Blake Johnson
9663980

ECE 156a HW6

Introduction:

This homework delves into the topic of using covergroups to verify that testing encompasses the entirety of the design. We also use Design Compiler to synthesize a design using a custom class to implement custom primitives.

Procedure:

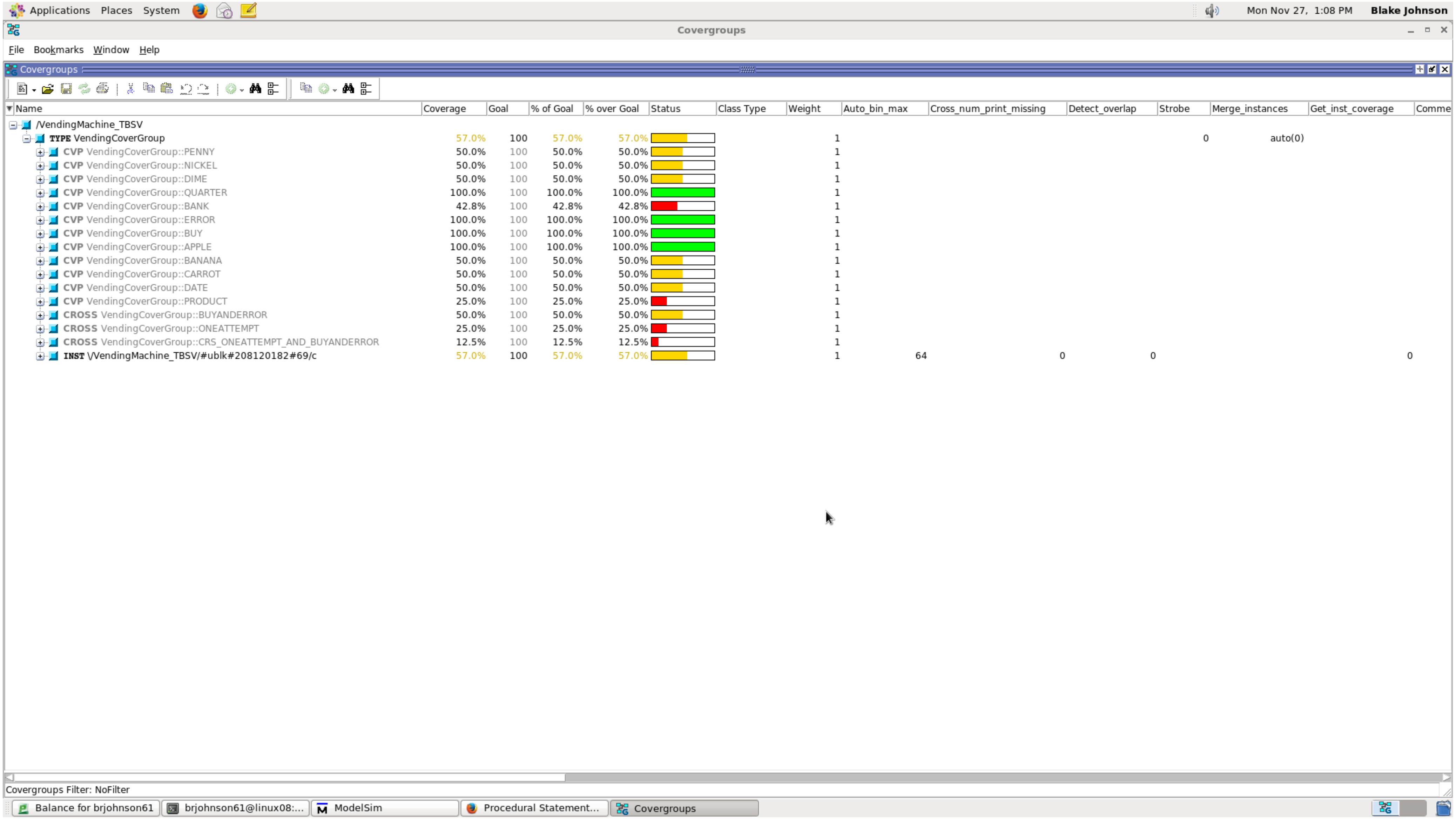
To begin, I implemented the cover group to meet the required specifications and tested my vending machine from the last project again. The result was low, but there are valid reasons for why the testing coverage was low. Due to my testbench implementation, I have lines annotated with comments where I can change the values to quickly change the overall behavior of the testbench. I did this to simplify taking screenshots so that each screenshot would display a different feature of the test/design. However, the covergroup covers one compiled file as it is, and does not take into account the user text changes. This is bad testbench implementation in the long run on my part, despite the flexibility I had on the last homework. Next, I created a new testbench that used systemverilog's rand features to generate controlled pseudo-random scenarios, and applied the same test coverage. This time my test worked 100% as I knew they would. Next, I created the simple 10 value upcounter and used design compiler to synthesize the design. Despite the extremely underwhelming instructions I was able to piece together what was asked of me and miraculously complete the assignment.

What's included in this report:

In this report I include the necessary files to run my VendingMachine module as well as the two test benches used with my covergroup. I also include my upcounter, the synthesized version of my upcounter, as well as the upcounter miter testbench for the upcounter modules. I also included 3 screenshots. The first is the less covered testbench from the last homework assignment, and the second is the more thoroughly covered testbench coverage report. The third screenshot is the waveform of the upcounter miter, that shows both the behavioral upcounter and the compiled structural upcounter I made using Design Compiler.

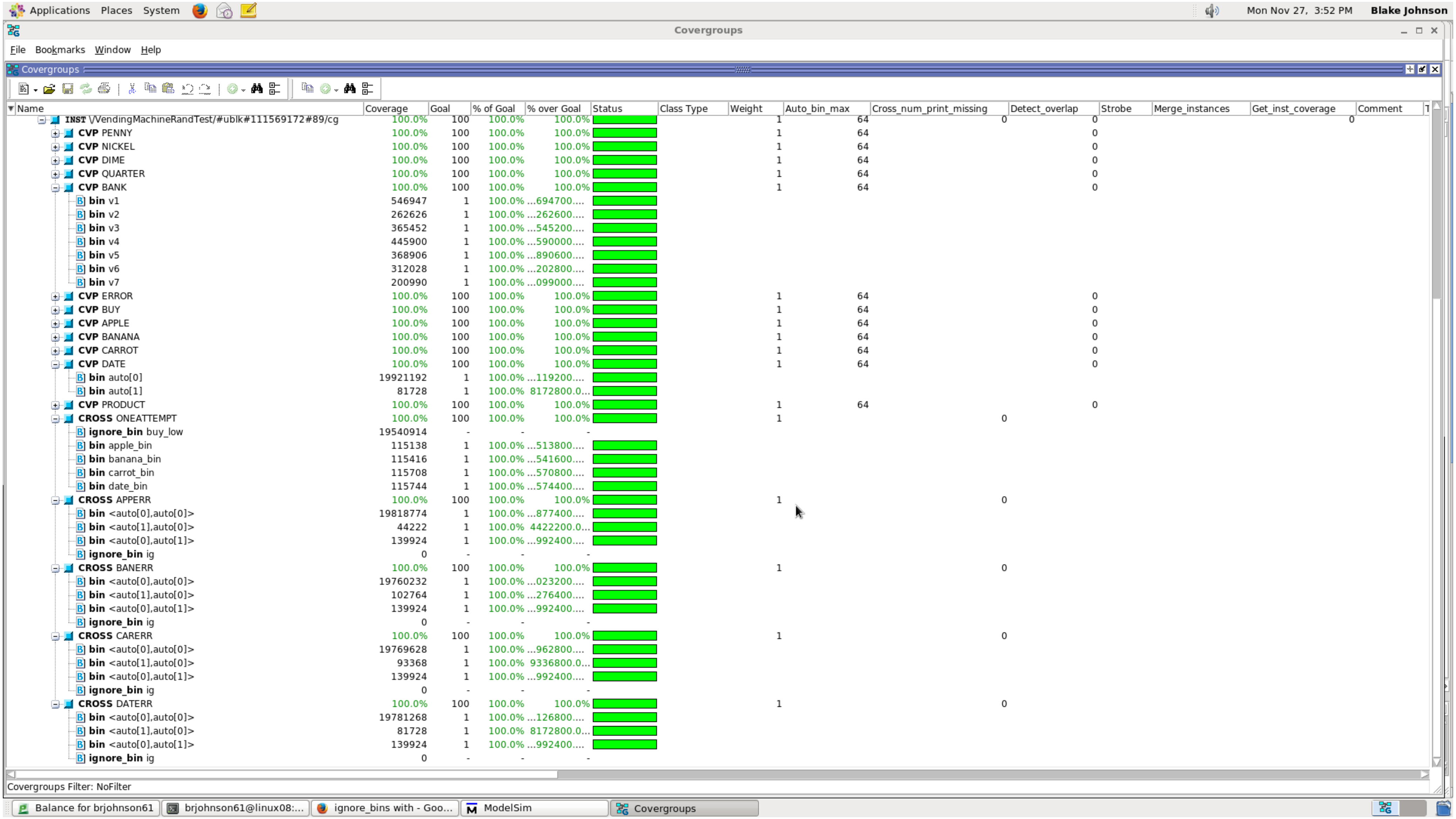
Conclusion:

In this homework I reworked my testbench for the vending machine to make it cover many more cases, and in doing so I learned how to use covergroups and randomization. I also used Design Compiler to synthesize my design using a custom library of primitives. This lab was extremely challenging, not because of the actual content of the lab, but because of the poor documentation of basic starting procedures, and the fact that ModelSim PE could not be used to synthesize. For example, the only thing that lets you know that you need to use Design Compiler is the subheader of the latter half of the assignment, and some vague piazza posts about .bashrc. Thanks for taking the time to read this constructive criticism, have a wonderful day.



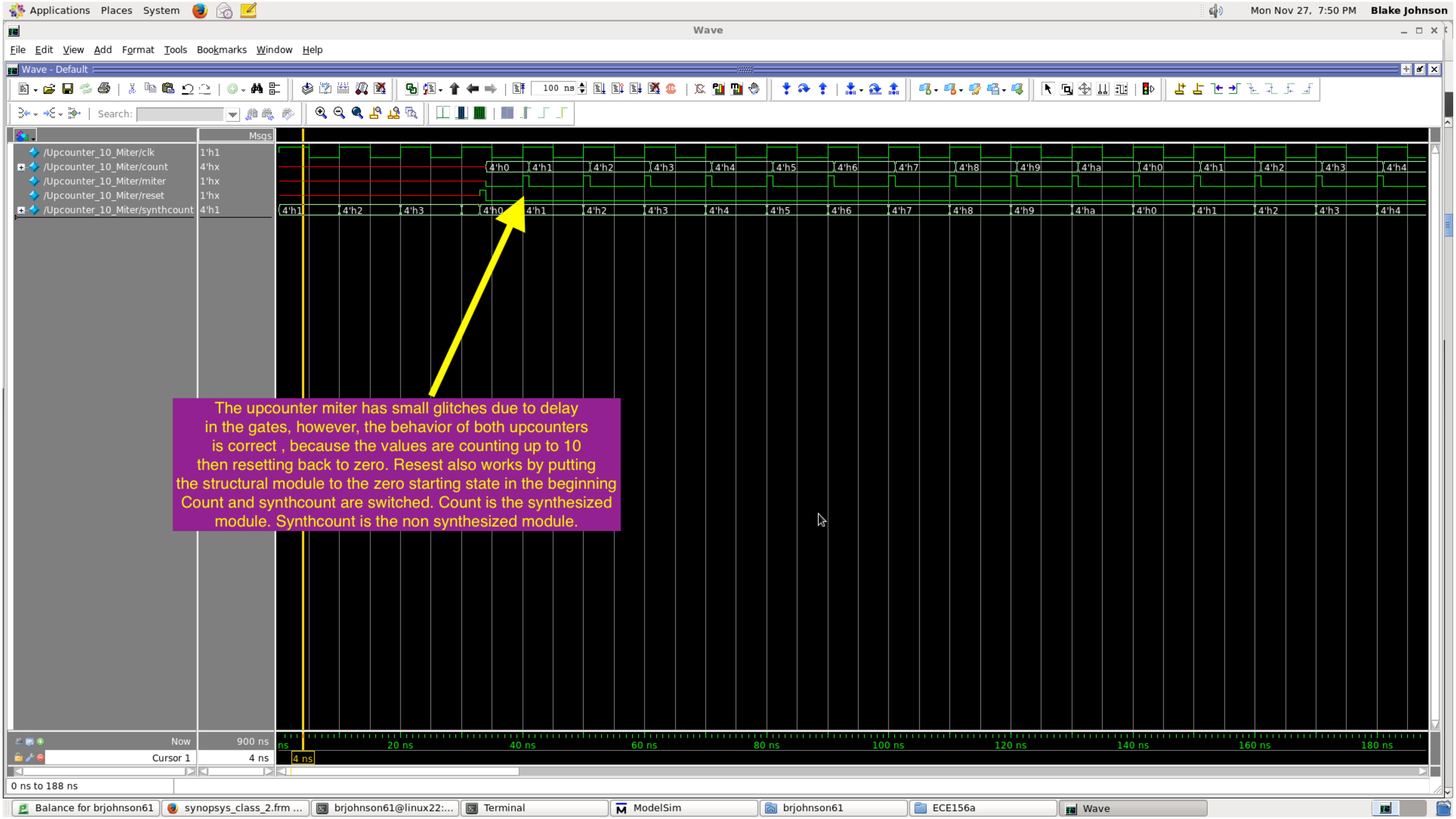
Name	Coverage	Goal	% of Goal	% over Goal	Status	Class Type	Weight	Auto_bin_max	Cross_num_print_missing	Detect_overlap	Strobe	Merge_instances	Get_inst_coverage	Comme
/VendingMachine_TBSV														
TYPE VendingCoverGroup	57.0%	100	57.0%	57.0%	🟨		1				0	auto(0)		
CVP VendingCoverGroup::PENNY	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::NICKEL	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::DIME	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::QUARTER	100.0%	100	100.0%	100.0%	🟩		1							
CVP VendingCoverGroup::BANK	42.8%	100	42.8%	42.8%	🟥		1							
CVP VendingCoverGroup::ERROR	100.0%	100	100.0%	100.0%	🟩		1							
CVP VendingCoverGroup::BUY	100.0%	100	100.0%	100.0%	🟩		1							
CVP VendingCoverGroup::APPLE	100.0%	100	100.0%	100.0%	🟩		1							
CVP VendingCoverGroup::BANANA	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::CARROT	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::DATE	50.0%	100	50.0%	50.0%	🟨		1							
CVP VendingCoverGroup::PRODUCT	25.0%	100	25.0%	25.0%	🟥		1							
CROSS VendingCoverGroup::BUYANDERROR	50.0%	100	50.0%	50.0%	🟨		1							
CROSS VendingCoverGroup::ONEATTEMPT	25.0%	100	25.0%	25.0%	🟥		1							
CROSS VendingCoverGroup::CRS_ONEATTEMPT_AND_BUYANDERROR	12.5%	100	12.5%	12.5%	🟥		1							
INST \/VendingMachine_TBSV/#ublk#208120182#69/c	57.0%	100	57.0%	57.0%	🟨		1	64	0	0			0	

Covergroups Filter: NoFilter



Name	Coverage	Goal	% of Goal	% over Goal	Status	Class Type	Weight	Auto_bin_max	Cross_num_print_missing	Detect_overlap	Strobe	Merge_instances	Get_inst_coverage	Comment
INST \/VendingMachineRandTest/#ublk#111569172#89/cg	100.0%	100	100.0%	100.0%	🟩		1	64	0	0			0	
CVP PENNY	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP NICKEL	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP DIME	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP QUARTER	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP BANK	100.0%	100	100.0%	100.0%	🟩		1	64		0				
bin v1	546947	1	100.0%	...694700....	🟩									
bin v2	262626	1	100.0%	...262600....	🟩									
bin v3	365452	1	100.0%	...545200....	🟩									
bin v4	445900	1	100.0%	...590000....	🟩									
bin v5	368906	1	100.0%	...890600....	🟩									
bin v6	312028	1	100.0%	...202800....	🟩									
bin v7	200990	1	100.0%	...099000....	🟩									
CVP ERROR	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP BUY	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP APPLE	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP BANANA	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP CARROT	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CVP DATE	100.0%	100	100.0%	100.0%	🟩		1	64		0				
bin auto[0]	19921192	1	100.0%	...119200....	🟩									
bin auto[1]	81728	1	100.0%	8172800.0...	🟩									
CVP PRODUCT	100.0%	100	100.0%	100.0%	🟩		1	64		0				
CROSS ONEATTEMPT	100.0%	100	100.0%	100.0%	🟩		1		0					
ignore_bin buy_low	19540914	-	-	-										
bin apple_bin	115138	1	100.0%	...513800....	🟩									
bin banana_bin	115416	1	100.0%	...541600....	🟩									
bin carrot_bin	115708	1	100.0%	...570800....	🟩									
bin date_bin	115744	1	100.0%	...574400....	🟩									
CROSS APPERR	100.0%	100	100.0%	100.0%	🟩		1		0					
bin <auto[0],auto[0]>	19818774	1	100.0%	...877400....	🟩									
bin <auto[1],auto[0]>	44222	1	100.0%	4422200.0...	🟩									
bin <auto[0],auto[1]>	139924	1	100.0%	...992400....	🟩									
ignore_bin ig	0	-	-	-										
CROSS BANERR	100.0%	100	100.0%	100.0%	🟩		1		0					
bin <auto[0],auto[0]>	19760232	1	100.0%	...023200....	🟩									
bin <auto[1],auto[0]>	102764	1	100.0%	...276400....	🟩									
bin <auto[0],auto[1]>	139924	1	100.0%	...992400....	🟩									
ignore_bin ig	0	-	-	-										
CROSS CARERR	100.0%	100	100.0%	100.0%	🟩		1		0					
bin <auto[0],auto[0]>	19769628	1	100.0%	...962800....	🟩									
bin <auto[1],auto[0]>	93368	1	100.0%	9336800.0...	🟩									
bin <auto[0],auto[1]>	139924	1	100.0%	...992400....	🟩									
ignore_bin ig	0	-	-	-										
CROSS DATERR	100.0%	100	100.0%	100.0%	🟩		1		0					
bin <auto[0],auto[0]>	19781268	1	100.0%	...126800....	🟩									
bin <auto[1],auto[0]>	81728	1	100.0%	8172800.0...	🟩									
bin <auto[0],auto[1]>	139924	1	100.0%	...992400....	🟩									
ignore_bin ig	0	-	-	-										

Covergroups Filter: NoFilter



/Upcounter_10_Miter/clk 1'h1
/Upcounter_10_Miter/count 4'hx
/Upcounter_10_Miter/miter 1'hx
/Upcounter_10_Miter/reset 1'hx
/Upcounter_10_Miter/synthcount 4'h1

Now 900 ns
Cursor 1 4 ns

0 ns to 188 ns

The upcounter miter has small glitches due to delay in the gates, however, the behavior of both upcounters is correct , because the values are counting up to 10 then resetting back to zero. Resest also works by putting the structural module to the zero starting state in the beginning Count and synthcount are switched. Count is the synthesized module. Synthcount is the non synthesized module.