California Institute of Technology
Department of Computer Science
Computer Architecture

CS184b, Spring 2005 # Assignment P1: VHDL and Queues Monday, April 4

Due: Monday, April 11, 9:00AM

Goals:

- Learn VHDL
- Become comfortable using CAD Tools from behavioral description to place-and-route
- Develop fast queues for use in project

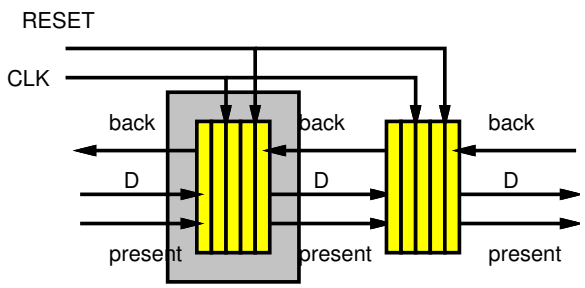
Collaboration: This assignment remains an individual assignment. We want **everyone** to be able to write VHDL and use the tools. You are required to do your own designs and complete your own writeup. You may help each other with the tools and help each other debug.

Target: All designs should target a Virtex2-6000-4.

Tasks:

1. Build a sorting box.
   Input: two 16b, unsigned integers (`in0` and `in1`).
   Outputs: max and min of the two inputs (`min` and `max`), also 16b integers
   Design should be pipelined to take one input every cycle
   (a) Write Debug/VHDL (turnin your VHDL code)
   (b) Verify correct operation (turnin your testing script)
   (c) Synthesize/place/route (turnin post-p&r report showing area and cycle time)

2. Build a FIFO queue.



Input: 16b data value (`din`), 1b data presence for din (`presentin`), 1b back pressure for dout (`backout`), Clock (`CLK`), Reset (`RESET`)
Outputs: 16b data value (`dout`), 1b back pressure for din (`backin`), 1b data presence for dout (`presentout`)
Depth: 16 (may make a parameter)
Operation:

- Assert data presence when not empty.
- Assert back pressure when prepared to take an input (*e.g.* not full).
- If back pressure is asserted and input data presence is asserted, the item is transfered between the source and sink (source pops data item and the sink pushes data item)

(a) Write Debug/VHDL (turnin your VHDL code)

(b) Verify correct operation (turnin your testing script)

- name inputs `din`, `presentin`, `backout`, `CLK`, `RESET` and outputs `dout`, `backin`, `presentout`

(c) Verify correct operation (turnin your testing script)

- single queue
- three queues back-to-back

(d) Synthesize/place/route (turnin post-p&r report showing area and cycle time)

3. Optimize FIFO queue for high clock rate. Pipeline and retime signals so queue as necessary. Try for 200MHz operation.

(a) Describe your pipelined handshaking discipline

(b) Write Debug/VHDL (turnin your VHDL code)

(c) Verify correct operation (turnin your testing script)

- single queue
- three queues back-to-back

(d) Synthesize/place/route (turnin post-p&r report showing area and cycle time; show **both** for single queue and three queues)

Turnin Instructions: Create a subdirectory of your CS computer account for this assignment (*e.g.* `/home/bitdiddle/cs184b/p1/`) that contains only the files requested above. Create a master file `p1.html` in that directory which points to all the constituent files and provides any necessary explanations. An html template is at:
`/cs/courses/cs184/spring2005/assign/p1/p1.html`.
Email the location of your assignment directory to `mdel@cs.caltech.edu`.

Resources:

- *Designer's Guide to VHDL* – two copies will be available in JRG 84
- For a more comprehensive reference:
  <http://tech-www.informatik.uni-hamburg.de/vhdl/doc/cookbook/VHDL-Cookbook.pdf>
  For examples look at:
  <http://www.vhdl-online.de/tutorial/>
- For information about how VHDL designs can use Xilinx parts, look at synplify_pro documentation:
  `/cs/research/ic/software/synplify/doc/synpro_ref.pdf`
- The most useful reference for Xilinx is the VirtexII data sheet:
  <http://direct.xilinx.com/bvdocs/publications/ds031.pdf>

Tool Setup and Use:

The main tool you will use is ise, which integrates editing, compilation and simulation tools. Other tools you may use are a program editor (probably emacs or vim), synplify_pro, vsim. The tool flow for compilation to an FPGA is:

VHDL (*.vhd)→synplify_pro→EDIF (*.edf)→ISE parts→bitstream.

For simulation, use vsim with VHDL input.

A small VHDL example is in `/cs/courses/cs184/spring2005/examples/mem_accum`. For each of this assignment's three questions, you should make a project. Steps for project setup and development are:

- Set environment variables for synplify_pro and vsim licenses. If you use bash for your shell, then add two lines to .bashrc:

  ```
  export MGLS_LICENSE_FILE=1717@squid.cs.caltech.edu
  export LM_LICENSE_FILE=20006@remus.caltech.edu:1717@squid.cs.caltech.edu:1709@sash.cs.caltech.edu:5219@squid.cs.caltech.edu:$LM_LICENSE_FILE
  ```

  Other shells may use the syntax:

  ```
  setenv MGLS_LICENSE_FILE 1717@squid.cs.caltech.edu
  setenv LM_LICENSE_FILE 20006@remus.caltech.edu:1717@squid.cs.caltech.edu:1709@sash.cs.caltech.edu:5219@squid.cs.caltech.edu:$LM_LICENSE_FILE
  ```

  Copy and paste from `/cs/courses/cs184/spring2005/examples/licenses`
- Setup a project in ISE.
  - For mandrake machines, start ISE:
    `ise &`
    If running redhat and using the bash shell, then start ISE with the command:
    `LD_ASSUME_KERNEL=2.4.1 ise &`
    If you're using another shell you may need to do:
    `env LD_ASSUME_KERNEL=2.4.1 ise &`

  - File→New Project
  - Initialize project settings:
    - Fill in Project Name and Project Location.
    - Top-Level Module Type is HDL.
    - Device Family is Virtex2.
    - Device is xc2v6000.
    - Package is bf957.
    - Speed Grade is -4.
    - Synthesis Tool is Synplify Pro (VHDL/Verilog)
    - Simulator is Modelsim
    - Generated Simulation Language is VHDL
    - Can add source files later.
  - In Sources in Project window, add VHDL source files you already have by right clicking on xc2v6000-4bf957.

- Write and syntax check VHDL.
  - Make a new source or add an existing source by right clicking on on xc2v6000-4bf957. You can edit VHDL files in ISE, or with vim or emacs which interpret `*.vhd` files as VHDL.
  - Syntax check using the synplify_pro synthesis tool: Highlight the top level VHDL file in the Sources in Project window. In the Processes for Source menu, right click on Synthesize, then choose rerun. The Console window refers to the log file (`mem_accum_2_16.srr`) which reports syntax errors or other output from synthesis.

- Simulate your design using vsim.
  - Under Design Entry Utilities, click Launch ModelSim Simulator.
  - In the menu Compile→Compile Options, check Use 1993 Syntax.
  - A script should be used to stimulate signal values and step time so you can check for correct behavior in the wave window. For mem_accum, in the model sim prompt type:
    `source ../mem_accum.vsim`
  - Modify `mem_accum.vsim` to change the set of signals that are displayed, and to change stimulator behavior. To view hex rather than binary, right click on the signal then modify Radix.
  - Although the `mem_accum.vsim` example should be adequate, documentation is provided at:
    <http://www.model.com/support/documentation.asp>

- Run your design through synthesis.
  - Same as syntax checking, use Synplfy Pro.
  - Set Synthesize properties, by right clicking. Under Synthesis Options, Frequency should be about 200 MHz. Under Device Options Retiming should be on.
  - Look at the end of the log file (`mem_accum_2_16.srr`) for resource usage: Total LUTs is most interesting. Approximate timing is reported also.

    - Synplify generates the file `mem_accum_2_16.ncf` for later stages of the compilation. Remove this file after running Synplify, otherwise later stages will not be able to meet timing requirements due to some OFFSETs. Alternatively, comment out the OFFSET lines in the .ncf file.

- Floorplan and compile to a bitstream.
    - Set timing constraints the rest of the compilation: Choose rerun for Create Timing Constraints. In the main table, in the CLK x Period field enter 5 then press return. This sets the target clock period to 5 ns, which is 200 MHz.
    - Right click on Implement Design and choose rerun to compile to a bitstream.
    - Under Place & Route, Place & Route Report gives the time and area requirements of your design. This is one of the files you should submit. For the cycle time, which should be less than 5ns for 200MHz, look at the Constraint table at the end of the report. Slices used is reported near the top (One slice contains two LUTs).
    - Generate Post-Place & Route Static Timing→Analyze Post-Place & Route Static Timing can be used to view the critical path.
    - In order to floorplan, run stages Translate and Map first. Under Map, use the Floorplanner to constrain parts of your design with bounding boxes.

Relevant files for mem_accum are:

- **mem_accum.npl**: Xilinx project file.
- **VHDL files *.vhd**: main source code.
- **mem_accum.vsim**: stimulator script for model sim.
- **mem_accum.srr**: synthesis log which reports any compile time errors.
- **mem_accum.ucf**: place & route timing and area constraints. You may use ISE timing and floorplanning tools to modify constraints, rather than modifying this file directly.