autoVHDL: A Domain-Specific Modeling Language for the Auto-Generation of VHDL Core Wrappers

Erica Jones, Jonathan Sprinkle
Domain motivation for “correct by construction” embedded systems
Before they fly....a PRE-prototype is tested -- HARDWARE IN THE LOOP!

NOTE: Images are representative of hardware in the loop test cases, but are images gathered from various contextual searches, not representing systems discussed in this paper.
But the problem doesn’t have to be missiles---all HWIL domains apply:

• Just pretend like we are using this technology to make computing safe for kittens.
• HWIL prototypes must realistically capture the information flow architecture of the designed system
  – Bus width/frequency must be matched
  – Proper voltages, interfacing
  – Communications engines must comply with standards
• “Core” technology is deployed into reconfigurable hardware
  – Speedy execution
  – Realistic power draw
  – Componentized behaviors, available for reconfigurable hardware execution
    • e.g., 8b10b, UART, SDLC
• “Core” technology difficult to rapidly reuse, due to subtle differences in the interface specification, depending on the domain-specific application of the cores
• Much of the difficulty lies in rapidly rehosting a core on different hardware, with different communication specification pathways, due to the distributed specification of the communication pathways in hardware
Why hasn’t it been done already?

- Significant effort in developing VHDL
- How to improve on VHDL?
  - Many folks tried to create “simply” more clever graphical versions
  - However, all of these approaches resulted in clumsier languages, that were unable to solve the entire design space---so they were “failures”
- This approach *limits* the intended application space
- The *domain-specific* approach permits us to focus on a particular application where we can have impact
  - Thus, the fact that there are some things that “raw” VHDL does better than us is still OK---we are not trying to do that
  - Likewise, those folks who are actually working in this domain can utilize this tool to more rapidly stand up hardware pre-prototypes for testing
Contribution of this work

- A domain-specific modeling approach to generate the high-level description language for hardware deployment, based on an application model
- The language relies on centralized specification of the domain concepts, and uses a graphical syntax
- The generated files are able to be synthesized on board standard hardware, and executed in order to validate design criteria
- **VHDL**: VHSIC Hardware Description Language
- **Hmmmmmm:**
  - VHSIC: Very-High-Speed Integrated Circuits
- An architecture description, to tell a chip how to arrange itself

```vhdl
-- sqrt8m.vhdl unsigned integer sqrt 8-bits computing unsigned integer
-- sqrt(00000100) = 0010  sqrt(4)=2
-- sqrt(01000000) = 1000  sqrt(64)=8
-- modification of sqrt8.vhdl with specialized circuits

library IEEE;
use IEEE.std_logic_1164.all;

entity Sm is  -- subtractor multiplexor
  port ( x  : in  std_logic;
         y  : in  std_logic;
         b  : in  std_logic;
         u  : in  std_logic;
         d  : out std_logic;
         bo : out std_logic);
end Sm;

architecture circuits of Sm is
  signal t011, t111, t010, t001, t100, td : std_logic;
begin  -- circuits of Sm
  t011 <= (not x) and y and b;
  t111 <= x and y and b;
  t010 <= (not x) and y and (not b);
  t001 <= (not x) and (not y) and b;
  t100 <= x and (not y) and (not b);
  bo   <= t011 or t111 or t010 or t001;
  td   <= t100 or t001 or t010 or t111;
  d    <= td when u='1' else x;
end architecture circuits;  -- of Sm

library IEEE;
use IEEE.std_logic_1164.all;

entity Sb is
  port ( x  : in  std_logic;
         y  : in  std_logic;
         b  : in  std_logic;
         bo : out std_logic);
end Sb;

-- VHDL code for Sm and Sb entities for subtractor and multiplexor
```
This paper: NOT a tutorial on VHDL...let's look at the domain:

**Bus Module**
- **8b10b Data Register**
- **Control Register**

**Top Module**
- **Xilinx CoreGen 8b10b Encoder**
  - **Data[7..0]**
  - **kin[0]**
  - **ce[1]**

**Connections**
- **clock**
- **start**
- **addressBus**
- **rd/Wr’**
- **dataBus**
- **ack**
- **dout**
- **kerr**
- **disp_out**
- **clk8b10b**
Metamodel: Interesting Subset

 Electrical and Computer Engineering

autoVHDL
<<Model>>

Bus Module
<<Model>>
componentName : Field

FIFO
<<Model>>
componentName : Field

Core
<<Model>>
componentName : Field
Example Model: 8b10b

Bus Module

Top Module

Simple Bus Timing Diagram

Timing Diagram Defines Hi/Lo specs, etc.
Applications with FIFO buffer require extra details---separated as in #defines (#dsm11)

FIFO Model Atomic Parts (Subset Shown)

- **FIFO**
  - RD EMPTY
  - WR REQ
  - Q
  - ED FULL

- **DATA**
  - data
  - wrClk
  - rdClk
  - wrFull

- **FIFO**
  - rdEmpty
  - wrReq
  - q
  - rdFull

Bus Timing Diagram

- Clock
- Start
- Data/Addr
- X"0004" X"F00D" X"ZZZZ"
- Addr Sel
- Data Sel
- Write Sel
- Read Sel
- Acknowledge

Top Module

- Bus Module
  - Tx Data Register
  - Tx FIFO Status Register
  - Interface Control Register

- Async FIFO
  - txFifoWrClk
  - txFifoWriteReq
  - txFifoOutputData
  - wrEmpty
  - wrFull

- Custom Serial Tx Interface
  - txDataOut
  - txClkOut

- Enable
- errorInject
- bitRateSel
- modeSel

Clock
- start
- dataAddrBus
  - addrSel
  - dataSel
- wrSel
- rdSel
- ack
### Results from 8b10b Example

<table>
<thead>
<tr>
<th>Slice Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers</td>
<td>1,296</td>
<td>19,200</td>
<td>6%</td>
</tr>
<tr>
<td>Number used as Flip Flops</td>
<td>1,294</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as Latches</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Slice LUTs</td>
<td>478</td>
<td>19,200</td>
<td>2%</td>
</tr>
<tr>
<td>Number used as logic</td>
<td>478</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number using O6 output only</td>
<td>446</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number using O6 and O6</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slice Logic Distribution</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Occupied Slices</td>
<td>423</td>
<td>4,800</td>
<td>8%</td>
</tr>
<tr>
<td>Num. of LUT FF pairs used</td>
<td>1,347</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number with an unused FF</td>
<td>51</td>
<td>1,347</td>
<td>3%</td>
</tr>
<tr>
<td>Number with an unused LUT</td>
<td>869</td>
<td>1,347</td>
<td>64%</td>
</tr>
<tr>
<td>Num. fully used LUT-FF pairs</td>
<td>427</td>
<td>1,347</td>
<td>31%</td>
</tr>
<tr>
<td>Number of unique control sets</td>
<td>68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IO Utilization</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of bonded IOBs</td>
<td>27</td>
<td>220</td>
<td>12%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specific Feature Utilization</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of BUFG/BUFGCTRLs</td>
<td>2</td>
<td>32</td>
<td>6%</td>
</tr>
<tr>
<td>Number used as BUFGs</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
And...it actually synthesizes onto hardware

**Synplify Timing Report**

<table>
<thead>
<tr>
<th>Starting Clock</th>
<th>Requested Frequency</th>
<th>Estimated Frequency</th>
<th>Requested Period</th>
<th>Estimated Period</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>bus_module</td>
<td>clkIn</td>
<td>377.4 MHz</td>
<td>320.8 MHz</td>
<td>2.650</td>
<td>3.118</td>
</tr>
</tbody>
</table>
Note: ‘rstin’ --> Reset/Config (added by default to reset application)
• The domain-specific modeling language autoVHDL has shown proof of concept for rapid automation of VHDL core reuse
  – Use of domain types to take bus timing diagrams and structural models and generate synthesizable VHDL files
  – Code generator uses best practices for utilization, naming conventions, allocation practices, etc.
• Does not replace or supplant existing visual tools for VHDL
  – Rather, focuses on core module reuse across buses for HWIL
  – Reduces specification time for test engineers, without requiring them to clone existing tests
• This work supported by the National Science Foundation under awards CNS-0915010, CNS-0930919, and by AFOSR award FA9550-091-0519
We are always looking for good graduate students.

http://ece.arizona.edu/