

**Final Project Report**

**on**

**ASIC Implementation of Motion Estimator in 32/28nm CMOS**

**Guided by Prof. Hamid Mahmoodi**

**Advanced Digital Design**

**Fall 2021**

By:

**Chirag C Adiga (922059109**)

ACKNOWLEDGEMENT

Finishing the semester with this project report is an appropriate opportunity to express our gratitude to Dr. **Hamid Mahmoodi** for his invaluable guidance & support.

Without his incredible inputs, patience, excellent guidance and motivation at each and every step of the project we could not have completed this project.

A token of appreciation towards Lovpreet our teaching assistant who helped us at multiple occasions when we were confused about implementing the various concepts.

We are immensely grateful to our friends and classmates. They have supported and guided us at various stops and faults during the project.

.

Table of Contents

1. Understanding the Motion Estimator............................................................4
2. Design Specifications.....................................................................................6
3. Testbench for RTL verification......................................................................7
4. DC Synthesis................................................................................................11
5. Pre-Layout Timing Verification using Prime Time…….…………………18
6. Physical Design............................................................................................19
   1. Floor planning
   2. Placement
   3. Clock tree synthesis
   4. Routing
   5. Extraction
7. Post-Layout Timing Verification using Prime Time……..………….…….29
8. Results……………………………………………………………………..32
9. References…………………………………………………………………33

**1. What is Motion Estimator**

**Motion estimation** is the process of determining **motion** vectors that describe the transformation from one 2D image to another, usually from adjacent frames in a video sequence. Motion estimation is one of the key components of high compression video CODECS. The motion vectors may relate to the whole image (global motion estimation) or specific parts, such as rectangular blocks, arbitrary shaped patches or even per [pixel](https://en.wikipedia.org/wiki/Pixel). The motion vectors may be represented by a translational model or many other models that can approximate the motion of a real video camera, such as rotation and translation in all three dimensions and zoom.

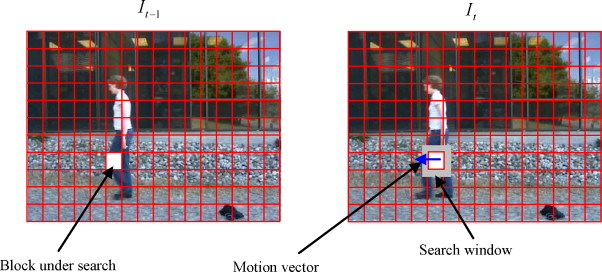


Fig 1: Concept of Motion Estimator

Figure 1 shows the concept of Motion Estimator, it illustrates a process of block-match algorithm. The most popular algorithm for motion estimation is block matching algorithm (BMA) due to simple hardware implementation. In a typical BMA, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. The motion vector and the resulting error can be transmitted instead of the original luminance block; thus, interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames. The summation gives an exact replica of the current frame. The better the prediction the smaller the error signal and hence the transmission bit rate.

**2.Design Specifications**

The goal of the design is to achieve the following specifications:

* The design is expected to meet the speed of 15 frames per second.
* The picture is gray-scale coded (8 bits per block).
* The reference block is 16x16 pixels and the search window is 32x32 pixels. There are 4096 reference blocks in a frame.
* To minimize the total area and energy consumed by the logic over the entire simulation of the test set.
* Must operate at the speed given above.

**Verification Plan:**

Several test benches will be developed with known motion vectors as well as random stimulus. These test benches will be applied to the design and the results will be compared against expected outcomes. The expected outcomes will be produced using a reference model or manually and compared against the outcomes produced by the design under test.

**Design Plan:**

After verification of the RTL code, the design is taken to the ASIC flow starting with front-end synthesis and timing check, followed by back-end physical design involving placement, clock-tree synthesis, and routing. The parasitics will be extracted from layout for final timing check using primetime. Efforts will be made to improve the design in terms of power consumption and area, while meeting the timing.

**3. Test Bench Verification**

Test bench provides the top-level design that stimulus the design. It is an environment in which the product under development is tested with the help of software and hardware tools. Individual modules were verified first by separate testbenches. Then a to-level test bench was implemented to verify the overall functioning of the complete design.

**Verification scripts**

source /packages/synopsys/setup/synopsys\_setup.tcl

Verification of Processing Element module: Arbitrary inputs were assigned within the testbench to inputs and observe the generated outputs. The idea was to verify the logical functioning of the S1S2mux signal such that it selects value S2 rather than S1 on active high signal. Also, arithmetic correctness of the accumulate signal is verified through waveforms.

* For PE verification:

vcs –V –R –full64 PE\_tb.v PE.v –debug\_pp –o simv –gui

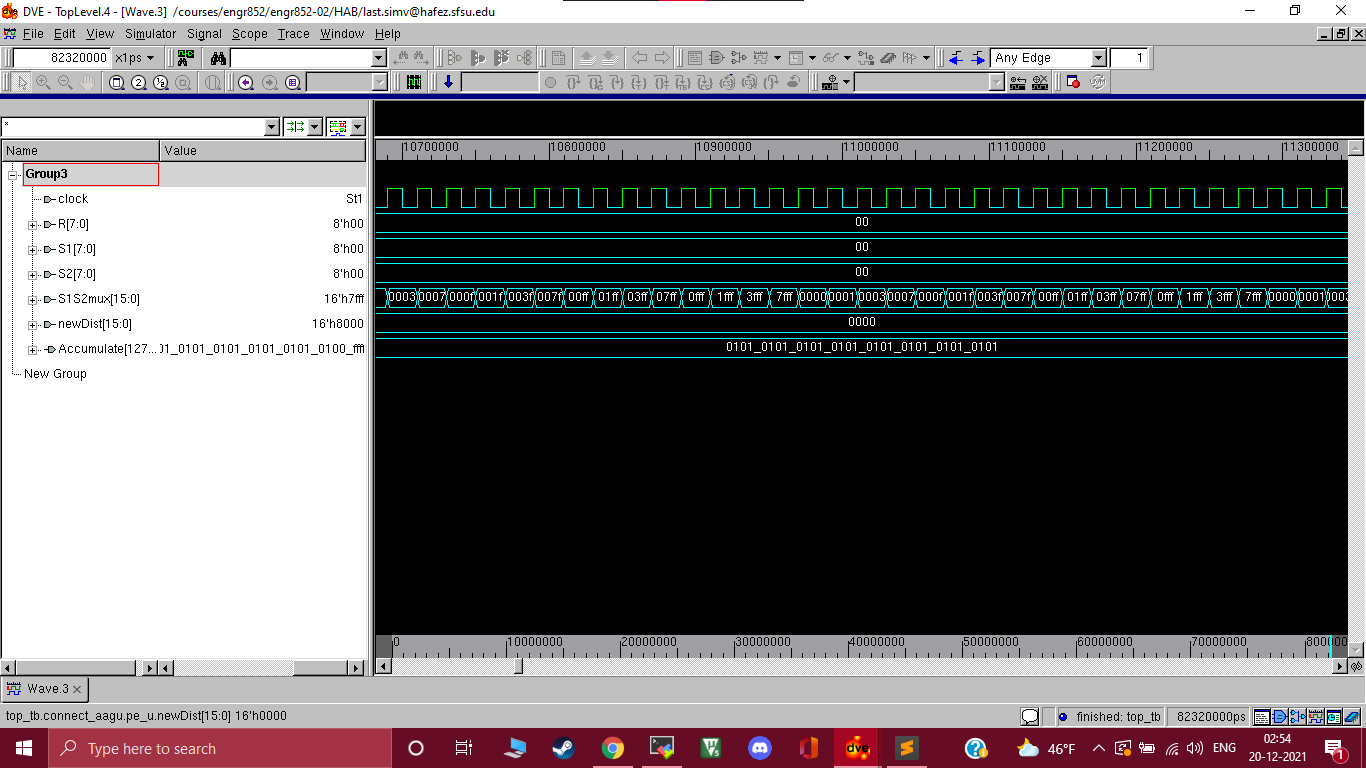


Fig 2: Simulation Results of Processing Element

Verification of Control Unit: The control implemented a counter to generate control signals and the value of these signals were dependent on count register of the control unit. The testbench simply initializes the control unit through a start signal and then monitors the generated signals. The generated signals were observed in the waveforms and the values for all the control signals were as desired.

* For control module verification:

vcs –V –R –full64 ctrl\_tb.v control.v –debug\_pp –o simv –gui

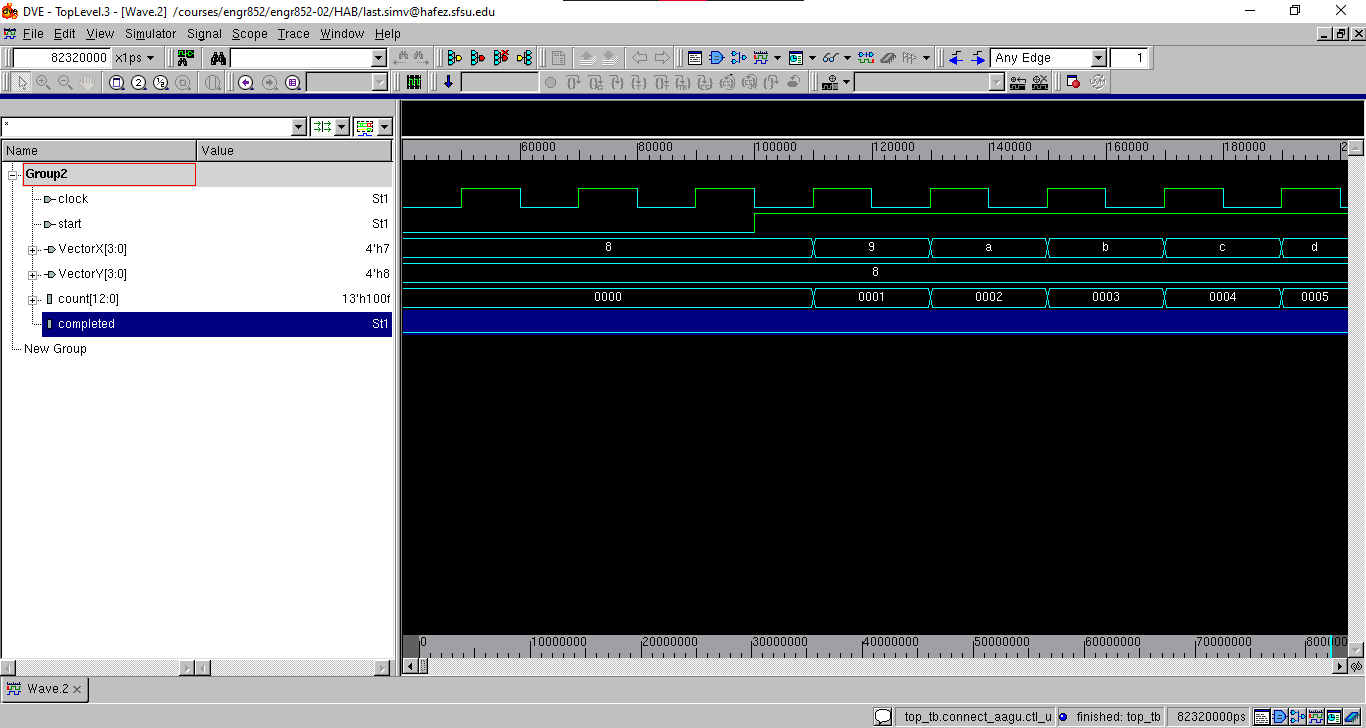


Fig 3: Simulation Results of Control Unit

**Verification of Comparator Unit:** The testbench loads PEout register is loaded with and arbitrary value and then random PEready signals, VectorX, VectorY values are applied by testbench to imitate an input of the PE and Control unit. The idea is to observe the BestDist values and Motion vectors are generated as desired from these input vectors. The values observed in waveforms verified the logical functioning of the comparator unit as the intended values were recorded.

* For comparator module verification:

vcs –V –R –full64 comp\_tb.v comp.v –debug\_pp –o simv –gui

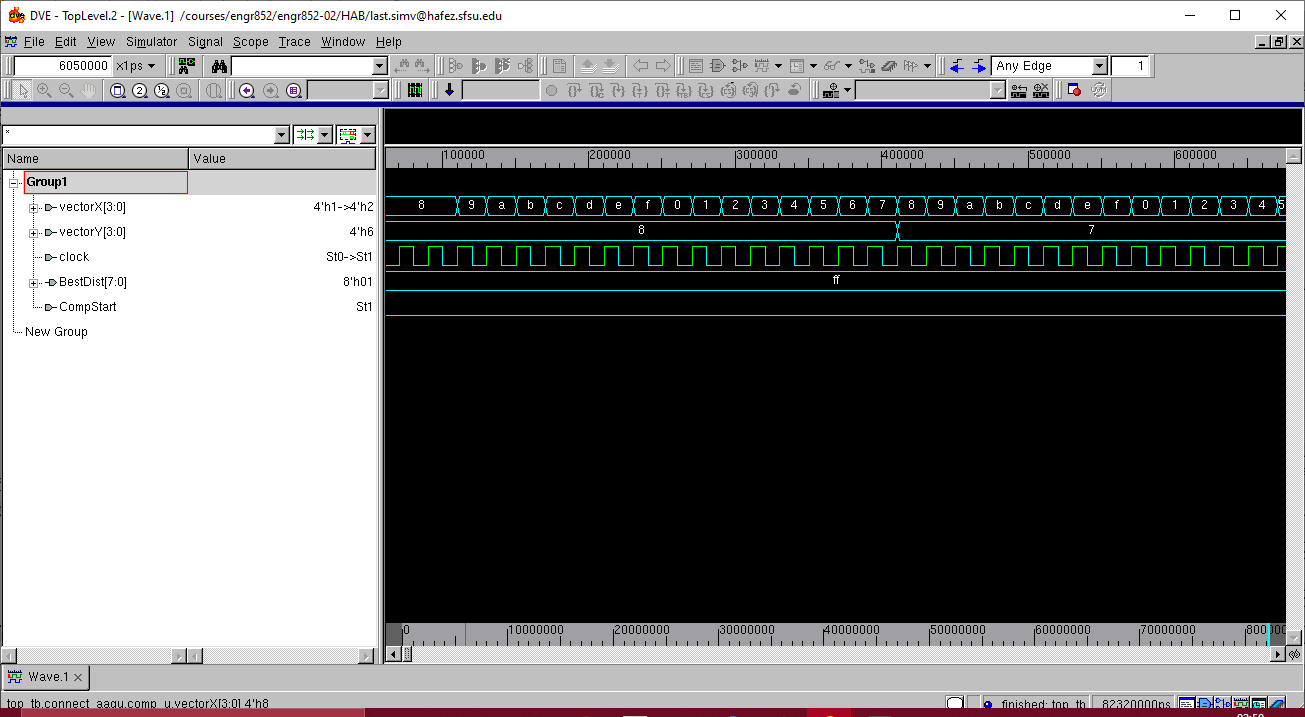


Fig 4: Simulation Results of Comparator

**Verification of top level testbench:**

Top test bench is connected to the top module of RTL and is providing the Reference and Search values to the RTL by reading external .txt files. The first reference value is set to ‘1’ and all other values are set to 0 while search values are all set to ‘0’. The intention is to produce a distortion for initial values which the comparator will output as best distortion value once the CompStart signal goes high and updates the motionX and motionY values as (-8,8).

Since the distortion occurs at first value, the reference window will be at the top left corner of the search window, so the motion for X= -8 and Y =8 should be observed in waveforms. The observed values of motionX in waveforms is 8 and it is a 2’s complement value of (-8). The initial VectorX values from the control RTL are 2’s complement of (-8, -7, -6, -5, -4, -3, -2, -1) as (8, 9, a, b, c, d, e, f, ) when compared with the VectorX expression in control RTL ( these values can be observed in control unit testbench waveforms while verification of control module). Since the motionX value is directly latched from the values of VectorX within the comparator, it verifies the motionX value ‘8’ in waveforms is a 2’s complement of ‘-8’.

The intended values for motion vectors were observed as outputs to verify that the design generates correct motion values for the least distortion observed during overlays.

* For top level verification:

vcs –V –R –full64 top\_tb.v rtl.v –debug\_pp –o simv –gui

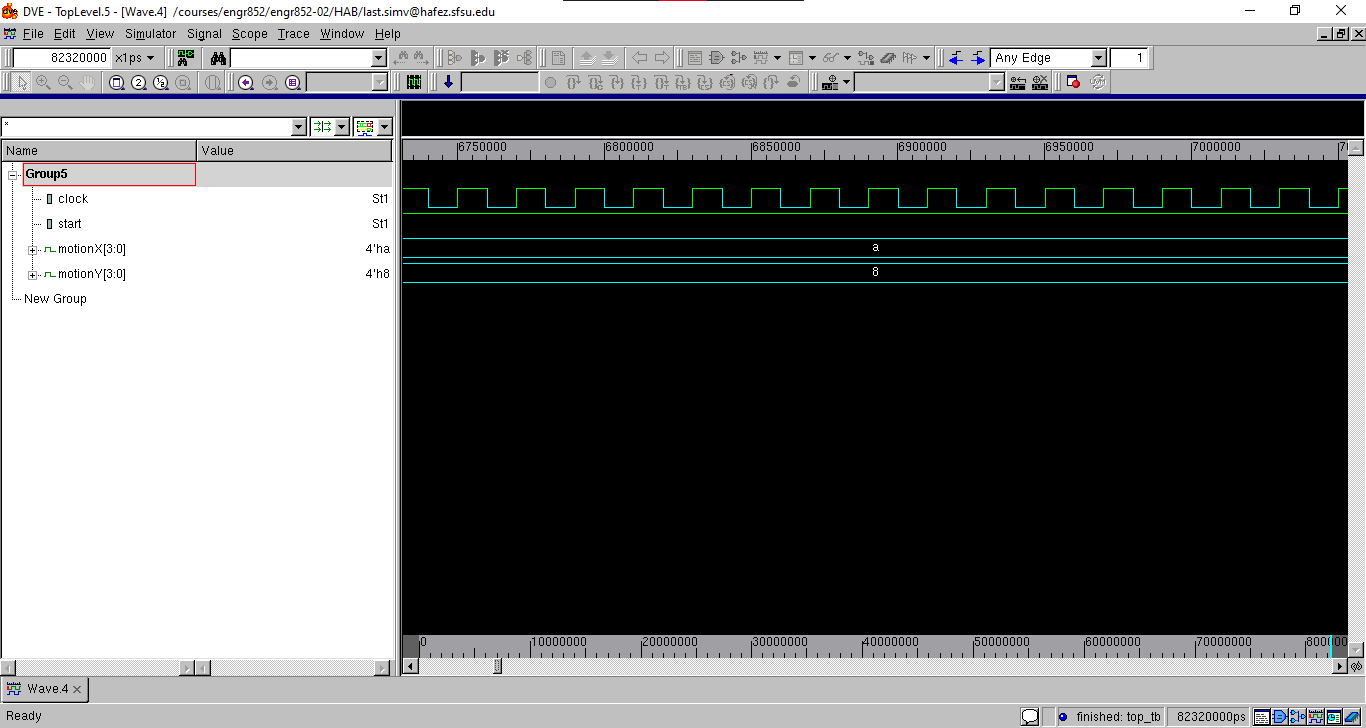


Fig 5: Simulation Results of Top Level Test bench

# 4. DC Synthesis

Design Compiler is a synthesis tool that synthesizes RTL [Register Transfer Logic] hardware description (Verilog/VHDL) which has a standard cell of library as input which provides output of a gate-level-netlist. Gate-level-netlist is a structural representation of standard cells based in the standard cell library.

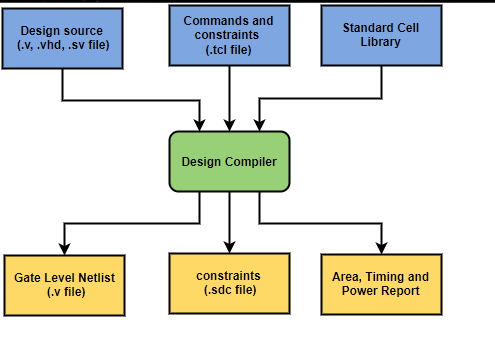


Fig 6: Synthesis Flow

The design on synthesis gave a negative slack for set-up time approx. 3.04 in magnitude. The issue was significant since the value was almost equal to clock frequency. The issue was resolved by using LVT cells for the design working in ff (fast-fast) configuration. The post synthesis set-up time was reported with a slack of 2.95 after the implementation of LVT cells in the design.

**Synthesis scripts**

dc\_shell –xg –t

lappend search\_path path of RTL

define\_design\_lib WORK –path “work”

source ../../../../lib\_setup.tcl

options selected [yes (multi-threshold design), ccs, lvt, ff]

source …/…/…/../lib\_container.tcl

analyze –library WORK –format Verilog rtl.v

elaborate –architecture Verilog –library WORK Switch

create\_clock clock –name ideal\_clock1 –period 3.846

compile –incremental

write\_sdc estimator.sdc

write –f ddc –hierarchy –output estimator.ddc

report\_area > reports/estimator\_area.rpt

report\_cell > reports/estimator\_cell.rpt

report\_design > reports/estimator\_design.rpt

report\_power > reports/estimator\_synthesis\_power.rpt

report\_timing > reports/estimator\_synthesis\_timing.rpt

report\_timing –delay min > reports/estimator\_synthesis\_hold.rpt

**DC synthesis Timing Report:**

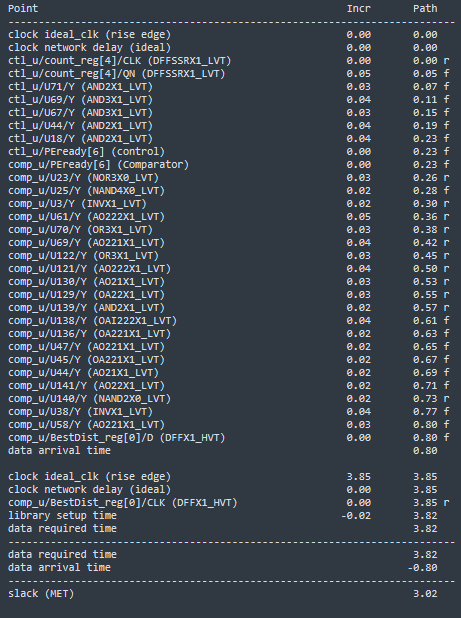


Fig 7 : Maximum Timing Report ( Post Synthesis)

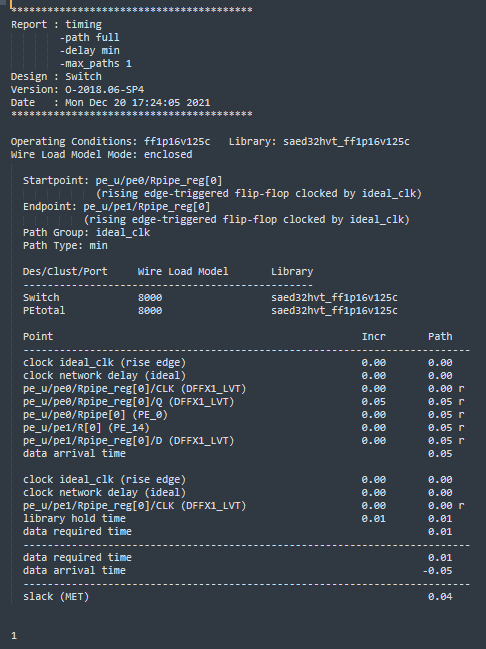
****

Fig 8: Minimum Timing Slack Report ( Post Synthesis)

**Reports of Area Post Synthesis:**

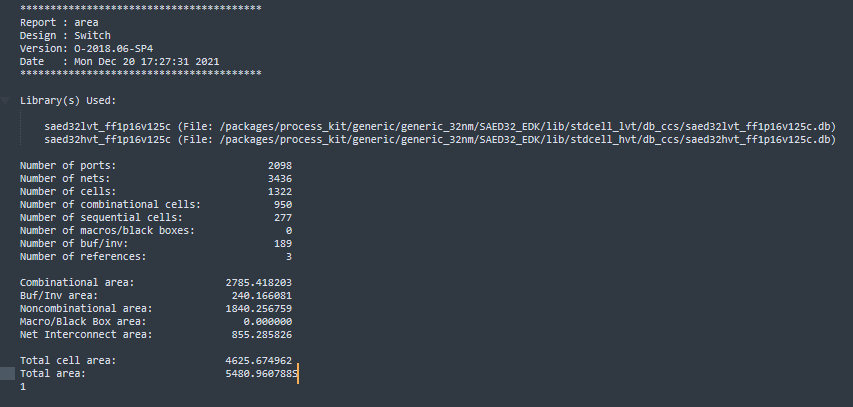


Fig 9. Total Area

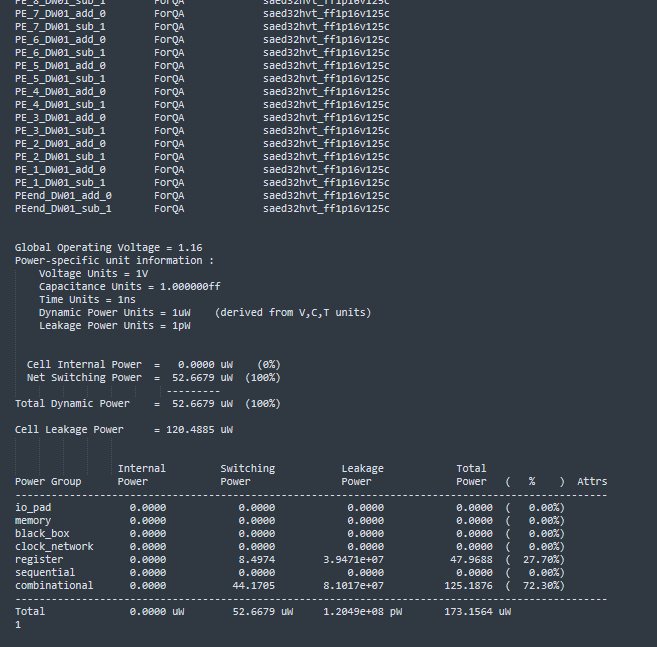


Fig 10: Post Synthesis Power report

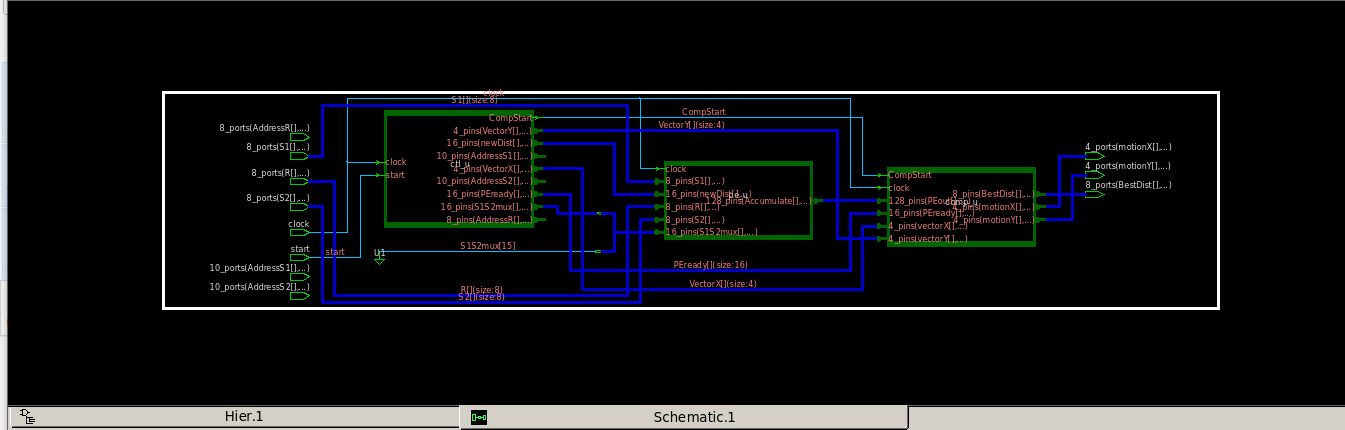


Fig 11: Schematic post synthesis

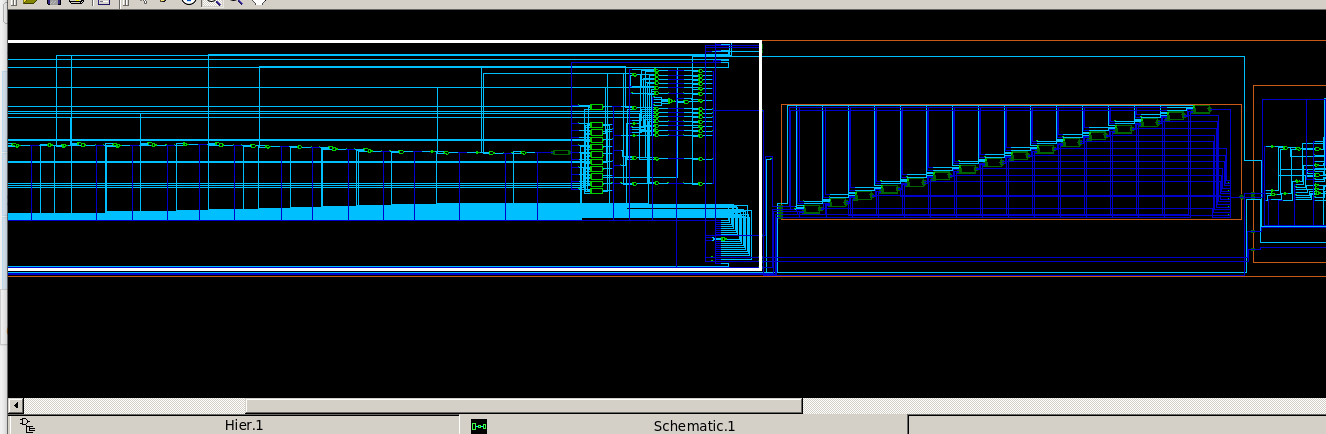
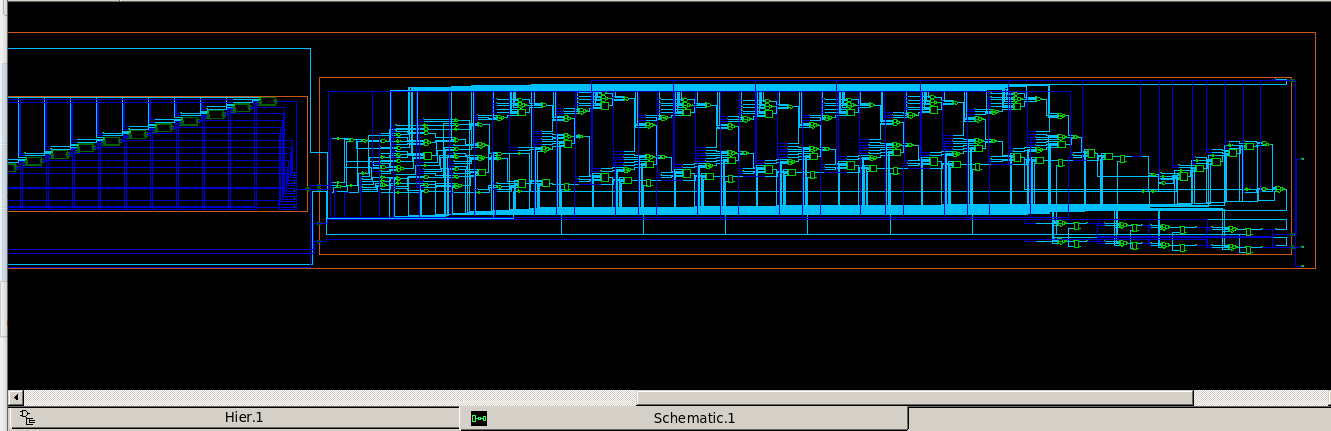


Fig 12: Schematic of Control unit and PE

Fig 13: Comparator Schematic

# 5. Pre-Layout Verification using Prime Time

Pre-Layout PrimeTime Script:

This step is performed for an added confirmation of the slack /

pt\_shell

source …/…/…/../lib\_container.tcl

read\_verilog ../../../estimator\_net.v

current\_design Switch

link\_design

source ../../../estimator.sdc

report\_timimg (setup)

report\_timing –delay\_type min (hold)

**6. Physical Design:**

In Physical design the circuit description is transformed into a physical design where the position of the cell is described and routes for the interconnection between them. ASIC physical design is a part where the design meets the physical world and constraints are added. Below is detailed flowchart of ASIC flow which consists of Initialization, Floorplanning, Placement, Clock Tress Synthesis, Routing and Extraction.

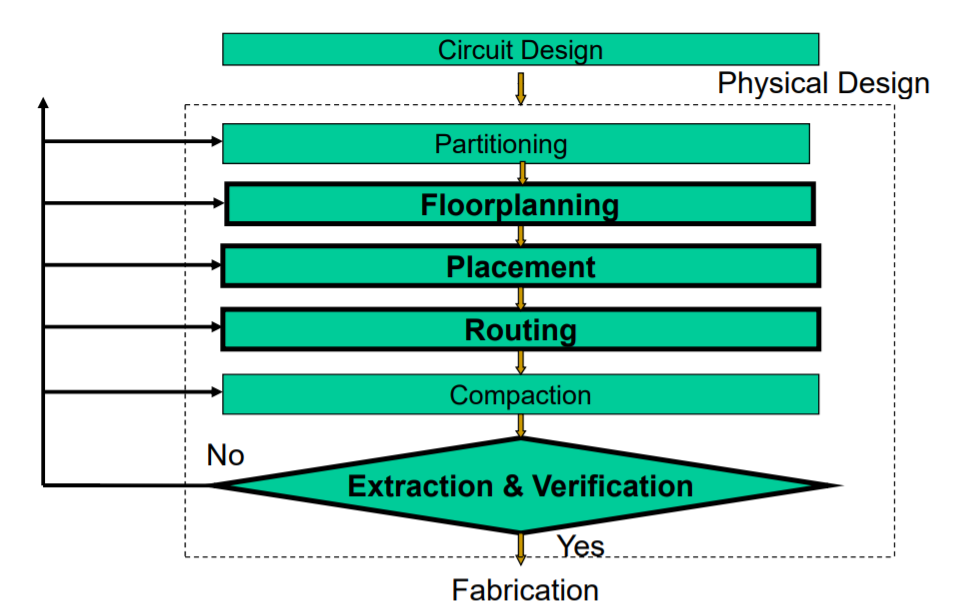


Fig 14: Physical Design Flow Chart

The designing in GUI is more educating, due to availability of multiple options such as view by hierarchy, view by color at the placement, CTS and routing steps of the design flow. These options help us to visually understand the cell placement and wiring connections of various modules.

**Physical Design Scripts**

**Initial environment setup**

icc\_shell

gui\_start

set mw\_logic0\_net VSS

set mw\_logic1\_net VDD

set\_tlu\_plus\_files -max\_tluplus /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_1p9m\_Cmax.tluplus -min\_tluplus /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_1p9m\_Cmin.tluplus -tech2itf\_map /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_tf\_itf\_tluplus.mp

create\_mw\_lib -technology /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/milkyway/saed32nm\_1p9m\_mw.tf -mw\_reference\_library {/packages/process\_kit/generic/generic\_32nm/EDK\_updated\_april2012/SAED32\_EDK/lib/stdcell\_lvt/milkyway/saed32nm\_lvt\_1p9m} estimator\_design.mw

open\_mw\_lib estimator\_design.mw

read\_verilog ../../output/estimator\_net.v

uniquify\_fp\_mw\_cel

link

read\_sdc ../…/…/estimator.sdc

save\_mw\_cel –as estimator\_inital\_design\_setup

**Floor planning**

At the stage of floor planning, we have the netlist that describes design and interconnection between different blocks. It defines the I/O ports, power supply connections. The final timing, quality of the chip depends on the floorplan design.

**Script for Floor planning:**

create\_floorplan -core\_utilization 0.6 -start\_first\_row -left\_io2core 5.0 -bottom\_io2core 5.0 -right\_io2core 5.0 -top\_io2core 5.0

derive\_pg\_connection -power\_net VDD –ground\_net VSS

derive\_pg\_connection -power\_net VDD –ground\_net VSS –tie

//Create VSS ring

create\_rectangular\_rings -nets {VSS} -left\_offset 0.5 -left\_segment\_layer M6 -left\_segment\_width 1.0 -extend\_ll -extend\_lh -right\_offset 0.5 -right\_segment\_layer M6 -right\_segment\_width 1.0 -extend\_rl -extend\_rh -bottom\_offset 0.5 -bottom\_segment\_layer M7 -bottom\_segment\_width 1.0 -extend\_bl -extend\_bh -top\_offset 0.5 -top\_segment\_layer M7 -top\_segment\_width 1.0 -extend\_tl -extend\_th

// Create VDD Ring

create\_rectangular\_rings -nets {VDD} -left\_offset 1.8 -left\_segment\_layer M6 -left\_segment\_width 1.0 -extend\_ll -extend\_lh -right\_offset 1.8 -right\_segment\_layer M6 -right\_segment\_width 1.0 -extend\_rl -extend\_rh -bottom\_offset 1.8 -bottom\_segment\_layer M7 -bottom\_segment\_width 1.0 -extend\_bl -extend\_bh -top\_offset 1.8 -top\_segment\_layer M7 -top\_segment\_width 1.0 -extend\_tl -extend\_th

create\_power\_strap -nets { VDD } -layer M6 -direction vertical -width 3

create\_power\_strap -nets { VSS } -layer M6 -direction vertical -width 3

save\_mw\_cel –as estimator\_floorplan

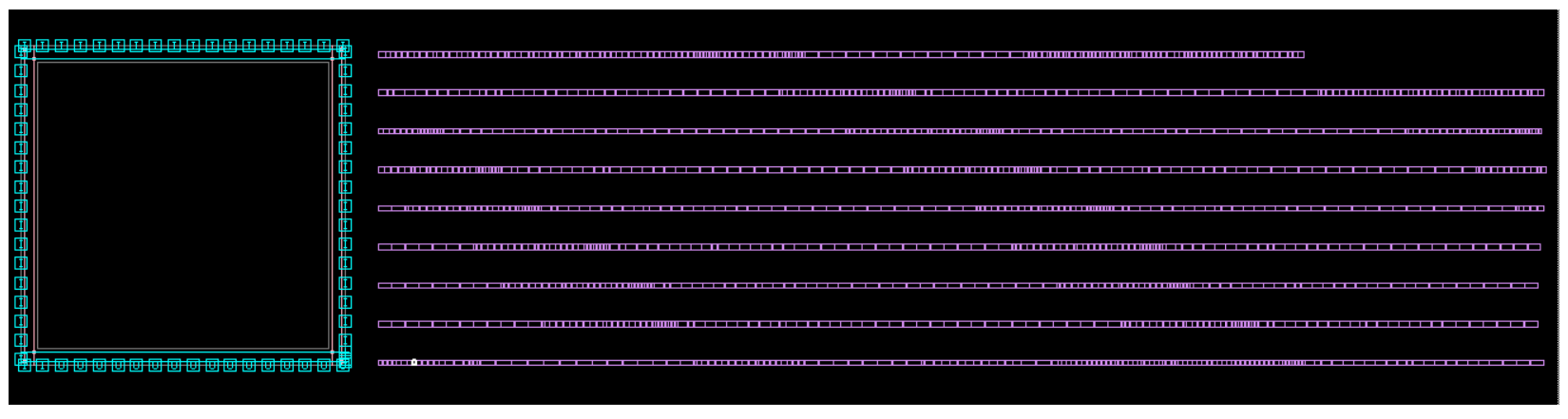


Fig 15: Floor Planning

**Placement**

The standard cell locations are defined to a particular position in a row. It is a process of placing the design cells in the floorplan in the most optimal way. The main algorithm is to make the chip as dense as possible. To minimize the total wire length. The number horizontal/vertical wire segments crossing a line.

**Scripts for Placement:**

set\_buffer\_opt\_strategy -effort low

set\_tlu\_plus\_files -max\_tluplus /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_1p9m\_Cmax.tluplus -min\_tluplus /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_1p9m\_Cmin.tluplus -tech2itf\_map /packages/process\_kit/generic/generic\_32nm/SAED32\_EDK/tech/star\_rcxt/saed32nm\_tf\_itf\_tluplus.map

place\_opt –congestion

report\_placement\_utilization > reports/estimator\_place\_utilization.rpt

report\_timing -delay max -max\_paths 1 > reports/estimator\_placement.setup\_time.rpt

report\_timing -delay min -max\_paths 1 > reports/estimator\_placement.hold\_time.rpt

save\_mw\_cel -as estimator\_placement



Fig 16: Placement

**Clock Tree Synthesis:**

CTS is a process which makes sure that the clock gets distributed evenly to all sequential elements in a design. During this process, depending on the clock skew, buffers are added to the different clock paths in the design. More delays (buffers) are added on the launching side to have a better hold time. More delays (buffers) are added to latching side to have a better Set up time.

**Clock Tree Synthesis Scripts**

*Clock > Core CTS and Optimization* action in GUI.

report\_timing -delay max -max\_paths 1 > reports/estimator\_cts.setup.rpt

report\_timing -delay min -max\_paths 1 > reports/estimator\_cts.hold.rpt

save\_mw\_cel –as estimator\_cts

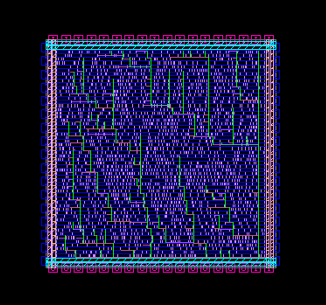


Figure 17: Clock Tree Synthesis

**Routing**

After the floor planning and placement steps in the design, routing needs to be done. Routing is the process of connecting the various blocks in the chip with one another.

**Routing Scripts:**

*Route > Core Routing and Optimization* action in GUI.

report\_placement\_utilization > reports/estimator\_route\_utilization.rpt

report\_timing -delay max -max\_paths 1 > reports/estimator\_route.setup.rpt

report\_timing -delay min -max\_paths 1 > reports/estimator\_route.hold.rpt

save\_mw\_cel –as estimator\_route

A screenshot of a computer

Description automatically generated with medium confidence

Figure 18 . Routing

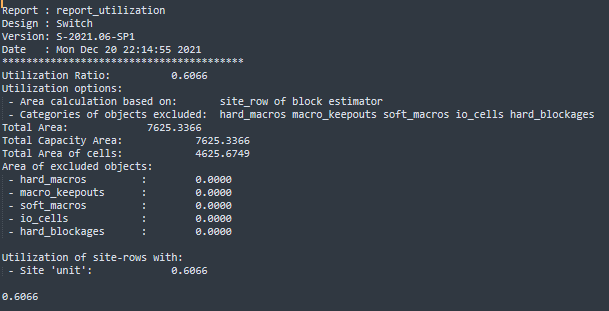


Fig 19: Utilization Report

**Extraction:**

Extraction is generating RC parasitics post routing and generating a new netlist. All layouts can extract the layout database using various algorithms and these algorithms define the accuracy of extracted values.

**Extraction Scripts:**

extract\_rc -cou

pling\_cap -routed\_nets\_only -incremental

write\_parasitics -output ./outputs/estimator\_extracted.spef -format SPEF

write\_sdf ./outputs/estimator\_extracted.sdf

write\_sdc ./outputs/estimator\_extracted.sdc

write\_verilog ./outputs/estimator\_extracted\_net.v

report\_timing -delay max -max\_paths 1 > reports/estimator\_extraction.setup.rpt

report\_timing -delay min -max\_paths 1 > reports/estimator\_extraction.hold.rpt

report\_power > reports/estimator\_extracted\_power.rpt

save\_mw\_cel –as estimator\_extracted

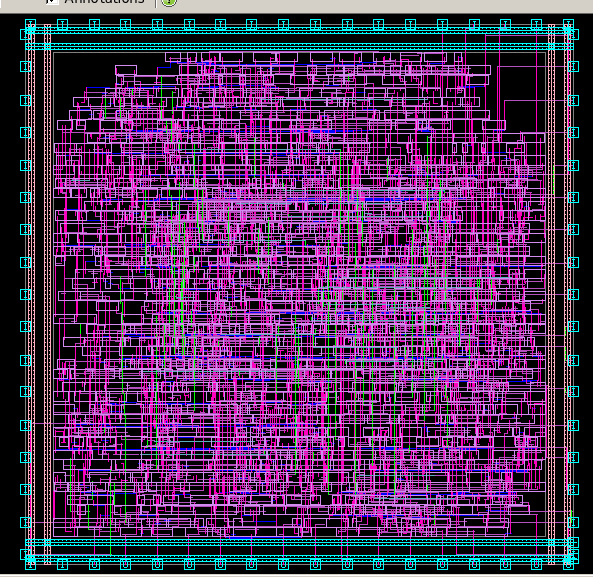


Fig 20: Placement of Cell in Different Colors

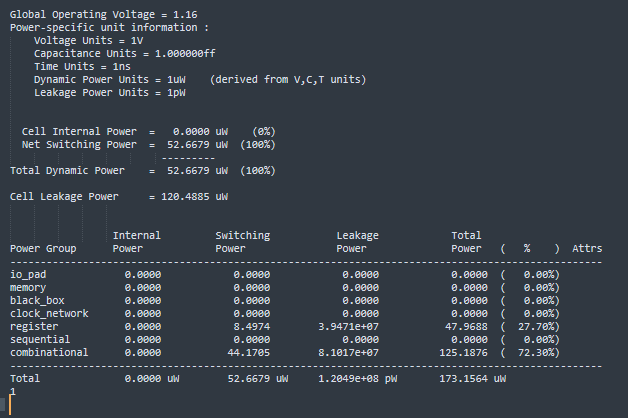


Fig 21: Power report after extraction

**7. Post Layout Timing Reports (Prime Time):**

To perform timing verification on the extracted netlist we run primetime. We use the post level netlist as well as parasitic information at this stage.

Primetime verification post layout presented challenges in analysis due to unresolved links between cells and reference library. We need to figure out the sourcing and linking appropriately of the required files. Out of the four available modes for the STA, check carried out in mode of primary inputs to primary outputs.

**Post-Layout Primetime Scripts:**

pt\_shell

source …/…/…/../lib\_container.tcl

read\_verilog ../../../estimator\_extracted\_net.v

current\_design Switch

FOR SETUP

read\_parasitics -format SPEF ../../outputs/estimator\_extracted.spef.max

read\_sdc ../../estimator\_extracted.sdc

report\_timing -from [all\_inputs] -to [all\_outputs] -max\_paths 1 > reports/ timing\_setup.pt\_postlayout.rpt

FOR HOLD

read\_parasitics -format SPEF ../../outputs/estimator\_extracted.spef.min

read\_sdc ../../estimator\_extracted.sdc

report\_timing -from [all\_inputs] -to [all\_outputs] -max\_paths 1 -delay\_type min > reports/ timing\_hold.pt\_postlayout.rpt

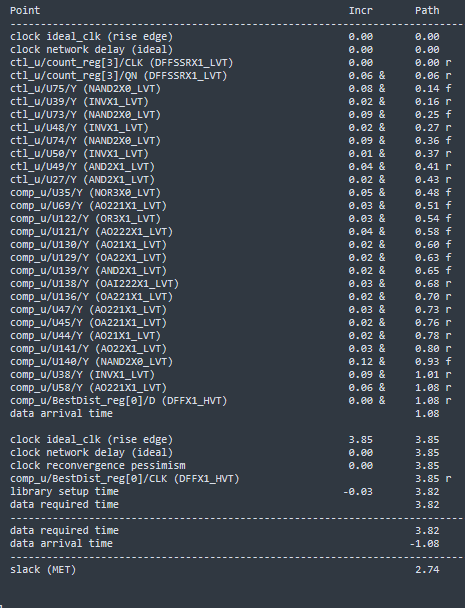
**Post-Physical Design Timing Reports:**

Fig 22: Maximum Timing Report (PD)

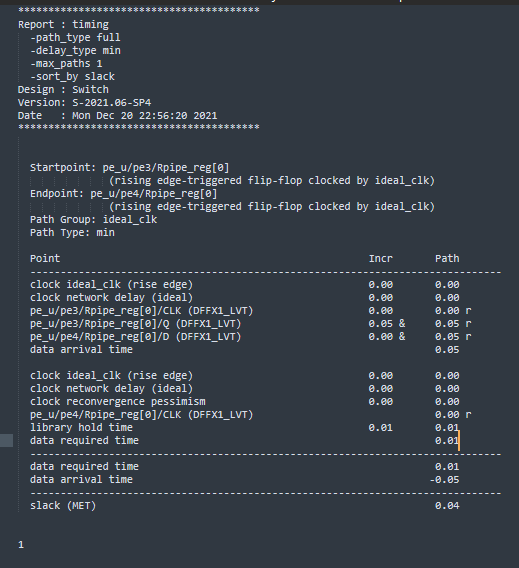


Fig 23: Minimum Timing Report (PD)

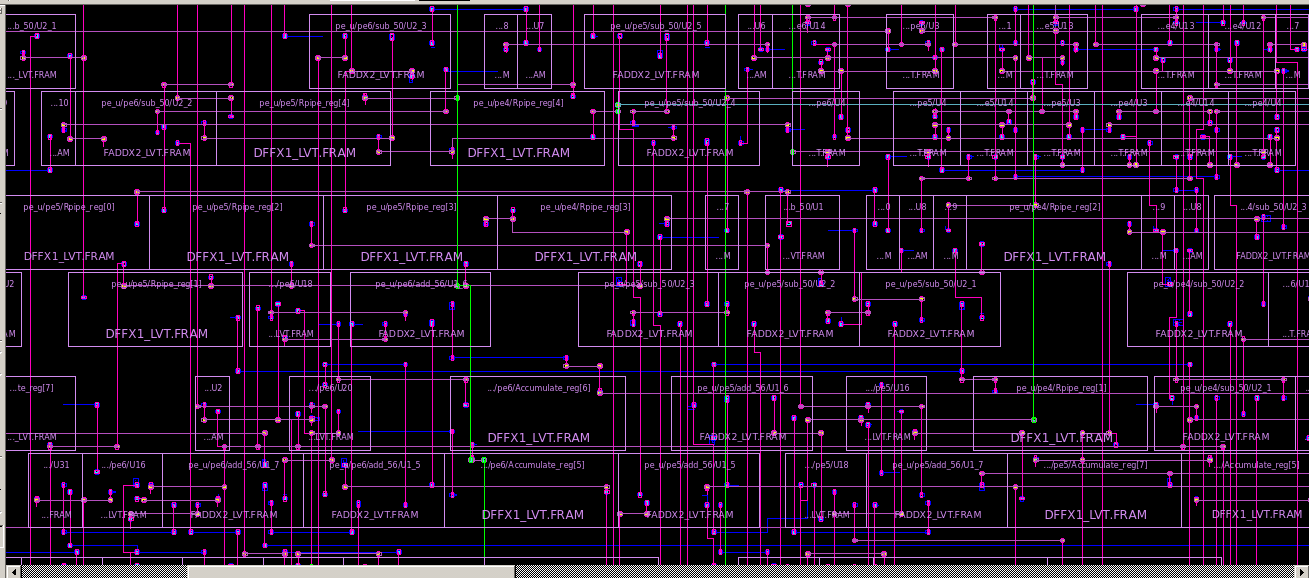


Fig 24: Final Layout

**Results**

Having completed the ASIC flow from RTL to GDS, We Verified the design by simulation.

We have met the timing for both setup and hold in post layout timing verification.

We have optimized the Area and Power.

Total Area: 5480.9607 um2 (after synthesis)

Cell leakage Power: 147.3350 uW (after synthesis)

Dynamic Power: 52.0520 uW (after synthesis)

Total Power: 173.1546 uW (after synthesis)

Chip Area: 9471.817 um2 (post pnr)

Cell leakage Power: 1.16e+08 pW (post layout and extraction)

Dynamic Power: 2.91e+07 pW (post layout and extraction)

Total Power: 210.2477 uW (post layout and extraction)

Set up time violations: No setup time violations in post layout timing analysis and have a slack value of (2.74).

Hold Time violations: No hold time violations in post layout timing analysis with slack value of (0.04).

**References**

[1] M. E. Rizkalla, et. al. “Hardware Implementation of Block-based Motion Estimation for Real Time Applications,” Journal of VLSI Signal Processing, pp. 139-159, 2007

[2] R. Meagher, et. al, “VHDL Design for Real Time Motion Estimation Video Applications,” Journal of Signal Processing Systems, Oct. 2008

[3] S. Lee, J. Kim, and S. Chae, “New motion estimation algorithm using adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding,” IEEE Transactions on Circuits and Systems for Video Technology, vol.8, no. 6, pp. 734-744, 1998

[4] Engr852 Lecture Notes from Professor Hamid Mahmoodi