Computer-Aided VLSI System Design
Final Report

Due Tuesday, May 23, 14:00

Team Number: 44

Team Member: B08505039 林楷崴 / B08505049 劉旻鑫

Algorithm

根據參考論文，我們可以根據 s1~s4 建出 4 層 level 的 tree，s4 在最上層，s1 在最下層，且每經過一個 node，我們只須計算對應的 partial Euclidean distance (PED) $d_i(x_i)$ (在解調過程是使用 s 來代表 x 所以求的是 $d_i(s_i)$)，並取有最小的 PED 的 node 繼續往下 traverse，我們目標在 traverse 完到 tree 的 leaf 時能找到最短距離，也就是 Formula. 1 的最小值。

$$\sum_{i=1}^{4} \left| \left[\hat{y}_i - \sum_{j=i}^{4} R_{ij} s_j \right] \right|^2$$

Formula. 1

Partial Euclidean Distance (PED) $d_i(s_i)$的算法如下：
最上層 s4 固定為，

$$d_4 = |[\hat{y}_4 - R_{44} S_4]|^2$$

往下每一層d_i為每層的e_i加上上一層的 d 值(d_{i+1})：

$$d_i = d_{i+1} + |e_i|^2 \; for \; i = 3,2,1$$

Formula. 2

$$|e_i|^2 = \left| \left[\hat{y}_i - \sum_{j=i}^{4} R_{ij} s_j \right] \right|^2 for \; i = 3,2,1$$

Formula. 3

從算式累加可得與 Formula. 1 一樣的結果，故可以得出最小值去計算 LLR。

Tree Search

Search Tree 的方法有 DFS 與 BFS，另外過程中會訂定一個 boundry 值，當距離(PED)超過就會直接 prune tree，以下簡單講述各法對應硬體實作的可能：

Depth-First Search Algorithm：

DFS 會搭配 backtracking，一開始便會直接 visit 到 leaf，便 backtrack visit 新的 node 直到 full explore。DFS 只需要固定 memory 大小，由於 Last-in First out 的特性，size 會固定。此外也能動態更新 boundry 值加速 tree search。然而當有較多的資源能拿來做 computation 時，此法就不能做 parallel。

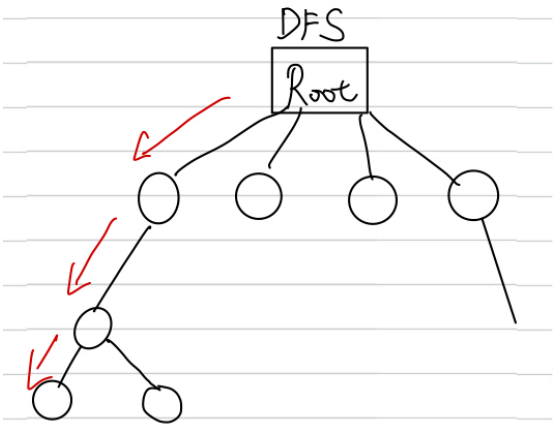


Figure. 1

Breadth-First Search Algorithm：

BFS 探索整個 tree level by level，同個 level 的 node 必須都被 visit 過才能到下一層。此法很適合結合 parallel implementation 因為同個 level 的 node 彼此是獨立的(不會同時成立)。相反地，此法的缺點就是 memory overhead。另外此法也不能隨時更新 boundry 值，因為最後才會 visit 到 leaf 的 level。

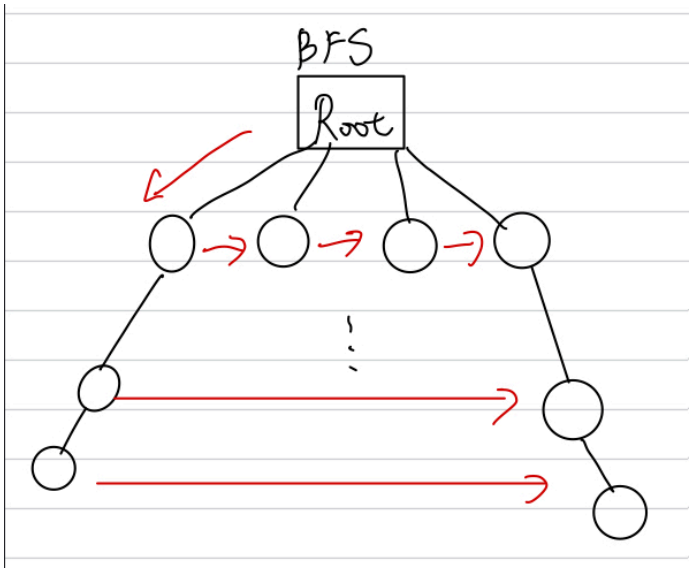


Figure. 2

也有其他的 tree search 方式但都是基於上述的 DFS 或 BFS，使得能更快找到最小值或能更加利用所有硬體資源。相關演算法的複雜度會跟 signal 數量(對應到 tree 的 level)與偏移調變的方式(此採 QPSK 故有四種)有指數相關。

LLR Implementation

我們在使用 Matlab 進行 Full Search 的模擬時有發現，計算的 16 組 min 中(見 Figure. 3)，分別 8 列的最小的 min 會是一樣的，也就是所有 computation 中的最小值，因此我們只要透過演算法得到最小的 min，並藉 traverse tree 時紀錄的 s 座標，即可知道該 min 要替換的是每列中左邊的 min 還右邊的 min。

我們預設每一組的 min 一開始都為最大值(=8'h7f)，若最後得到的 x1~x4 座標皆為(0,0)，便把計算出來的 min 代到每一列中右邊的 min，這樣計算出的 LLR 皆為正，故 hard bit 皆為 0。

Formula

- LLR for $x_{1,1}$: $L(x_{1,1}|\underline{y}) = \min_{\underline{x}\in X_{1,1,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{1,1,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{1,2}$: $L(x_{1,2}|\underline{y}) = \min_{\underline{x}\in X_{1,2,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{1,2,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{2,1}$: $L(x_{2,1}|\underline{y}) = \min_{\underline{x}\in X_{2,1,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{2,1,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{2,2}$: $L(x_{2,2}|\underline{y}) = \min_{\underline{x}\in X_{2,2,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{2,2,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{3,1}$: $L(x_{3,1}|\underline{y}) = \min_{\underline{x}\in X_{3,1,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{3,1,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{3,2}$: $L(x_{3,2}|\underline{y}) = \min_{\underline{x}\in X_{3,2,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{3,2,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{4,1}$: $L(x_{4,1}|\underline{y}) = \min_{\underline{x}\in X_{4,1,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{4,1,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$
- LLR for $x_{4,2}$: $L(x_{4,2}|\underline{y}) = \min_{\underline{x}\in X_{4,2,1}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2 - \min_{\underline{x}\in X_{4,2,0}} \sum_{i=1}^{4}|[\hat{y}_i - \sum_{j=i}^{4} R_{ij}s_j]|^2$

Figure. 3

Algorithm Implementation

本組 ML 解調器第一次採用[3,1,1,1]的方式 explore 本次由 signal 的規定建出來的 tree ([3,1,1,1]代表第一層(s4)取三個 min PED，第二層(s3)、第三層(s2)、第四層(s1)各取一個 min PED 依序往下 traverse)，依照此法只須計算 40 個 nodes 即能找到的最小的 PED，即找到 s4~s1 分別的座標。

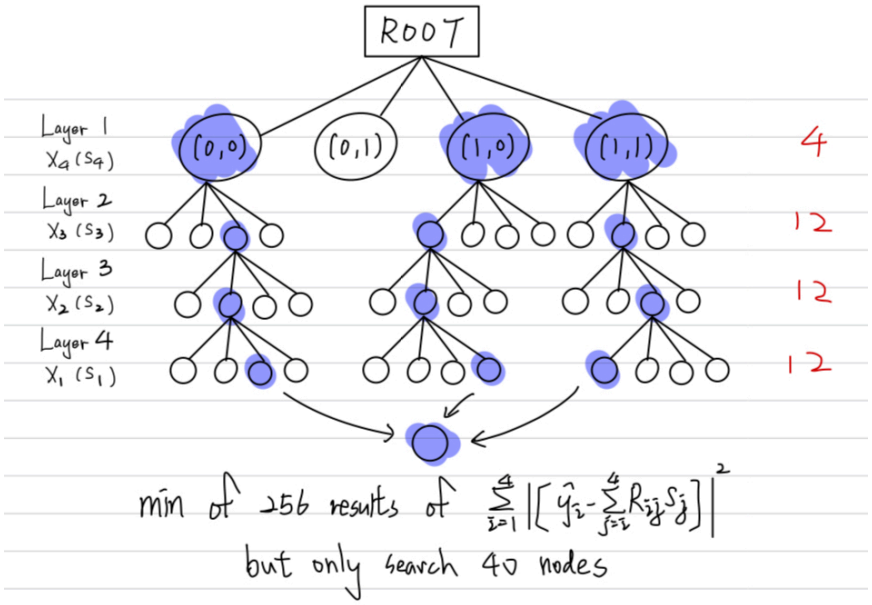


Figure. 4

不過測試結果不符合規定 error rate，故我們針對每一層取的數量去做優化，以能達到我們所需要的 error rate。最後我們 search tree 的方法如下圖：

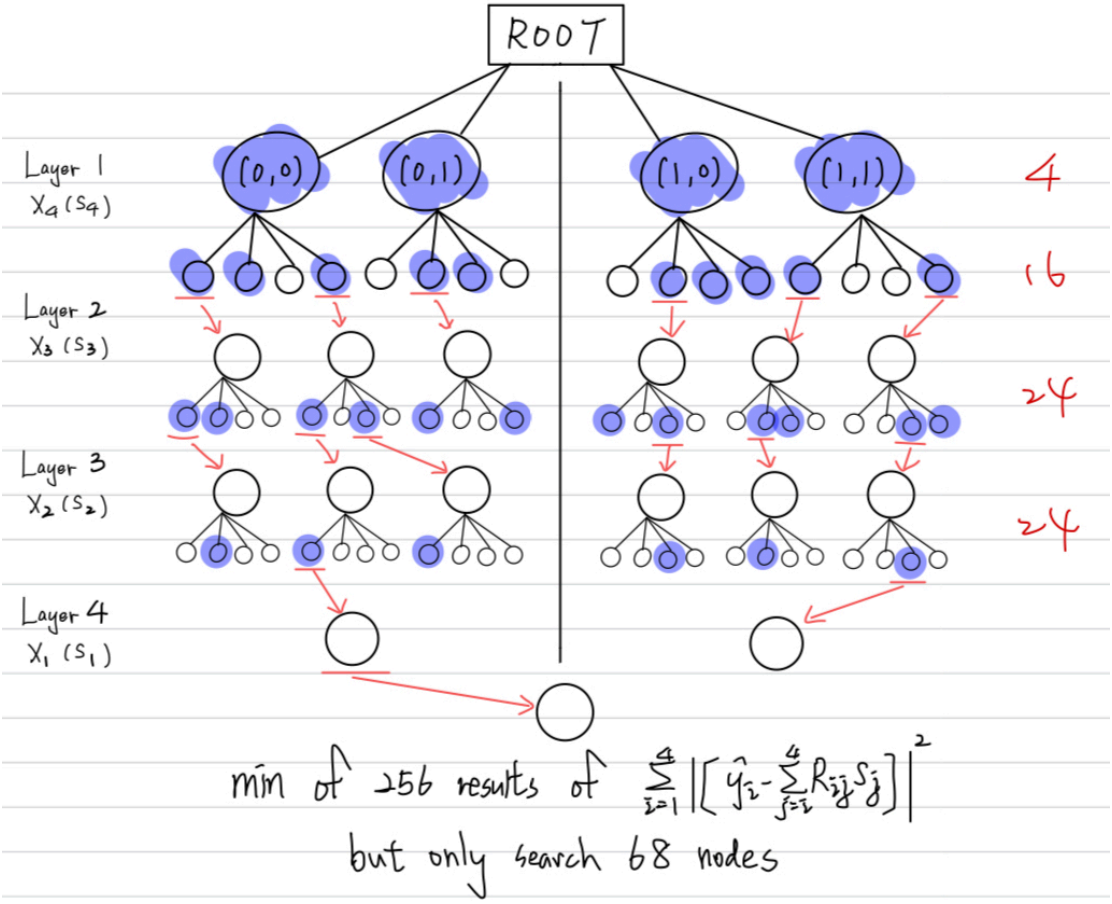


Figure. 5

由於此更動會使 cycle 數超過 64，故我們搭配 parallel，將 tree 拆成兩邊同步進行運算。s4 四個座標都需要保留向下 traverse(COMPUTE_S4)，第一組 s3 計算的 PED 取三個最小(COMPUTE_S3_1ST -> COMPARE_4to2)，第二組 s3 計算的 PED 取二個最小(COMPUTE_S3_2ND -> COMPARE_4to2)，五個當中再取最小三個往下 traverse(COMPARE)。三組 s2 計算分別取兩個最小(COMPUTE_S2_1ST -> COMPARE_4to2 -> COMPUTE_S2_2ND -> COMPARE_4to2 -> COMPUTE_S3_3RD -> COMPARE_4to2)，最後六個再取三個最小往下 traverse(COMPARE)。最後三組 s1 計算分別取一個最小(COMPUTE_S1_1ST -> COMPARE_4to1 -> COMPUTE_S1_2ND -> COMPARE_4to1 -> COMPUTE_S1_3RD -> COMPARE_4to1)，再三個取一個最小(COMPARE_3to1)，最後跟 parallel 的另一個結果取最小並計算 LLR(LLR)，即能得到符合 error rate 的 min。(括弧代表我們實作的 state)。此更動使得我們要找的 node 數增至 68 個。

PED Computation Method

我們在計算 PED 時採取 parallel 與 pipeline 的方式，分四個 cycle 完成，根據 Formula. 3，我們在第一個 cycle 先判斷 S 的正負，並將對應正負的預先計算好的 RS 值存至暫存器(我們將 R 值各自向右 shift 1、向右 shift 3、向右 shift 4 的值相加來近似 R 乘根號二分之一)，第二個 cycle 再一併用 y 值減去上一步的 RS 值，第三個 cycle 計算出平方，第四個 cycle 相加結果。

由於本處牽扯多項計算，此處使用 pipeline 也能適當縮短 critical path。

```
COMPUTE_s4:
begin
    case(s_choose)
        2'b00:
        begin
            i_r_01_w = i_r_re_44_pos;
            i_r_02_w = i_r_re_44_neg;
        end
        2'b01:
        begin
            i_r_01_w = i_y_hat_re_4 - i_r_01;
            i_r_02_w = i_y_hat_im_4 - i_r_01;
            i_r_03_w = i_y_hat_re_4 - i_r_02;
            i_r_04_w = i_y_hat_im_4 - i_r_02;
        end
        2'b10:
        begin
            mul_temp_01_w = trunc(i_r_01 * i_r_01);
            mul_temp_02_w = trunc(i_r_02 * i_r_02);
            mul_temp_03_w = trunc(i_r_03 * i_r_03);
            mul_temp_04_w = trunc(i_r_04 * i_r_04);
        end
        2'b11:
        begin
            min_PED_1st_w = mul_temp_01 + mul_temp_02;
            min_PED_2nd_w = mul_temp_01 + mul_temp_04;
            min_PED_1st_p_w = mul_temp_03 + mul_temp_02;
            min_PED_2nd_p_w = mul_temp_03 + mul_temp_04;
        end
    endcase
end
```

Hardware Optimization

由於本次接收的 data 位元眾多，若能適當只取用一定數量的 bits 數即可滿足 error rate 則能減少不少硬體使用量。透過 trial and error，我們發現套用我們的做法，一開始的 data 接收只需用 4 bits 整數部分與 7 bits 小數部分。另外由於有乘法的使用，我們在乘法結束也能適當 truncate 一些不必要的位元，透過 trial and error，我們能減少因為乘法造成的 22 bits 結果(11 bits * 11 bits)至 11 位，分別 truncate 整數 2 位與小數 9 位。

```
// BitLength
parameter   INT = 4;
parameter   FRAC = 7;
parameter   INT_TRUNC = 2;
parameter   FRAC_TRUNC = 9;
parameter   LENGTH = INT + FRAC;
parameter   TRUNCATION = INT_TRUNC + FRAC_TRUNC;
parameter   LENGTH_MUL = 2*LENGTH;
```

Hardware Scheduling

下圖展示我們本設計遵循的 state machine，各 state 進行的操作在上頁演算法的實作已說明，共使用 50 個 cycle 來找到我們要的 min 並計算 LLR。我們將計算部分(COMPUTE)寫在 Combinential，賦值與比較(COMPARE)寫在 Sequential。

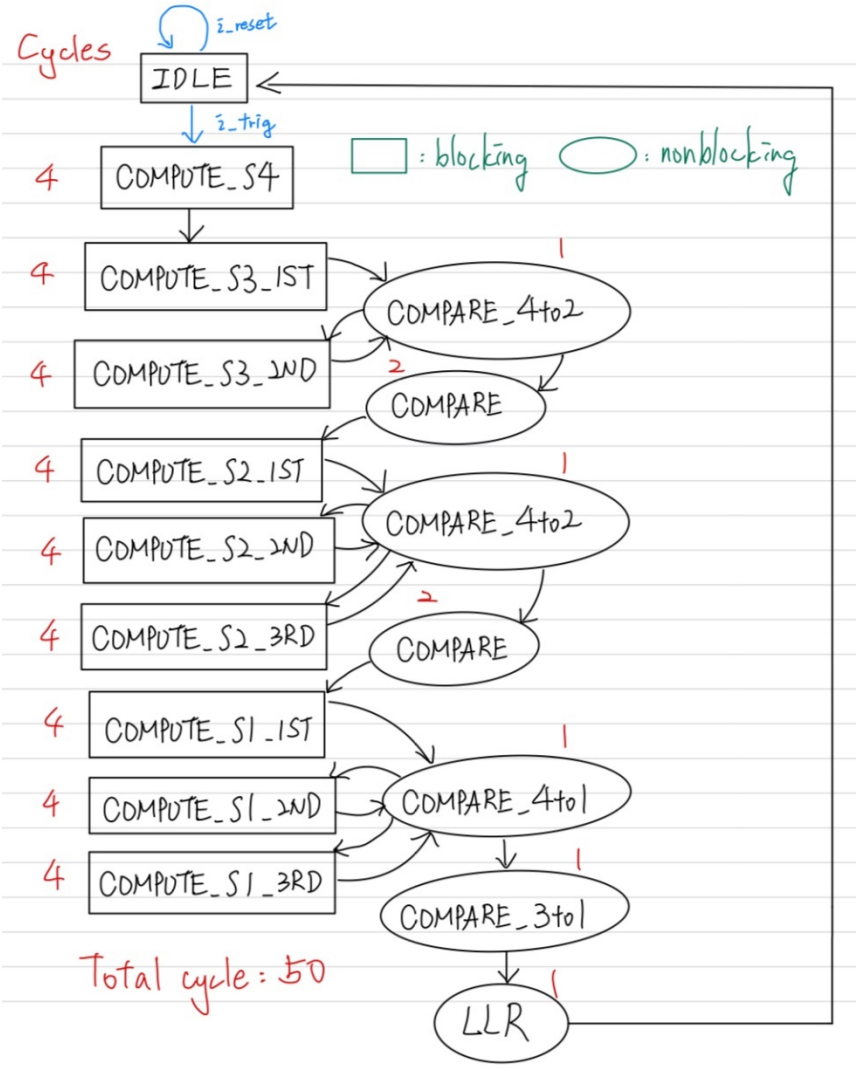


Figure. 6

Performance & APR Results

GSIM cycle: 12.8 (ns)

GSIM latency: 824991(ns) (64452 cycles)

POST-SIM cycle: 12.8 (ns)

POST-SIM latency: 824991(ns) (64452 cycles)

Post layout area: 275332.43 (um^2)

Post layout PrimeTime active power: 2.414 (mW)

# of DRC violations: 0

Status of LVS check: pass

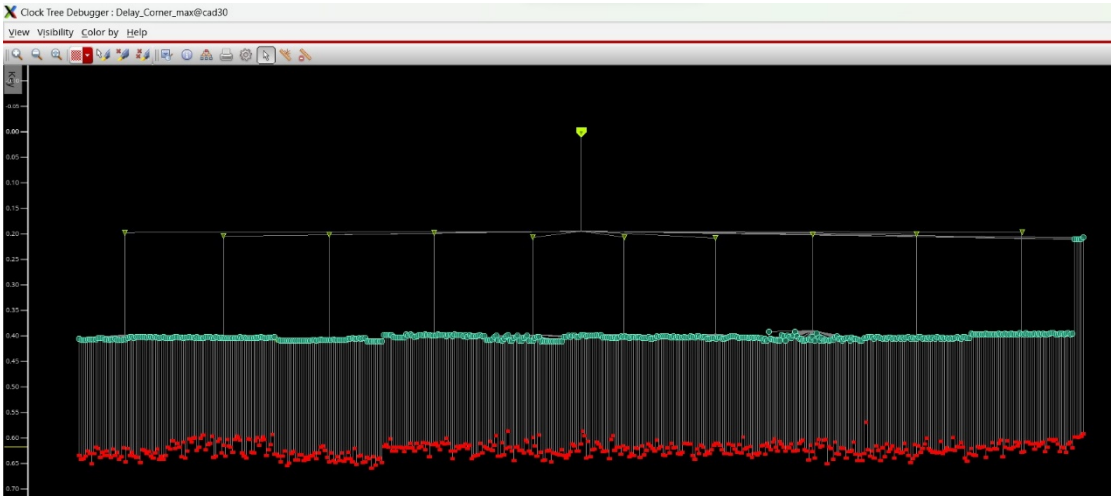
Design Stage	Description	Value
P&R	Number of DRC violations (ex: 0) (Verify -> Verify Geometry…)	0
	Number of LVS violations (ex: 0) (Verify -> Verify Connectivity…)	0
	Die Area (um^2)	275332.43
	Core Area (um^2)	250001.56
Post-layout Simulation	Clock Period for Post-layout Simulation (ex. 10ns)	12.8

Work Distribution

	B08505039 林楷崴	B08505049 劉旻鑫
Content	RTL (Algorithm, FSM)	RTL (I/O, TB)
	Report	APR
	Code Debugging and Optimization, Gate Simulation	
Percentage	50%	50%

Screen Shot

1. CCOpt Clock Tree Debugger result



2. DRC and LVS checking after routing.

```
innovus 20> verify_drc
 *** Starting Verify DRC (MEM: 1817.8) ***

  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area: {0.000 0.000 176.800 176.800} 1 of 9
  VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {176.800 0.000 353.600 176.800} 2 of 9
  VERIFY DRC ..... Sub-Area : 2 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {353.600 0.000 526.700 176.800} 3 of 9
  VERIFY DRC ..... Sub-Area : 3 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {0.000 176.800 176.800 353.600} 4 of 9
  VERIFY DRC ..... Sub-Area : 4 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {176.800 176.800 353.600 353.600} 5 of 9
  VERIFY DRC ..... Sub-Area : 5 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {353.600 176.800 526.700 353.600} 6 of 9
  VERIFY DRC ..... Sub-Area : 6 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {0.000 353.600 176.800 522.750} 7 of 9
  VERIFY DRC ..... Sub-Area : 7 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {176.800 353.600 353.600 522.750} 8 of 9
  VERIFY DRC ..... Sub-Area : 8 complete 0 Viols.
  VERIFY DRC ..... Sub-Area: {353.600 353.600 526.700 522.750} 9 of 9
  VERIFY DRC ..... Sub-Area : 9 complete 0 Viols.

  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:04.3  ELAPSED TIME: 4.00  MEM: 45.2M) ***
```

```
innovus 21> VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Mon Jun  5 15:54:00 2023

Design Name: ml_demodulator
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (526.7000, 522.7500)
Error Limit = 1000; Warning Limit = 50
Check all nets
**** 15:54:00 **** Processed 5000 nets.
**** 15:54:00 **** Processed 10000 nets.
**** 15:54:01 **** Processed 15000 nets.

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Mon Jun  5 15:54:01 2023
Time Elapsed: 0:00:01.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:01.2  MEM: 21.625M)
```

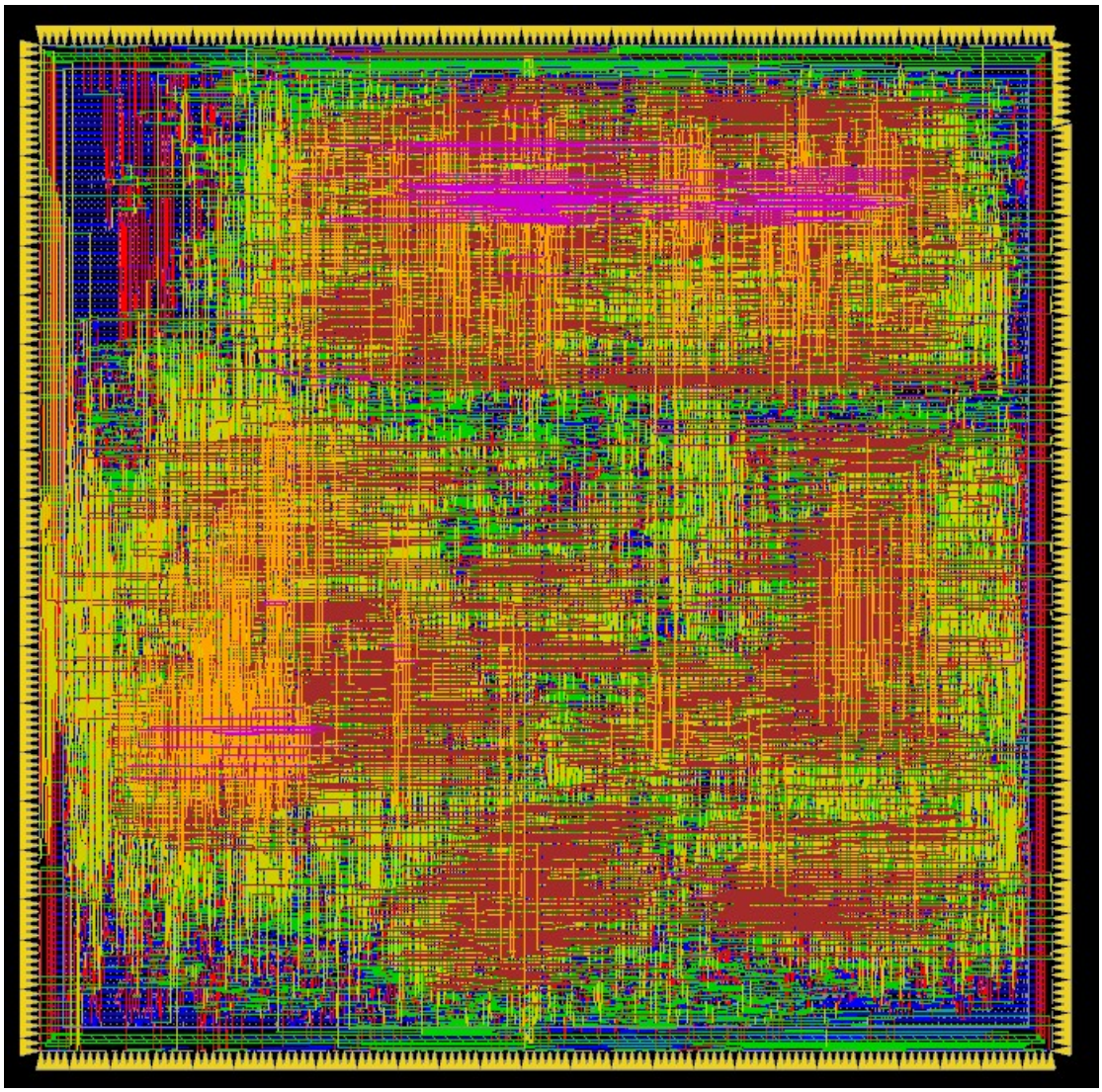
3. Timing report for **setup time and hold time** with no timing violation (post-route)

```
------------------------------------------------------------
     optDesign Final SI Timing Summary
------------------------------------------------------------

Setup views included:
 av_func_mode_max
Hold  views included:
 av_func_mode_max

+--------------------+---------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg |reg2cgate| in2reg  | reg2out | in2out  | default |
+--------------------+---------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  3.933  |  3.933  |  7.111  | 10.464  |  7.662  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |    0    |    0    |   N/A   |    0    |
|          All Paths:|  4314   |  1641   |   428   |  2239   |   10    |   N/A   |    0    |
+--------------------+---------+---------+---------+---------+---------+---------+---------+

+--------------------+---------+---------+---------+---------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg |reg2cgate| in2reg  | reg2out | in2out  | default |
+--------------------+---------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):|  0.998  |  1.133  |  1.967  |  0.998  |  2.341  |   N/A   |  0.000  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |    0    |    0    |   N/A   |    0    |
|          All Paths:|  4314   |  1641   |   428   |  2239   |   10    |   N/A   |    0    |
+--------------------+---------+---------+---------+---------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +-------------------+-----------+------------------|
|                |  Nr nets(terms)   | Worst Vio |  Nr nets(terms)  |
+----------------+-------------------+-----------+------------------+
|   max_cap      |      0 (0)        |   0.000   |      0 (0)       |
|   max_tran     |      0 (0)        |   0.000   |      0 (0)       |
|   max_fanout   |      0 (0)        |     0     |      0 (0)       |
|   max_length   |      0 (0)        |     0     |      0 (0)       |
+----------------+-------------------+-----------+------------------+

Density: 70.160%
Total number of glitch violations: 0
------------------------------------------------------------
```

4. Final area result

```
innovus 27> analyzeFloorplan
**WARN: (IMPAFPU-9006): Command 'analyzeFloorplan' is obsolete. Please use commands 'placeDesign + trialR
 + load_timing_debug_report -proto' to analyze congestion and timing for the floorplan.
Start to collect the design information.
Build netlist information for Cell ml_demodulator.
Finished collecting the design information.
Average module density = 1.000.
Density for the design = 1.000.
       = stdcell_area 147285 sites (250002 um^2) / alloc_area 147285 sites (250002 um^2).
Pin Density = 0.3493.
            = total # of pins 51441 / total area 147285.
************************ Analyze Floorplan *************************
    Die Area(um^2)             : 275332.43
    Core Area(um^2)            : 250001.56
    Chip Density (Counting Std Cells and MACROs and IOs): 90.800%
    Core Density (Counting Std Cells and MACROs): 100.000%
    Average utilization        : 100.000%
    Number of instance(s)      : 24342
    Number of Macro(s)         : 0
    Number of IO Pin(s)        : 494
    Number of Power Domain(s) : 0
************************ Estimation Results ************************
********************************************************************
```

5. Final layout after adding core filler



Reference

1. Parallel High Throughput Soft-Output Sphere Decoder
   https://www.public.asu.edu/~chaitali/confpapers/qi_sips10.pdf
2. Complexity Assesment of Sphere Decoding Methods for MIMO Detection
   https://www.cel.kit.edu/download/articulo_isspit09.pdf
3. A Novel VLSI Architecture of Fixed-Complexity Sphere Decoder
   https://arxiv.org/ftp/arxiv/papers/1006/1006.4030.pdf
4. A Fixed-Complexity MIMO Detector Based on the Compex Sphere Decoder
   https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4153876
5. Reduced-Complexity Sphere Decoding Algorithm Based on Adaptive Radius in Each Dimension
   https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7387587
6. 基於軟式解調輸出球面解碼法之效能改進研究
   https://ir.nctu.edu.tw/bitstream/11536/44544/1/352601.pdf