HOMEWORK – TARGETING XILINX & ALTERA

Prof. Don Bouldin – 26 Sept. 2011

Purposes: Perform pre-synthesis simulation using *ModelSim*, synthesis using *Synplify_Premier_DP*, place & route for both Xilinx (Virtex) and Altera (Cyclone) parts and perform post-layout simulation using *ModelSim*.

Part A – Tutorial

Download: **http://web.eecs.utk.edu/~bouldin/protected/551-hw6.tar.gz**

or copy the files on ada3.eecs.utk.edu:

1. cp ~bouldin/webhome/protected/551-hw6.tar.gz .

2. gunzip 551-hw6.tar.gz

3. tar –xvf 551-hw6.tar

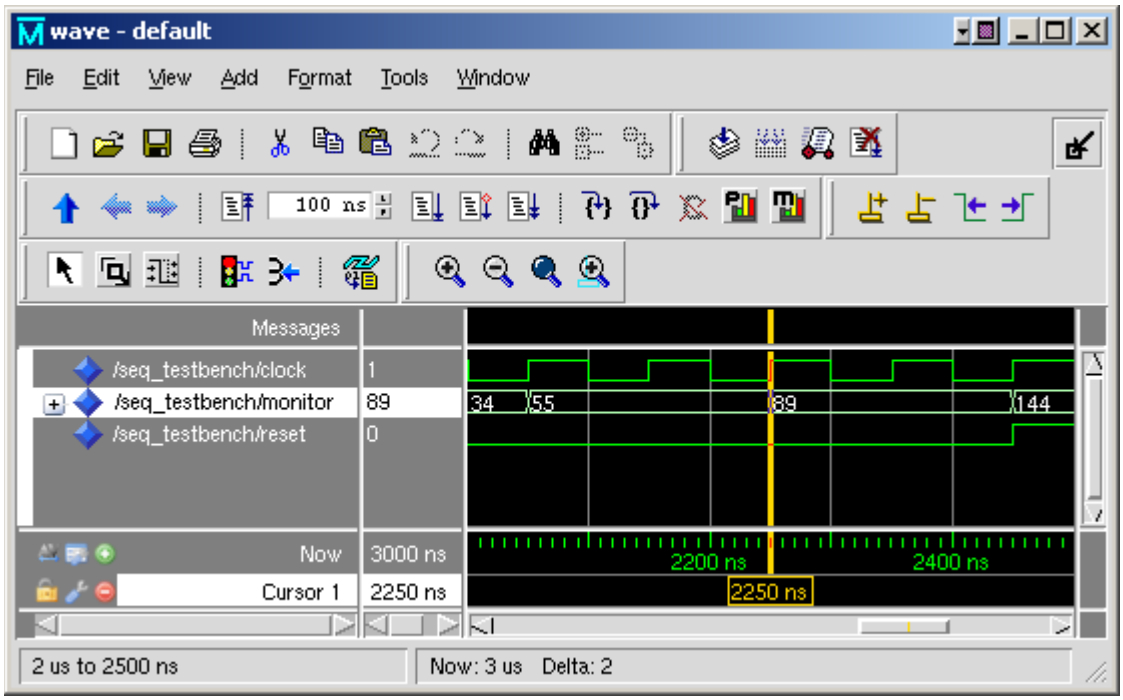
Now, move down to the subdirectory:

4. cd 551-hw6

Now, perform pre-synthesis simulation by typing:

5. ./presynth_sim

This will bring up the following windows with the radix of “monitor” manually changed to “unsigned” and selecting View→Zoom→Range→2000ns to 2500ns:



Expand the “uut” to show the coverage:

Instance	Design unit	De	V	Stmt count	Stmt hits	Stmt misses	Stmt %	Stmt graph
seq_testbench	seq_testbench(struct)	...		42	38	4	90.5%	
uut	seq_generator(struct)	...		38	34	4	89.5%	
acc_a	accumulator(spec)	...		4	3	1	75%	
truth_pr...	accumulator(spec)	...						
acc_b	accumulator(spec)	...		4	4	0	100%	
truth_pr...	accumulator(spec)	...						
acc_sum	accumulator(spec)	...		4	3	1	75%	
truth_pr...	accumulator(spec)	...						
fsm	control(fsm)	...		23	21	2	91.3%	
line__217	seq_generator(struct)	...						
line__220	seq_generator(struct)	...						
line__260	seq_generator(struct)	...						

Now, quit ModelSim and synthesize the VHDL source file using *Synplify_Pro* into the Xilinx Virtex part (xc2vp30ff896-7) by typing:

6. synplify_premier_dp –batch -tcl synplify-virtex.tcl

7. The results are reported in **rev_1/Seq_Generator.srr**: LUTs: 22 (0%)

Copy the required files to the rev_1 directory and then generate the Xilinx layout using the Xilinx fitter:

8. cp stim.do rev_1

9. cp Seq_TestBench.vhd rev_1

10. cp vsim-post-virtex rev_1

11. cp virtex-fit rev_1

12. cp virtex-view rev_1

13. cd rev_1

14. ./virtex-fit Seq_Generator

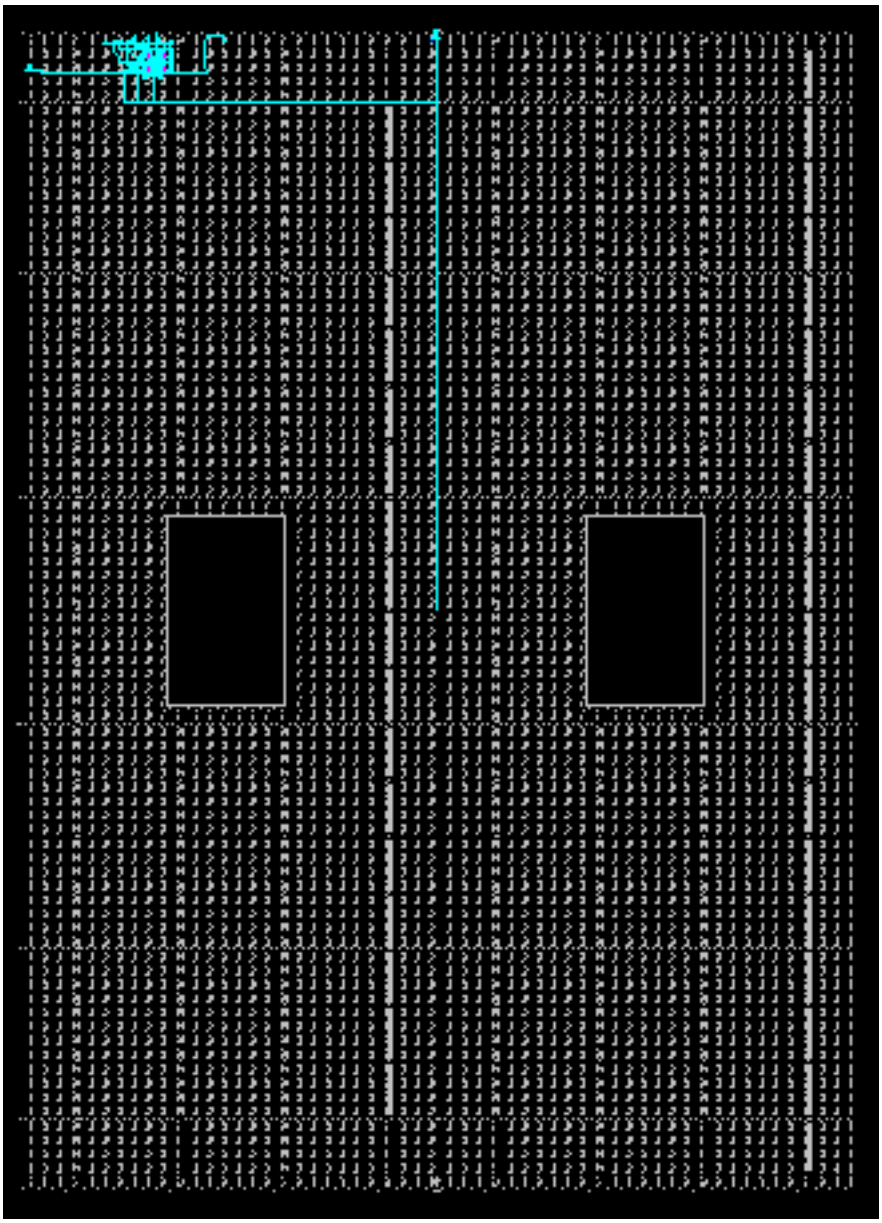
To determine the resources used, type:

15. grep SLICE Seq_Generator_r.par

Number of SLICEs 14 out of 13696 1%

To view the layout, type:

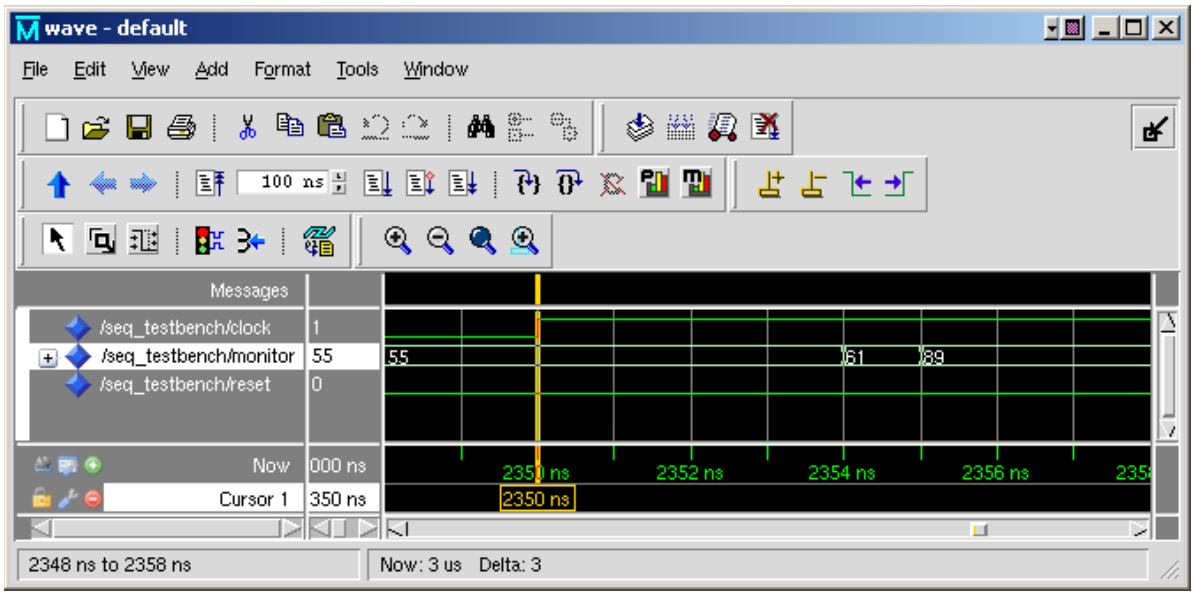
16. ./virtex-view Seq_Generator



Perform post-layout simulation using *ModelSim*:

17. ./vsim-post-virtex

This will bring up the following window:



18. Capture the waveform and note the signal begins to change at 2354 ns and stabilizes at 2355 ns instead of 2250 ns for a delay of **5 ns** + 100ns (initialization).

Now, move up one directory **(cd ..)** and synthesize the VHDL source file using *Synplify_Premier_DP* into the Altera (Cyclone II) part by typing:

19. synplify_premier_dp –batch -tcl synplify-altera.tcl

The resulting net-list will be placed in a newly created subdirectory, rev_2.

20. View the results in **rev_2/Seq_Generator.srr**: registers *26 of 33216 (0%)*

Copy several files to rev_2 and generate the Altera layout using the Altera fitter:

21. cp altera-fit.tcl rev_2

Move down to the directory:

22. cd rev_2

Now, generate the Altera layout:

23. quartus_sh –t altera-fit.tcl

24. View the results in **Seq_Generator.fit.rpt**:

Total logic elements: 20 / 33,216 (< 1%)

25. Now, move to the simulation directory and copy the required files:

cd simulation/modelsim

cp ../../../Seq_TestBench.vhd .

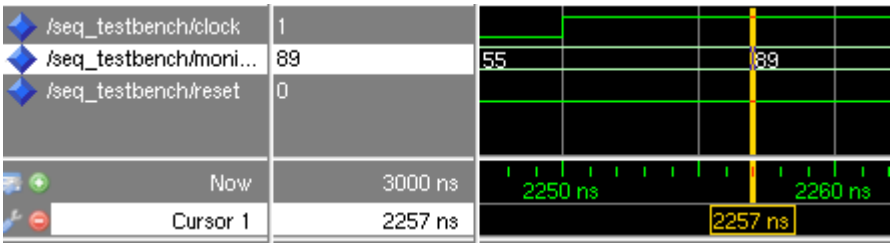
cp ../../../stim.do .

cp ../../../vsim-post-altera .

26. Perform the post-layout simulation by typing:

./vsim-post-altera

27. Capture the waveform and note the signal stabilizes at 2257 ns instead of 2250 ns for a delay of **7 ns**.



APPENDIX

hw6-demo

```
cp ~bouldin/webhome/protected/551-hw6.tar.gz   .
gunzip   551-hw6.tar.gz
tar   –xvf   551-hw6.tar
cd        551-hw6
./presynth_sim
synplify_premier_dp –batch -tcl synplify-virtex.tcl
vi    rev_1/Seq_Generator.srr
cp   stim.do                    rev_1
cp   Seq_TestBench.vhd   rev_1
cp   vsim-post-virtex        rev_1
cp   virtex-fit                 rev_1
cp   virtex-view              rev_1
cd   rev_1
 ./virtex-fit Seq_Generator
grep   SLICE   Seq_Generator_r.par
./vsim-post-virtex
./virtex-view Seq_Generator
cd ..
synplify_pro –batch -tcl synplify-altera.tcl
vi    rev_2/Seq_Generator.srr
cp    altera-fit.tcl                rev_2
cd    rev_2
quartus_sh    –t    altera-fit.tcl
vi       Seq_Generator.fit.rpt
cd   simulation/modelsim
cp  ../../../Seq_TestBench.vhd .
cp ../../../stim.do .
cp  ../../../vsim-post-altera .
./vsim-post-altera
```

presynth_sim

```
#./presynth_sim
vlib work
vcom -work work Seq_Generator.vhd
vcom -work work Seq_TestBench.vhd
vsim -coverage Seq_TestBench -do stim.do
```

stim.do

```
add wave *
run 3000
```

synplify-virtex.tcl

```
#synplify_premier_dp -batch -tcl synplify-virtex.tcl
project -new proj.prj
add_file Seq_Generator.vhd
impl -add rev_1
impl -active "rev_1"
set_option -technology VIRTEX2P
set_option -part XC2VP30
set_option -package FF896
set_option -speed_grade -7
set_option -synthesis_onoff_pragma 0
#map options
set_option -frequency 25.00
set_option -fanout_limit 500
set_option -domap 1
set_option -cliquing 1
set_option -pipe 0
set_option -retiming 0
set_option -fixgatedclocks 0
project -run synthesis
```

virtex-fit

```
#./virtex-fit Seq_Generator
source /usr/local/xilinx/10.1/ISE/settings64.csh
ngdbuild   $1.edf
map -cm speed  -timing $1.ngd
par $1.ncd -w $1_r.ncd
trce -u 100  $1_r.ncd -o $1_r.twr
netgen -sta -w $1_r.ncd $1_sta.v -ofmt verilog
netgen -sim -tb -w $1_r.ncd $1_sim.vhd -ofmt vhdl
```

vsim-post-virtex

```
#./vsim-post-virtex
vlib work
vcom -work work Seq_Generator_sim.vhd
vcom -work work Seq_TestBench.vhd
vsim Seq_TestBench -coverage -do stim.do -sdftyp UUT=Seq_Generator_sim.sdf
```

synplify-altera.tcl

```
#synplify_premier_dp -batch -tcl synplify-altera.tcl
project -new proj.prj
add_file Seq_Generator.vhd
impl -add rev_2
impl -active "rev_2"
set_option -technology CYCLONEII
set_option -part EP2C35
set_option -package FC672
set_option -grade -6
set_option -synthesis_onoff_pragma 0
set_option –result_file Seq_Generator.vqm
#map options
set_option -frequency 25.00
set_option -fanout_limit 500
set_option -pipe 0
set_option -retiming 0
set_option -fixgatedclocks 0
project -run synthesis
```

altera-fit.tcl

```
#quartus_sh -t altera-fit.tcl
set project_name Seq_Generator
# Open the Project. If it does not already exist, create it
if [catch {project_open $project_name}] {project_new $project_name}
set_global_assignment -name family CYCLONEII
set_global_assignment -name device EP2C35F672C6
set_global_assignment -name optimization_technique speed
set_global_assignment -name fast_fit_compilation on
set_global_assignment -name eda_simulation_tool "ModelSim (VHDL output from
Quartus II)"
create_base_clock -fmax 25MHz -target clk clk
package require ::quartus::flow
execute_flow -compile
project_close
```

vsim-post-altera

```
#./vsim-post-altera
Vlib work
Vcom /usr/local/quartus/eda/sim_lib/cycloneii_atoms.vhd
Vcom /usr/local/quartus/eda/sim_lib/cycloneii_components.vhd
Vmap cycloneii work
Vcom –work work Seq_Generator.vho
Vcom –work work Seq_TestBench.vhd
Vsim Seq_TestBench –coverage –do stim.do –sdftyp UUT=Seq_Generator_vhd.sdo
```

Seq_Generator.vhd

```
1   -- VHDL Fibonacci Sequencer Design
2
3   -- VHDL Entity control
4
5   LIBRARY ieee ;
6   USE ieee.std_logic_1164.all;
7   USE ieee.std_logic_arith.all;
8
9   ENTITY control IS
10     PORT(
11        clock  : IN     std_logic  ;
12        reset  : IN     std_logic  ;
13        clr    : OUT    std_logic  ;
14        inc    : OUT    std_logic  ;
15        ld_A_B : OUT    std_logic  ;
16        ld_sum : OUT    std_logic
17     );
18
19  END control ;
20
21  -- VHDL Architecture control
22
23  LIBRARY ieee ;
24  USE ieee.std_logic_1164.all;
25  USE ieee.std_logic_arith.all;
26
27  ARCHITECTURE fsm OF control IS
28
29     -- Architecture Declarations
30     TYPE state_type IS (
31        clr_regs,
32        inc_accb,
33        load_acc_sum,
34        load_acc_A_B
35     );
36
37     -- State vector declaration
38     ATTRIBUTE state_vector : string;
39     ATTRIBUTE state_vector OF fsm : architecture IS "current_state" ;
40
41     -- Declare current and next state signals
42     SIGNAL current_state, next_state : state_type ;
43
44  BEGIN
45
46     clocked : PROCESS (clock, reset)
47     BEGIN
48        IF (reset = '1') THEN
49           current_state <= clr_regs;
50           -- Reset Values
51        ELSIF (clock'EVENT AND clock = '1') THEN
52           current_state <= next_state;
53           -- Default Assignment To Internals
54
55        END IF;
56
57     END PROCESS clocked;
58
59     nextstate : PROCESS (current_state)
60     BEGIN
61        CASE current_state IS
62        WHEN clr_regs =>
63              next_state <= inc_accb;
64        WHEN inc_accb =>
65              next_state <= load_acc_sum;
66        WHEN load_acc_sum =>
67              next_state <= load_acc_A_B;
68        WHEN load_acc_A_B =>
69              next_state <= load_acc_sum;
70        WHEN OTHERS =>
71           next_state <= clr_regs;
72
73        END CASE;
74
75     END PROCESS nextstate;
```

```
 76
 77        output : PROCESS (current_state)
 78        BEGIN
 79           -- Default Assignment
 80           clr <= '0';
 81           inc <= '0';
 82           ld_A_B <= '0';
 83           ld_sum <= '0';
 84
 85           -- State Actions
 86           CASE current_state IS
 87           WHEN clr_regs =>
 88              clr <= '1' ; -- Corrected error on 10/29/Monday
 89              inc <= '0' ;
 90              ld_A_B <= '0' ;
 91              ld_sum <= '0' ;
 92           WHEN inc_accb =>
 93              clr <= '0' ; -- Corrected error on 10/29/Monday
 94              inc <= '1' ;
 95           WHEN load_acc_sum =>
 96              inc <= '0' ;
 97              ld_A_B <= '0' ;
 98              ld_sum <= '1' ;
 99           WHEN load_acc_A_B =>
100              ld_A_B <= '1' ;
101              ld_sum <= '0' ;
102           WHEN OTHERS =>
103              NULL;
104           END CASE;
105
106        END PROCESS output;
107
108     END fsm;
109
110     -- VHDL Entity accumulator
111
112     LIBRARY ieee ;
113     USE ieee.std_logic_1164.all;
114     USE ieee.std_logic_arith.all;
115
116     ENTITY accumulator IS
117        PORT(
118           clock : IN     std_logic  ;
119           clr   : IN     std_logic  ;
120           inc   : IN     std_logic  ;
121           ip    : IN     std_logic_vector (7 DOWNTO 0) ;
122           ld    : IN     std_logic  ;
123           op    : BUFFER std_logic_vector (7 DOWNTO 0)
124        );
125
126     END accumulator ;
127
```

```
128   -- VHDL Architecture accumulator
129
130   ARCHITECTURE spec OF accumulator IS
131
132   BEGIN
133
134      truth_process: PROCESS(clock)
135
136      BEGIN
137         IF (clock'EVENT AND clock = '1') THEN
138            IF (clr = '1') THEN
139               -- Reset Actions
140               op <= "00000000";
141            ELSE
142               IF (ld = '1') THEN
143                  op <= ip;
144               ELSIF (inc = '1') THEN
145                  op <= unsigned(op)+1;
146               ELSE
147                  op <= op;
148               END IF;
149
150            END IF;
151         END IF;
152      END PROCESS truth_process;
153
154   END spec;
155
156
157   -- VHDL Entity Seq_Generator
158
159   LIBRARY ieee ;
160   USE ieee.std_logic_1164.all;
161   USE ieee.std_logic_arith.all;
162
163   ENTITY Seq_Generator IS
164      PORT(
165         clk   : IN     std_logic  ;
166         reset  : IN     std_logic  ;
167         fibout : OUT    std_logic_vector (7 DOWNTO 0)
168      );
169
170   END Seq_Generator ;
171
172   LIBRARY ieee ;
173   USE ieee.std_logic_1164.ALL;
174   USE ieee.std_logic_arith.ALL;
175
176   ARCHITECTURE struct OF Seq_Generator IS
177
178   -- Internal signal declarations
179   SIGNAL A       : std_logic_vector(7 DOWNTO 0);
180   SIGNAL B       : std_logic_vector(7 DOWNTO 0);
181   SIGNAL clr     : std_logic;
182   SIGNAL gnd     : std_logic;
183   SIGNAL inc     : std_logic;
184   SIGNAL ld_A_B  : std_logic;
185   SIGNAL ld_sum  : std_logic;
186   SIGNAL sum     : std_logic_vector(7 DOWNTO 0);
187
188   -- Implicit buffer signal declarations
189   SIGNAL fibout_internal : std_logic_vector (7 DOWNTO 0);
190
```

```
191     -- Component Declarations
192     COMPONENT accumulator
193        PORT (
194           clock : IN     std_logic ;
195           clr   : IN     std_logic ;
196           inc   : IN     std_logic ;
197           ip    : IN     std_logic_vector (7 DOWNTO 0);
198           ld    : IN     std_logic ;
199           op    : BUFFER std_logic_vector (7 DOWNTO 0)
200        );
201     END COMPONENT;
202     COMPONENT control
203        PORT (
204           clock  : IN     std_logic ;
205           reset  : IN     std_logic ;
206           clr    : OUT    std_logic ;
207           inc    : OUT    std_logic ;
208           ld_A_B : OUT    std_logic ;
209           ld_sum : OUT    std_logic
210        );
211     END COMPONENT;
212
213     BEGIN
214        -- Architecture concurrent statements
215
216        -- Add signals A and B together
217        sum <= unsigned(A) + unsigned(B) ;
218
219        -- Tie signal gnd to ground
220        gnd <= '0' ;
221
222        acc_A : accumulator
223           PORT MAP (
224              clock => clk,
225              clr   => clr,
226              inc   => gnd,
227              ip    => fibout_internal,
228              ld    => ld_A_B,
229              op    => A
230           );
231        acc_B : accumulator
232           PORT MAP (
233              clock => clk,
234              clr   => clr,
235              inc   => inc,
236              ip    => A,
237              ld    => ld_A_B,
238              op    => B
239           );
240        acc_sum : accumulator
241           PORT MAP (
242              clock => clk,
243              clr   => clr,
244              inc   => gnd,
245              ip    => sum,
246              ld    => ld_sum,
247              op    => fibout_internal
248           );
249        FSM: control
250           PORT MAP (
251              clock  => clk,
252              reset  => reset,
253              clr    => clr,
254              inc    => inc,
255              ld_A_B => ld_A_B,
256              ld_sum => ld_sum
257           );
258
259        -- Implicit buffered output assignments
260        fibout <= fibout_internal;
261
262     END struct;
```

Seq_TestBench.vhd

```
1   -- VHDL Test Bench for Fibonacci Sequencer Design
2
3   -- VHDL Entity Seq_Generator_tester
4
5   LIBRARY ieee ;
6   USE ieee.std_logic_1164.all;
7   USE ieee.std_logic_arith.all;
8
9   ENTITY Seq_Generator_tester IS
10     PORT(
11        monitor : IN     std_logic_vector (7 DOWNTO 0) ;
12        clock   : OUT    std_logic  ;
13        reset   : OUT    std_logic
14     );
15
16  END Seq_Generator_tester ;
17
18  -- VHDL Architecture Seq_Generator_tester
19
20  ARCHITECTURE spec OF Seq_Generator_tester IS
21
22  -- Architecture declarations
23  CONSTANT clk_prd : time := 100 ns ;
24  SIGNAL int_clock : std_logic := '0';
25
26  BEGIN
27
28     process0 : PROCESS (monitor)
29     BEGIN
30        IF unsigned(monitor) > 128 THEN
31           reset <= '1' ;
32        ELSE
33           reset <= '0' ;
34        END IF;
35     END PROCESS process0;
36
37  -- Architecture concurrent statements
38  -- Clock Generator
39
40
41  int_clock <= not int_clock AFTER clk_prd / 2;
42  clock <= int_clock;
43
44  END spec;
45
46  LIBRARY ieee ;
47  USE ieee.std_logic_1164.all;
48  USE ieee.std_logic_arith.all;
49
50  ENTITY Seq_TestBench IS
51  -- Test bench has no external interface
52
53  END Seq_TestBench ;
54
55  LIBRARY ieee ;
56  USE ieee.std_logic_1164.ALL;
57  USE ieee.std_logic_arith.ALL;
58
59  ARCHITECTURE struct OF Seq_TestBench IS
60
61  -- Architecture declarations
62  -- Internal signal declarations
63  SIGNAL clock   : std_logic;
64  SIGNAL monitor : std_logic_vector(7 DOWNTO 0);
65  SIGNAL reset   : std_logic;
66
```

```
67    -- Component Declarations
68    COMPONENT Seq_Generator
69       PORT (
70          clk   : IN     std_logic ;
71          reset  : IN     std_logic ;
72          fibout : OUT    std_logic_vector (7 DOWNTO 0)
73       );
74    END COMPONENT;
75    COMPONENT Seq_Generator_tester
76       PORT (
77          monitor : IN     std_logic_vector (7 DOWNTO 0);
78          clock  : OUT    std_logic ;
79          reset  : OUT    std_logic
80       );
81    END COMPONENT;
82
83
84    BEGIN
85       UUT : Seq_Generator
86          PORT MAP (
87             clk  => clock,
88             reset  => reset,
89             fibout => monitor
90          );
91       Checker: Seq_Generator_tester
92          PORT MAP (
93             monitor => monitor,
94             clock  => clock,
95             reset  => reset
96          );
97
98    END struct;
```