GAELS Progress

Wei Song
31/08/2012
Content

• Tool flow
• Progress
  – Verilog Parser
  – Tcl user interface
  – Petri-Net graphic library
• Future works
• Issues
• Conclusion
Tool flow

- RTL Verilog HDL
- Cell Library
- VCD waveform

AVerilog synthesiser

- Timing info
- Pipeline usage

- Multiple smaller RTL Verilog HDL designs
- Async interfaces
- Sync elastic pipelines

Commercial tools

blackboxes
dont touch
Flow inside Synthesizer

- RTL Verilog
- Verilog Parser
- Elaborator
- Timed Colored Petri-Net (TCPN) Extraction
- TCPN Simplification
- GALS Partition
- Asynchronous Component Library
- Async Pipeline Insertion
- Netlist Writer
- Async Gate-Level Verilog
- Automatic Constraint Generation
- Syn, P&R constraints
- Frequency Constraints
- Area, Power constraints
- Simulation Waveform
Progress from Last Meeting

• Verilog Parser
  – More supported features

• User Interface
  – A fully embedded Tcl interpreter (v8.5)

• Petri-Net (PN) Library
  – Support hierarchical TCPN (expected)
  – PNML standard (and dot, GML, SVG)
  – Automatic layout for GUI
Verilog Parser

• Preprocessor (Macro support)
  – VPreProcessor from Perl-Verilog tool suite
  – https://github.com/wsong83/vpreproc
  – Full language features (SystemVerilog)

• Parser
  – Understand all synthesizable Verilog

• Semantic (Paring tree)
  – Parameter, module, input/output port, reg/wire/integer, always, <=, =, if/else, case
  – Features not supported yet:
    • Inout port, for loop, generation block, library gates
Verilog Parser

• Elaboration
  – Automatic parameter expansion
  – Module renaming (parameter suffix)
  – Hierarchical module linkage
  – Port direction check
  – Multi-driver, no-driver and no-load check
  – Conservative simplification (preserving logic rationales between signals and always blocks)
Verilog Parser

• Verification (no error coverage)
  – Read in OpenRISC 1200 processor
  – One line change in the source code:
    wire flag = 1'b1;
    Change to
    wire flag;
    assign flag = 1'b1;

• Small demo later (with Tcl UI)
User Interface

• Reasoning for CMD env.
  – Large scale designs (no schematic design view)
  – Command line environment is efficient and has a low memory footage
  – Synchronous users are familiar with it
  – GUI may not be useful when designs are large

• Solution
  – Full embedded Tcl interpreter
  – Extra tool related Tcl commands and global variables
  – Special support to display TCPNs
C++/Tcl

- C++/Tcl
  - A C++ / Tcl interface library
  - [https://github.com/wsong83/cpptcl](https://github.com/wsong83/cpptcl)
  - Design by Maciej Sobczak (2004-2006)

- Features:
  - C++ wrapper for Tcl C APIs
  - Easy command expansion (my addition)
  - Read/Write Tcl variables
  - Tracing Tcl variables (my addition)
Demo

• Parsing and elaboration of OR1200

  ./bin/avs_shell

  > source ../test/avs_test.tcl

  > elaborate or1200_top

  > write -hierarchy

  > exit
Petri-Net Library

- CppPNML library
  - https://github.com/wsong83/cppPNML
  - C++ PN Graphic library
    - Wrapper C++ classes to hide internals
    - Boost Graphic Library to store diagrams
    - Multi-maps/sets (associated containers) to store indices and identifiers
    - Open Graphic Design Framework (OGDF) for automatic layout
Example: a simple PT-net

```plaintext
cpynPlace pstart("p1", "start");
g.add(pstart);
pstart.setInitMarking(1);
g.add(pnTran("t1", "transition");
g.add(pnPlace("p2");
g.add(pnPlace("p3", "arbiter");
g.get<pnPlace>("p3").setInitMarking(2);
g.addF(pnTran("t2", "choice1")).add(pnTran("t3", "choice2");
g.add(pnPlace("p4");
g.add(pnTran("t4", "finish");
g.add(pnPlace("p5", "finish");
g.addF(pnArc("a1", "p1", "t1").add(pnArc("a2", "t1", "p2");
g.add(pnArc("", "p2", "t2", "choose 1");
g.add(pnArc("", "p2", "t3", "choose 2");
g.addF(pnArc("", "p3", "t2").add(pnArc("", "p3", "t3");
g.addF(pnArc("", "t2", "p4").add(pnArc("", "t3", "p4");
g.addF(pnArc("", "t2", "p3").add(pnArc("", "t3", "p3");
g.add(pnArc("", "p4", "t4");
g.add(pnArc("", "t4", "p5");
```
Dot vs OGDF

OGDF:Sugiyama Layout

GraphViz: Dot
MNMA allocator
MNMA: OGDF
MNMA: Dot
Format supported

• Output formats
  – PNML (place, initial marking, transition, arc) (color, set, guard, read arc)
  – GraphViz: Dot
  – GML and SVG (no token)

• Input formats
  – PNML (pugixml XML parser)
  – GML (internal use)

• pnml2pdf (Qt 4.7)
Tool summary

- AVS (Asynchronous Verilog Synthesiser)
  - [https://github.com/wsong83/Asynchronous-Verilog-Synthesiser](https://github.com/wsong83/Asynchronous-Verilog-Synthesiser)
  - Third party tools / libraries used:
    - GNU C++ / C++0x / Boost
    - Bison / Flex
    - GNU MP Lib
    - Tcl/Tk 8.5
    - C++/Tcl
    - VPreProcessor (embedded)
    - OGDF 2012.07
    - Pugixml (embedded)
    - Qt 4.7
Future works

- cppPNML library
  - couple of months
  - Reference node, color, set, guard, time, arcs
- TCPN extraction
  - Starting from September
  - Hopefully some results in next meeting
Issues: why TCPN?

- Place Transition (PN) net is difficult to represent conditions.

```vhdl
always @(posedge clock)
  if(a)
    b <= c;
  else
    b <= d;
```

![TCPN Diagram]
always @(posedge clock)
  if(a)
    b <= c;
  else
    b <= d;
always @(@posedge clock)
if(a)
    b <= c;
else
    b <= d;
always @(posedge clock)
    if(a)
        b <= c;
    else
        b <= d;
always @(posedge clock)
  if(a)
    b <= c;
  else
    b <= d;

Well, this works.
What is the practical meaning of sinks?
Anyway to simplify it?
Conditions using PN-Net

always @ (posedge clock)
if (a)
  b <= c;
else
  b <= d;

The CPN also clearly shows a, c, d
are related. Or, they can be bundled
together.
Issues: why TCPN?

• How to represent clock and flip-flops?
  – Does clock matter? YES
Issues: why TCPN?

\[ n \rightarrow n + 1 \]

If \( n = 5 \) then token
Else empty

\[ @ \rightarrow @ + 5 \]

If \( n = 5 \) then token
Else empty

\[ @ + 1 \]
Simplification using TCPN
Problem with PNML

- Petri-Nets described by PNML are uniquified!
  - Hierarchy is supported by page and refnode.
  - Refnode in PNML must reference to a unique node
    <referencePlace id="ref_id" ref="org_id">  
  - Every module represented by a page in PNML can have only one entity (uniquified).
  - Modular PNML
Conclusions

• Still working on tool preparation
  – Parser, UI and Graphic Library

• Try to extract Timed Colored PN from RTL designs.
  – Simple nets
  – Simplification