ETIN01 IC-project & Verification, digital

Assignment Report

Ma Ling (770821-8867)
Payman Pooyan (831230-6296)

December 2008

Contents

VHDL coding assignments

1. Describe your Finite State Machine(FSM) for assignment 1.

According to assignment 1's requirement, a state machine can be designed to control the alarm light while the sensors' output is changed. We develop the first solution which is a 9 states FSM and is shown in Figure 1.1.

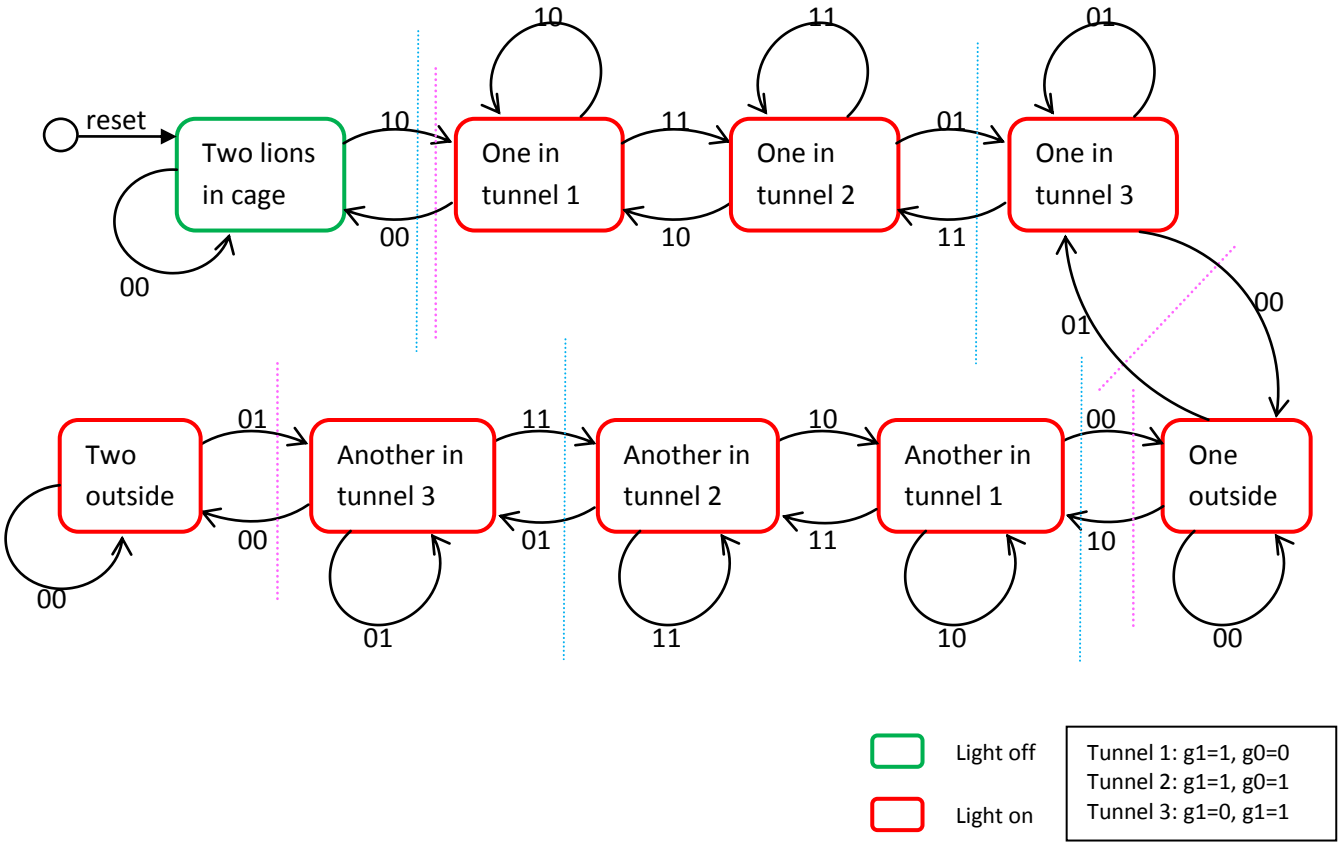


Figure 1.1 9 states FSM

Every possible state has been considered. This FSM can work well, but 9 states seem too many and we try to reduce them. Because lions can't back down or retreat while they enter the tunnel, we can merge some states about tunnel and modify the 9 states FSM to a 5 states FSM. If we merge the 3 states of tunnel into one state and the original FSM is divided by the pink lines in Figure 1.1, assuming that the current state is "One in tunnel", the next state which is "Two lions in cage" or "One outside" will be hard to decide when the sensors' output is 00. So we can't do

like the pink lines. Then we divide the FSM by the blue lines and merge the two states into one, it works and is redrawn as Figure 1.2.

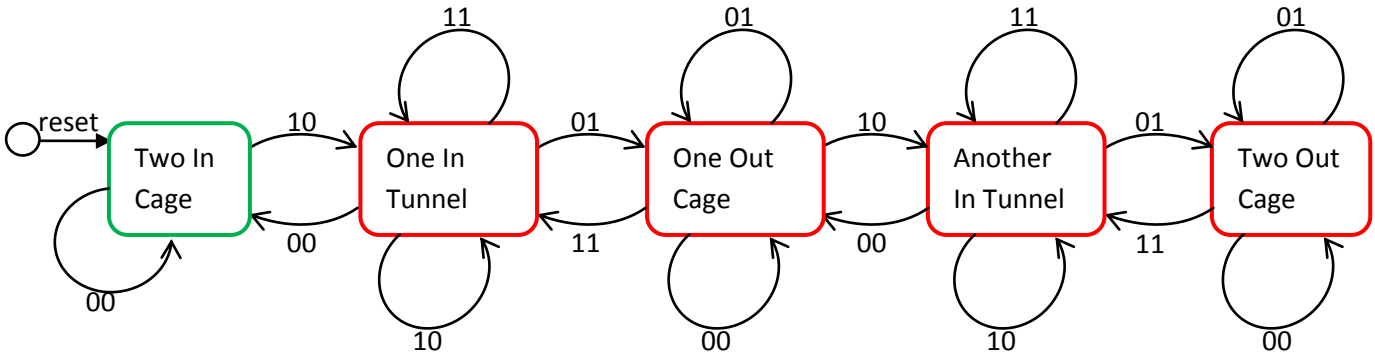


Figure 1.2 5 states FSM

Can we reduce the 5 states into less? The following 4 states FSM is designed like Figure 1.3. But at the state, " Two In Cage", we should control the light separately according to the sensors' output. In this state, the light will turn off when g1g0 is "00" and turn on when g1g0 is not "00". This 4 states FSM also works well.

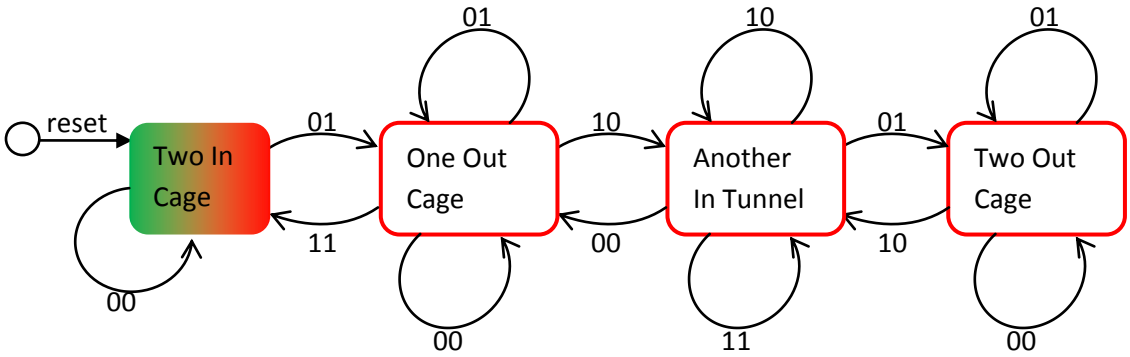


Figure 1.3 4 states FSM

2. Describe architecture including wordlengths (WL) with motivation for assignment 2.

The direct-form FIR filter consists of three basic blocks, which are D-flip-flop, multiplier, and adder. For D-flip-flop, the output and input have the same word length.

For multiplier, the output word length should equal to the sum of two inputs' word lengths to avoid overflow. For adder, one extra bit should be added in the output compared with input to get full precision.

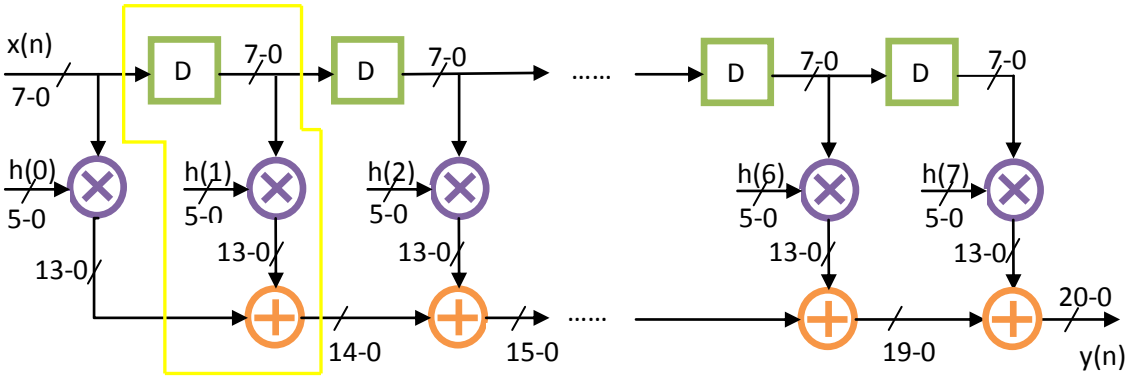


Figure 2 8-tap Direct-form FIR's architecture

A 8-tap direct-form FIR's architecture is shown in Figure 2. The first stage can be composed of only one multiplier. And the next 7 stages can be generated by using loops of one D-flip-flop, one adder and a multiplier. In the fir entity, some signals are created to connect these blocks together.

Assuming that there is a M-order FIR, the input's word length is INPUT_WIDTH and the coefficient's word length is COEFF_WIDTH, the output's word length is INPUT_WIDTH+COEFF_WIDTH+M for full precision.

3. Describe architecture, including WLs, and the FSM for assignment 3.

The q-tap finite impulse response (FIR) filter with unit-sample response $h(k) = b(k)$ can be defined by following difference equation

$$y(n) = \sum_{k=0}^{q-1} h(k) * x(n-k)$$

Therefore, the equations bellow describes a complete computation with 36 steps iteration.

$y(0) = h(0) \cdot x(0)$

$y(1) = h(0) \cdot x(1) + h(1) \cdot x(0)$

$y(2) = h(0) \cdot x(2) + h(1) \cdot x(1) + h(2) \cdot x(0)$

……

$y(34) = h(0) \cdot x(34) + h(1) \cdot x(33) + \cdots + h(33) \cdot x(1) + h(34) \cdot x(0)$

$y(35) = h(0) \cdot x(35) + h(1) \cdot x(34) + \cdots + h(33) \cdot x(2) + h(34) \cdot x(1) + h(35) \cdot x(0)$

We will use a time-multiplexed FIR filter to implement the above function. The key point is addressing and matching the coefficient and input data.

From the above steps, we can easily get the conclusion that the order of coefficient is increasing and the input data is decreasing. So, an addition can be used as ROM address generator. Considering that data shuffling is a major power dissipation source, when RAM is full of 36 data, the next input data will replace the oldest data in RAM. Then we should divide RAM into two parts. One is from the newest data position to the top of RAM, and another is from the bottom of RAM to the newest data position. So, two subtractions should be used as RAM address generator. Meanwhile, these address generators should be included in a controller (FSM), in which other control signal is generated correspondingly.

What's more, because reading x data from RAM needs two clocks, it is possible to share the address line between ROM and RAM, which can save the interface of controller unit and reduce the processing time. Figure 3.1 shows the timing chart of address, CSN, WEN, input and coefficient data. "Aw" is the address for writing x to RAM, "Ar" is the address for reading x from RAM and "Addr" is the address for reading coefficient from ROM.

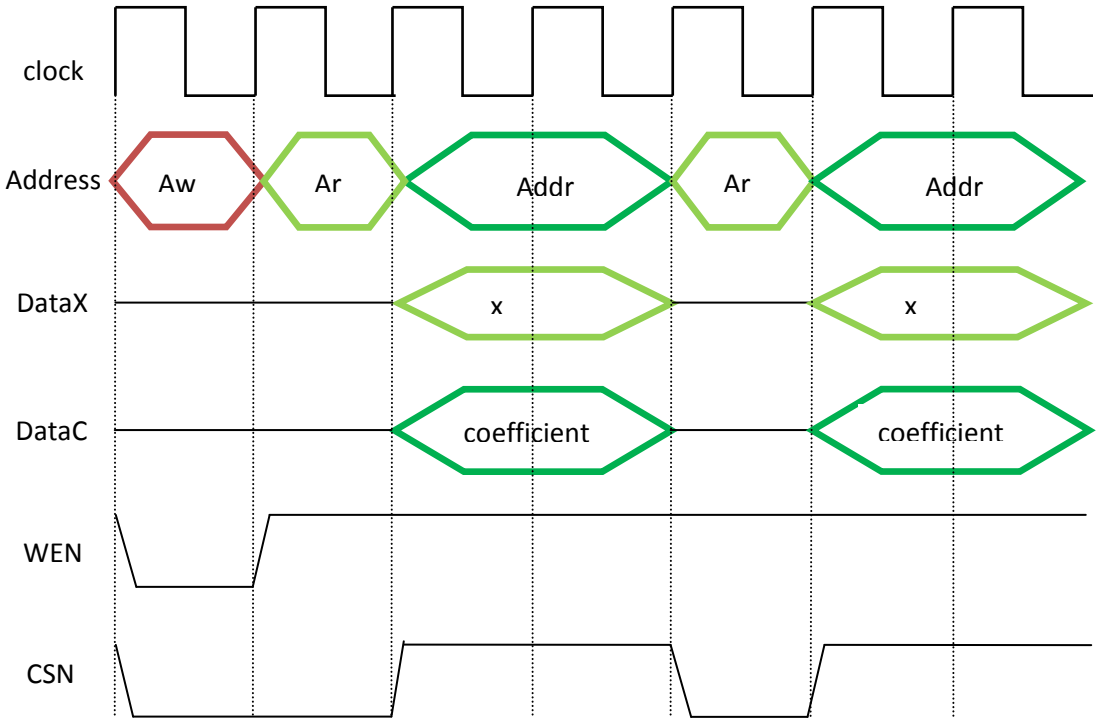


Figure 3.1 timing chart of sharing address line

In our first design, we didn't use a D-flipflop to connect the RAM and MAC unit. The correct output can be obtained but there is a short uncertain state before MAC unit. To eliminate the uncertain state, we use a D-flipflop between RAM and MAC.

The whole FIR's architecture is shown in Figure 3.2. The valid_i, data_i, clk and rst signals come from the testbench and the data_o output to testbench. All signals in our design active low. RAM and ROM share the address line.

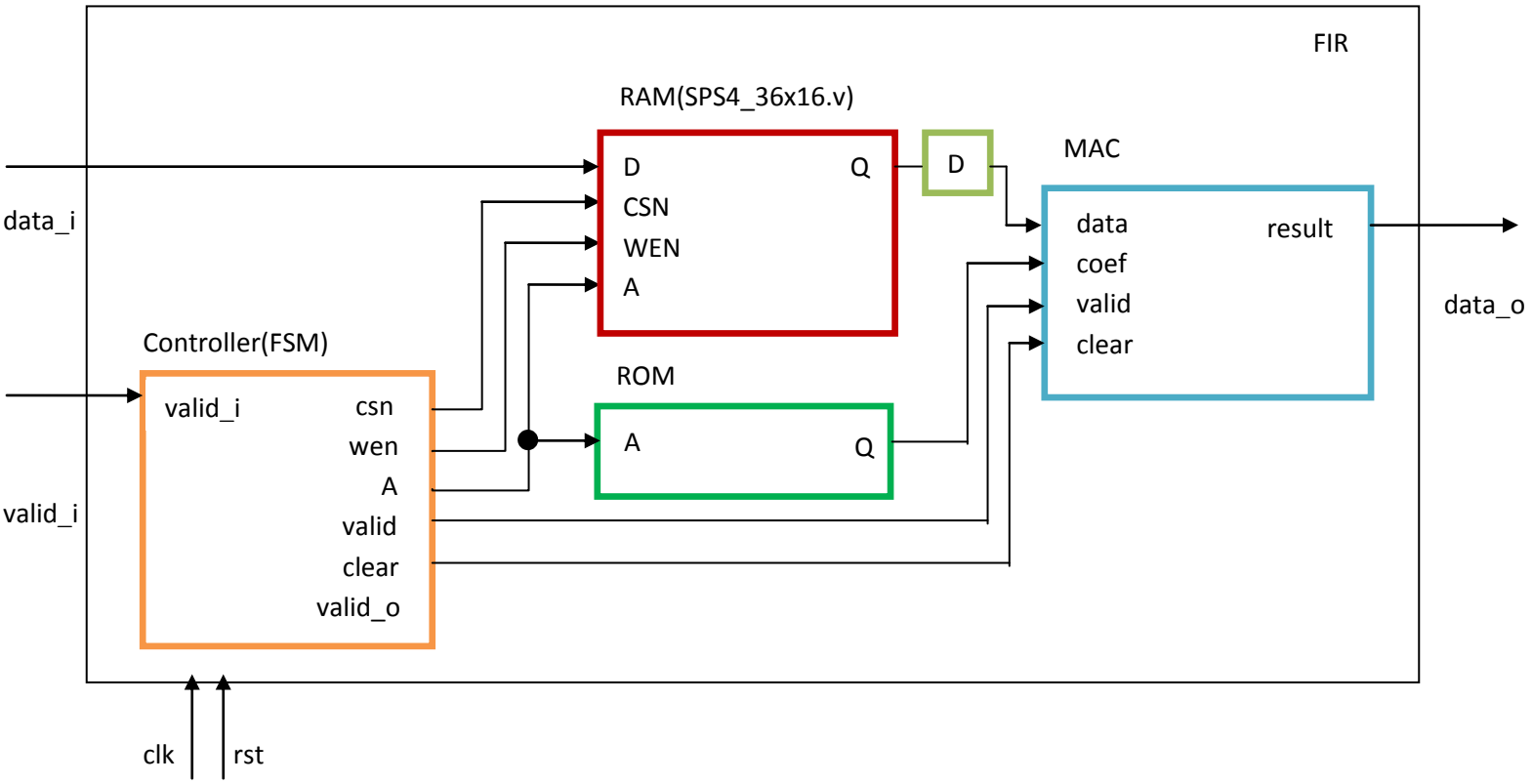


Figure 3.2 FIR architecture

When valid_i =’0’, it means a new input data is available and it can be written into RAM. At this time, controller unit should generate RAM’s address, enable CSN and WEN and write input x data to RAM. After finishing to store the input data , MAC unit begin to read x data from RAM. Then a coefficient address which matches the x data should be sent to ROM, MAC unit will calculate the additive and multiplicative operation by using the x and coefficient data. Then next x and coefficient data will be read and calculated , and them will be repeated by 36 times. When 36 iterations are completed, the signal valid_o should be set ‘0’, so the final result can be saved to the file. Then the filter will wait for the next new input data. So, the FSM has five state: Writer new x to RAM, Read X from RAM, Read Coeff from ROM, Mac Unit Calculate and Output Y. The block diagram of FSM is drawn in Figure 3.3.

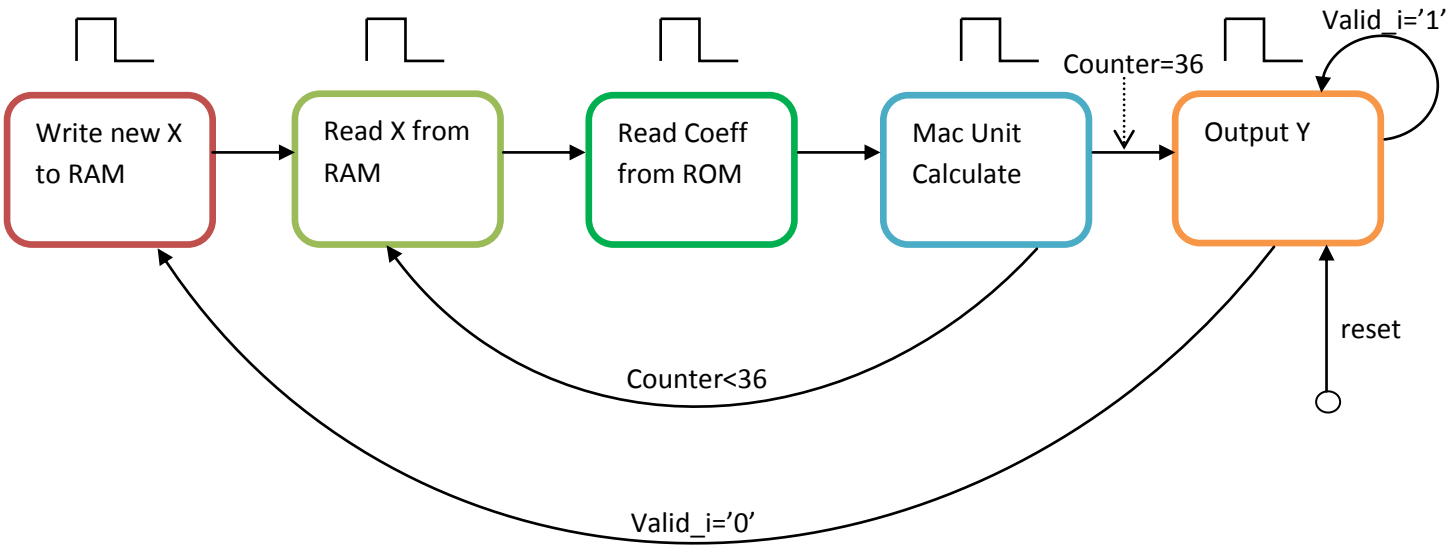


Figure 3.3 Controller (FSM)

In MAC unit, the word length (WL) of input data are INPUT_WIDTH bits, and the coefficient's WL are COEFF_WIDTH bits. In order to guarantee the full precision, the multiplier's WL should be INPUT_WIDTH + COEFF_WIDTH. For adder and D-flipflop, their WLs should be INPUT_WIDTH+COEFF_WIDTH+LOG2_ORDER. Therefore, the final result's WL is INPUT_WIDTH+COEFF_WIDTH+LOG2_ORDER. The actual word length value is shown in Figure 3.4.

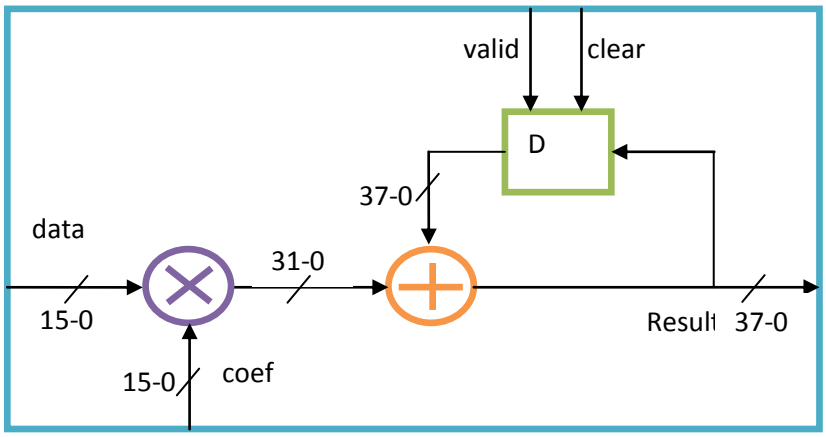


Figure 3.4 MAC unit's Word Length

4. How many clock cycles are needed to produce an output for assignments 2 & 3 ?

For assignment 2, the direct-form FIR filter has 7 D-flipflops, so the latency from input to output is 7 clock cycles. But, FIR unit will generate one output per clock cycle.

For assignment 3, time-multiplexed FIR filter needs 3 clock cycles to complete one calculation of MAC unit. To get an final result, MAC unit should calculate by 36 times. So it needs 36*3(=108) clock cycles to produce one output. In addition, sending output to testbench and writing new data into RAM need 2 clock cycles. Assuming that there is no waiting clock cycles between FIR and testbench, FIR will generate one output per 110 clock cycles.

Synthesis

1. What constraints were set in Design Compiler?

A) *Synthesizing for Min Area:*

create_clock "clock" -period 20 -name clock
set_clock_uncertainty 0.1 clock
set_max_area 0
compile -map_effort high
set_max_area 64839
compile -map_effort high

Result:

Data required time:	19.79
Data Arrival time:	-9.38
Slack:	10.41

B) *Synthesizing for High Speed:*

create_clock "clock" -period 4 -name clock
set_clock_uncertainty 0.1 clock
compile -map_effort high

Result:

Data required time:	3.78
Data Arrival time:	-3.78
Slack:	0

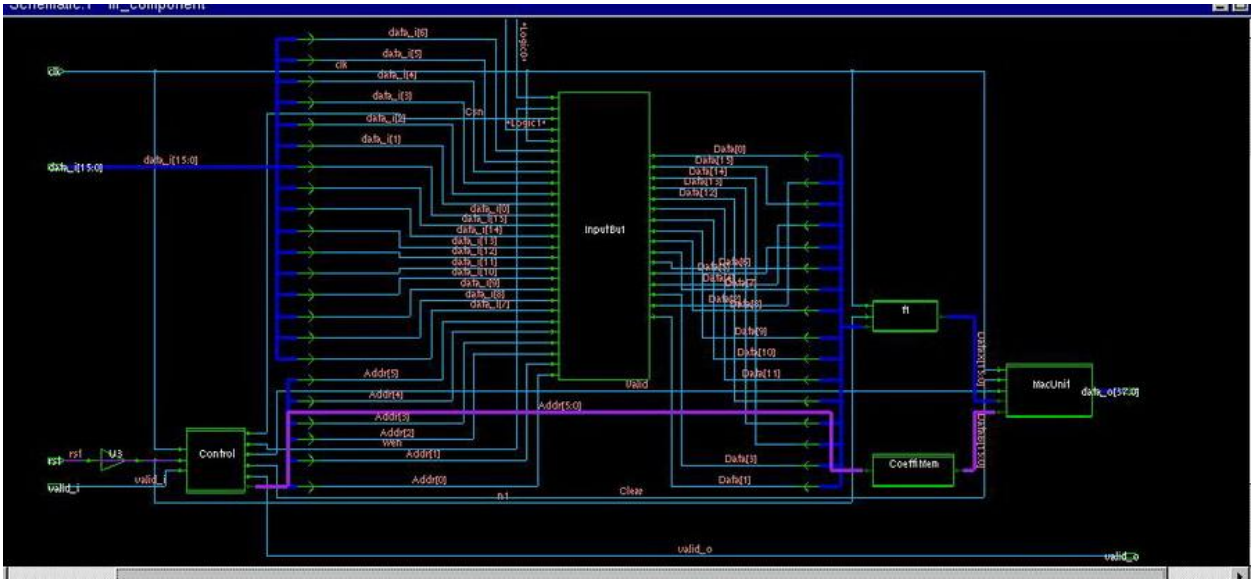
2. Report the area for the subblocks: mem, mac, rom, controller FSM and the pads.In the mac design unit you should report area for the multiplier, adder and register.

Design1 is for Min Area and Design 2 is for High speed.

	MAC			ROM	FSM	Memory	Pads
	Adder	Multiplier	Register				
Design1:	833	5448	1137	490	7774	39770	
Design2:	2125	7215	1140	704	8312	39770	
Design1:	7418						
Design2:	10480						

3. What is the critical path for the design?

Critical Path is shown below for clock period of 20.



4. Which blocks are in the critical path and what are the individual delays for each block?

We can see that FSM and Rom are in the critical path.

Delay for Rom in first Design: 2.92 f

Delay for Rom in second Design: 2.2f

5. Report maximal clock frequency and suggest how you could increase it at the register transfer level.

Our maximal clock frequency was achieved for clock period of 4.we can increase the clock period by optimizing the design and moving flip flops in the design. A good method for increasing the clock frequency is pipelining the design.

Frequency for the first design:342.465MHZ

Frequency for the second design:454.454MHZ

6. What are the differences between the high speed and low area synthesis results,i.e. what implementation of arithmetic blocks has been used.

In the upper parts of our reports we have compared the 2 designs but here are some more issues:

Design1:
Combinational area: 13826.000000
Noncombinational area: 50718.000000
Total cell area: 64544.000000

Design2:
Combinational area: 17589.000000
Noncombinational area: 50765.000000
Total cell area: 68354.000000

7. Present problems with synthesizing the original code.

We did not have problem while synthesizing the original code. But as we saw some people had some problems like having Latches in their design .

8. If you have synthesized several versions, compare results since depending on your VHDL code you will get different synthesis results.

We only synthesize one version.

Place and Route

1. Compare the final area with the estimated area from the synthesis tool. Only compare the core area.

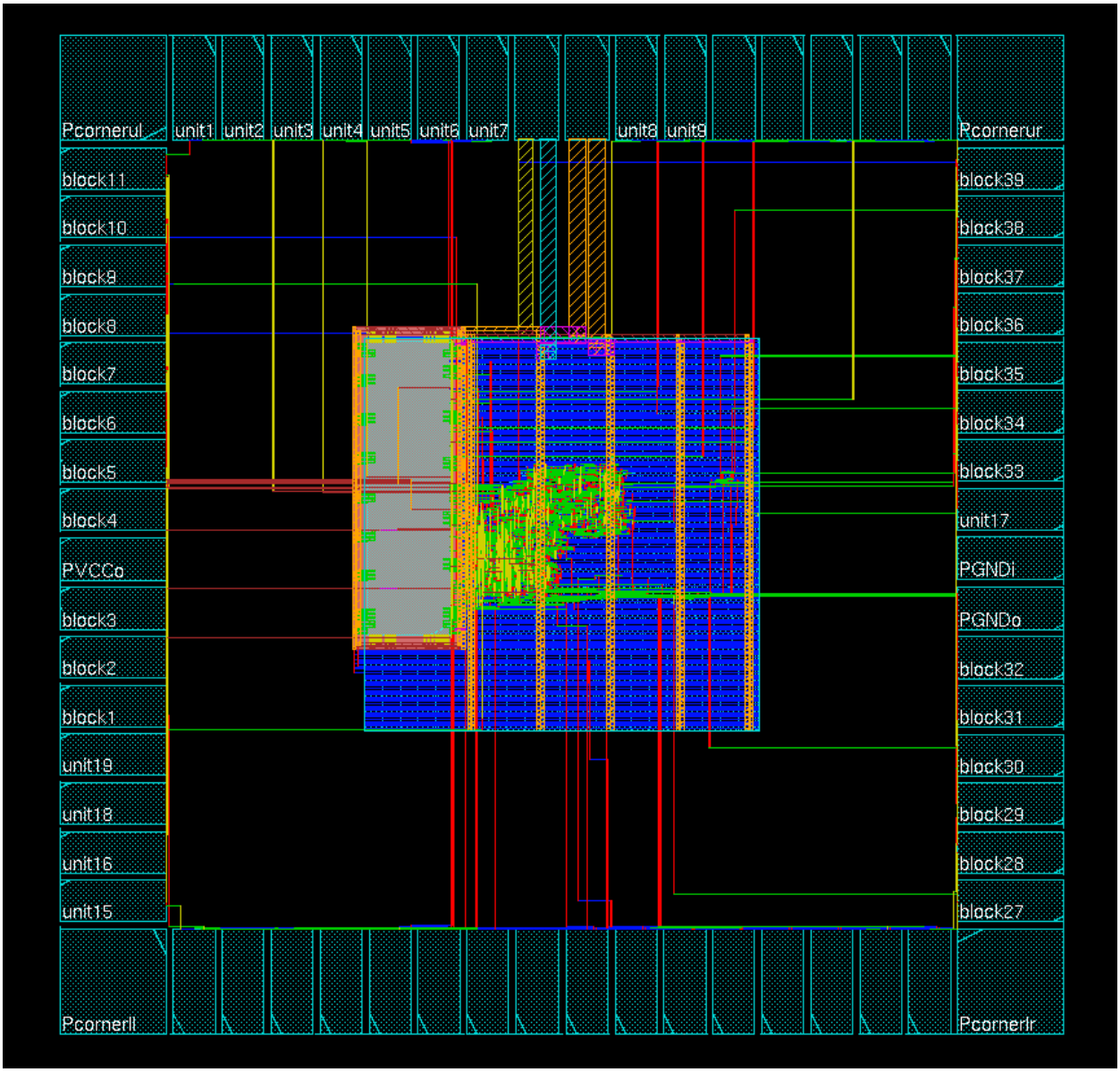
Synthesis:

Total cell area: 64544.0

Place & route:

Stand cell area: 20407 um^2 / Allocate area: 255386 um^2

2. Show layout.



3. Show script-files produced during P&R.

```
#########################################################
#                                                       #
#  Encounter Command Logging File                       #
#  Created on Fri Dec 12 11:55:39                       #
#                                                       #
#########################################################

loadConfig /h/d3/r/sx08lm4/pr4/Default.conf 0

commitConfig

fit

setDrawView fplan

selectInst fir_component/InputBuf

uiSetTool move

setObjFPlanBox Instance fir_component/InputBuf 348.92 784.777 771.72 905.177

uiSetTool move

fplanFlipOrRotateInstance -rotate 90

setObjFPlanBox Instance fir_component/InputBuf 355.392 488.668 475.792
911.468

uiSetTool move

setObjFPlanBox Module fir_component 514.4 521.6 888.0 888.041

uiSetTool move

floorPlan -site core -r 0.998581560284 0.224503 285.0 285 285.0 285

setRoutingStyle -top -style m

uiSetTool select

addHaloToBlock 20 20 20 20 -allBlock

cutRow

globalNetConnect VCC -type pgpin -pin VCC -inst *

globalNetConnect VCC -type tiehi -pin VCC -inst *

globalNetConnect GND -type pgpin -pin GND -inst *
```

```
globalNetConnect GND -type tielo -pin GND -inst *

addRing -spacing_bottom 1 -width_left 4.9 -width_bottom 4.9 -width_top 4.9 -
spacing_top 1 -layer_bottom metal5 -width_right 4.9 -around each_block -
offset_bottom 10 -layer_top metal5 -option_file
/h/d3/r/sx08lm4/pr4/soc/appOption.dat/top_fir.ppo.ppo -offset_left 10 -
spacing_right 1 -spacing_left 1 -type block_rings -offset_right 10 -
offset_top 10 -layer_right metal6 -nets {GND VCC } -layer_left metal6addRing
-spacing_bottom 1 -width_left 4.9 -width_bottom 4.9 -width_top 4.9 -
spacing_top 1 -layer_bottom metal5 -width_right 4.9 -around each_block -
offset_bottom 10 -layer_top metal5 -option_file
/h/d3/r/sx08lm4/pr4/soc/appOption.dat/top_fir.ppo.ppo -offset_left 10 -
spacing_right 1 -spacing_left 1 -type block_rings -offset_right 10 -
offset_top 10 -layer_right metal6 -nets {GND VCC } -layer_left metal6

addStripe -set_to_set_distance 100 -spacing 1 -option_file
/h/d3/r/sx08lm4/pr4/soc/appOption.dat/top_fir.ppo.ppo -direction vertical -
layer metal6 -stop_x 1012.759 -width 4.9 -nets {GND VCC } -start_x 585.374

setPlaceMode -congEffort medium -timingDriven 1 -modulePlan 1 -doCongOpt 0 -
clkGateAware 0 -powerDriven 0 -ignoreScan 1 -reorderScan 1 -ignoreSpare 1 -
placeIOPins 1 -moduleAwareSpare 0 -checkPinLayerForAccess {  1 } -
preserveRouting 0 -rmAffectedRouting 0 -checkRoute 0 -swapEEQ 0

setScanReorderMode -skipMode skipNone -clkAware false -defInForce false -
allowSwapping false -keepHierPorts false

setStreamOutMode -specifyViaName default -SEvianames false -virtualConnection
true -uniquifyCellNamesPrefix false

setTieHiLoMode -honorDontTouch false

setTrialRouteMode -highEffort false -floorPlanMode false -detour true -
maxRouteLayer 8 -minRouteLayer 1 -handlePreroute false -autoSkipTracks false
-handlePartition false -handlePartitionComplex false -useM1 false -
keepExistingRoutes false -ignoreAbutted2TermNet false -pinGuide true -
honorPin false -selNet {} -selNetOnly {} -selMarkedNet false -
selMarkedNetOnly false -ignoreObstruct false -PKS false -updateRemainTrks
false -ignoreDEFTrack false -printWiresOnRTBlk false -usePagedArray false -
routeObs true -routeGuide {} -blockageCostMultiple 1 -maxDetourRatio 0

placeDesign -prePlaceOpt

clockDesign -specFile ../design.cts -outDir clock_report -fixedInstBeforeCTS

getFillerMode -quiet

findCoreFillerCells

addFiller -cell FILLER8EHD FILLER64EHD FILLER4EHD FILLER3HD FILLER32EHD
FILLER2HD FILLER1HD FILLER16EHD FILLER8ELD FILLER64ELD FILLER4ELD FILLER3LD
FILLER32ELD FILLER2LD FILLER1LD FILLER16ELD -prefix FILLER -markFixed
```

```
sroute -noPadRings -jogControl { preferWithChanges differentLayer }
setNanoRouteMode -quiet -routeBottomRoutingLayer 1
setNanoRouteMode -quiet -routeTopRoutingLayer 8
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithSiDriven false
trialRoute -handlePreroute
setCteReport
writeDesignTiming .timing_file.tif
routeDesign -globalDetail
verifyGeometry
violationBrowser -all -no_display_false
```

4. Present and discuss violations.

We had Violation in our pads.The first time we had filled our pads and we had the violation which tells us we have overlap in our pads.The second time we did not fill them and we did not have any violations.

Acknowledgements

Thanks to Joachim Rodrigues, Deepak Dasalukunte, Isael Diaz, Chenxin Zhang and Johan Löfgren for helping us to complete our assignments. Thanks to Professor Viktor for the wonderful DSP design course. It's our pleasure to study with your group.

References

[1] IC-project & Verification , digital. http://www.eit.lth.se/index.php?id=241&ciuid=197&L=1

[2] Keshab K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation, John Wiley & Sons, 1999, ISBN Number: 0-471-24186-5.

[3] Matthias Kamuf, "DSP Design: Seminars and Labs", October 2005.