BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 468

VLSI Laboratory

Final Project Report
UART Design and Synthesis

Section: G2 Group: 06

Course Instructors:

Mr. Hamidur Rahman

Md. Rasel Mia

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Md. Ayenul Azim** **Student ID: 1806089**	**Signature: _____** **Full Name: Shahriar Khan** **Student ID: 1806092**
Signature: _____ **Full Name: Saif Ahmed Sunny** **Student ID: 1806097**	**Signature: _____** **Full Name: Shahriar Ahmed** **Student ID: 1806192**

Signature: _____

Full Name: Md. Sazzad Hossan

Student ID: 1506084

1 Abstract

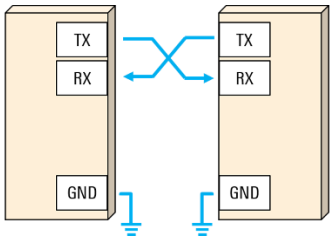
Universal Asynchronous Receiver/Transmitter (UART) technology serves as a cornerstone in the domain of digital communication, offering a strong framework for asynchronous serial data exchange between electronic devices. This abstract discusses the significance of UART in current technology, revealing its underlying concepts, operational procedures, and applications. By deconstructing UART's role in allowing serial communication, this abstract digs into crucial features such as data framing, baud rate tuning, and error detection. Furthermore, it illustrates the versatility and widespread acceptance of UART across varied industries, from embedded systems and microcontrollers to consumer electronics and industrial automation. As newer communication protocols arise, UART's lasting significance underlines its stability and simplicity, making it a sturdy component in the ever-evolving world of digital communication systems.

2 Introduction

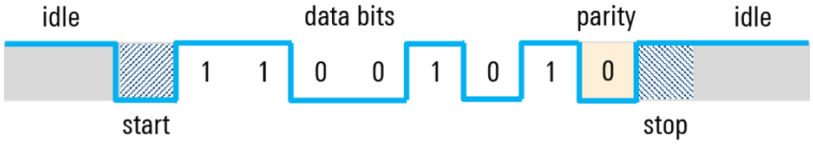
UART, or Universal Asynchronous Receiver/Transmitter, facilitates asynchronous serial communication between devices. It operates by sending data in packets containing a start bit, customizable data bits (typically 8), an optional parity bit for error checking, and one or more stop bits. Unlike synchronous protocols, UART doesn't require a shared clock signal but depends on a mutually agreed baud rate for transmission speed. Data is structured in bytes with a start bit, data bits, optional parity, and stop bits. UART is widely used in embedded systems and electronic devices for short-range communication due to its simplicity and versatility.

2.1Theory

UART stands for universal asynchronous receiver / transmitter and establishes a protocol, or set of rules, for transmitting serial data between two devices. UART is very simple and simply utilizes two wires between transmitter and receiver to transmit and receive in both directions. Both ends also have a ground connection. Communication in UART can be simplex (data is transferred in one direction only), half-duplex (both side talks but only one at a time), or full-duplex (both sides can transmit concurrently). Data in UART is transmitted in the form of frames.



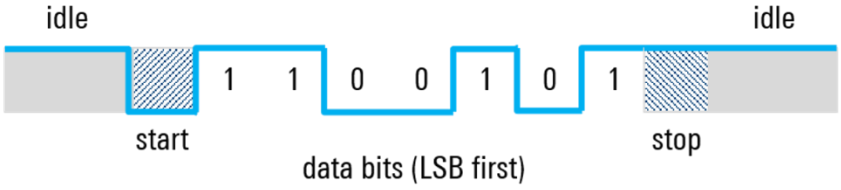
UART stands for universal asynchronous receiver/transmitter and establishes a protocol, or set of rules, for transferring serial data between two devices. UART is very simple and simply utilizes two wires between the transmitter and receiver to transmit and receive in both directions. Both ends also have a ground connection. Communication in UART can be simple (data is transferred in one direction only), half-duplex (both sides talk but only one at a time), or full-duplex (both sides can transmit concurrently). Data in UART is transmitted in the form of frames.



In UART protocol, high voltage indicates "1" and low voltage represents "0". While no exact voltage levels are stated, high is frequently labeled "mark" and low is termed "space". During idle situations, the line remains high, helpful in detecting line damage or transmitter difficulties.

In UART communication, the start bit announces the beginning of data transmission by transitioning from a high to a low state, followed by the data bits. Once the data bits are delivered, the stop bit signifies the end of the transmission, either by transitioning back to a high state or remaining high for an additional bit period. Optionally, a second stop bit can be specified to enable the receiver time to prepare for the following frame, though this is rarely utilized in practice.

The data bits are the user data or “useful” bits and occur immediately after the start bit. There can be 5 to 9 user data bits, however 7 or 8 bits is most typical. These data bits are normally delivered with the least significant bit first.



UART (Universal Asynchronous Receiver/Transmitter) is necessary for various purposes.

1. **Serial Communication**: Many devices communicate utilizing serial communication protocols due to their simplicity and efficiency, especially for short-distance communication. UART provides a standardized approach to implement serial communication between devices.

2. **Interfacing**: UART enables devices with different data formats, voltages, and speeds to communicate with one another. This versatility is crucial in several applications, such as attaching sensors, actuators, displays, and other peripherals to microcontrollers and other digital systems.

3. **Simplicity:** UART is comparatively straightforward to construct compared to other communication protocols like SPI (Serial Peripheral Interface) or I2C (Inter-Integrated Circuit). It has low hardware and software overhead, making it suited for resource-constrained embedded devices.

4. **Versatility:** UART can be utilized in many applications, from basic point-to-point communication between two devices to more complicated networks including several devices. It's often found in consumer electronics, industrial automation, automobile systems, and telecommunications equipment.

5. **Asynchronous Communication**: The asynchronous nature of UART allows devices to communicate with one another without the requirement for a shared clock signal, making it beneficial for applications where perfect synchronization is not required or if devices operate at various clock speeds.

6. **Cost-Effectiveness:** UART is a cost-effective alternative for establishing serial communication, particularly in mass-produced electronic products. Its simplicity and ubiquitous availability make it a popular choice for firms aiming to cut development expenses.

Overall, UART is vital for facilitating communication between diverse digital devices in various applications, delivering simplicity, versatility, and cost-effectiveness.

3 Methodology

3.1 Problem Formulation (PO(b))

3.1.1 Identification of Scope

UART synthesis involves translating behavioral descriptions from HDLs like Verilog into a structural representation, optimizing for area, timing, and power. Place-and-route (PNR) maps the synthesized design onto the physical layout of the IC, optimizing wire length and connectivity paths while meeting design constraints. A tool like Cadence Innovus is used for synthesis and PNR, respectively. Designers consider technology libraries, constraints, and performance specifications for successful implementation. The process aims to create an efficient and functional UART design suitable for fabrication.

3.2 Design Method (PO(a))

RTL synthesis for UART:

1. Convert behavioral description into RTL representation.

2. Optimize design for area, timing, and power.

3. Perform technology mapping to map logical elements to specific cells from the target technology library.

PNR (Place and Route) for UART:

1. Define floorplan to allocate space for UART and other functional blocks.

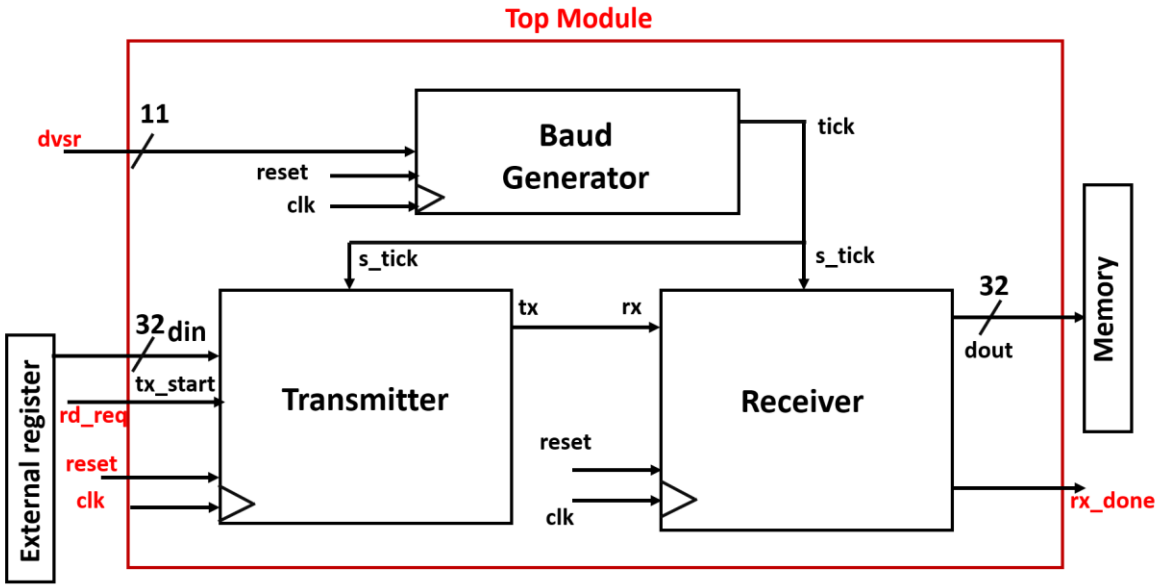
2. Place the synthesized RTL design onto the chip's layout, optimizing placement to minimize wire length.

3. Route interconnections between components while considering timing, power, and signal integrity.

4. Conduct physical verification checks, including DRC and LVS, to ensure compliance with design rules and functional correctness.

3.3Circuit Diagram

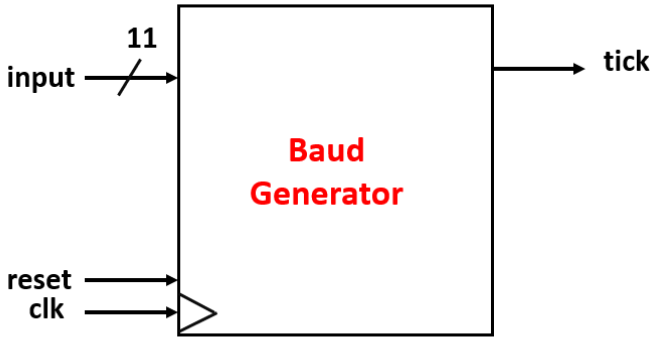
3.3.1 Top Module



Interaction with outside:
Input: 11 bit dvsr, 32 bit din from an external register, reset, clk and a rd_req

Output: rx_done, 32 bit dout to memory

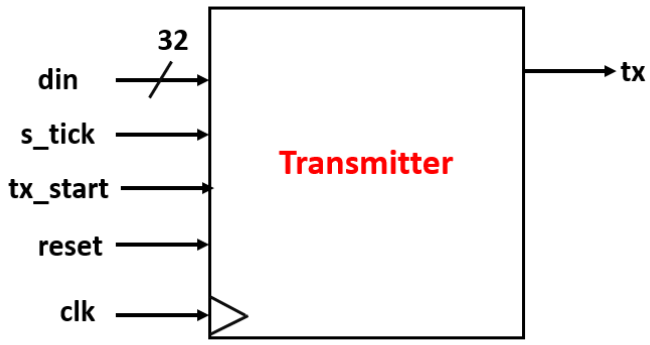
3.3.2 Baud Generator



Input: 11 bit from dvsr, reset, clk

Output: tick to Transmitter and Reciever

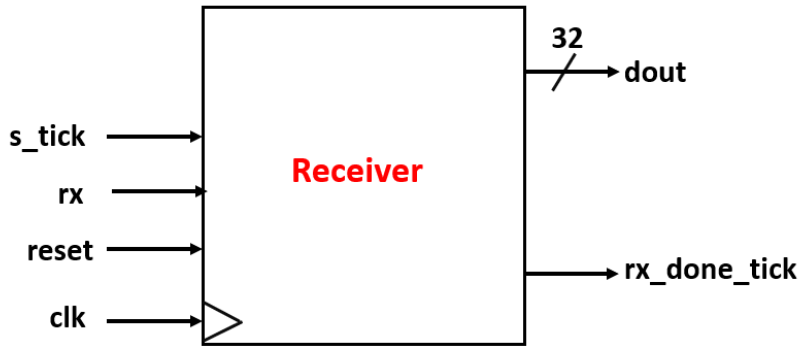
3.3.3 Transmitter



Input: 32 bit din from external register, s_tick from baud generator, reset, clk, tx_start from an external reas request rd_req

Output: tx to Reciever

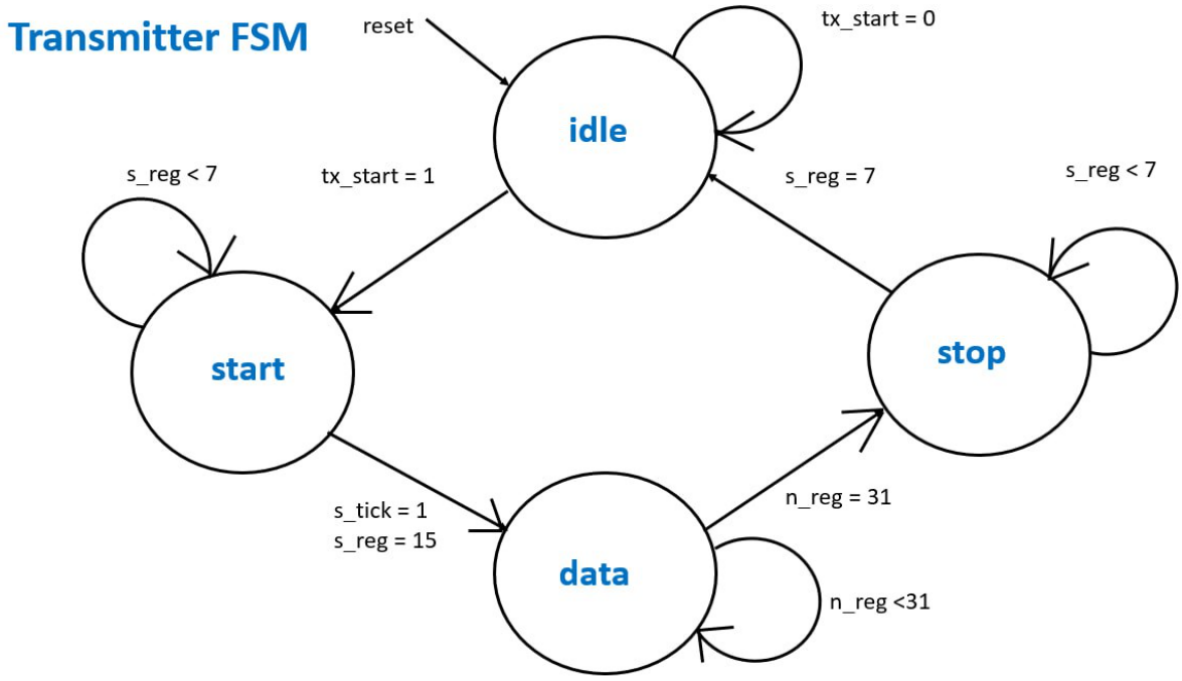
3.3.4 Receiver

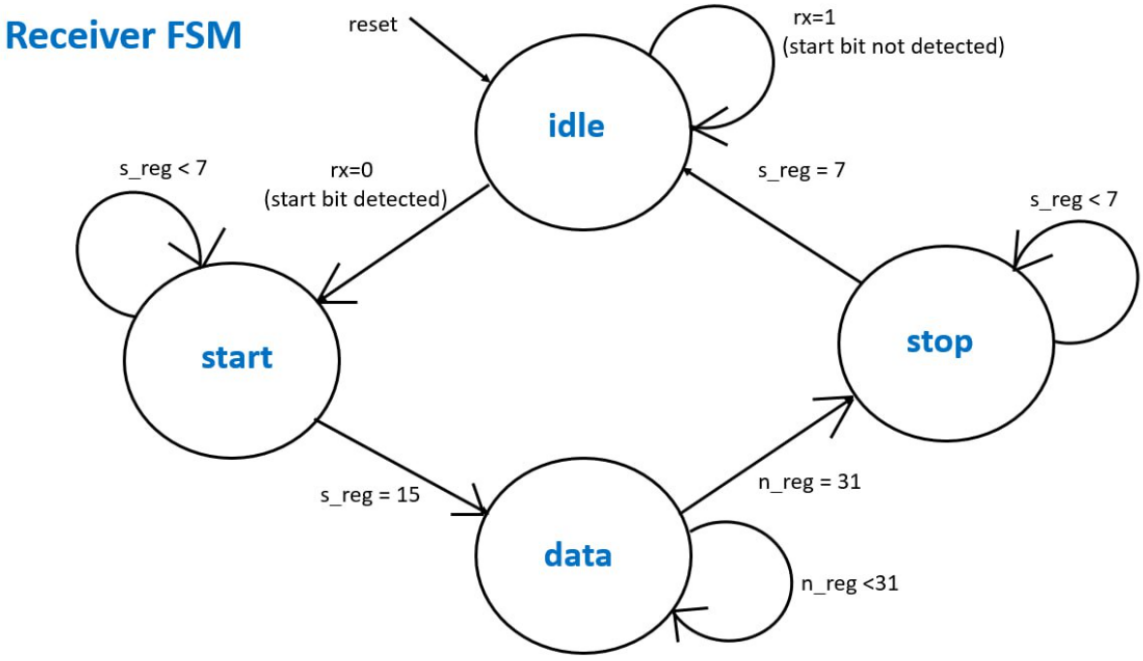


Input: s_tick from baud generator, reset, clk, rx from Transmitter tx.

Output: 32 bit dout to external register, rx_done_tick to outside.

3.4 FSM





4 Design Analysis and Evaluation

4.1 Novelty

Recent advancements in UART technology include the development of high-speed implementations for faster data transfer rates, low-power designs catering to energy-efficient applications, integration with emerging communication standards like Bluetooth Low Energy and USB, enhanced error detection and correction mechanisms for improved reliability, implementation of security features such as encryption and authentication, and integration into System-on-Chip platforms for streamlined system design. These improvements highlight the continual evolution and versatility of UART technology to satisfy the demands of modern communication systems across varied applications.

4.2 Design Considerations (PO(c), (PO(f)), (PO(f)))

4.2.1 Assessment of Safety Issues

Assessing the safety issues of UART communication entails evaluating potential risks associated with its operation and implementation, particularly in safety-critical systems. This assessment encompasses ensuring electrical safety standards are met to prevent hazards like electrical shock and short circuits, implementing robust error detection and correction mechanisms to maintain data integrity, and addressing susceptibility to electromagnetic interference. Functional safety considerations involve analyzing the impact of UART communication on overall system safety and implementing appropriate measures to mitigate risks. Additionally, safeguarding against security threats such as data breaches and unauthorized access is essential, along with considering environmental factors that may affect performance. By proactively identifying and addressing these safety concerns, developers can ensure the secure and reliable operation of UART communication in various applications.

4.2.2 Assessment of Legal/Ethical Issues

The assessment of legal and ethical concerns around UART technology involves verifying compliance with rules like as data protection laws and intellectual property rights, as well as evaluating ethical aspects such as privacy, transparency, and equitable access. This includes addressing potential hazards connected to surveillance, data management practices, and societal impact, while promoting responsible development and use of UART technology to safeguard individual rights and societal well-being.

External Register input through

```
reg [31:0] inputmem [10:0];
integer i;
$readmemh("externalregister.txt", inputmem);
```

```
[2024-03-06 06:44:58 UTC] iveri
data input: 00000000

After_receive:
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
data input: 00002197

After_receive:
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00002197
```

```
data input: 94018193

After_receive:
00000000
00000000
00000000
00000000
00000000
00000000
94018193
00002197
data input: 00000117

After_receive:
00000000
00000000
00000000
00000000
00000000
00000117
94018193
00002197
data input: 7f810113

After_receive:
00000000
00000000
00000000
00000000
7f810113
00000117
94018193
00002197
```

```
data input: 00010433

After_receive:
00000000
00000000
00000000
00010433
7f810113
00000117
94018193
00002197
```

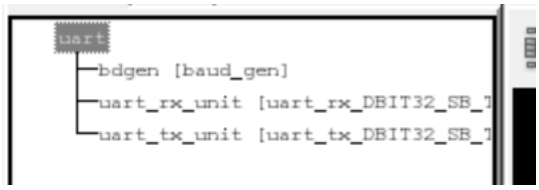
```
data input: 00010433

After_receive:
00000000
00000000
00010433
00010433
7f810113
00000117
94018193
00002197
```

```
data input: 0c80006f

After_receive:
00000000
0c80006f
00010433
00010433
7f810113
00000117
94018193
00002197
```

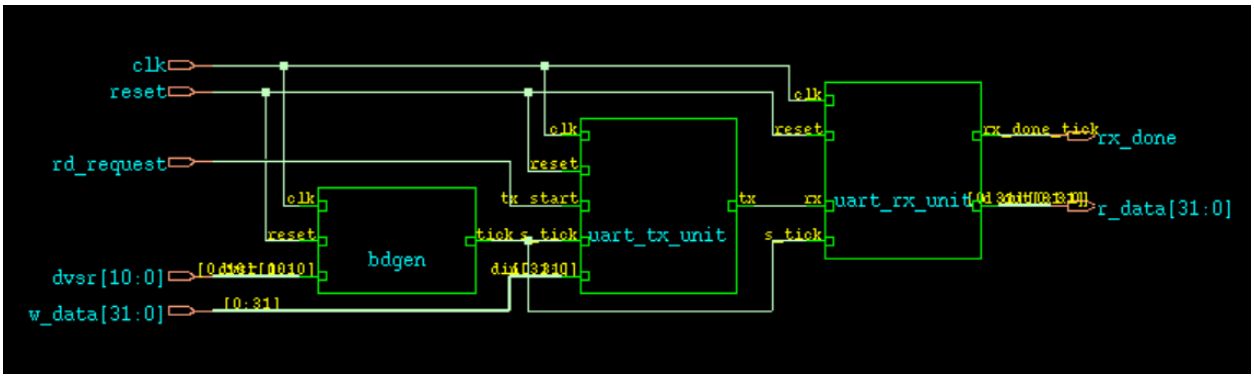
```
data input: 0c80006f

After_receive:
0c80006f
0c80006f
00010433
00010433
7f810113
00000117
94018193
00002197
```

x

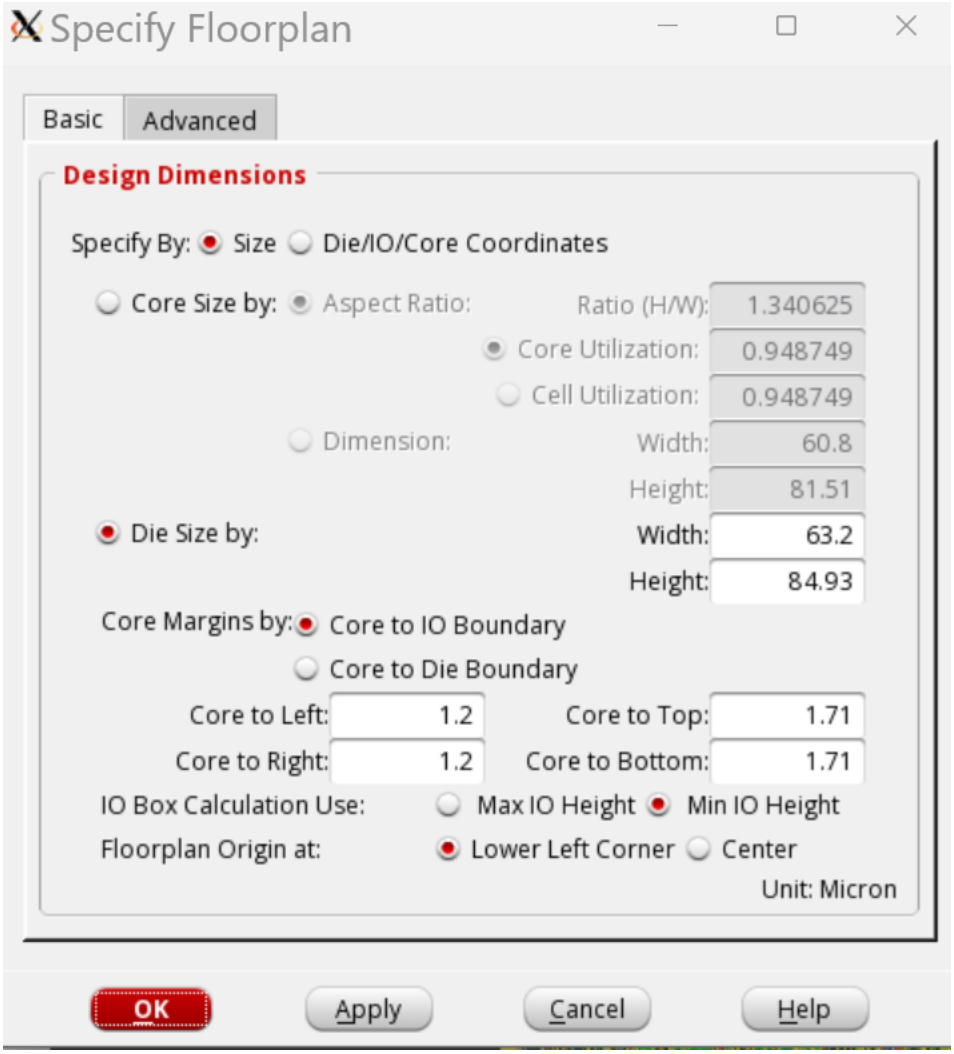
UART Hierarchy



UART top module

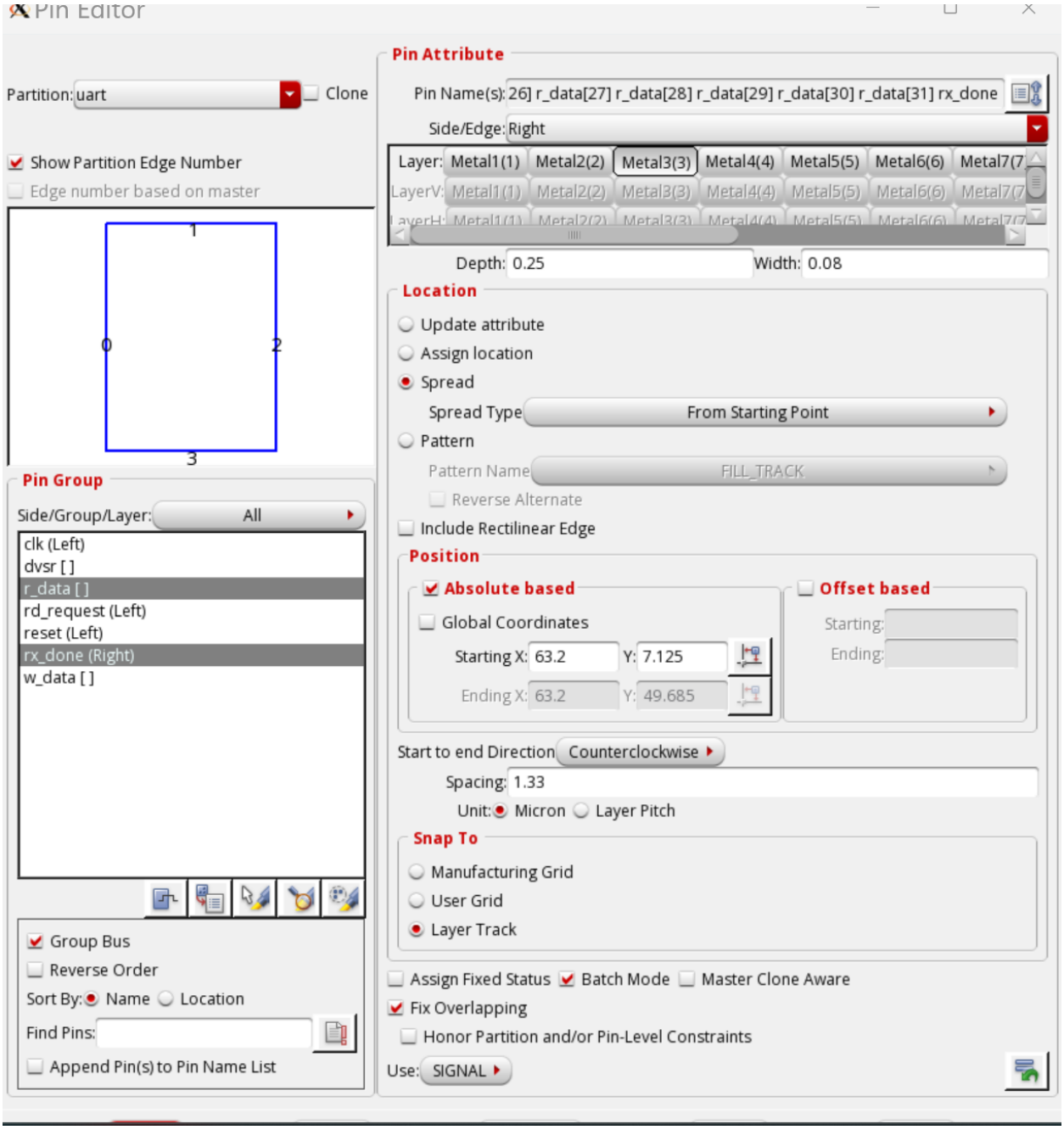


Specify Floorplan

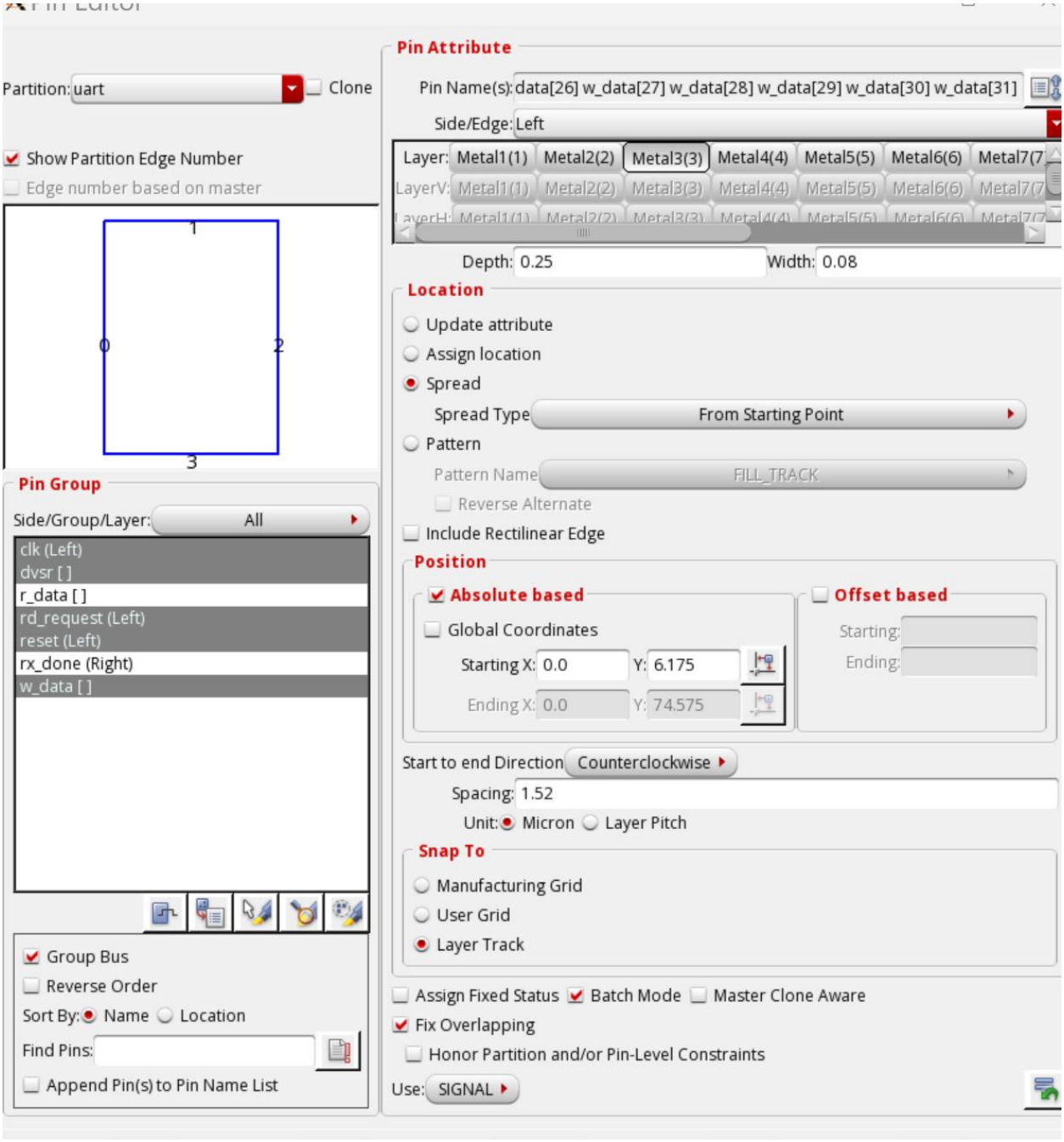


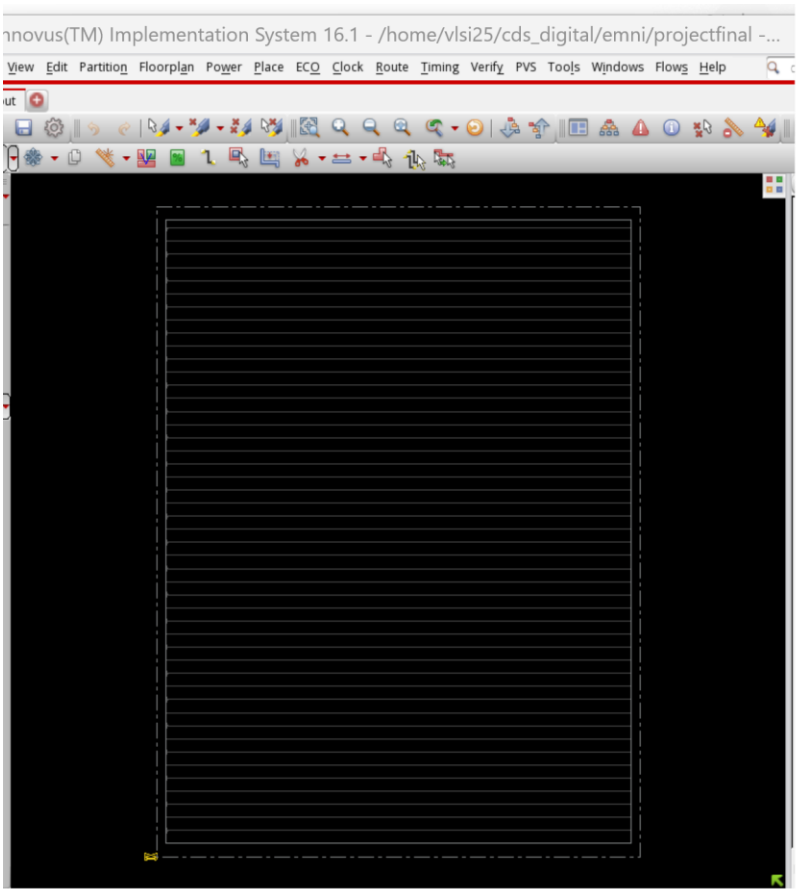
Pin editor

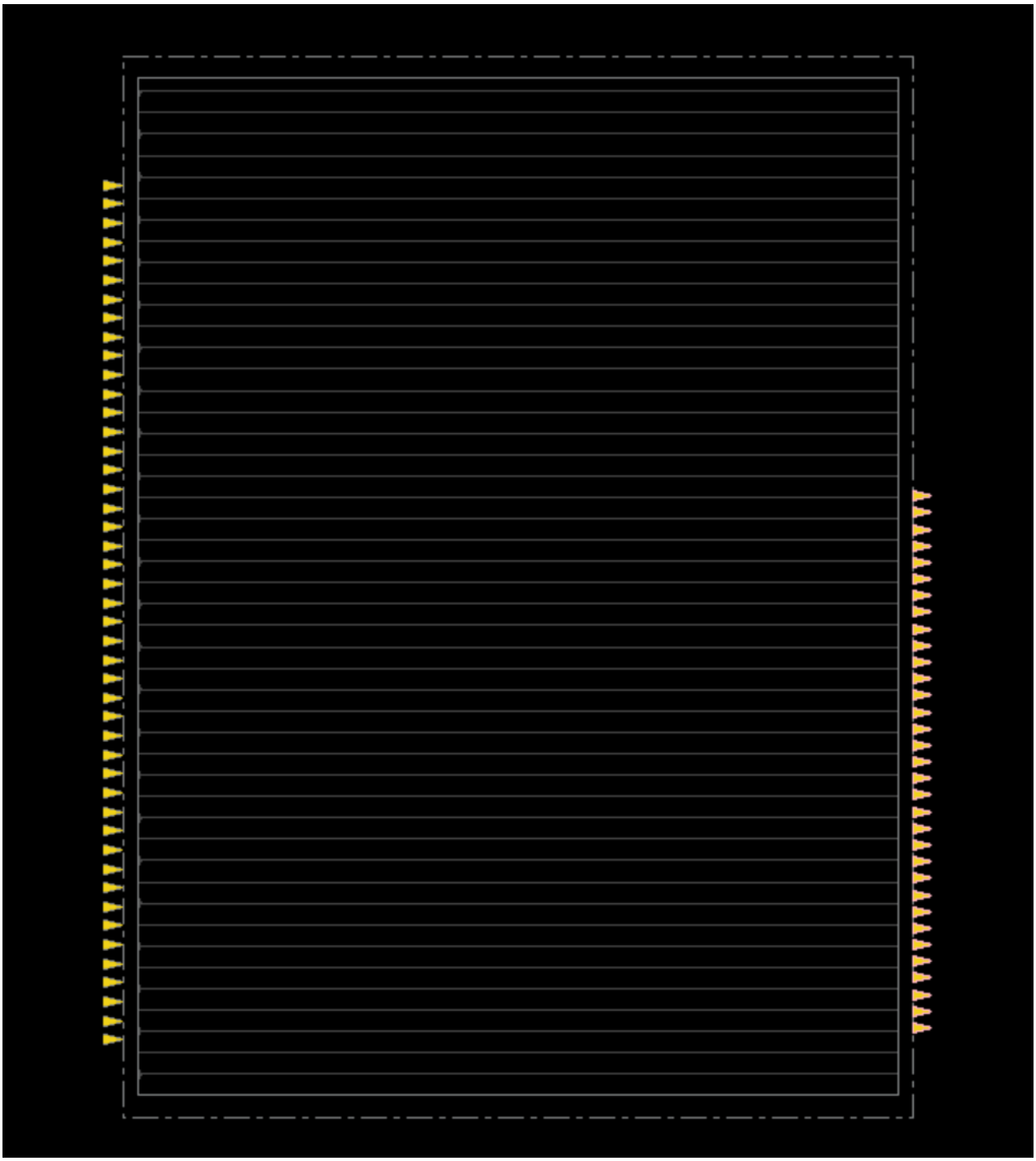
(Output pin)


Pin Editor
Partition: uart
Clone
Show Partition Edge Number
Edge number based on master
1
0
2
3
Pin Group
Side/Group/Layer: All
clk (Left)
dvsr [ ]
r_data [ ]
rd_request (Left)
reset (Left)
rx_done (Right)
w_data [ ]
Group Bus
Reverse Order
Sort By: Name Location
Find Pins:
Append Pin(s) to Pin Name List
Pin Attribute
Pin Name(s): 26] r_data[27] r_data[28] r_data[29] r_data[30] r_data[31] rx_done
Side/Edge: Right
Layer: Metal1(1) Metal2(2) Metal3(3) Metal4(4) Metal5(5) Metal6(6) Metal7(7
LayerV: Metal1(1) Metal2(2) Metal3(3) Metal4(4) Metal5(5) Metal6(6) Metal7(7
Depth: 0.25
Width: 0.08
Location
Update attribute
Assign location
Spread
Spread Type From Starting Point
Pattern
Pattern Name FILL_TRACK
Reverse Alternate
Include Rectilinear Edge
Position
Absolute based
Global Coordinates
Starting X: 63.2
Y: 7.125
Ending X: 63.2
Y: 49.685
Offset based
Starting:
Ending:
Start to end Direction Counterclockwise
Spacing: 1.33
Unit: Micron Layer Pitch
Snap To
Manufacturing Grid
User Grid
Layer Track
Assign Fixed Status
Batch Mode
Master Clone Aware
Fix Overlapping
Honor Partition and/or Pin-Level Constraints
Use: SIGNAL

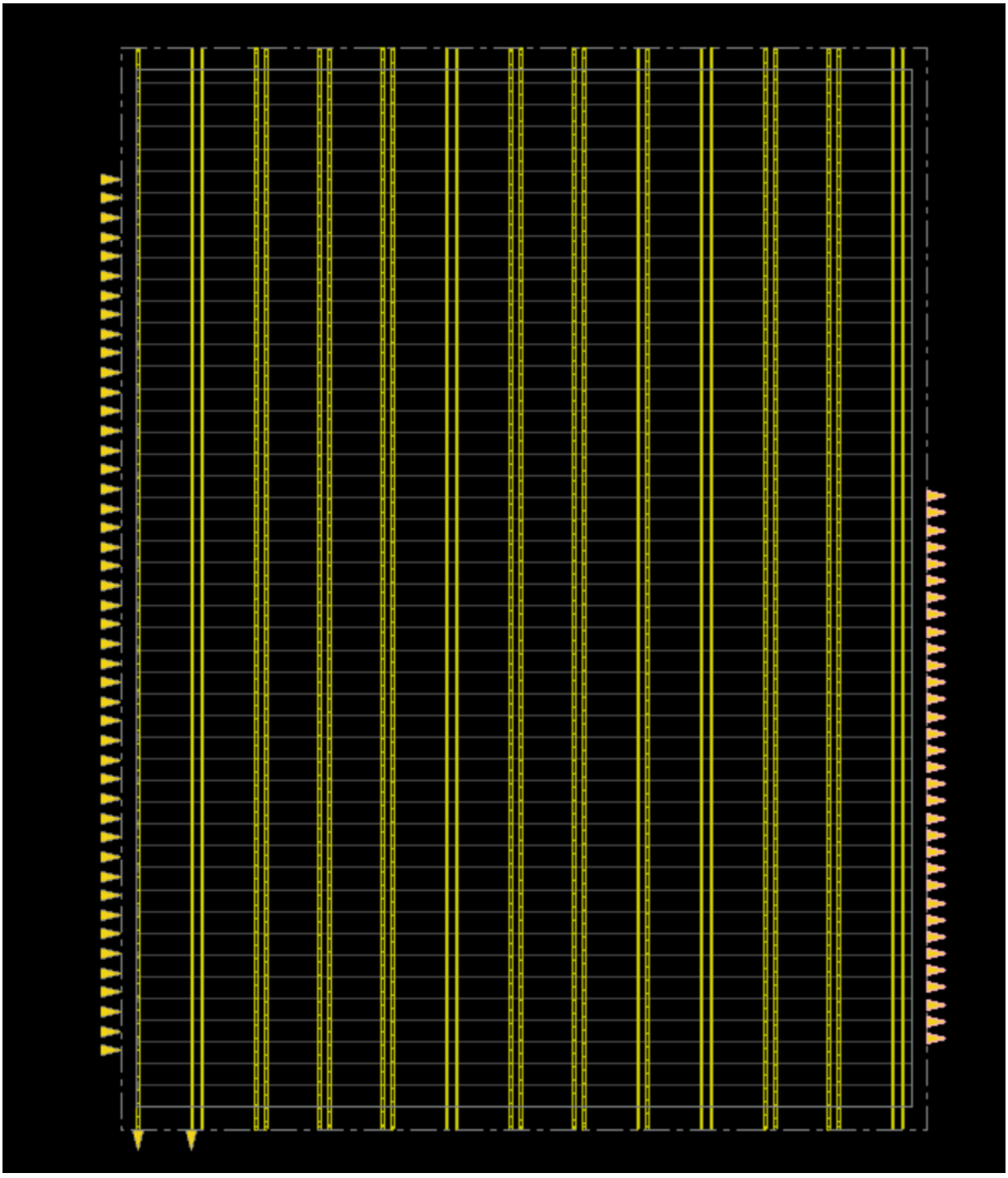
(Input pin)


Pin Editor
Partition: uart
Clone
Show Partition Edge Number
Edge number based on master
1
0
2
3
Pin Group
Side/Group/Layer: All
clk (Left)
dvsr [ ]
r_data [ ]
rd_request (Left)
reset (Left)
rx_done (Right)
w_data [ ]
Group Bus
Reverse Order
Sort By: Name Location
Find Pins:
Append Pin(s) to Pin Name List
Pin Attribute
Pin Name(s): data[26] w_data[27] w_data[28] w_data[29] w_data[30] w_data[31]
Side/Edge: Left
Layer: Metal1(1) Metal2(2) Metal3(3) Metal4(4) Metal5(5) Metal6(6) Metal7(7
LayerV: Metal1(1) Metal2(2) Metal3(3) Metal4(4) Metal5(5) Metal6(6) Metal7(7
Depth: 0.25
Width: 0.08
Location
Update attribute
Assign location
Spread
Spread Type From Starting Point
Pattern
Pattern Name FILL_TRACK
Reverse Alternate
Include Rectilinear Edge
Position
Absolute based
Global Coordinates
Starting X: 0.0 Y: 6.175
Ending X: 0.0 Y: 74.575
Offset based
Starting:
Ending:
Start to end Direction Counterclockwise
Spacing: 1.52
Unit: Micron Layer Pitch
Snap To
Manufacturing Grid
User Grid
Layer Track
Assign Fixed Status
Batch Mode
Master Clone Aware
Fix Overlapping
Honor Partition and/or Pin-Level Constraints
Use: SIGNAL

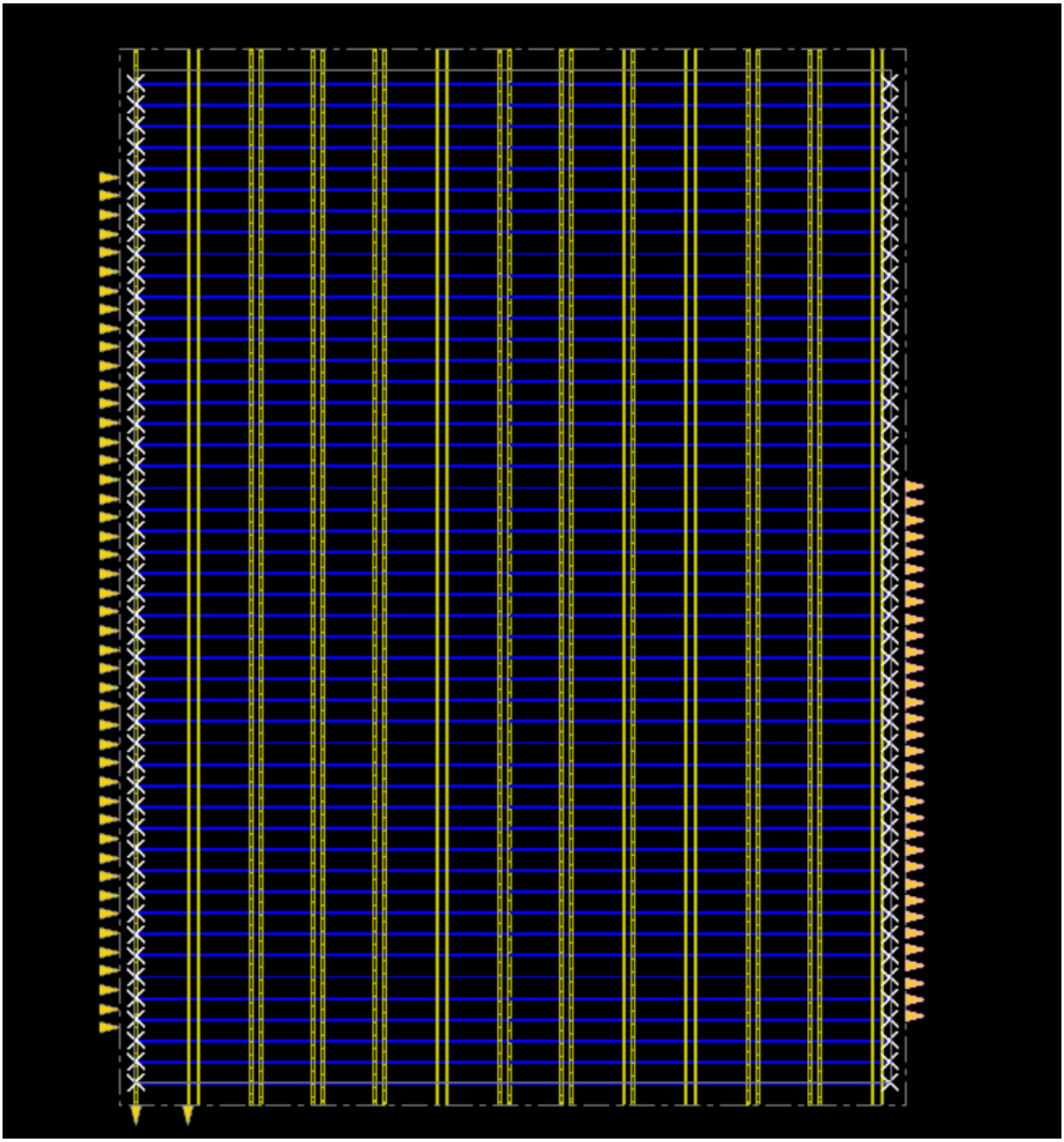

nnovus(TM) Implementation System 16.1 - /home/vlsi25/cds_digital/emni/projectfinal -...
View Edit Partition Floorplan Power Place ECO Clock Route Timing Verify PVS Tools Windows Flows Help



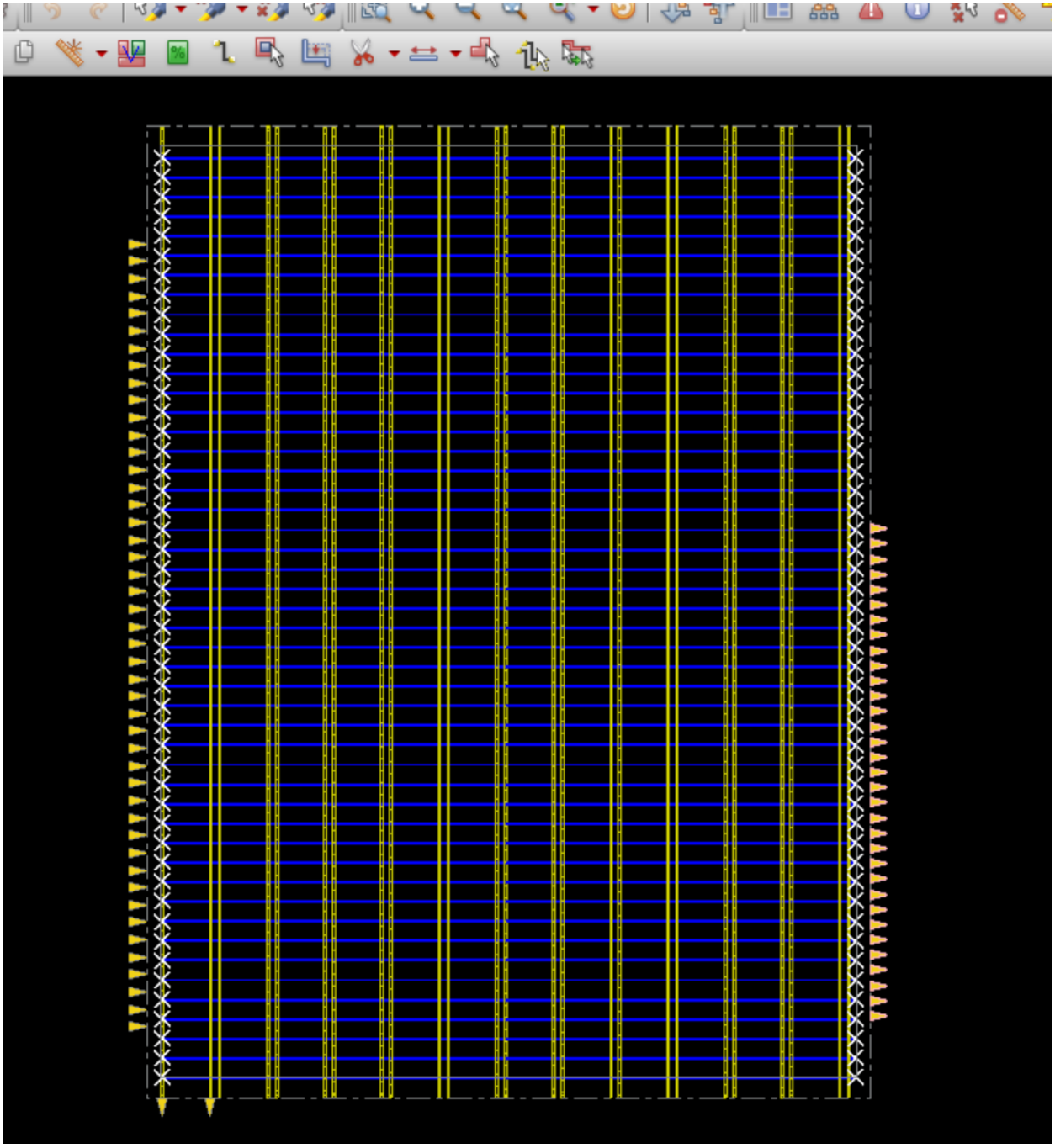
Stripe



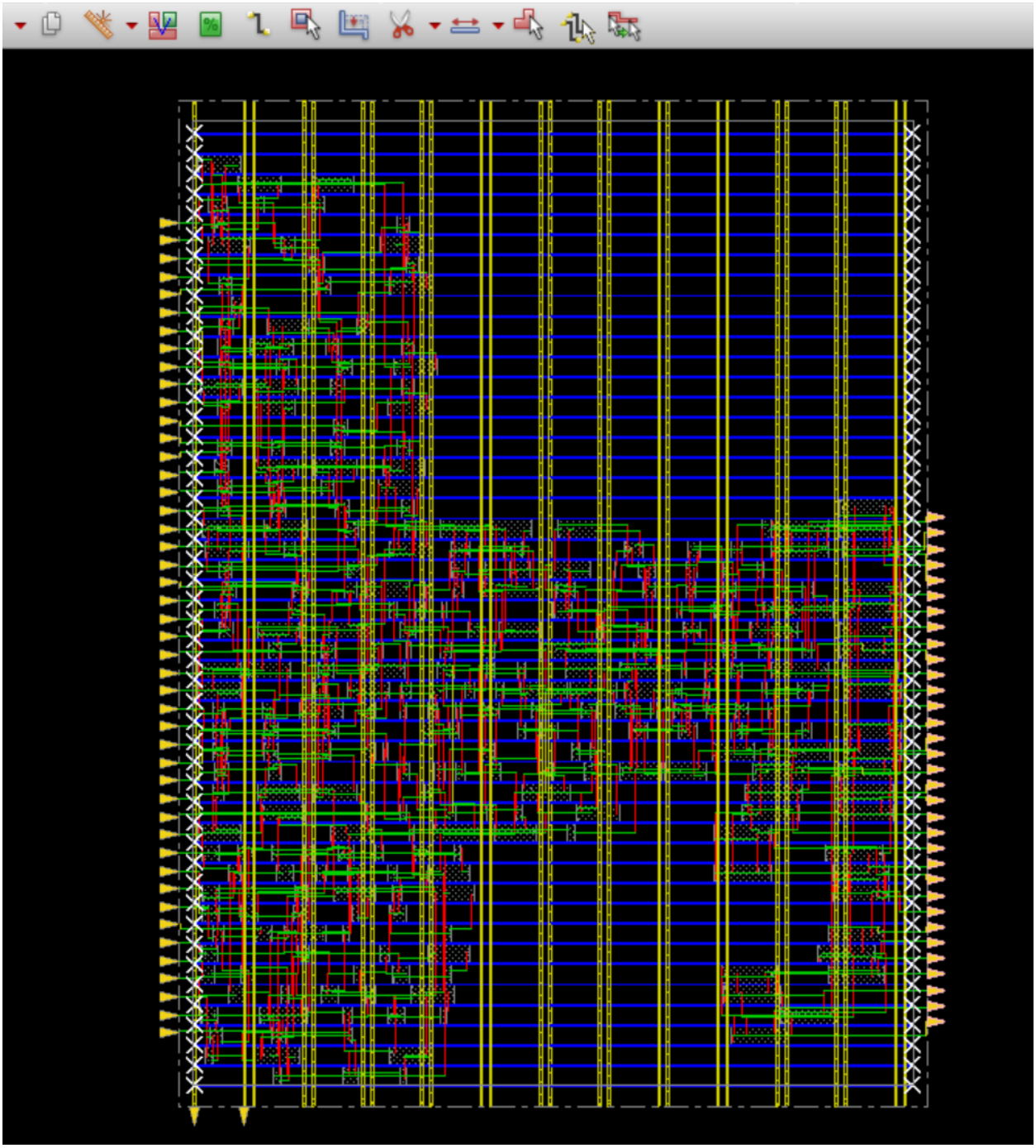
Special route



Connecting M1 rails to respective M4 stripe



Place opt design



Opt design final summary

```
------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  1.837  |  1.837  |  2.162  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   266   |   168   |   146   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              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: 21.263%
Routing Overflow: 0.00% H and 0.00% V
```

Clock Tree Convergence done

```
innovus 13> source ccopt.spec
Extracting original clock gating for func_clk...
  clock_tree func_clk contains 98 sinks and 0 clock gates.
  Extraction for func_clk complete.
Extracting original clock gating for func_clk done.
Checking clock tree convergence...
Checking clock tree convergence done.
innovus 14>
```

Clock Tree Synthesis No Violation

```
Setting all clocks to propagated mode.
Resetting all latency settings from fanout cone of clock 'func_clk'
Clock DAG stats after update timingGraph:
  cell counts      : b=0, i=0, icg=0, nicg=0, l=0, total=0
  cell areas       : b=0.000um^2, i=0.000um^2, icg=0.000um^2, nicg=0.000um^2, l=0.000um^2, total=0.000um^2
  cell capacitance : b=0.000pF, i=0.000pF, icg=0.000pF, nicg=0.000pF, l=0.000pF, total=0.000pF
  sink capacitance : count=98, total=0.028pF, avg=0.000pF, sd=0.000pF, min=0.000pF, max=0.000pF
  wire capacitance : top=0.000pF, trunk=0.000pF, leaf=0.037pF, total=0.037pF
  wire lengths     : top=0.000um, trunk=0.000um, leaf=462.145um, total=462.145um
Clock DAG net violations after update timingGraph: none
Clock DAG transition distribution after update timingGraph:
  Leaf : target=0.035ns count=1 avg=0.010ns sd=0.000ns min=0.010ns max=0.010ns {1 < 0.014ns}
Skew group summary after update timingGraph:
  skew_group func_clk/func: insertion delay [min=0.001, max=0.006, avg=0.003, sd=0.002], skew [0.005 vs 0.
026, 100% {0.001, 0.003, 0.006}] (wid=0.006 ws=0.005) (gid=0.000 gs=0.000)
Clock network insertion delays are now [0.001ns, 0.006ns] average 0.003ns std.dev 0.002ns
Logging CTS constraint violations...
  No violations found.
Logging CTS constraint violations done.
Synthesizing clock trees with CCOpt done.

*** Summary of all messages that are not suppressed in this session:
Severity  ID               Count  Summary
WARNING   IMPEXT-3530          5  The process node is not set. Use the com...
ERROR     IMPCCOPT-3092        1  Couldn't load external LP solver library...
WARNING   IMPCCOPT-1361        3  Routing configuration for %s nets in clo...
*** Message Summary: 8 warning(s), 1 error(s)

**ccopt_design ... cpu = 0:00:05, real = 0:00:05, mem = 1262.6M, totSessionCpu=0:01:12 **
innovus 15>
```

Report Timing After CTS:

```
#####################################################################
Path 1: MET Setup Check with Pin uart_rx_unit/n_reg_reg[3]/CK
Endpoint:   uart_rx_unit/n_reg_reg[3]/D (^) checked with  leading edge of 'func_
clk'
Beginpoint: bdgen/r_reg_reg[4]/Q       (v) triggered by  leading edge of 'func_
clk'
Path Groups: {func_clk}
Analysis View: func@BC_rcbest0.hold
Other End Arrival Time          0.002
- Setup                         0.031
+ Phase Shift                   2.500
= Required Time                 2.471
- Arrival Time                  0.631
= Slack Time                    1.840
     Clock Rise Edge                 0.000
     + Clock Network Latency (Prop)  -0.003
     = Beginpoint Arrival Time       -0.003
     +-------------------------------------------------------------------------------------+
     |         Instance          |     Arc      |   Cell    | Delay | Arrival | Required |
     |                           |              |           |       |  Time   |   Time   |
     |---------------------------+--------------+-----------+-------+---------+----------|
     | bdgen/r_reg_reg[4]        | CK ^         |           |       |  -0.003 |    1.838 |
     | bdgen/r_reg_reg[4]        | CK ^ -> Q v  | DFFRHQX1  | 0.060 |   0.058 |    1.898 |
     | bdgen/g175                | A v -> Y ^   | NOR4X1    | 0.043 |   0.101 |    1.941 |
     | bdgen/g174                | C ^ -> Y ^   | AND4X1    | 0.088 |   0.188 |    2.029 |
     | uart_rx_unit/g1163        | AN ^ -> Y ^  | NOR2BX1   | 0.039 |   0.228 |    2.068 |
     | uart_rx_unit/g2470        | B ^ -> Y v   | NAND2X2   | 0.135 |   0.363 |    2.203 |
     | uart_rx_unit/g2468        | A v -> Y ^   | INVX1     | 0.082 |   0.445 |    2.285 |
     | uart_rx_unit/g2467        | B ^ -> Y v   | NAND2XL   | 0.050 |   0.495 |    2.335 |
     | uart_rx_unit/g2421        | AN v -> Y v  | NAND3BXL  | 0.049 |   0.543 |    2.384 |
     | uart_rx_unit/g2414        | C v -> Y ^   | NAND3XL   | 0.040 |   0.583 |    2.424 |
     | uart_rx_unit/g2408        | B ^ -> Y v   | NAND2XL   | 0.031 |   0.615 |    2.455 |
     | uart_rx_unit/g2399        | C0 v -> Y ^  | OAI221X1  | 0.016 |   0.631 |    2.471 |
     | uart_rx_unit/n_reg_reg[3] | D ^          | DFFRHQX1  | 0.000 |   0.631 |    2.471 |
     +-------------------------------------------------------------------------------------+

innovus 19>
```

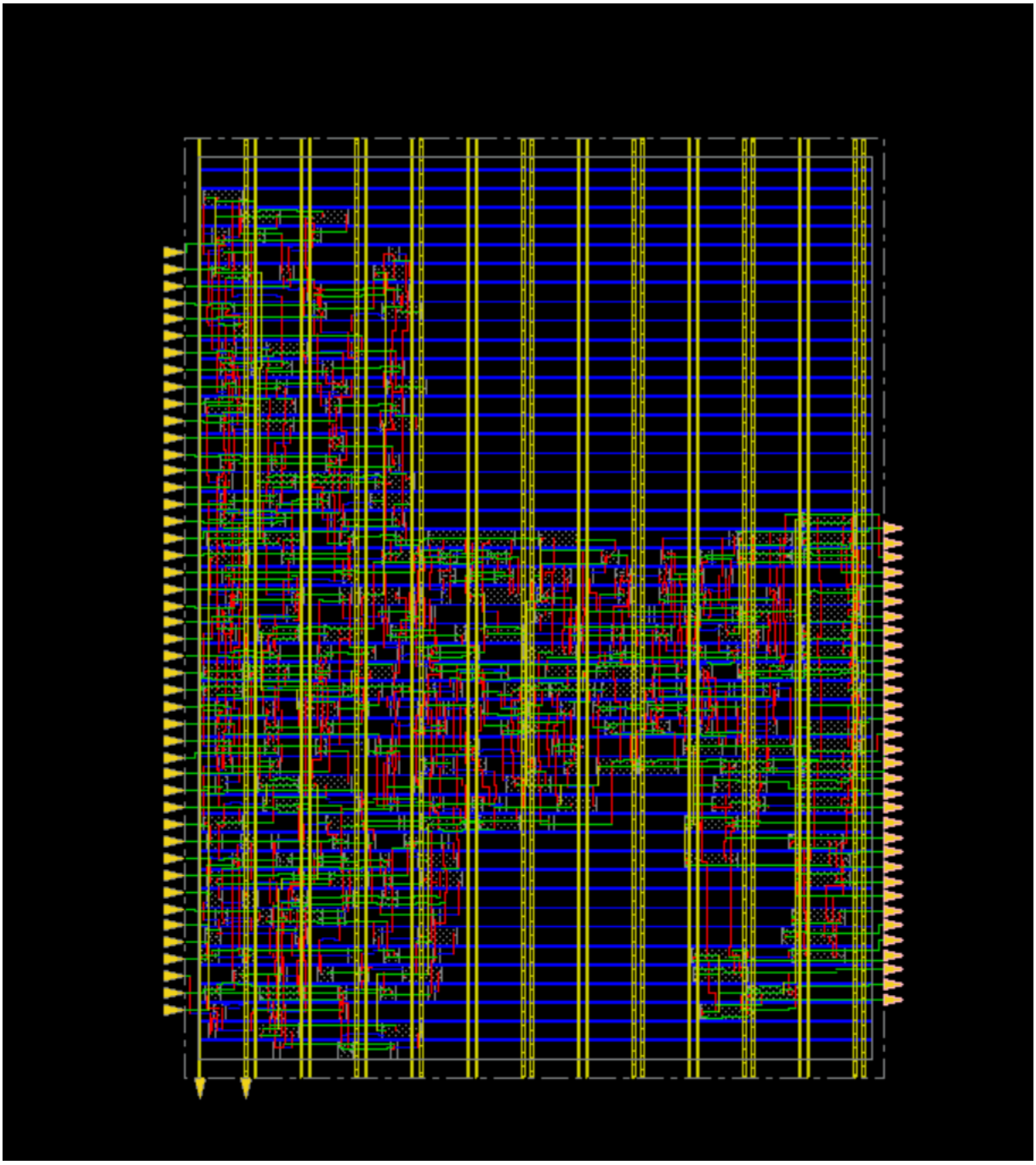
Core Utilization

```
Core utilization  = 94.874862
Effective Utilizations
Average module density = 0.962.
Density for the design = 0.962.
       = stdcell_area 13748 sites (4702 um^2) / alloc_area 14288 sites (4886 um^2).
Pin Density = 0.09323.
            = total # of pins 1332 / total area 14288.
innovus 30>
```

Power Report

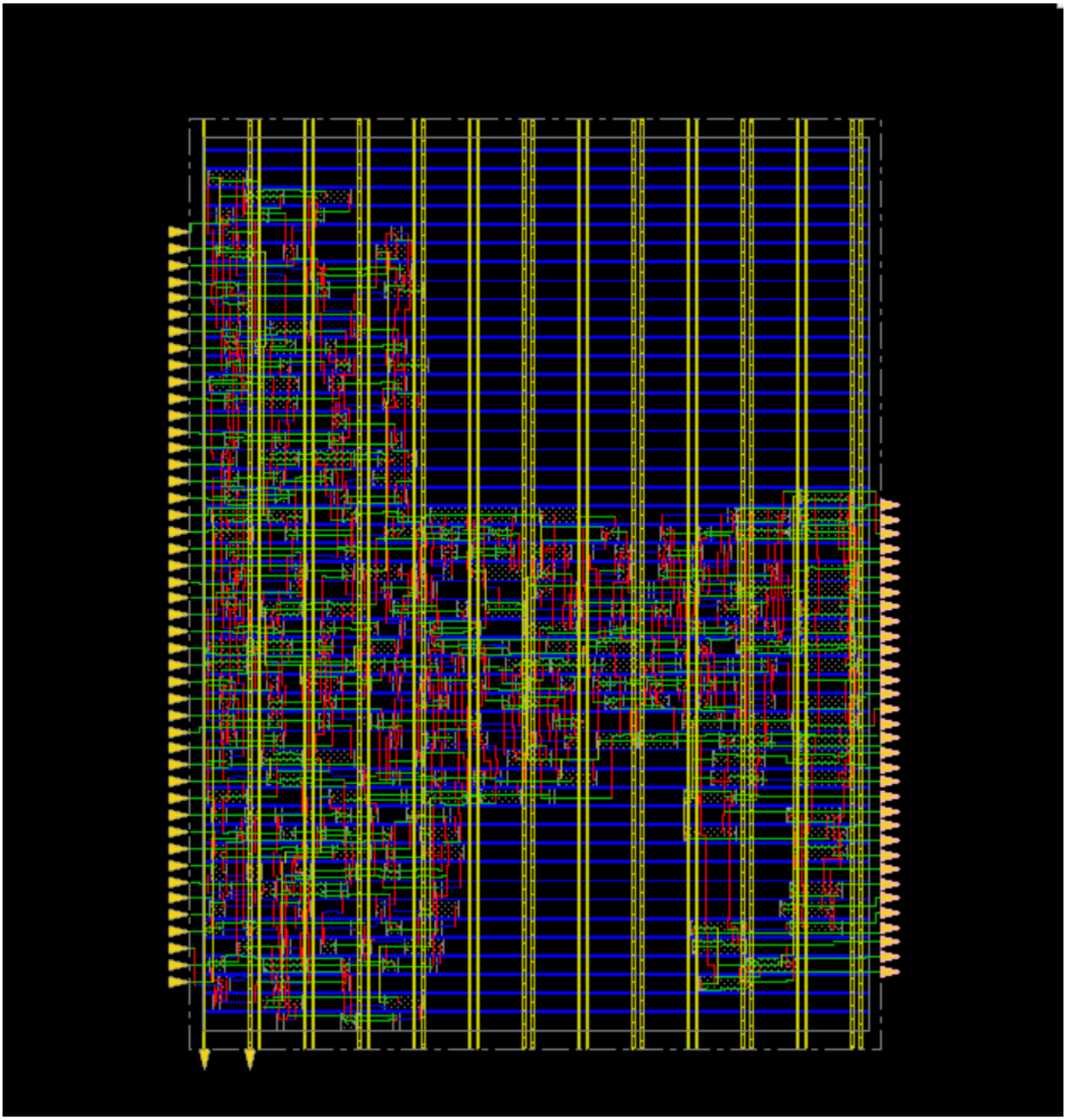
```
Total Power
--------------------------------------------------------------------------------
Total Internal Power:        0.21666625            90.4362%
Total Switching Power:       0.02285054             9.5378%
Total Leakage Power:         0.00006221             0.0260%
Total Power:                 0.23957899
--------------------------------------------------------------------------------


Group                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
--------------------------------------------------------------------------------
Sequential                           0.2    0.008819   4.129e-05        0.2089       87.17
Macro                                  0           0           0             0           0
IO                                     0           0           0             0           0
Combinational                    0.01668     0.01403   2.092e-05       0.03073       12.83
Clock (Combinational)                  0           0           0             0           0
Clock (Sequential)                     0           0           0             0           0
--------------------------------------------------------------------------------
Total                              0.2167     0.02285   6.221e-05        0.2396         100
--------------------------------------------------------------------------------


Rail                  Voltage   Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
--------------------------------------------------------------------------------
VDD                       1.1     0.2167     0.02285   6.221e-05        0.2396         100


--------------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power:        uart_tx_unit/g1751 (INVX2):         0.002406
*               Highest Leakage Power: uart_rx_unit/n_reg_reg[0] (SDFFRHQX1):    5.036e-07
*               Total Cap:      5.96897e-13 F
*               Total instances in design:   305
*               Total instances in design with no power:     0
*                Total instances in design with no activty:     0

*               Total Fillers and Decap:     0
--------------------------------------------------------------------------------

Ended Static Power Report Generation: (cpu=0:00:00, real=0:00:00,
mem(process/total)=1121.21MB/1121.21MB)

1
innovus 32>
```

Nano route



Post route

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  1.843  |  1.843  |  2.163  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   266   |   168   |   146   |
+--------------------+---------+---------+---------+

+----------------+--------------------------------+------------------+
|                |              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: 21.263%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 3.28 sec
Total Real time: 3.0 sec
Total Memory Usage: 1393.597656 Mbytes
Reset AAE Options
innovus 23> innovus 23>
```



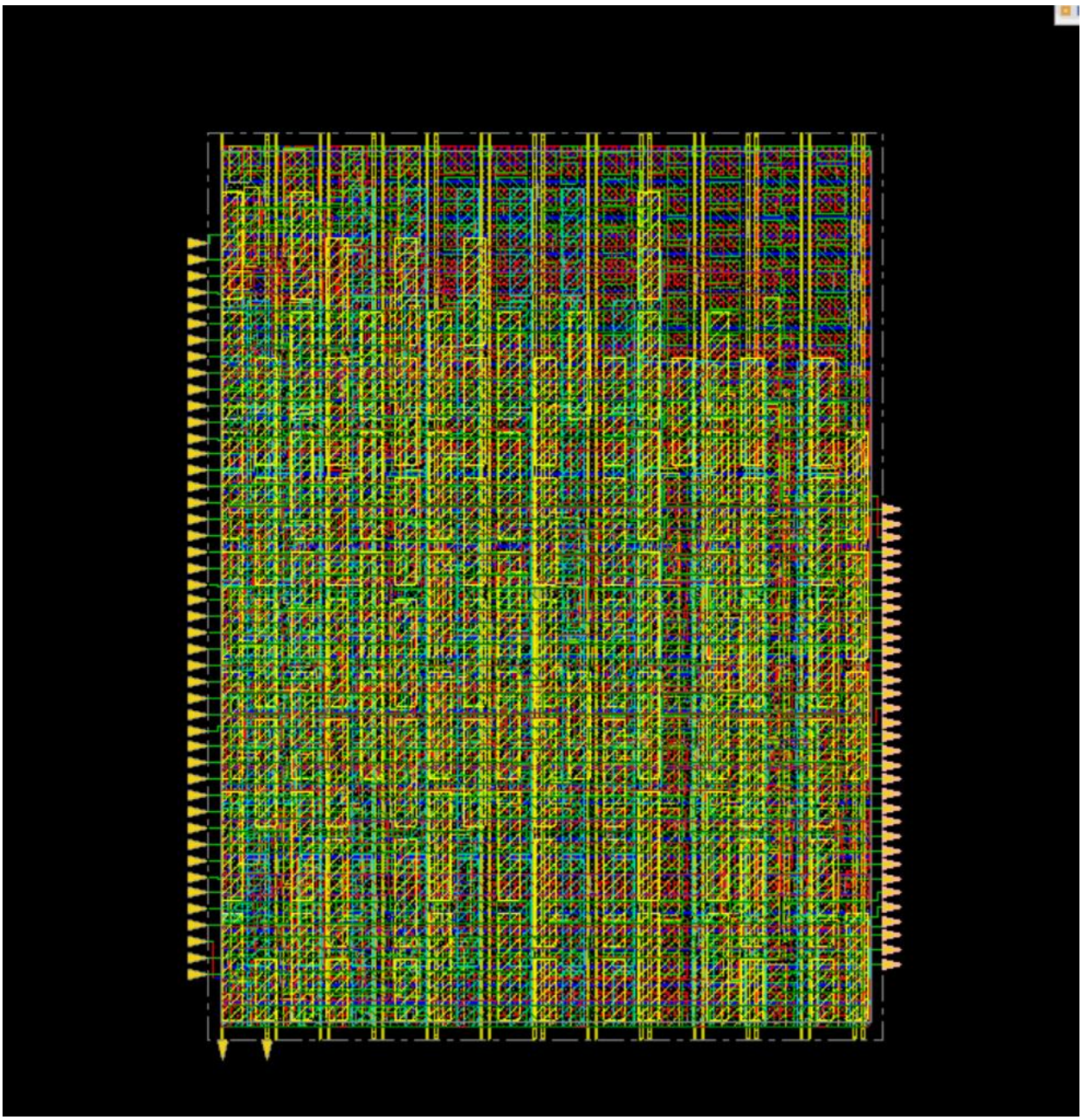
Opt design final summary

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

Setup views included:
 func@BC_rcbest0.hold
Hold  views included:
 func@BC_rcbest0.hold

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  1.843  |  1.843  |  2.163  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   266   |   168   |   146   |
+--------------------+---------+---------+---------+

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.035  |  0.069  |  0.035  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|   266   |   168   |   146   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              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: 21.263%
Total number of glitch violations: 0
------------------------------------------------------------
**optDesign ... cpu = 0:00:10, real = 0:00:12, mem = 1486.7M, totSessionCpu=0:01:56 **
 ReSet Options after AAE Based Opt flow
```

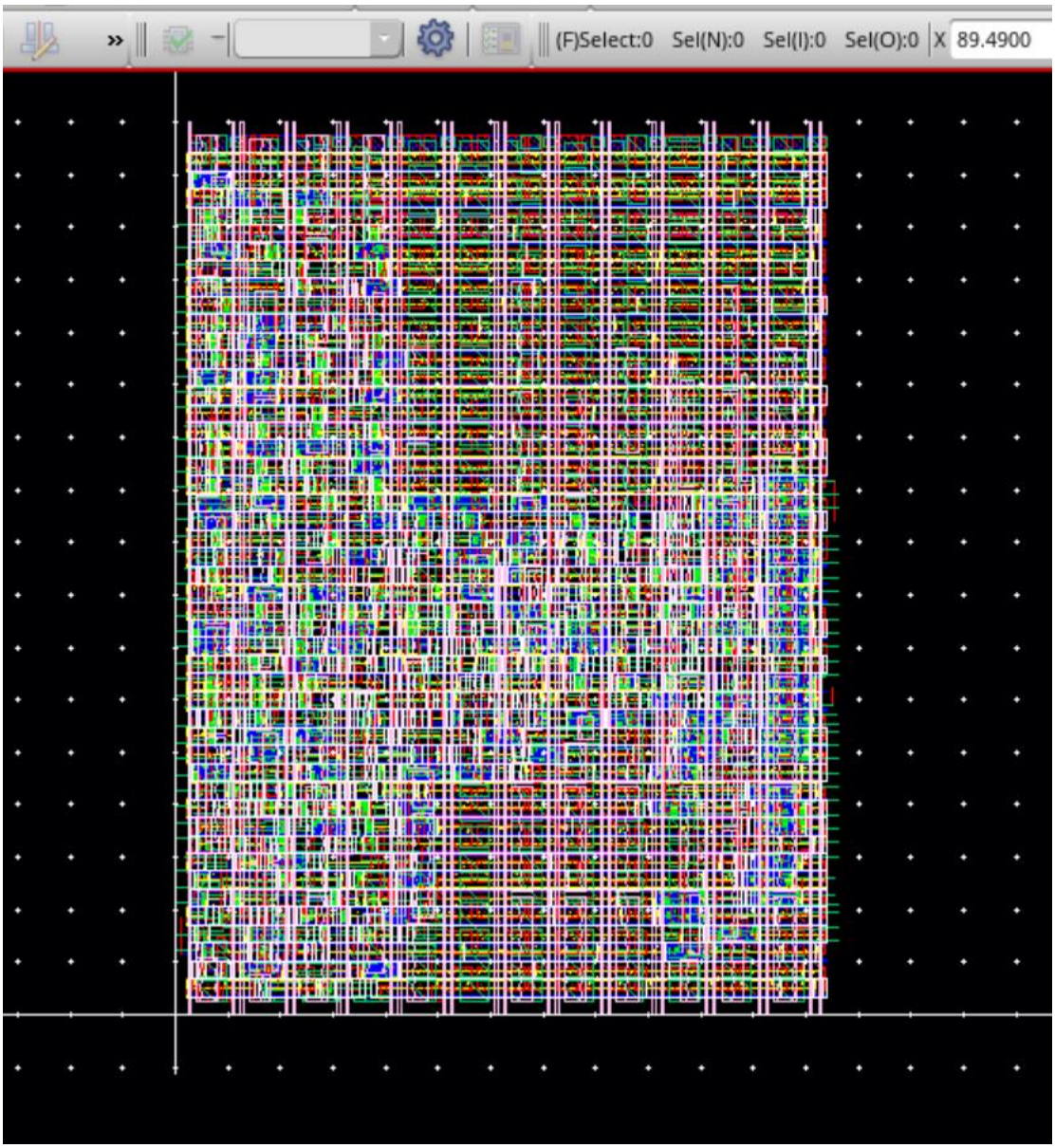
Geometry verification

```
innovus 25> innovus 25>  *** Starting Verify Geometry (MEM: 1486.7) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this r
elease but will be removed in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.3  MEM: 168.1M)

 *** Starting Verify Geometry (MEM: 1654.9) ***

**WARN: (IMPVFG-257):   verifyGeometry command is replaced by verify_drc command. It still works in this r
elease but will be removed in future release. Please update your script to use the new command.
  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
```

Metal filling



Strmout



Code:

```
// Code your design here
module uart
 #(
  parameter DBIT = 32, // # data bits
     SB_TICK = 16// # 16 ticks for 1 stop bit
 )

 (
  input clk, reset,
  input rd_request,
  input [31:0] w_data,
  input [10:0] dvsr,
  output rx_done,
  output [31:0] r_data
 );

 // signal declaration
 wire tx_rx, tick ;

 // body
 baud_gen bdgen (.clk(clk),.reset(reset),.dvsr(dvsr),.tick(tick));

 uart_rx #(.DBIT(DBIT), .SB_TICK(SB_TICK)) uart_rx_unit
  (.clk(clk),.reset(reset),.rx(tx_rx),.s_tick(tick),.rx_done_tick(rx_done),.d_out(r_d
ata));

 uart_tx #(.DBIT(DBIT), .SB_TICK(SB_TICK)) uart_tx_unit
  (.clk(clk),.reset(reset),.tx_start(rd_request),.s_tick(tick),.din(w_data),.tx(tx_rx))
;

endmodule
```

```
module baud_gen
 (
  input clk, reset,
  input [10:0] dvsr,
  output tick
 );

 // declaration
 reg [10:0] r_reg;
 reg [10:0] r_next;

 //body
 //register
 always @(posedge clk or posedge reset)
  if(reset)
   r_reg <= 0;
  else
   r_reg <= r_next;

 always@(*)
  r_next = (r_reg==dvsr) ? 0 : r_reg+1;

 // output reg
 assign tick = (r_reg==1);
endmodule


// Code your design here
module uart_rx
 #(
  parameter DBIT = 32, // # data bits
     SB_TICK = 16 // # 16 ticks for 1 stop bit
 )
```

```
(
 input clk, reset,
 input rx, s_tick,
 output reg rx_done_tick,
 output [31:0] d_out
);

// fsm state type
//typedef enum {idle, start, data, stop} state_type;
localparam idle =2'b00, start =2'b01, data = 2'b10, stop = 2'b11;

// signal declaration
reg [1:0 ] state_reg, state_next;
reg [3:0] s_reg, s_next;
reg [4:0] n_reg, n_next;
reg [31:0] b_reg, b_next;

// body
// FSMD state & data registers
 always @(posedge clk or posedge reset)
 if(reset) begin
  state_reg <= idle;
  s_reg <= 0;
  n_reg <= 0;
  b_reg <= 0;
 end
 else begin
  state_reg <= state_next;
  s_reg <= s_next;
  n_reg <= n_next;
  b_reg <= b_next;
 end

// FSMD next-state reg
```

```
always@(*)
begin
 state_next = state_reg;
 rx_done_tick = 1'b0;
 s_next = s_reg;
 n_next = n_reg;
 b_next = b_reg;
 case (state_reg)
  idle:
   if(~rx) begin
    state_next = start;
    s_next = 0;
   end
  start:
   if(s_tick)
    if(s_reg==7) begin
     state_next = data;
     s_next = 0;
     n_next = 0;
    end
    else
     s_next = s_reg + 1;
  data:
   if(s_tick)
    if(s_reg==15) begin
     s_next = 0;
     b_next = {rx, b_reg[31:1]};
     if (n_reg==(DBIT-1))
      state_next = stop;
     else
      n_next = n_reg + 1;
    end
    else
     s_next = s_reg + 1;
```

```
  stop:
   if(s_tick)
    if(s_reg==SB_TICK-1) begin
     state_next = idle;
     rx_done_tick = 1'b1;
    end
    else
     s_next = s_reg + 1;
 endcase
end
// output
assign d_out = b_reg;
endmodule




module uart_tx
 #(
  parameter DBIT = 32, // # data bits
     SB_TICK = 16 // # 16 ticks for 1 stop bit
 )
 (
  input clk, reset,
  input tx_start, s_tick,
  input [31:0] din,
 // output reg tx_done_tick,
  output tx
 );

 // fsm state type
 //typedef enum {idle, start, data, stop} state_type;
 localparam idle =2'b00, start =2'b01, data = 2'b10, stop = 2'b11;
```

```
// signal declaration
reg [1:0] state_reg, state_next;
reg [4:0] s_reg, s_next;
reg [3:0] n_reg, n_next;
reg [31:0] b_reg, b_next;
reg tx_reg, tx_next;

// body
// FSMD state & data registers
always @(posedge clk or posedge reset)
 if(reset) begin
  state_reg <= idle;
  s_reg <= 0;
  n_reg <= 0;
  b_reg <= 0;
  tx_reg <= 1'b1;
 end
 else begin
  state_reg <= state_next;
  s_reg <= s_next;
  n_reg <= n_next;
  b_reg <= b_next;
  tx_reg <= tx_next;
 end

// FSMD next-state reg
always @(*)
begin
 state_next = state_reg;
 //tx_done_tick = 1'b0;
 s_next = s_reg;
 n_next = n_reg;
 b_next = b_reg;
 tx_next = tx_reg;
```

```
case (state_reg)
 idle: begin
  tx_next = 1'b1;
  if(tx_start) begin
   state_next = start;
   s_next = 0;
   b_next = din;
  end
```