

**SAN FRANCISCO STATE UNIVERSITY**

**SCHOOL OF ENGINEERING**

**Final Project Report: Motion Estimator**

ENGR 852-01

Fall 2025

By

Chuan Li, #924320342

[cli43@sfsu.edu](mailto:cli43@sfsu.edu)

Dahlen Lakin, #911161763

[dlakin@sfsu.edu](mailto:dlakin@sfsu.edu)

Oscar Barajas, #916364337

[obarajas2@mail.sfsu.edu](mailto:obarajas2@mail.sfsu.edu)

Date report submitted: Dec. 20, 2020

## Table of Contents

|                        |    |
|------------------------|----|
| Problem Analysis ..... | 3  |
| Hardware Design.....   | 3  |
| Verification.....      | 5  |
| Synthesis.....         | 10 |
| Physical Design .....  | 14 |
| Sign-Off.....          | 19 |
| Discussion .....       | 21 |
| Conclusion.....        | 21 |

**Problem Analysis:** Today's world depends very heavily on video files especially since a majority of human communication is currently virtual. Videos are now being used everywhere, from the movie industry all the way to education. Especially with the rise of 4K (and soon 8K), video sizes are getting larger than ever, making it increasingly difficult to share and store on one's computer. This is what makes the practice of Motion Estimation so important. Instead of simply storing each pixel for each frame, one can instead calculate and store the much more portable motion vector of subsequent frames, drastically decreasing file size.

We are tasked to implement this device using an Application Specific Integrated Circuit. By implementing the Motion Estimator using dedicated hardware, we have the potential to increase efficiency over a software implementation using algorithms. The 32/28nm CMOS Library will be used when designing the physical layout of the chip. As for the motion estimator itself, the design will be capable of reading a 16x16 Reference Frame coded in an 8-bit Grayscale. The subsequent Search Frame for which we will be determining the motion displacement vector will have dimensions 31x31 also coded in Grayscale. Only the 260MHz Clock is available to us, and we will have to meet the minimum search speed of 15 Frames/Second.

**Hardware Design:** Before writing a Verilog module to implement this Motion Estimator, we must first plan out the behavioral structure of our design. This will reduce the likelihood of creating bugs in the design. A Black-Box representation of our design can be seen below in Figure 1.

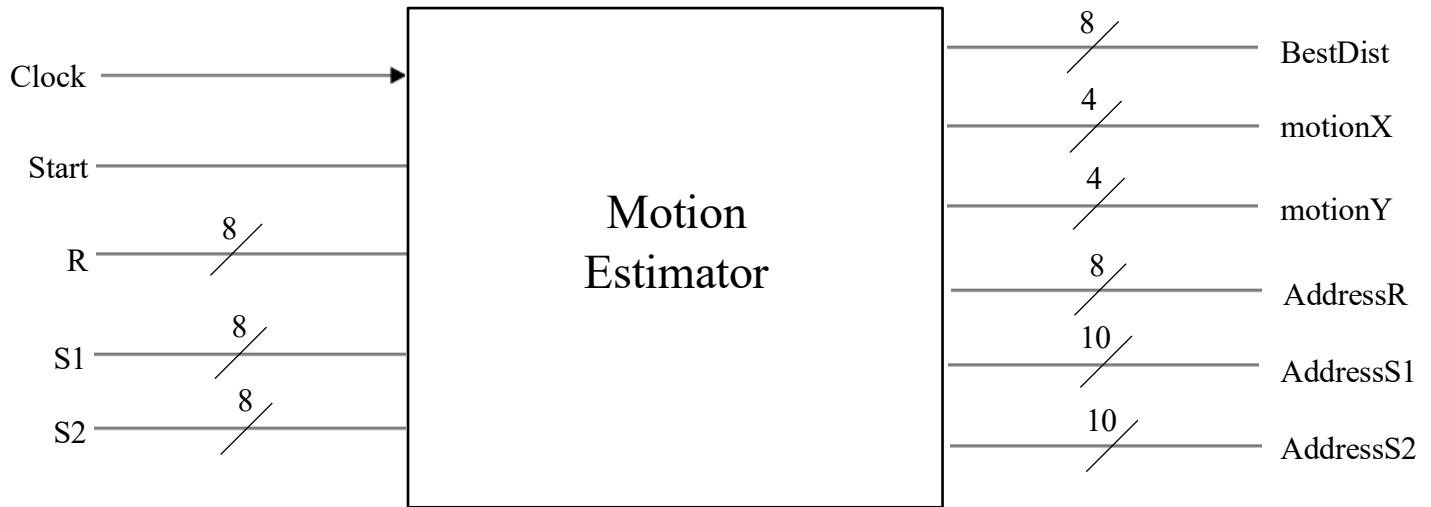


Figure 1 - Motion Estimator, Black Box Representation

As indicated above, all operations will take place on the rising edge of the clock signal as opposed to using a level-sensitive latch-based design. A start signal will signal the Motion Estimator to begin executing calculations when the start signal is high. Note, our memories R, S1, and S2 (representing Reference Frame and two Possible Search Frames) are external to the

Motion Estimator and will not be included in our final design. Therefore, we will have to access

the memory contents as input signals to the Motion Estimator, making it possible to switch out different memories to compute Motion Estimation on. Our design will be capable of calculating the amount of distortion seen on subsequent frames, allowing us to deduce how similar two frames really are. The estimated motion vectors for the X and Y axis will be in Two's Compliment form, meaning that motion can range anywhere from -8 to 7 pixels.

Fortunately, we have several reference materials available to us on how to compute the Motion Vectors. We will be using the most commonly used Block Matching Algorithm to implement the Motion Estimator. The way that this algorithm works is that a frame comparison must be made between Reference and Search frames for each possible position of the Reference Frame. If the frame comparison yields low distortion, the vector describing the movement of the original reference frame to that position on the search frame will be saved as the estimated motion until a comparison yielding an even lower distortion is found.

In order to calculate this distortion, we will first have to implement a Processing Element capable of determining an accumulated distortion value based on the difference between the R memory and either S1 or S2 depending on which memory we want to use. If the distortion is greater than the maximum value allowed by 8 bits (255), distortion will simply saturate at this maximum value. To achieve the required speed of our motion estimator, we will utilize Pipelining by having 16 Processing Elements working in Parallel, though with the Reference Memory

‘pipelining’ through each one.

Then, we must make a Comparator Module that will receive the 16 distortion values from the Processing Elements (all accumulated into one long vector) and determine whether they are smaller than the currently saved value, initially set to the maximum value of 255. If distortion is smaller, we will save the frame displacement as our motion X and Y vectors.

Next, to generate all of the signals that will start calculating and comparing new distortion values, accessing memory addresses, etc., we will implement a controller that will mimic the behavior of a reference block moving through each possible position of a search block.

Finally, all of these modules will be brought together in one top module like the one shown in Figure 1.

## I . Verification

### A. VCS

```

13 70 fd e2 97 f1 c5 ec 48 0c 2c 6b 1b 45 f4 6c 67 8c 4a a6 a3 9d 7c b8 eb 5b f3 4d 5c f6 d9 a9
aa 9e c8 ad 26 f6 60 5b 42 21 91 38 4f 6d c2 79 7c 2e 73 1d e3 fc 0a 13 aa 3c 3a 78 e7 97 7f e3
20 c1 bf 91 49 4c 76 62 4c 0c 8d c8 1f ed 23 99 42 8f f6 15 f7 0e ca 6d 06 50 1a 8f 5d 2f 21 4e
0f 74 04 47 3d 22 9c 3e 36 94 a5 2b e2 41 ab 35 1d 02 c3 3d 05 be 8c 01 38 ce 91 65 0c 45 f2 80
77 10 66 ba 56 b3 ca 46 74 26 24 6f 24 a7 bf 99 12 3a b9 6d be 68 d1 70 42 b4 95 3f 1a b8 15 9d
2a 2b cc 23 b7 d1 4a 9a d4 40 2f a3 7d e5 bb 72 37 a8 69 4f cd 8b f9 6c 5b 53 ea 99 47 bc 2d 9d
a7 33 18 e8 89 1a 2a ac ca 8f d3 8a ec 92 ee 6d 68 0a a0 66 fb 7e 88 47 2c 16 36 71 f7 a6 1c 0f
ed 7c 34 40 23 d2 b3 3a 18 41 f1 67 56 5b 6d 69 85 07 67 76 9f fc db af db 8c 5e b1 cd bc f8 12
35 64 8e b7 5c df 80 2b 0b 08 6e 05 87 5d c1 bb 40 cc 35 0b dc f6 e6 43 30 82 7d dd dd ca 4a 1c
80 f7 8c 39 7e 1a 31 10 81 ba 12 05 c4 20 0b ca 6a c6 4b a1 4c 72 07 b1 2e ee 80 70 a9 4f 68 01
84 39 e7 89 61 71 79 9d 50 25 08 fc 96 5b a0 b6 a0 3d 97 37 22 b8 38 0e bd 6d f3 1c 3b 3c 2e
ba 7a 86 a0 f1 6a a7 b6 3d 8e 0c bf 4f 4a 33 ac 6b 96 f0 8b 1e 38 52 3c 87 c0 6c 49 84 1c fa d2
bf 9d 93 3c 8f 8d 16 26 f6 63 1d ad 4e 07 2b b4 59 62 a0 67 24 ae 85 b5 ba e4 dd d2 9f 99 66 70
51 ef 74 5a 9a d5 60 4d 2e 22 a8 32 2d be 1a 9b ca 5d 9a e1 67 50 c0 21 78 ab e7 08 b1 74 75 08
47 2f 6c f0 58 c2 45 57 46 95 de 5a b4 dc e1 f3 67 8e 43 ce 5e 54 2f b8 24 09 d0 fc bb c0 2a b1
7c e5 ba c8 41 cf f3 3b c2 83 fd 0e 4f d9 8d 76 0f b1 b6 9a a8 6d 24 df 58 58 3e 5b 4a c5 f2 e5
41 3d 31 4e 02 f6 96 14 c9 be 8b 3d 86 a6 07 52 8c 07 0b 89 be c9 00 69 c6 e5 52 55 30 ad 18 21
bf e1 6c 87 3e 2a 03 56 9e bb 55 8d d2 ed 43 c1 0c b9 c5 64 a1 8d ec a0 b3 88 99 0b d5 ea 62 7d
3c 28 3b 0b 99 97 00 9e 81 8f 9d 8e b2 00 b4 6a 1a c5 16 4b 7f 00 6d ba bc 43 f5 93 a3 49 67 e6
3d 48 c3 ed b6 bc 30 5d 06 37 12 4f 27 c5 ad 93 0c 5f fb 0b c7 80 a7 b7 6a d5 9b 72 e4 8d 27 cd
26 7d 90 c4 65 e3 96 a1 20 3f be 9e 9e 2a 6a 03 95 f0 44 d5 d0 30 4b 4e 53 99 03 47 1f fb 90 d7
ca e3 56 19 03 43 13 e0 3b 4a 6a 00 1c b2 6e 03 8a c3 5b 01 61 25 8f 40 19 e3 c8 1a 9a 98 ac 36
a9 49 60 cf c9 52 fd 7f 22 7d 3c e3 9a 49 b6 6a c4 39 7b 38 0f 0f 75 77 da 4e 53 f6 40 7e 37 16
f9 05 f4 b8 cd a1 8b 54 96 cf 29 20 69 85 ed 52 86 00 65 3f 32 19 60 53 6f bd c2 32 f7 1d cf 7c
b6 ee 96 dc 42 5b 94 e3 cb 24 4d 85 6e 7c 02 21 a9 f2 14 d7 ac e2 47 f3 01 1c 97 36 21 cb 2b 68
a8 f4 3a 39 a0 2e 88 a3 ab e1 68 80 33 d0 67 83 95 ce bf 86 f0 83 4e fc 1e d3 66 76 2c e4 c2 a4
a2 d4 b6 fa 8d 7f 14 96 63 c4 da 78 57 23 28 1f 02 29 75 a0 31 7e 1e ef 6b b0 19 c5 d0 7e 99 ca
1b 59 8d 68 23 cf 4d 98 97 ab 70 37 f7 eb 93 71 87 49 f7 61 1b 87 78 22 66 f3 dd c3 d7 ff 46 36
7e 75 48 23 53 b0 b7 76 6a d0 4c f1 34 ea 69 90 ae b0 b1 5a 1e 52 0e 0c e4 00 c9 12 f3 39 5f 14
bd 8a d6 c0 1b 20 43 95 e9 bd 86 44 85 3a 85 c8 50 53 b1 7e bc 36 97 b2 c7 34 8a a3 79 d6 90 bb
0d cd 47 01 7f 93 d9 63 69 2b 45 b8 35 93 7f ec e3 95 ba d4 6a b5 36 b3 3d 97 cf 6b 6b 31 44 5b
07 77 08 7a bd e7 49 99 1a 54 ac 82 04 8f 8a b6 47 c6 47 75 1a 09 46 bd 24 f7 3d 46 be 7b d7 80
Perfect Match Found for Reference Memory in the Search Window : BestDist = -2 , motionX = -2 , motionY = -3 Expected motion
nX = -2 , Expected motionY = -3
DUT motion outputs Do match expected motions: DUT motionX = -2 DUT motionY = -3 Expected_motionX = -2 Expected_motionY = -3
onY = -3
All tests completed

```

Figure 2 – VCS Verification

There are three main steps in debugging the design, which are as follows

1. Compiling the Verilog/VHDL source code.
2. Running the Simulation.
3. Viewing and debugging the generated waveforms.

In this step, the design and testbench are compiled using Synopsys VCS. During compilation, VCS checks the design for syntax errors, module connectivity, and signal mismatches as shown in Figure 2. If the design does not compile successfully, the simulation cannot proceed, so this step ensures that the RTL is structurally correct and ready to execute.

## B. Testing Input:

```
// First initialize all registers
$readmemh("ref.txt", memR_u.Rmem); // Load the reference block pixel values from ref.txt
into the memory which is 16 x 16
$readmemh("search.txt", memS_u.Smem); // Load the search window pixel values from
search.txt into the which is a 32 x 32 block
clock = 1'b0; // Initialize the clock
start = 1'b0;
completed = 1'b1;

@(posedge clock);#10;
start = 1'b1;
completed = 1'b0;

for(i=0;i<4112;i=i+1) // Run the DUT for exactly 4112 clock cycles count 0- to 4112
begin

    if (dut.comp_u.newBest == 1'b1) // Check if the comparator has found a new
best distortion value.
    begin
        $display("New Result Coming!");
    end
    @(posedge clock);#10;
End
```

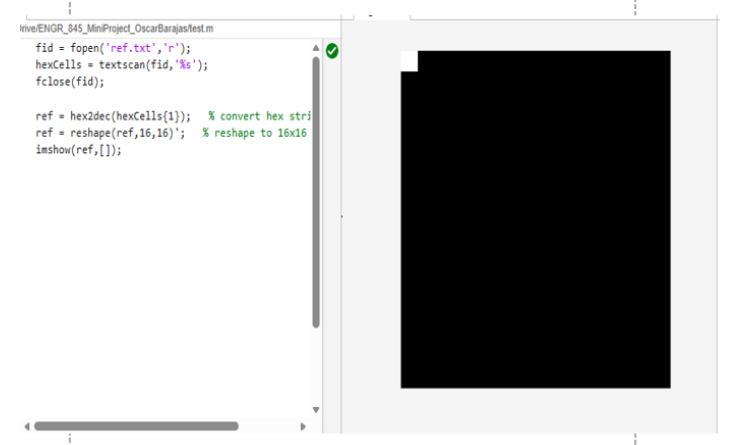


Figure 3 – Input Text Files

The testbench begins by loading two input files using the \$readmemh command.

The first file in Figure 3, ref.txt, is loaded into the reference memory and represents a 16 by 16 reference image block. This block contains the pixel pattern that the design is trying to locate.

The second file in Figure 3, search.txt, is loaded into the search memory and represents a 32 by 32 search window. This larger window defines the region where the reference block is searched for possible matches.

### C. Functional Verification

#### a. Worst case

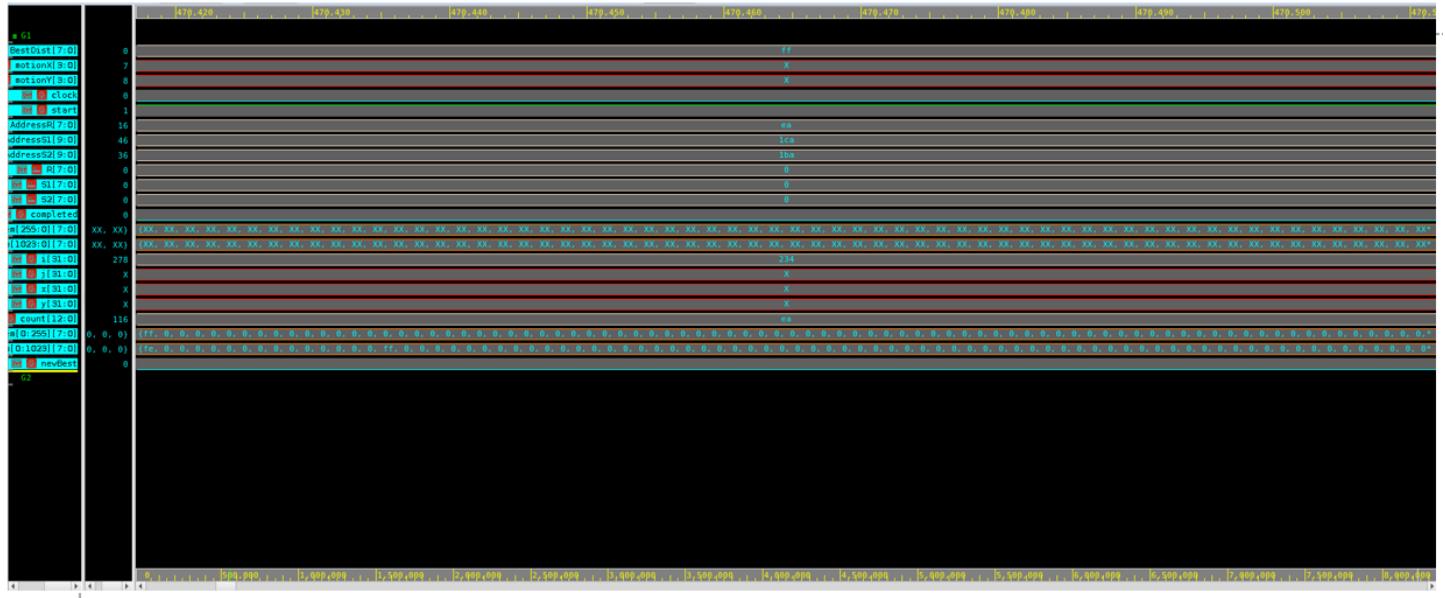
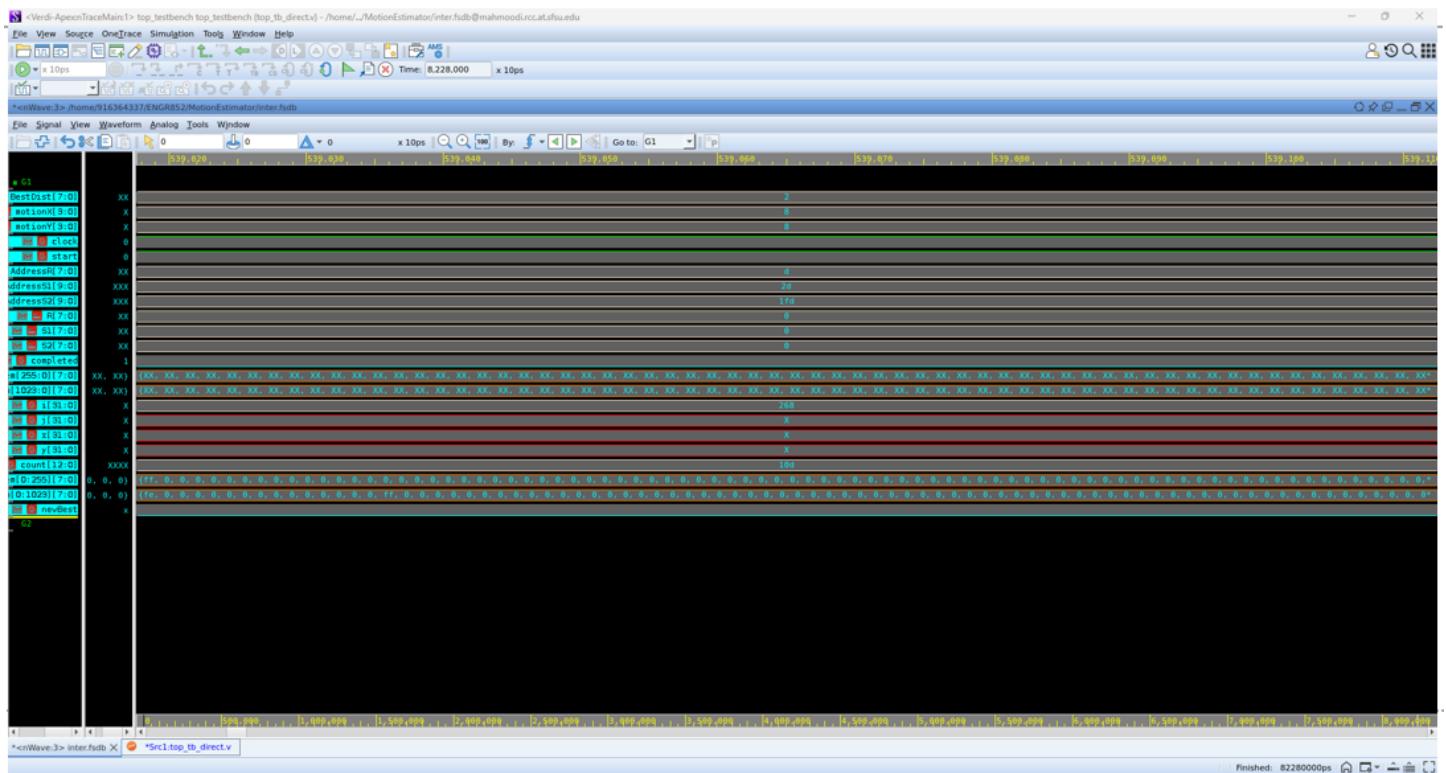


Figure 4 – Worst Case Waveform

Figure 4 shows the waveform during the first few clock cycles of the motion estimation process. At this early stage, we can see that the BestDist signal is equal to 8'hFF, or 255, which corresponds to the output message, “Not Found the frame in the Search Window.” This waveform represents the worst-case scenario during functional verification of the motion estimator.

b. Better case



*Figure 5 – Better Case Waveform*

In this Figure 5, BestDist equals 2 which is non zero number but not zero , indicating a partial match that is close to the reference image, which correctly triggers the “Almost Found the frame in the Search Window” message.

## c. Best Case

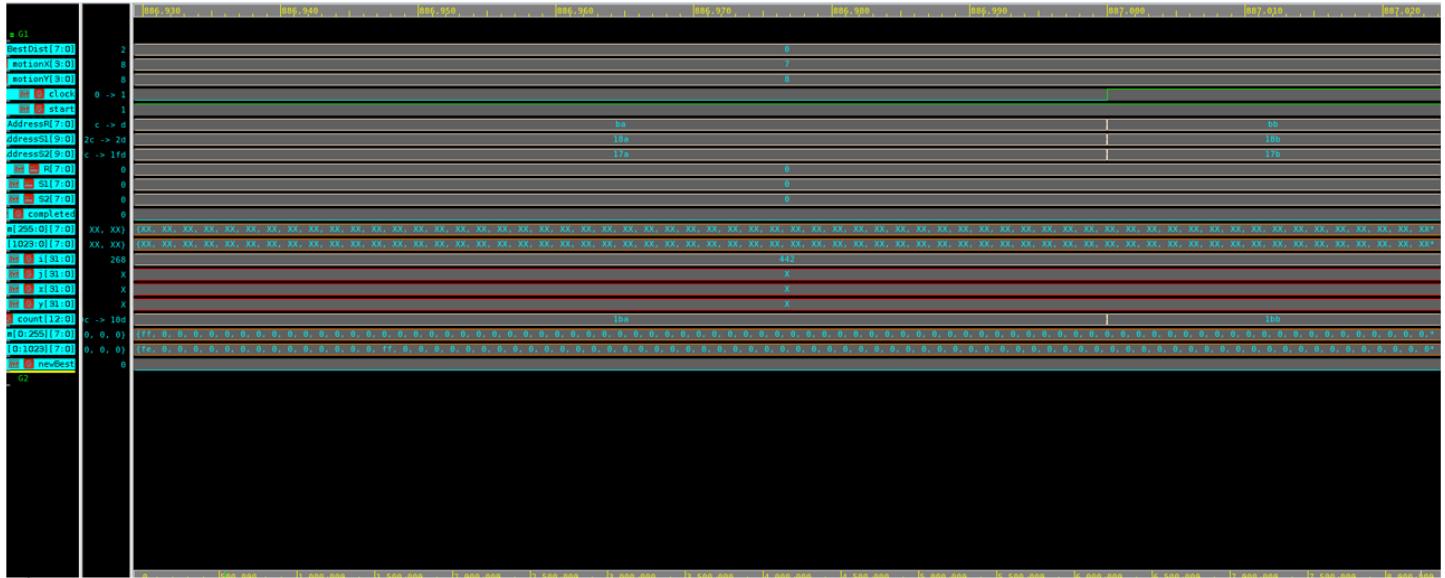


Figure 6 – Best Case Waveform

Figure 6 represents the best-case scenario in the functional verification of the motion estimator. In this case, the BestDist signal equals zero, meaning the value of BestDist is exactly 0. A BestDist value of zero indicates a perfect match between the  $16 \times 16$  reference block and a selected  $16 \times 16$  block within the search window.

## II. Synthesis

### A. Constraints

```
tmp38.sdc - Notepad
File Edit Format View Help
#####
# Created by write_sdc on Wed Dec 3 15:54:23 2025
#####
set sdc_version 2.1
set_units -time ns -resistance MOhm -capacitance fF -voltage V -current uA
create_clock [get_ports clock] -name ideal_clock1 -period 3.8 -waveform {0 1.9}
```

Figure 7 – sdc File

For synthesis, we use Synopsys Design Compiler within the dc\_shell-xg-t environment, which provides access to the technology libraries containing all standard cells and gates. We then import our RTL using the *analyze* and *elaborate* lines of code to build the structural representation of the design as seen in Figure 7. After applying a clock constraint with *create\_clock*, we evaluate the timing slack. Through this analysis, we determined that a 3.8-nanosecond clock period was appropriate for our design.

## B. Power and Area

Area Report

```
*synth_area.rpt - Notepad
File Edit Format View Help
*****
Report : area
Design : top
Version: 0-2018.06-SP4
Date   : Wed Dec 3 15:59:02 2025
*****
Number of ports: 2100
Number of nets: 3536
Number of cells: 1409
Number of combinational cells: 1080
Number of sequential cells: 277
Number of macros/black boxes: 0
Number of buf/inv: 383
Number of references: 5
Combinational area: 578.176789
Buf/Inv area: 86.446799
Noncombinational area: 311.110801
Macro/Black Box area: 0.000000
Net Interconnect area: 880.765264
Total cell area: 889.287590
Total area: 1770.052854
1
```

Power Report

```
*synth_power.rpt - Notepad
File Edit Format View Help
*****
Report : power
      -analysis_effort low
Design : top
Version: 0-2018.06-SP4
Date   : Wed Dec 3 16:00:33 2025
*****
Power Group          Internal Power    Switching Power    Leakage Power    Total Power  ( % ) Attrs
-----|-----|-----|-----|-----|-----|-----|
io_pad              0.0000            0.0000            0.0000            0.0000    ( 0.00%)
memory              0.0000            0.0000            0.0000            0.0000    ( 0.00%)
black_box           0.0000            0.0000            0.0000            0.0000    ( 0.00%)
clock_network       0.0000            0.0000            0.0000            0.0000    ( 0.00%)
register            72.8992           2.4409           1.7857e+08        253.9066    ( 25.66%)
sequential           0.0000            0.0000            0.0000            0.0000    ( 0.00%)
combinational        36.8496           14.7350           6.8419e+08        735.7791    ( 74.34%)
-----|-----|-----|-----|-----|-----|-----|
Total               109.7487 uW     17.1759 uW     8.6276e+08 pW    989.6857 uW
```

Figure 8 – Synthesized Power and Area Reports

The total cell area is 889.287590 nm<sup>2</sup> while the total area is 1770.052854 nm<sup>2</sup>. Total power consumed is 989.6857 μW. These values are satisfactory for our project but could be optimized a bit further.

### C. Post Synthesis Static Timing Analysis

**Set-Up**

```

*presetup.rpt - Notepad
File Edit Format View Help
-max_paths 1
-sort_by slack
Design : top
Version: 5-2021.06-SP4
Date : Wed Dec 10 12:40:59 2025
*
Point           Incr   Path
-----
clock ideal_clock1 (rise edge)      0.00  0.00
clock network delay (ideal)        0.00  0.00
ctl_u/count_reg[3]/CK (SAEDSLVT14_FDPHQ_1) 0.00  0.00 r
ctl_u/count_reg[3]/D (SAEDSLVT14_FDPHQ_1)  0.01  0.01 r
ctl_u/u76/X (SAEDHVT14_INV_0P5)       0.03  0.04 f
ctl_u/u64/X (SAEDHVT14_N02_CD0_0P5)    0.03  0.07 r
ctl_u/u63/X (SAEDHVT14_INV_0P5)       0.02  0.09 f
ctl_u/u56/X (SAEDHVT14_N02_CD0_0P5)    0.02  0.11 r
ctl_u/u55/X (SAEDHVT14_INV_0P5)       0.02  0.13 f
ctl_u/u53/X (SAEDHVT14_N02_CD0_0P5)    0.02  0.16 r
ctl_u/u52/X (SAEDHVT14_NR2_2)         0.03  0.19 f
ctl_u/u35/X (SAEDHVT14_AN2_U_0P5)     0.03  0.22 f
comp_u/u26/X (SAEDSLVT14_N02B_0P5)    0.02  0.24 f
comp_u/u20/X (SAEDSLVT14_A021B_0P5)   0.01  0.25 r
comp_u/u17/X (SAEDSLVT14_A021B_0P5)   0.01  0.27 r
comp_u/u36/X (SAEDSLVT14_A0121_0P5)   0.01  0.28 f
comp_u/u28/X (SAEDSLVT14_A021B_0P5)   0.01  0.29 r
comp_u/u27/X (SAEDSLVT14_A0221_0P5)   0.01  0.30 r
comp_u/u21/X (SAEDSLVT14_A021B_U_0P5) 0.02  0.32 r
comp_u/u64/X (SAEDSLVT14_A0221_2)     0.01  0.33 r
comp_u/u16/X (SAEDRVVT14_OR2B_PWV_2)  0.02  0.34 r
comp_u/u100/X (SAEDSLVT14_A021B_1)    0.01  0.36 r
comp_u/u148/X (SAEDSLVT14_A01221_0P5) 0.02  0.38 f
comp_u/u141/X (SAEDSLVT14_A0221_0P5)  0.01  0.39 f
comp_u/u143/X (SAEDSLVT14_A0221_U_0P5) 0.01  0.41 f
comp_u/u146/X (SAEDSLVT14_A01221_0P5) 0.02  0.43 r
comp_u/u139/X (SAEDSLVT14_A021_0P5)   0.01  0.44 r
comp_u/u150/X (SAEDSLVT14_A0121_0P5)  0.01  0.45 f
comp_u/u149/X (SAEDSLVT14_A021B_0P5)  0.01  0.46 r
comp_u/u37/X (SAEDSLVT14_A0121_0P5)   0.01  0.47 f
comp_u/u51/X (SAEDSLVT14_A01311_4)    0.02  0.49 r
comp_u/u132/X (SAEDSLVT14_INV_S_0P5)  0.03  0.52 f
comp_u/u24/X (SAEDHVT14_A0221_0P5)   0.04  0.55 f
comp_u/BestDist_reg[2]/D (SAEDHVT14_FDP_V2_0P5) 0.01  0.56 f
data arrival time                      0.56
data arrival time                      3.23
slack (MET)                           -0.56
-----
```

**Hold**

```

*prehold.rpt - Notepad
File Edit Format View Help
*****
Report : timing
-path_type full
-delay_type min
-max_paths 1
-sort_by slack
Design : top
Version: 5-2021.06-SP4
Date : Wed Dec 10 12:41:09 2025
*****
Startpoint: pe_u/pe0/Rpipe_Reg[2]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Endpoint: pe_u/pe1/Rpipe_Reg[2]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Path Group: ideal_clock1
Path Type: min

Point           Incr   Path
-----
clock ideal_clock1 (rise edge)      0.00  0.00
clock network delay (ideal)        0.00  0.00
pe_u/pe0/Rpipe_Reg[2]/CK (SAEDSLVT14_FDP_V2LP_0P5) 0.00  0.00 r
pe_u/pe0/Rpipe_Reg[2]/Q (SAEDSLVT14_FDP_V2LP_0P5)  0.01  0.01 r
pe_u/pe1/Rpipe_Reg[2]/D (SAEDHVT14_FDP_V2LP_1)    0.01  0.02 r
data arrival time                  0.02
-----
```

```

clock ideal_clock1 (rise edge)      0.00  0.00
clock network delay (ideal)        0.00  0.00
clock reconvergence pessimism    0.00  0.00
pe_u/pe1/Rpipe_Reg[2]/CK (SAEDHVT14_FDP_V2LP_1)  0.00  0.00 r
library hold time                 -0.00 -0.00
data required time                -0.00
-----
```

```

data required time                -0.00
data arrival time                 -0.02
-----
```

```

slack (MET)                       0.02
-----
```

Figure 9 – Set-Up and Hold Times

Figure 9 shows our post-synthesis static timing analysis (STA). Before entering physical design, we run PrimeTime to evaluate the timing quality of the netlist. We check both setup and hold times to identify early timing violations. The reports here summarizes the worst path's slack values, giving us a baseline before placement and routing.

Before entering the physical design stage, the first thing we do is run static timing analysis, or STA. We use PrimeTime, which is a timing verification tool that evaluates whether our design meets setup and hold requirements based on the synthesized netlist and constraints. At this stage, the design has no physical information yet—no placement, no routing, no parasitics. So this pre-layout STA gives us an initial understanding of the timing quality of our netlist.

We generate two main timing reports:

A setup timing report, which checks whether signals can arrive in time.

A hold timing report, which ensures signals do not arrive too early.

These reports help us confirm that the design is ready to move into physical implementation.

### III. Physical Design

#### A. Constraints

```
*top_post.sdc - Notepad
File Edit Format View Help
#####
#
# Design name: top
#
# Created by icc2 write_sdc on Wed Dec 10 12:57:12 2025
#
#####
|
# /home/924320342/Engr852/HW/tmp38.sdc, line 9
create_clock -name ideal_clock1 -period 3.8 -waveform {0 1.9} [get_ports \
{clock}]
set_propagated_clock [get_clocks {ideal_clock1}]
# Warning: Libcell power domain derates are skipped!

# Set latency for io paths.
# -origin user
set_clock_latency -min 0.0156593 [get_clocks {ideal_clock1}]
# -origin user
set_clock_latency -max 0.0157356 [get_clocks {ideal_clock1}]
# Set propagated on clock sources to avoid removing latency for IO paths.
set_propagated_clock [get_ports {clock}]
```

Figure 10 – Post Synthesis sdc File

Physical design follows several key stages: floorplanning, placement, clock tree synthesis, and routing. Figure 10 provides the timing constraints used throughout these steps. It links back to the 3.8-nanosecond clock period defined in our earlier timing design, ensuring that the entire physical implementation is built on consistent timing assumptions.

### Physical Design Stages

Once we finish pre-layout STA, we begin the actual physical design, using ICC2. This stage is where the netlist is transformed into a real physical chip layout.

We can divide the physical design process into several major steps:

#### (1) *Floorplanning*

We define the overall structure of the chip—its area, core boundary, and the locations of IO ports and power structures.

#### (2) *Placement*

Next, all the standard cells are placed onto the chip.

After placement, we run timing checks again, which give us updated setup and hold reports based on cell positions.

#### (3) *Clock Tree Synthesis (CTS)*

We then build the clock tree to ensure the clock signal arrives uniformly across the design.

After CTS, we evaluate the clock quality and check timing again.

#### (4) *Routing*

Finally, we route all the wires that connect the cells.

This includes global routing, detailed routing, and routing optimization.

After routing, we generate a complete set of reports—area, power, QoR, utilization, and updated timing reports. By the end of this stage, the design is fully placed and routed, but not yet analyzed with real parasitics.

## B. Power and Area

**Power Report**

```
*power.rpt - Notepad
File Edit Format View Help
*****
Report : power
-significant_digits 2
Design : top
Version: S-2021.06-SP5-1
Date   : Wed Dec 10 12:55:44 2025
*****
| -----
u - User defined power group
i - Includes clock pin internal power

Power Group      Internal Power      Switching Power      Leakage Power      Total Power      ( % )      Attrs
-----
io_pad           0.00e+00            0.00e+00            0.00e+00            0.00e+00            ( 0.0%)
memory          0.00e+00            0.00e+00            0.00e+00            0.00e+00            ( 0.0%)
black_box        0.00e+00            0.00e+00            0.00e+00            0.00e+00            ( 0.0%)
clock_network   6.45e+07            2.37e+07            4.31e+05            8.87e+07            ( 53.8%)      i
register         7.99e+06            5.39e+06            1.25e+07            2.59e+07            ( 15.7%)
sequential       0.00e+00            0.00e+00            0.00e+00            0.00e+00            ( 0.0%)
combinational   1.11e+07            2.31e+07            1.60e+07            5.02e+07            ( 30.5%)
-----
Total           8.36e+07 pW          5.22e+07 pW          2.89e+07 pW          1.65e+08 pW
1
```

**Area Report**

```
*area.rpt - Notepad
File Edit Format View Help
Design : top
Version: S-2021.06-SP5-1
Date   : Wed Dec 10 12:55:25 2025
*****
----- FLOORPLAN INFORMATION -----
----- CORE AND CHIP AREA INFORMATION -----
Core Area is : 1260.028
Chip Area is : 1451.372
-----
----- SITE ROW INFORMATION -----
Site Name      Width     Height    Total Rows    Total Tiles    Area
-----
unit          0.07      0.60       59           28379        1260.028
-----
```

Figure 11 – Post Synthesis Power and Area Reports

Figure 11 show the physical design power and area reports. Compared with the synthesis results, the metrics changed because placement, routing, and real parasitics were introduced. These reports provide a more accurate view of the chip's final power consumption and physical area. Core area: 1260.028 nm<sup>2</sup>, Chip Area: 1451 nm<sup>2</sup>, Total Power: 1.65e8 pW. These values are suspiciously lower than the pre-synthesis values which is abnormal. We chalked it up to error in the estimation process in the pre synthesis reports.

### C. Board

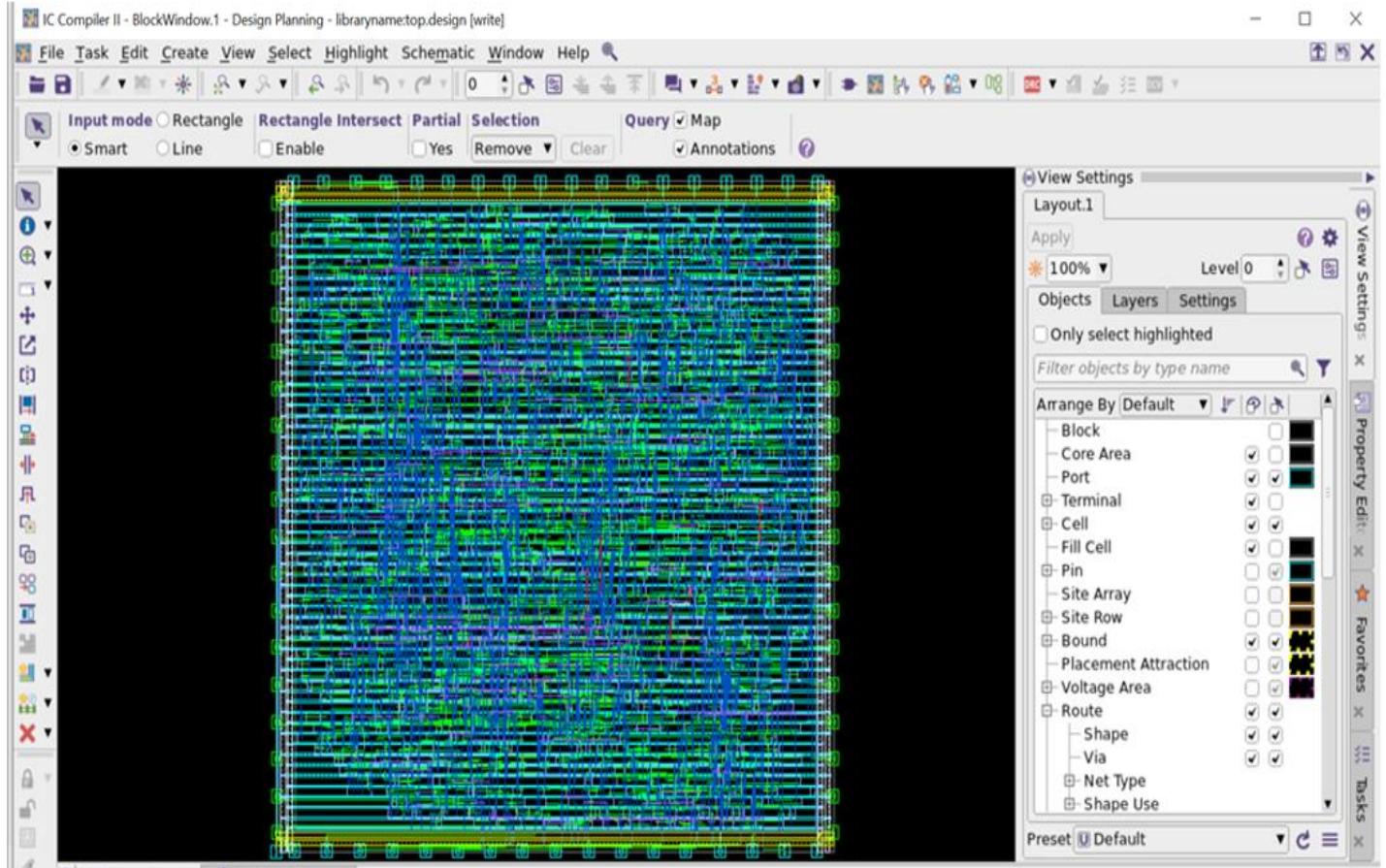


Figure 12 – Physical Design Board

Figure 12 shows the ICC2 layout board, displaying the placed cells and routed interconnects that form the final physical chip structure of our design.

## D. Output

The screenshot shows two Notepad windows side-by-side. The left window is titled "top\_parasitic.tlup\_max\_125.spef - Notepad" and the right window is titled "top\_parasitic.tlup\_min\_125.spef - Notepad". Both windows display SPEF (SPICE Parameter Exchange Format) files. The right window also contains a portion of a Verilog netlist labeled "Netlist(top\_post.v)".

```

// max common. mite_verilog_top_module_start //outputs top_post.v
7 module top ( clock , start , BestDist , motionX , motionY , AddressR ,
8   AddressS1 , AddressS2 , R , S1 , S2 , completed );
9   input clock ;
10  input start ;
11  output [7:0] Bestdist ;
12  output [3:0] motionX ;
13  output [3:0] motionY ;
14  output [7:0] AddressR ;
15  output [9:0] AddressS1 ;
16  output [9:0] AddressS2 ;
17  input [7:0] R ;
18  input [7:0] S1 ;
19  input [7:0] S2 ;
20  output completed ;
21
22  wire [15:1] S1S2MUX ;
23  wire [15:0] newDist ;
24  wire [15:0] PReady ;
25  wire [3:0] VectorX ;
26  wire [3:0] VectorY ;
27  wire [127:0] Accumulate ;
28
29  control ctl_u (.clock ( clock ) , .start ( start ) ,
30   .S1S2MUX ( [ S1S2MUX[15] , S1S2MUX[14] , S1S2MUX[13] , S1S2MUX[12] ,
31   | S1S2MUX[11] , S1S2MUX[10] , S1S2MUX[9] , S1S2MUX[8] , S1S2MUX[7] ,
32   | S1S2MUX[6] , S1S2MUX[5] , S1S2MUX[4] , S1S2MUX[3] , S1S2MUX[2] ,
33   | S1S2MUX[1] , SYNOPSYS_UNCONNECTED 1 ) ) ,
34   .newDist ( newDist ) , .PCompStart ( CompStart ) , .PReady ( PReady ) ,
35   .VectorX ( VectorX ) , .VectorY ( VectorY ) , .AddressR ( AddressR ) ,
36   .AddressS1 ( { AddressS1[9] , AddressS1[8] , AddressS1[2] , AddressS1[6] ,
37   | AddressS1[5] , SYNOPSYS_UNCONNECTED 2 , AddressS1[3] , AddressS1[2] ,
38   | AddressS1[1] } ) );

```

Figure 13 – Physical Design Output

Figure 13 shows the physical design outputs. After routing, we extract detailed RC parasitics for both max and min corners, capturing real wire resistance and capacitance. After extracting these parasitics, ICC2 generates a post-layout netlist and updated SDC. This netlist reflects the actual routed design and is used for final timing analysis in PrimeTime.

Once routing is complete, we extract the actual parasitic effects from the layout.

These parasitics—such as real wire capacitance and resistance—have a direct impact on timing. We generate an extracted version of the design, including: a post-layout netlist, a post-layout SDC, and a SPEF file containing parasitic information. This extracted data allows us to run a much more accurate timing analysis.

## IV. Sign Off

```
*setup_max.rpt - Notepad
File Edit Format View Help
Date : Wed Dec 10 13:03:06 2025
*****
Startpoint: ctl_u/count_reg[3]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Endpoint: comp_u/motiony_reg[8]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Last common pin: clock
Path Group: ideal_clock1
Path Type: max

Point           Incr      Path
-----
clock ideal_clock1 (rise edge)    0.00    0.00
clock network delay (propagated)  0.00    0.00
ctl_u/count_reg[3]/CK (SAEDRV14_FSDPQ_V2LP_0P5) 0.00    0.00 r
ctl_u/count_reg[3]/Q (SAEDRV14_FSDPQ_V2LP_0P5)   0.07 & 0.07 r
ctl_u/U76/X (SAEDRV14_INV_S_0P5)   0.05 & 0.13 f
ctl_u/U64/X (SAEDRV14_INV_M_0P5)   0.05 & 0.18 r
ctl_u/U63/X (SAEDRV14_INV_S_0P5)   0.03 & 0.20 f
ctl_u/U62/X (SAEDRV14_INV_M_0P5)   0.04 & 0.24 r
ctl_u/U56/X (SAEDRV14_INV_S_0P5)   0.02 & 0.26 f
ctl_u/U54/X (SAEDRV14_AN3_0P5)    0.03 & 0.29 f
ctl_u/U50/X (SAEDRV14_AN2_M_0P5)  0.02 & 0.31 f
comp_u/U12/X (SAEDRV14_NR2_1)     0.02 & 0.33 r
comp_u/U26/X (SAEDRV14_NR2B_U_0P5) 0.04 & 0.38 f
comp_u/U20/X (SAEDRV14_A021B_0P5) 0.02 & 0.39 r
comp_u/U17/X (SAEDRV14_A0221_0P5) 0.02 & 0.41 r
comp_u/U36/X (SAEDRV14_A0121_0P5) 0.01 & 0.42 f
comp_u/U28/X (SAEDRV14_A021B_0P5) 0.01 & 0.43 r
comp_u/U27/X (SAEDRV14_A0221_0P5) 0.02 & 0.45 r
comp_u/U21/X (SAEDRV14_A021B_1)   0.02 & 0.47 r
comp_u/U64/X (SAEDRV14_A0221_0P5) 0.02 & 0.48 r
comp_u/U16/X (SAEDRV14_NR2B_PSEC0_1) 0.01 & 0.49 r
comp_u/ctmTdsLR_1_14/ (SAEDRV14_A022_0P5) 0.01 & 0.51 r
comp_u/ctmTdsLR_2_30_roptpi_2638/X (SAEDRV14_A022_U_0P5) 0.01 & 0.52 r
comp_u/ctmTdsLR_2_2616/X (SAEDRV14_A021_U_0P5) 0.01 & 0.53 r
comp_u/ctmTdsLR_2_60/ (SAEDRV14_A0221_0P5) 0.01 & 0.54 r
comp_u/ctmTdsLR_1_2615/X (SAEDRV14_A021_U_0P5) 0.01 & 0.55 r
comp_u/ctmTdsLR_1_66/ (SAEDRV14_A021B_0P5) 0.01 & 0.56 f
comp_u/U150/X (SAEDRV14_A021_0P5)   0.01 & 0.56 r
comp_u/U37/X (SAEDRV14_A0121_0P5)   0.01 & 0.57 f
comp_u/U151/X (SAEDRV14_A0331_0P5)  0.06 & 0.63 r
comp_u/U132/X (SAEDRV14_INV_M_0P5)  0.07 & 0.70 f
comp_u/U8/X (SAEDRV14_A022_0P75)   0.02 & 0.72 f
comp_u/motiony_reg[8]/D (SAEDRV14_FDP_V2LP_1) 0.00 & 0.72 f
data arrival time                   0.72

clock ideal_clock1 (rise edge)    3.80    3.80
clock network delay (propagated)  0.01    3.81
clock reconvergence pessimism   0.00    3.81
comp_u/motiony_reg[8]/CK (SAEDRV14_FDP_V2LP_1) 3.81 r
library setup time                -0.01   3.81
data required time                3.81
```

## Post Layout Timing Analysis & Sign-Off

### Timing Report

```
*hold_min.rpt - Notepad
File Edit Format View Help
Date : Wed Dec 10 13:05:06 2025
*****
Startpoint: pe_u/pe12/Rpipe_Reg[7]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Endpoint: pe_u/pe13/Rpipe_Reg[7]
(rising edge-triggered flip-flop clocked by ideal_clock1)
Last common pin: clock
Path Group: ideal_clock1
Path Type: min

Point           Incr      Path
-----
clock ideal_clock1 (rise edge)    0.00    0.00
clock network delay (propagated)  0.00    0.00
pe_u/pe12/Rpipe_Reg[7]/CK (SAEDRV14_FDP_V2LP_0P5) 0.00    0.00 r
pe_u/pe12/Rpipe_Reg[7]/Q (SAEDRV14_FDP_V2LP_0P5)   0.02 & 0.02 f
pe_u/pe13/Rpipe_Reg[7]/D (SAEDRV14_FDP_V2LP_0P5)  0.00 & 0.02 f
data arrival time                   0.02

clock ideal_clock1 (rise edge)    0.00    0.00
clock network delay (propagated)  0.01    0.01
clock reconvergence pessimism   0.00    0.01
pe_u/pe12/Rpipe_Reg[7]/CK (SAEDRV14_FDP_V2LP_0P5) 0.01 r
library hold time                0.00    0.02
data required time                0.02

data required time                0.02
data arrival time                 -0.02

slack (MET)                      0.01
```

Figure 14 – Post Layout Timing Analysis & Sign-Off

Using the extracted parasitics from the SPEF files together with the post-layout netlist, we perform final setup and hold analysis in PrimeTime. This post-layout STA reflects actual routed delays, including wire resistance and capacitance. The reports shown here in Figure 14 represent our sign-off results, confirming timing closure for both the MAX corner (setup) and the MIN corner (hold).

### Post-layout STA

After combining the layout parasitics with the netlist and constraints, we perform post-layout timing signoff in PrimeTime.

Here we analyze two corners:

(1) The MAX delay corner

Used for checking setup timing, ensuring signals can travel fast enough across long wires with parasitic loading.

(2) The MIN delay corner

Used for hold timing, ensuring signals do not arrive too early due to short paths.

At the end of this stage, we produce two final signoff reports:

A post-layout setup timing report

A post-layout hold timing report

These reports represent the true timing behavior of the chip and determine whether the design is ready for tape-out.

**Discussion:** Overall, this project showed that a block-matching motion estimator can be implemented as a relatively small, timing-closed ASIC using an industry-standard 32/28 nm flow, while still exposing several non-idealities and open questions. Functionally, the VCS simulations with worst, better, and best cases confirmed that the controller, PE array, and comparator interact correctly: the design cleanly distinguishes between “no match” ( $\text{BestDist} = 255$ ), “almost found” ( $\text{BestDist} = 2$ ), and a perfect match ( $\text{BestDist} = 0$ ), which gives strong evidence that the search window is being scanned as intended and that the distortion metric is implemented correctly.

From a physical perspective, synthesis and pre-layout STA showed that the design can meet a 3.8 ns clock period in the chosen technology, which is slightly more aggressive than the nominal 260 MHz target and suggests that the 16-PE pipelined architecture is fast enough to support the required frame rate, assuming reasonable control overhead. Post-layout results in ICC2 and PrimeTime further confirmed that setup and hold constraints are satisfied even after routing and parasitic extraction, indicating that there are no hidden critical paths that only appear once wiring is modeled.

On the other hand, the discrepancy between pre-synthesis and post-layout power estimates—where post-layout power appears unrealistically lower—highlights limitations in our power analysis flow, such as simplified switching activity assumptions and possible mis-configuration of reporting units, and would need to be resolved in a real tape-out scenario. Finally, while the achieved area and power numbers are acceptable for the course project, they were not aggressively optimized; future work could explore alternative search strategies, deeper pipelining, or clock-gating to trade off area, power, and throughput more systematically, as well as expand the testbench (more image patterns and corner cases) to increase coverage and further validate the robustness of the motion estimator architecture.

**Conclusion:** By following the ASIC Design Flow, we were able to successfully implement a Motion Estimator capable of drastically reducing video compression time using the Block-Matching Algorithm. The RTL Code we had designed showed successful results using both Waveform Simulation and FPGA Emulation. By the end of this project, we had designed a fully operational low power/area chip capable of performing the task we were assigned.

To summarize:

We start with pre-layout STA to verify the netlist. Then we go through the physical design process—floorplan, placement, CTS, and routing—evaluating timing along the way. After routing, we extract real parasitics from the layout. Finally, we run post-layout STA, which incorporates those parasitics and generates our final timing signoff. This complete flow ensures that the design meets timing requirements under realistic conditions before it can move to manufacturing.