several levels of memory
- Few configuration parameters (e.g. external memory latency)
- Little information on expected use profile available
- Configuration uses proprietary syntax

Prioritized Multithreaded Processor

- Maria Gabrani & Andreas Döring
- 6200 LoC
- Simulate a superscalar, multithreaded processor with instruction-granularity prioritization based on
  - Input from executed thread
  - Steering from external sources (semaphor manager)
  - Under Control of the Operating System
- Use stochastical program model (distribution of instructions, register dependencies, cache hit rate)

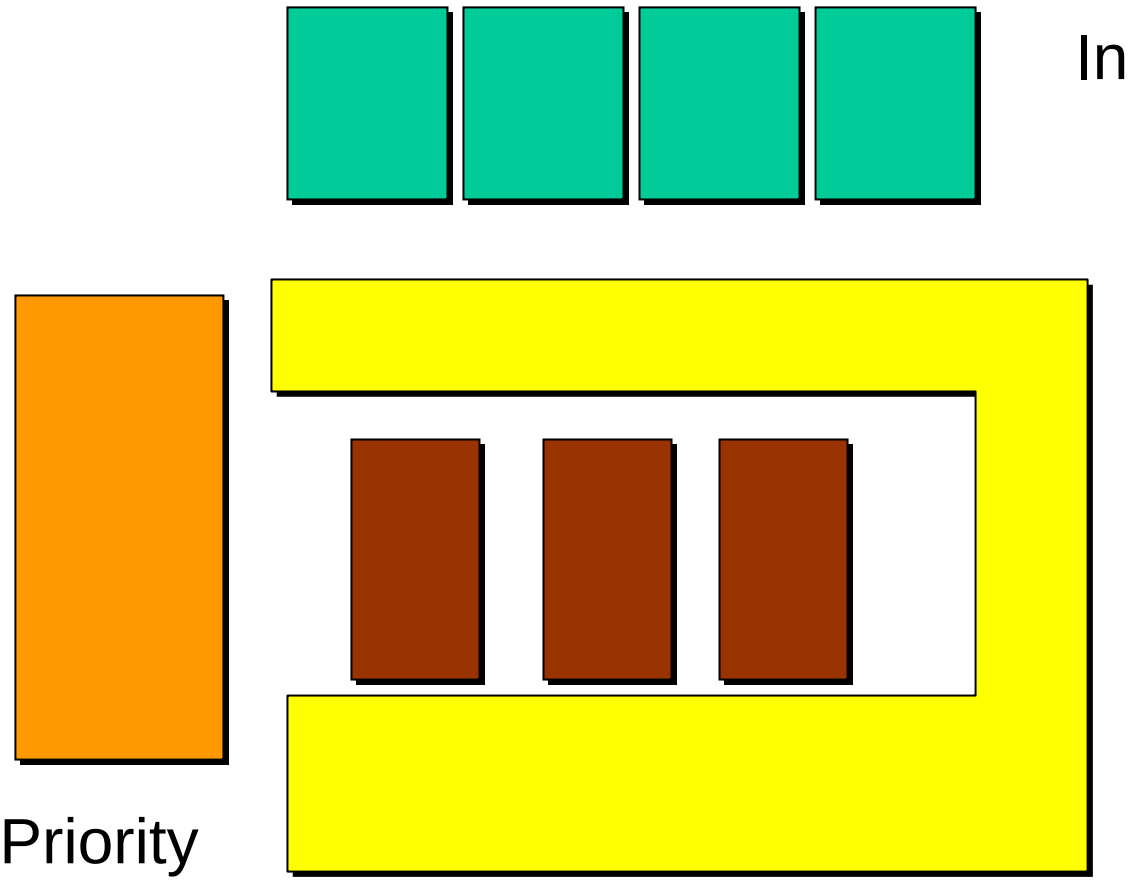
PSMT Configuration

- Configurable Structure
  - Number of threads, number and types of pipelines
- Configurable Parameters
  - Unit behavior (latency, capacity, instruction distribution, branch prediction, issue policy)
- Configurable output detail

PSMT Libraries

- SystemC (from systemc.org) 2.01
- Newrand (TBD)
- XerxesC (apache.org)
- STL (sgi.com)

PSMT Structure



XML Configuration

- Call parser in top-file on startup
  - ```
    XMLPlatformUtils::Initialize();
    cfg = new Config();
    cfg->Init(argv[1]);
    ```
- Command-line argument selects parameter set
- Wrapper class
  - Translates SystemC module name to hierarchical XML query (s/\./\//g)
  - Member function for each desired type (double, int, string)
    ```
    cfg->Get(namebuf,decodewidth);
    ```
    ```

Example

```
<run_8_4_5ii
    DecodeWidth="4"
    PipelineNumber="5"
    ThreadNumber="8">
    <Issue
        Verbose="$verbose"
        Prioritized-Complete="no"
        Issuewidth="3"
        Completionwidth="4"
        Issuebuffersize="4"
       Out-of-Order-Issue="n"
       Out-of-Order-Complete="n"
       SpeculateBranches="yes"
       StallOnLoad="n">
  </Issue>
...
< /run_8_4_5ii>
```

XML output

- Simple “<<”
- Arrays? (e.g. histogram on completion buffer use)
- Concatenation of results of individual simulation runs
- Shell script adds file header/footer

Multiple Threads with one function

- Needed a configurable number of instances of the same thread (discarding of instructions from scoreboard for each execution pipeline)
- Each thread needs to get its index
- More general: pass parameters through

Outlook

- Use XML processing tools for evaluation and visualization, e.g. OpenOffice Spreadsheet
- Standardize presentation for typical result structures, e.g. histogram
- Allow more reuse of monitoring functions

Backup SC_THREAD_I Macro

```
#define SC_THREAD_I(func,i)                    \
   char strgbuf[30];                           \
   sprintf(strgbuf,#func  "_%d",i);            \
   declare_thread_process( func ##             \
  _handle,                                      \
   strgbuf,                                    \
   SC_CURRENT_USER_MODULE,                     \
   func )
```

Parameter passing

```
private:
map<sc_thread_handle,int,lt_sc_thread_handl
  e,malloc_alloc> parameter_passing;
…
SC_HAS_PROCESS(Cissue) …
for (i=0;i<numberofpipes;i++)
{SC_THREAD_I(branch_discard,i);
   sensitive << Ibranch_failed_perpipe[i];
              parameter_passing
  [branch_discard_handle]=i;
```

Picking up parameters

```
sc_thread_handle myhandle;
    myhandle =
  (sc_thread_handle)sc_get_curr_simc
  ontext()->get_curr_proc_info()->
  process_handle;
parameter_passing.find(myhandle);
    discard_pipe =
  parameter_passing[myhandle];
```