3 Cycle Non-Pipeline Harvard Structure Microcontroller

Md. Jannatun Nayem Tonmoy, Anwar Hakim Tamim, Abu Muhammad Shafayat Kabir, Asaduzzaman Seam, Md. Khairul Islam Ratul

Department of EEE, Ahsanullah University of Science and Technology

180105091@aust.edu, 180105092@aust.edu, 180105095@aust.edu, 180105100@aust.edu, 180105198@aust.edu

***Abstract*— This document notes step by step procedures of synthesizing and physical design of a 3 cycle non-pipelined Harvard Structure Micro-Controller. This document also gives an idea of problems can be occurred during the process and how to solve those problems. We used Genus tool to synthesize and Encounter tool to physically design the microcontroller. Encounter tool is also used for removing violations and errors. In our design we used both command and manual method to remove violations and errors. This document also tries to solve problems and violations occurring when using Encounter tool as it is designed for professional uses and not for educational purposes. As this is a group project, this paper is divided into several parts to better suit individual group member. This document also gives a rough idea for future aspect of the project, as Microcontrollers are used in almost every aspect of electronics field. But future problems are not described in detail in this document.**

***Keywords*— Microcontroller, Genus, Encounter, DRC, Connectivity, Via, Violation.**

I. Introduction

A. Microcontroller

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).

The Harvard architecture stores machine instructions and data in separate memory units that are connected by different busses. In this case, there are at least two memory address spaces to work with, so there is a memory register for machine instructions and another memory register for data. Computers designed with the Harvard architecture are able to run a program and access data independently, and therefore simultaneously. Harvard architecture has a strict separation between data and code. In a Non-Pipelining system, processes like decoding, fetching, execution and writing memory are merged into a single unit or a single step. Only one instruction is executed at the same time.

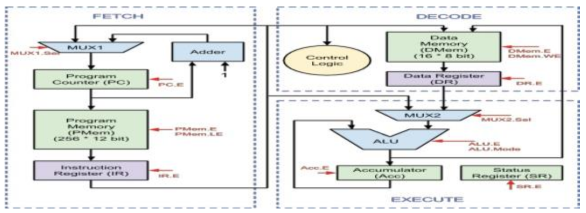


Fig I 1: Architecture of Microcontroller

The following diagram is the architecture of the microcontroller. The datapath is shown asblack arrows and control signals are red arrows.

The following two type of components holds programming context –

- Program counter, program memory, data memory, accumulator, status register (green boxes). They are programmer visible registers and memories.
- Instruction register and data register (purple boxes). They are programmer invisibleregisters.

The following two type of components is Boolean logics that do the actual computationwork. They are stateless -

- ALU, MUX1, MUX2, Adder (blue boxes), used as a functional unit.
- Control Logic (yellow box), used to denote all control signals (red signal).

B. Instruction Set

There are 3 sets of instruction. Each instruction is 12 bits. There are 3 types of instructions by encoding, shown as following:

- M type: one operand is accumulator (sometimes ignored) and the other operand is from data memory; the result can be stored into accumulator or the data memory entry (same entry as the second operand).
- I type**:** one operand is accumulator and the other operand is immediate number encoded in instruction; the result is stored into accumulator.
- S type**:** special instruction, no operand required. (e.g., NOP).

The instruction encoding is given in following table:

Table I 1: Instruction Encoding

Code space (binary)	Code space (hex)	Type	Number of Instructions	Note
0000_0000_0000 - 0000_1111_1111	000-0FF	special instructions (S type)	256	Currently only NOP used,255 free slots
0001_0000_0000 - 0001_1111_1111	100-1FF	unconditional jump (I type)	1	GOTO
0010_0000_0000 - 0011_1111_1111	200-3FF	ALU instructions (M type)	32	16 instructions, 2 destination choices
0100_0000_0000 - 0111_1111_1111	400-7FF	conditional jump (I type)	4	JZ, JC, JS, JC
1000_0000_0000 - 1111_1111_1111	800-FFF	ALU instructions (I type)	8	Currently 7 used, 1 free slot

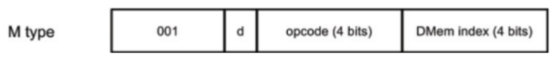
These instructions can be grouped into 4 categories by function -

1. ALU instruction: using ALU to compute result;
2. Unconditional branch: the GOTO instruction;
3. Conditional branch: the JZ, JC, JS, JO instruction;
4. Special instruction: the NOP.

Short description of each instructions are given below:

i) M Type Instructions

The general format of M type instruction is shown as following –



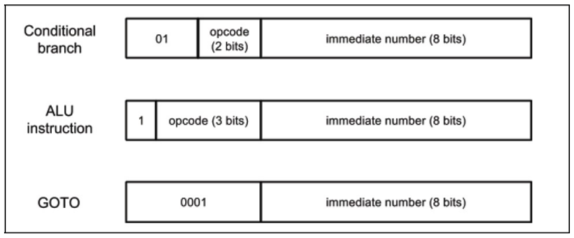
The tables below have detailed information:

Table I 2: M Type Instruction

instruction mnemonics	function	encoding (binary)	status affected	example (encoding) (assembly) (meaning)
ADD	add a memory entry with accumulator	001d_0000_aaaa	Z, C, S, O	0011_0000_0001 ADD Acc, Acc, DMem[1] (Acc = Acc + DMem[1])
SUBAM	subtract accumulator by a memory entry	001d_0001_aaaa	Z, C, S, O	0011_0001_0000 SUBAM Acc, Acc, DMem[0] (Acc = Acc - DMem[0])
MOVAM	move the value of accumulator to a memory entry	0010_0010_aaaa	none	0010_0010_0000 MOVAM DMem[0], Acc (Acc = DMem[0])
MOVMA	move the value of a memory entry to accumulator	0011_0011_aaaa	none	0011_0011_0000 MOVMA Acc, DMem[0] (DMem[0] = Acc)
AND	bitwise AND a memory entry with accumulator	001d_0100_aaaa	Z	0010_0100_0000 AND DMem[0], Acc, DMem[0] (DMem[0] = DMem[0] AND Acc)
OR	bitwise OR a memory entry with accumulator	001d_0101_aaaa	Z	0010_0101_0000 OR DMem[0], Acc, DMem[0] (DMem[0] = DMem[0] OR Acc)
XOR	bitwise XOR a memory entry with accumulator	001d_0110_aaaa	Z	0010_0110_0000 XOR DMem[0], Acc, DMem[0] (DMem[0] = DMem[0] XOR Acc)
SUBMA	subtract a memory entry by accumulator	001d_0111_aaaa	Z, C, S, O	0010_0001_0000 SUBAM DMem[0], DMem[0], Acc (DMem[0] = DMem[0] - Acc)
INC	increment a memory entry	0010_1000_aaaa	Z, C, S, O	0011_1000_0000 INC DMem[0] (DMem[0] = DMem[0] + 1)
DEC	decrement a memory entry	0010_1001_aaaa	Z, C, S, O	0011_1001_0000 DEC DMem[0] (DMem[0] = DMem[0] - 1)
ROTATEL	circulative shift left a memory entry, by the number of bits specified by accumulator	0010_1010_aaaa	none	0010_1010_0000 ROTATEL DMem[0] (see comments below)
ROTATER	circulative shift left a memory entry, by the number of bits specified by accumulator	0010_1011_aaaa	none	0010_1011_0000 ROTATER DMem[0] (see comments below)
SLL	shift a memory entry left, by the number of bits specified by accumulator	0010_1100_aaaa	Z, C	0010_1100_0000 SLL DMem[0] (see comments below)
SRL	shift a memory entry right, logical (fill 0), by the number of bits specified by accumulator	0010_1101_aaaa	Z, C	0010_1101_0000 SRL DMem[0] (see comments below)
SRA	shift a memory entry right, arithmetic (fill original MSB), by the number of bits specified by accumulator	0010_1110_aaaa	Z, C, S	0010_1110_0000 SRA DMem[0] (see comments below)
COMP	take 2's complement of a memory entry, i.e. 0 subtracted by the memory entry	0010_1111_aaaa	Z, C, S, O	0010_1111_0000 COMP DMem[0] (DMem[0] = - DMem[0])

ii) I Type Instructions

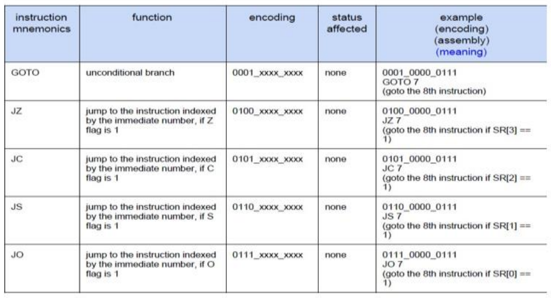
The general format of I type instruction is shown as following:



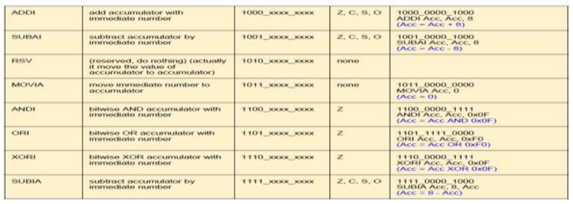
Conditional branch
01
opcode (2 bits)
immediate number (8 bits)
ALU instruction
1
opcode (3 bits)
immediate number (8 bits)
GOTO
0001
immediate number (8 bits)

The pictures below have detailed information:

Table I 3: I Type Instruction



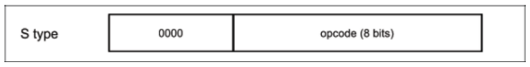
instruction mnemonics	function	encoding	status affected	example (encoding) (assembly) (meaning)
GOTO	unconditional branch	0001_xxxx_xxxx	none	0001_0000_0111 GOTO 7 (goto the 8th instruction)
JZ	jump to the instruction indexed by the immediate number, if Z flag is 1	0100_xxxx_xxxx	none	0100_0000_0111 JZ 7 (goto the 8th instruction if SR[3] == 1)
JC	jump to the instruction indexed by the immediate number, if C flag is 1	0101_xxxx_xxxx	none	0101_0000_0111 JC 7 (goto the 8th instruction if SR[2] == 1)
JS	jump to the instruction indexed by the immediate number, if S flag is 1	0110_xxxx_xxxx	none	0110_0000_0111 JS 7 (goto the 8th instruction if SR[1] == 1)
JO	jump to the instruction indexed by the immediate number, if O flag is 1	0111_xxxx_xxxx	none	0111_0000_0111 JO 7 (goto the 8th instruction if SR[0] == 1)



ADDI	add accumulator with immediate number	1000_xxxx_xxxx	Z, C, S, O	1000_0000_1000 ADDI Acc, Acc, 8 (Acc = Acc + 8)
SUBAI	subtract accumulator by immediate number	1001_xxxx_xxxx	Z, C, S, O	1001_0000_1000 SUBAI Acc, Acc, 8 (Acc = Acc - 8)
RSV	(reserved, do nothing) (actually it move the value of accumulator to accumulator)	1010_xxxx_xxxx	none	
MOVIA	move immediate number to accumulator	1011_xxxx_xxxx	none	1011_0000_0000 MOVIA Acc, 0 (Acc = 0)
ANDI	bitwise AND accumulator with immediate number	1100_xxxx_xxxx	Z	1100_0000_1111 ANDI Acc, Acc, 0x0F (Acc = Acc AND 0x0F)
ORI	bitwise OR accumulator with immediate number	1101_xxxx_xxxx	Z	1101_1111_0000 ORI Acc, Acc, 0xF0 (Acc = Acc OR 0xF0)
XORI	bitwise XOR accumulator with immediate number	1110_xxxx_xxxx	Z	1110_0000_1111 XORI Acc, Acc, 0x0F (Acc = Acc XOR 0x0F)
SUBIA	subtract accumulator by immediate number	1111_xxxx_xxxx	Z, C, S, O	1111_0000_1000 SUBIA Acc, 8, Acc (Acc = 8 - Acc)

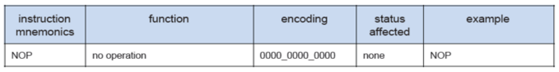
iii) S Type Instructions

The general format of S type instruction is shown as following –



There is only one S type instruction i.e., NOP instruction.

Table I 4: S Type Instruction



instruction mnemonics	function	encoding	status affected	example
NOP	no operation	0000_0000_0000	none	NOP

C. Timing and State Transition

The Timing and State Transition are as follows:

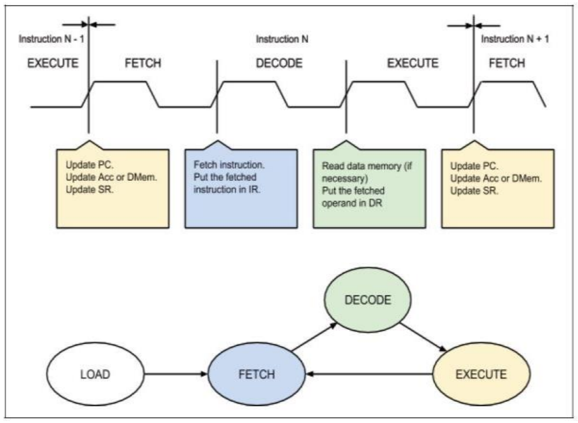


Fig I 2: Timing and State Transition

Each instruction needs 3 clock cycles to finish, i.e. FETCH stage, DECODE stage, and EXECUTE stage. It is not pipelined. Together with the initial LOAD state, it can be considered as an FSM of 3 states (technically 4 states).

There are 4 states. They are:

- <u>LOAD (initial state):</u> load program to program memory, which takes 1 cycle per instruction loaded;
- <u>FETCH (first cycle):</u> fetch current instruction from program memory;
- <u>DECODE (second cycle):</u> decode instruction to generate control logic, read data memory for operand;
- <u>EXECUTE (of the third cycle):</u> execute instruction;

D. Transitions

1. LOAD → FETCH (initialization finish):
2. Clear content of PC, IR, DR, Acc, SR; DMem is not required to be cleared.
3. FETCH → DECODE (rising edge of second cycle):
4. IR = PMem [PC]
5. DECODE → EXECUTE
6. DR = DMem [IR[3:0]]
7. EXECUTE → FETCH (rising edge of first cycle and fourth cycle):

For non-branch instruction, PC = PC + 1; for branch instruction, if branch is taken,PC = IR [7:0], otherwise PC = PC + 1;

For ALU instruction, if the result destination is accumulator, Acc = ALU.Out; ifthe result destination is data memory, DMem [IR[3:0]] = ALU.Out.

For ALU instruction, SR = ALU.Status;

The transitions can be simplified using enable port of corresponding registers, e.g., assignALU.Out to Acc at every clock rising edge if Acc.E is set to 1. Such control signals as Acc.E are generated as a Boolean function of both current state and the current instruction.

E. Components

1) Register

The microcontroller has 3 programmer visible register:

1. Program Counter (8 bit, denoted as PC): Contains the index of current executing instruction.
2. Accumulator (8 bit, denoted as Acc): Holds result and 1 operand of the arithmeticor logic calculation.
3. Status Register (4 bit, denoted as SR): Holds 4 status bit, i.e. Z, C, S, O.
    - Z (zero flag, SR[3]): 1 if result is zero, 0 otherwise.
    - C (carry flag, SR[2]): 1 if carry is generated, 0 otherwise.
    - S (sign flag, SR[1]): 1 if result is negative (as 2's complement), 0 otherwise.
    - (overflow flag, SR[0]): 1 if result generates overflow, 0 otherwise.

Each of these registers has an enable port, as a flag for whether the value of the registershould be updated in state transition. They are denoted as PC.E, Acc.E, and SR.E.

The microcontroller has 2 programmer invisible registers:

1. Instruction Register (12 bit, denoted as IR): Contains the current executinginstruction.
2. Data Register (8 bit, denoted as DR): Contains the operand read from datamemory.

Similarly, each of these registers has an enable port as a flag for whether the value of theregister should be updated in state transition. They are denoted as IR.E and DR.E

2) Program Memory

The microcontroller has a 256-entry program memory that stores program instructions, denoted as PMem. Each entry is 12 bits, the ith entry is denoted as PMem[i]. The program memory has the following input/output ports.

1. Enable port (1 bit, input, denoted as PMem.E): enable the device, i.e., if it is 1, thenthe entry specified by the address port will be read out, otherwise, nothing is read out.
2. Address port (8-bit, input, denoted as PMem.Addr): specify which instruction entryis read out, connected to PC.
3. Instruction port (12-bit, output, denoted as PMem.**I**): the instruction entry that is read out, connected to IR.
4. 3 special ports are used to load program to the memory, not used for executing instructions.
5. Load enable port (1 bit, input, denoted as PMem.LE): enable the load, i.e., if it is 1,then the entry specified by the address port will be load with the value specified by the load instruction input port and the instruction port is supplied with the same value; otherwise, the entry specified by the address port will be read out oninstruction port, and value on instruction load port is ignored.
6. Load address port (8-bit, input, denoted as PMem.LA): specify which instruction entry is loaded.
7. Load instruction port (12-bit, input, denoted as PMem.LI): the instruction that is loaded.

3) Data Memory

The microcontroller has a 16-entry data memory, denoted as DMem. Each entry is 8 bits,the i-th entry is denoted as DMem[i**]**. The program memory has the following input/outputports -

1. Enable port (1 bit, input, denoted as DMem.E): Enable the device, i.e. if it is 1, thenthe entry specified by the address port will be read out or written in; otherwise nothing is read out or written in.
2. Write enable port (1 bit, input, denoted as DMem.WE)**:** Enable the write, i.e., if it is 1, then the entry specified by the address port will be written with the value specified by the data input port and the data output port is supplied with the same value; otherwise, the entry specified by the address port will be read out on data output port, and value on data input port is ignored.
3. Address port (4-bit, input, denoted as DMem.Addr): Specify which data entry is read out, connected to IR[3:0].

4. Data input port (8-bit, input, denoted as DMem.DI): The value that is written in, connected to ALU.Out.
5. Data output port (8-bit, output, denoted as DMem.DO): The data entry that isread out, connected to MUX2.In1.

4) PC Adder

PC adder is used to add PC by 1, i.e., move to the next instruction. This component is purecombinational. It has the following ports —

1. Adder input port (8-bit, input, denoted as Adder.In): Connected to PC.
2. Adder output port (8-bit, output, denoted as Adder.Out): Connected to MUX1.In2.

5) MUX1

MUX1 is used to choose the source for updating PC. If the current instruction is not a branch or it is a branch but the branch is not taken, PC is incremented by 1; otherwise PCis set to the jumping target, i.e. IR [7:0]. It has the following ports —

1. MUX1 input 1 port (8-bit, input, denoted as MUX1.In1): Connected to IR [7:0].
2. MUX1 input 2 port (8-bit, input, denoted as MUX1.In2): Connected to Adder.Out.
3. MUX1 selection port (1-bit, input, denoted as MUX1.Sel): Connected to controllogic.
4. MUX1 output port (8-bit, output, denoted as MUX1.Out): Connected to PC.

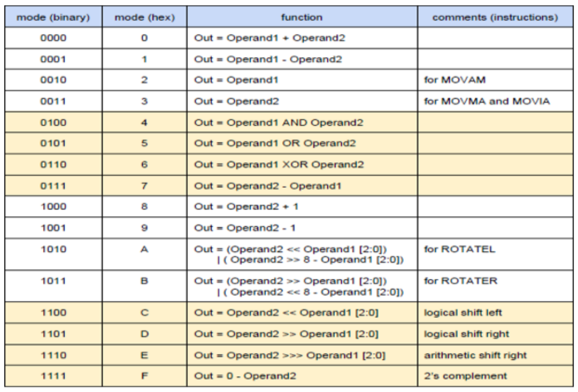
6) ALU

ALU is used to do the actual computation for the current instruction. This component is pure combinational. It has the following ports.

1. ALU operand 1 port (8-bit, input, denoted as ALU.Operand1): connected to Acc.
2. ALU operand 2 port (8-bit, input, denoted as ALU.Operand2): connected toMUX2.Out.
3. ALU enable port (1-bit, input, denoted as ALU.E): connected to -control logic.
4. ALU mode port (4-bit, input, denoted as ALU.Mode): connected to control logic.
5. Current flags port (4-bit, input, denoted as ALU.CFlags): connected to SR.
6. ALU output port (8 bit, output, denoted as ALU.Out): connected to DMem.DI.
7. ALU flags port (4-bit, output, denoted as ALU.Flags): the Z (zero), C (carry), S(sign), O (overflow) bits, from MSB to LSB, connected to status register.

The mode of ALU is listed in the following table:

Table I 5: Modes of ALU



mode (binary)	mode (hex)	function	comments (instructions)
0000	0	Out = Operand1 + Operand2	
0001	1	Out = Operand1 - Operand2	
0010	2	Out = Operand1	for MOVAM
0011	3	Out = Operand2	for MOVMA and MOVIA
0100	4	Out = Operand1 AND Operand2	
0101	5	Out = Operand1 OR Operand2	
0110	6	Out = Operand1 XOR Operand2	
0111	7	Out = Operand2 - Operand1	
1000	8	Out = Operand2 + 1	
1001	9	Out = Operand2 - 1	
1010	A	Out = (Operand2 << Operand1 [2:0]) | (Operand2 >> 8 - Operand1 [2:0])	for ROTATEL
1011	B	Out = (Operand2 >> Operand1 [2:0]) | (Operand2 << 8 - Operand1 [2:0])	for ROTATER
1100	C	Out = Operand2 << Operand1 [2:0]	logical shift left
1101	D	Out = Operand2 >> Operand1 [2:0]	logical shift right
1110	E	Out = Operand2 >>> Operand1 [2:0]	arithmetic shift right
1111	F	Out = 0 - Operand2	2's complement

7) MUX2

MUX2 is used to choose the source for operand 2 of ALU. If the current instruction is M type, operand 2 of ALU comes from data memory; if the current instruction is I type, operand 2 of ALU comes from the instruction, i.e., IR [7:0]. It has the following ports —

1. MUX2 input 1 port (8-bit, input, denoted as MUX2.In1): connected to IR [7:0].
2. MUX2 input 2 port (8-bit, input, denoted as MUX2.In2): connected to DR.
3. MUX2 selection port (1-bit, input, denoted as MUX2.Sel): connected to controllogic.
4. MUX2 output port (8-bit, output, denoted as MUX2.Out): connected to ALU. Operand2.

F. Control Unit Design

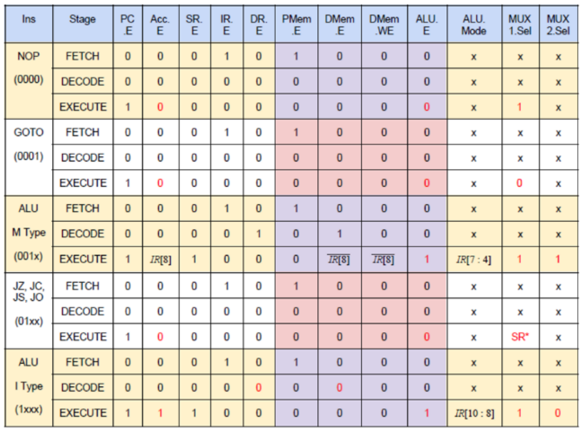
Control signal is derived from the current state and current instruction. The control logiccomponent is purely combinational. There are in total 12 control signals, listed as follows-

1. PC.E**:** enable port of program counter (PC);
2. Acc.E**:** enable port of accumulator (Acc);
3. SR.E: enable port of status register (SR)**;**
4. IR.E: enable port of instruction register (IR);
5. DR.E**:** enable port of data register (DR);
6. PMem.E**:** enable port of program memory (PMem);
7. DMem.E**:** enable port of data memory (DMem);

8. DMem.WE**:** write enable port of data memory (DMem);
9. ALU.E**:** enable port of ALU;
10. ALU.Mode**:** mode selection port of ALU;
11. MUX1.Sel: selection port of MUX1;
12. MUX2.Sel: selection port of MUX2;

The following table documents the detail of how these control signals are generatedimportant signals are marked in red:

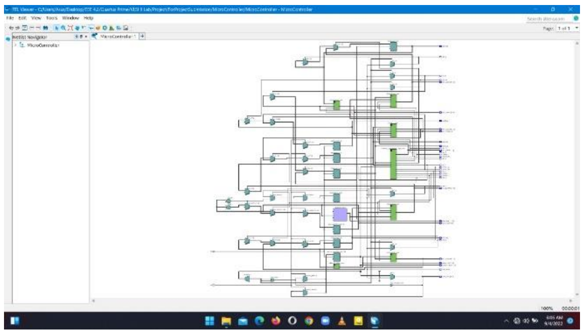
TABLE I 6: Control Signal



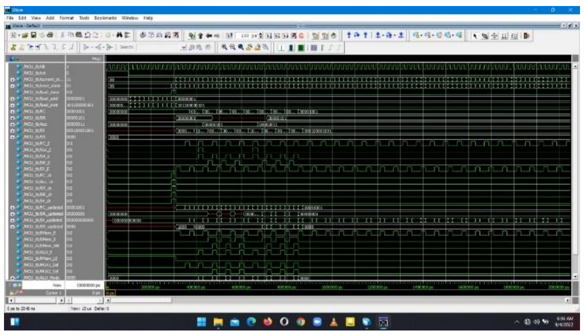
Ins	Stage	PC .E	Acc. E	SR. E	IR. E	DR. E	PMem .E	DMem .E	DMem .WE	ALU. E	ALU. Mode	MUX 1.Sel	MUX 2.Sel
NOP	FETCH	0	0	0	1	0	1	0	0	0	x	x	x
(0000)	DECODE	0	0	0	0	0	0	0	0	0	x	x	x
	EXECUTE	1	0	0	0	0	0	0	0	0	x	1	x
GOTO	FETCH	0	0	0	1	0	1	0	0	0	x	x	x
(0001)	DECODE	0	0	0	0	0	0	0	0	0	x	x	x
	EXECUTE	1	0	0	0	0	0	0	0	0	x	0	x
ALU	FETCH	0	0	0	1	0	1	0	0	0	x	x	x
M Type	DECODE	0	0	0	0	1	0	1	0	0	x	x	x
(001x)	EXECUTE	1	$IR[8]$	1	0	0	0	$\overline{IR}[8]$	$\overline{IR}[8]$	1	$IR[7:4]$	1	1
JZ, JC, JS, JO	FETCH	0	0	0	1	0	1	0	0	0	x	x	x
(01xx)	DECODE	0	0	0	0	0	0	0	0	0	x	x	x
	EXECUTE	1	0	0	0	0	0	0	0	0	x	SR*	x
ALU	FETCH	0	0	0	1	0	1	0	0	0	x	x	x
I Type	DECODE	0	0	0	0	0	0	0	0	0	x	x	x
(1xxx)	EXECUTE	1	1	1	0	0	0	0	0	1	$IR[10:8]$	1	0

II. RTL (Netlist) and Test Bench

RTL view of the netlist is given below:



Test bench output of the netlist of microcontroller:



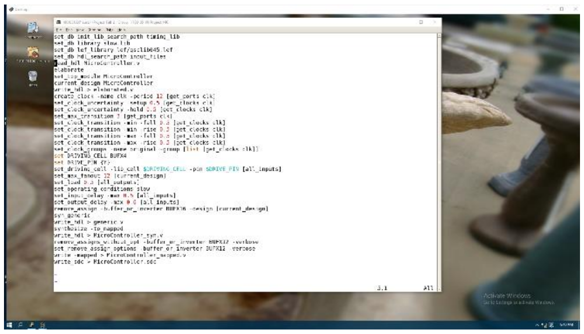
III. Synthesis Result

Our group has following synthesis constraints:

TABLE II 1: S synthesis Constraints

Parameter	**Value**
Clock Frequency (MHz)	83.33
Maximum Transition	3
Driving Cell	BUFX4
Operating Condition	Slow
Output Delay (ms)	0.6
Max Fanout	12

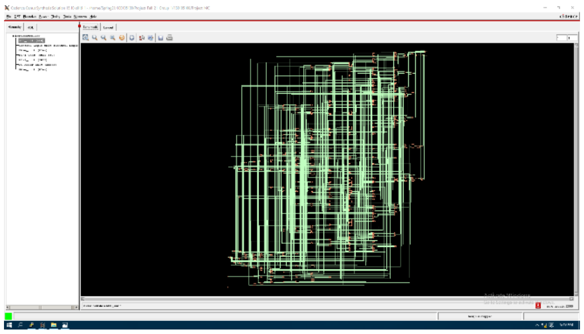
[1] Screenshot of VIM editor



```
set_db init_lib_search_path timing_lib
set_db library slow.lib
set_db lef_library lef/gsclib045.lef
set_db hdl_search_path input_files
load_hdl MicroController.v
elaborate
set_top_module MicroController
current_design MicroController
write_hdl > elaborated.v
create_clock -name clk -period 12 [get_ports clk]
set_clock_uncertainty -setup 0.5 [get_clocks clk]
set_clock_uncertainty -hold 0.5 [get_clocks clk]
set_max_transition 3 [get_ports clk]
set_clock_transition -min -fall 0.3 [get_clocks clk]
set_clock_transition -min -rise 0.3 [get_clocks clk]
set_clock_transition -max -fall 0.3 [get_clocks clk]
set_clock_transition -max -rise 0.3 [get_clocks clk]
set_clock_groups -name original -group [list [get_clocks clk]]
set DRIVING_CELL BUFX4
set DRIVE_PIN {Y}
set_driving_cell -lib_cell $DRIVING_CELL -pin $DRIVE_PIN [all_inputs]
set_max_fanout 12 [current_design]
set_load 0.5 [all_outputs]
set_operating_conditions slow
set_input_delay -max 0.5 [all_inputs]
set_output_delay -max 0.6 [all_inputs]
remove_assign -buffer_or_inverter BUFX16 -design [current_design]
syn_generic
write_hdl > generic.v
synthesize -to_mapped
write_hdl > MicroController_syn.v
remove_assigns_without_opt -buffer_or_inverter BUFX12 -verbose
set_remove_assign_options -buffer_or_inverter BUFX12 -verbose
write -mapped > MicroController_mapped.v
write_sdc > MicroController.sdc
```

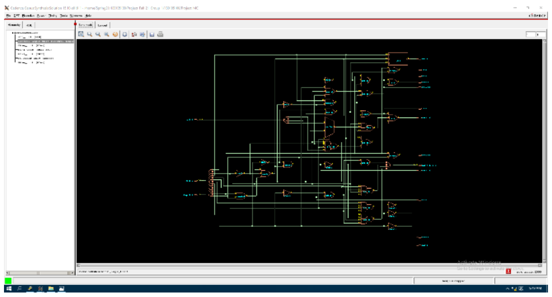
Synthesis is done using genus tool. Here are the screenshots while using VIM editor. We used design constraints given above.

[2] GUI Show

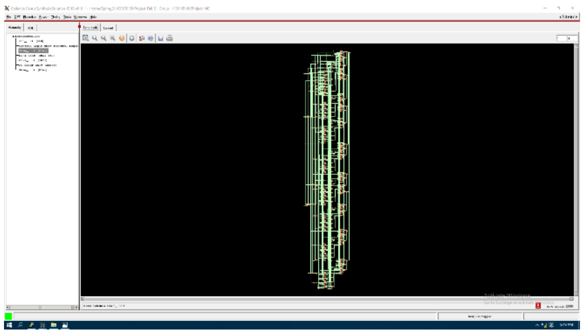


A screenshot of using gui_show command.

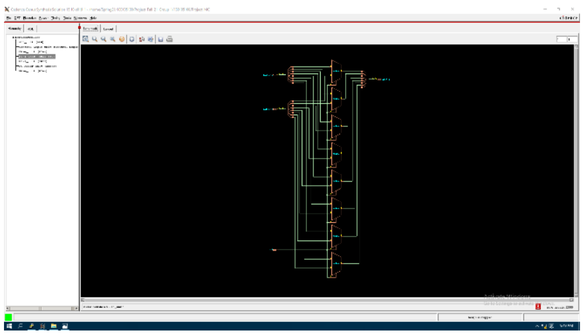
[3] Control Logic Unit



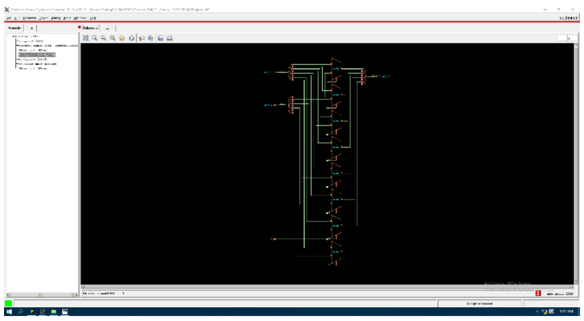
[4] Data Memory Unit



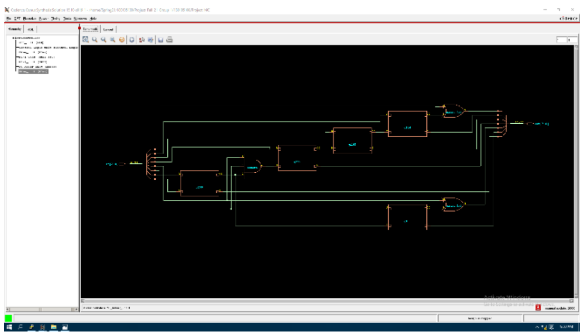
[5] MUX 1 Unit



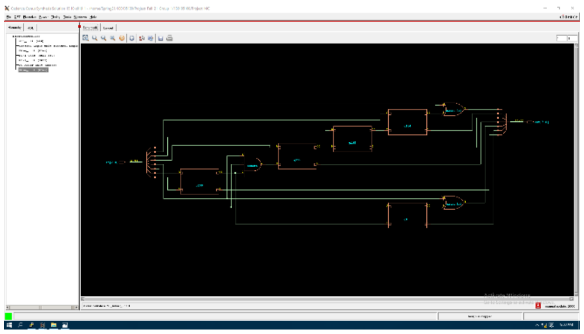
[6] MUX 2 Unit



[7] PC Adder Unit



[8] Program Memory Unit



IV. PNR OUTPUTS

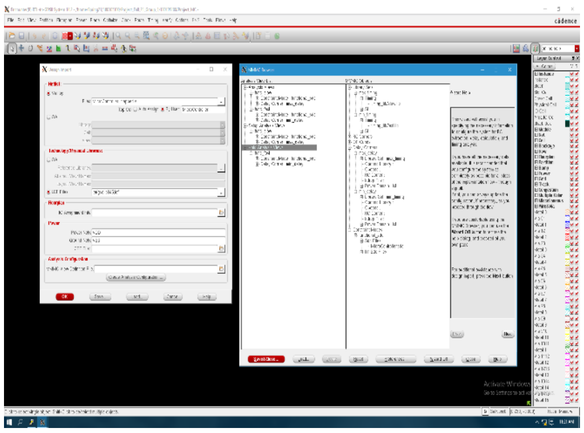
Our group had following design constraints:

Table III 1: Design Constraints

Parameter	Value
Distance (Die to Core)	11
Ring (W, D)	3,2
Stripe	3
Initial Density	40%

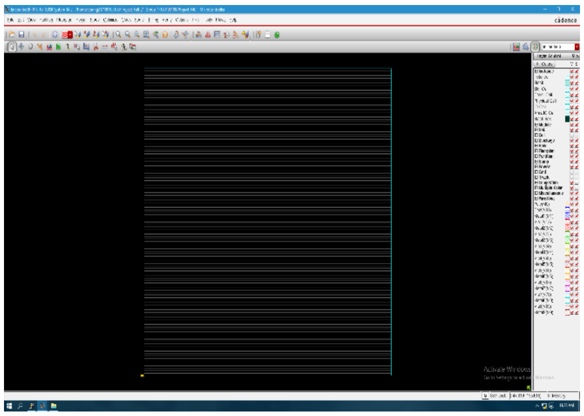
Below screen-shots are given with short description:

1. Importing Design and MMMC Browser Window



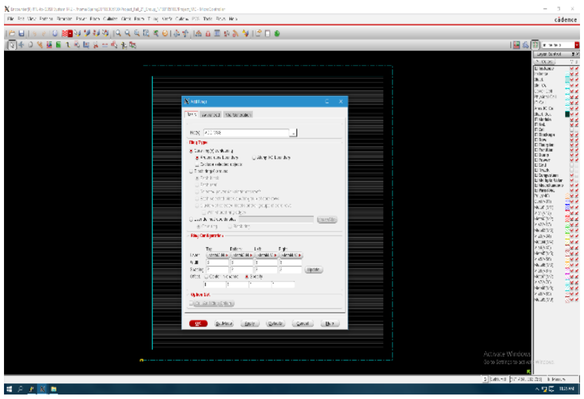
Here a screenshot MMMC window after all the inputs of analysis views are given.

2. Initial Floorplan



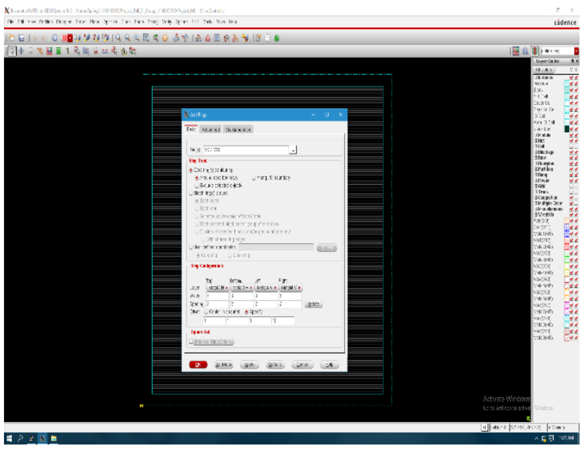
Initial design of the floorplan after given MMMC analysis view inputs.

3. Floorplan with Specification



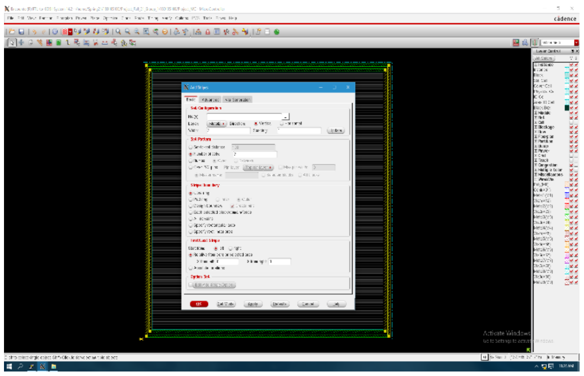
Floorplan of the design within given constraints.

4. Adding Ring



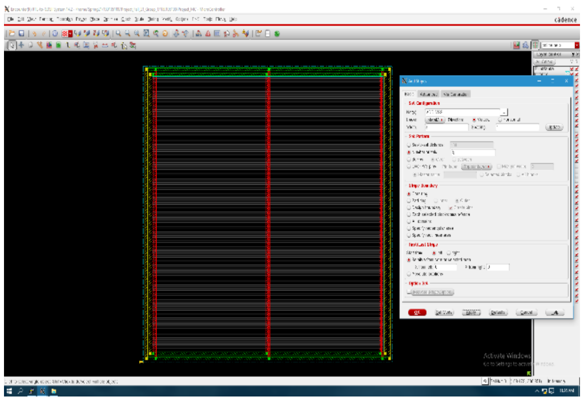
Adding ring in dye area.

5. Adding Stripes



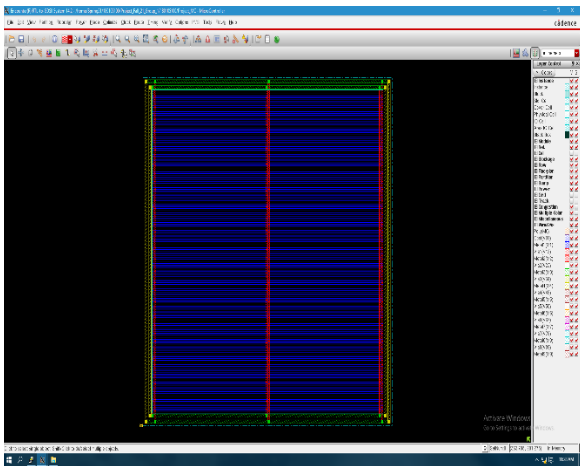
Here we are adding 3 stripes as given constraints.

6. After Adding Stripes



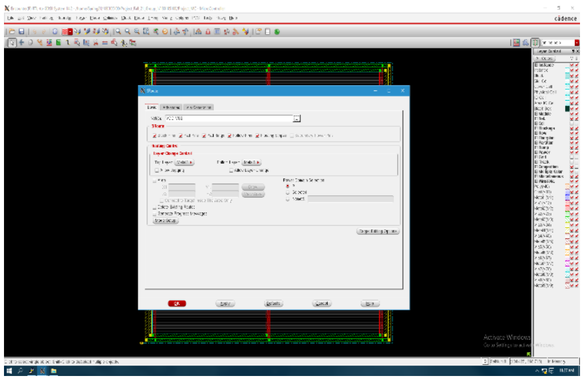
Adding 3 stripes.

7. Power Planning



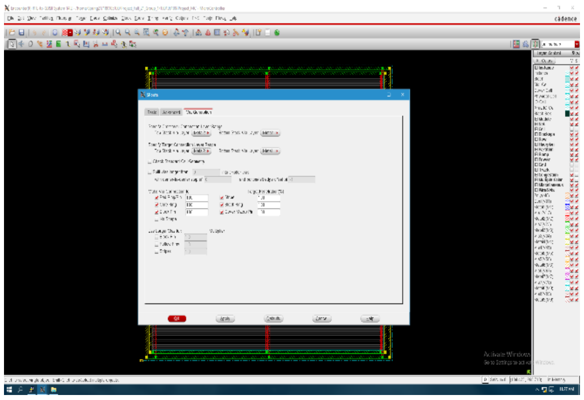
Here, we a final screenshot after power planning within given constraints.

8. SR Route



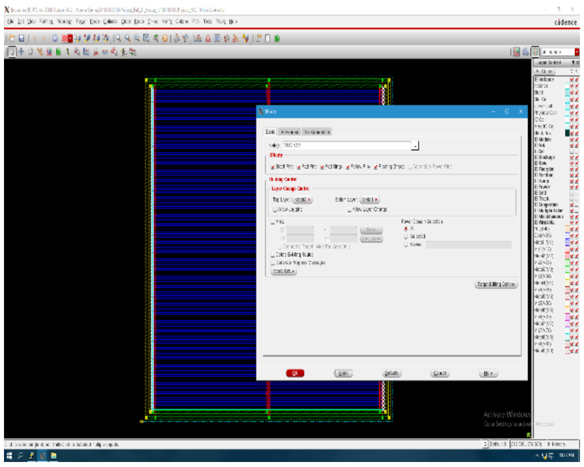
Starting SR routing here.

9. SR Route (Via Generation)



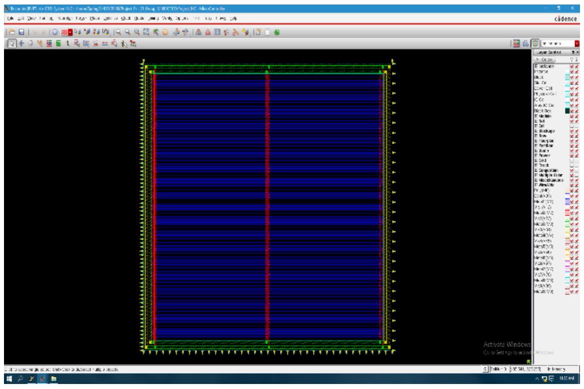
A screenshot of Via generation tab.

10. SR Route Final



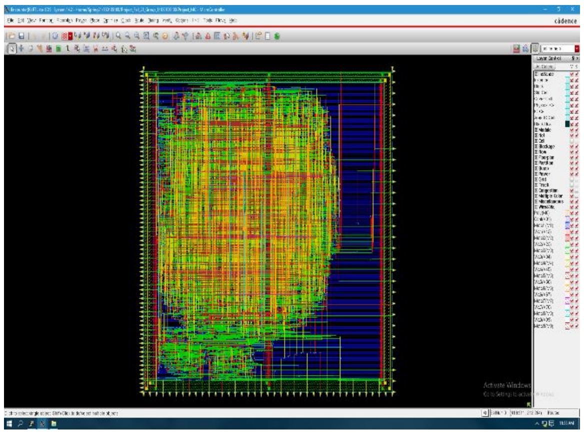
Final screenshot of routing window.

11. Pin Placement



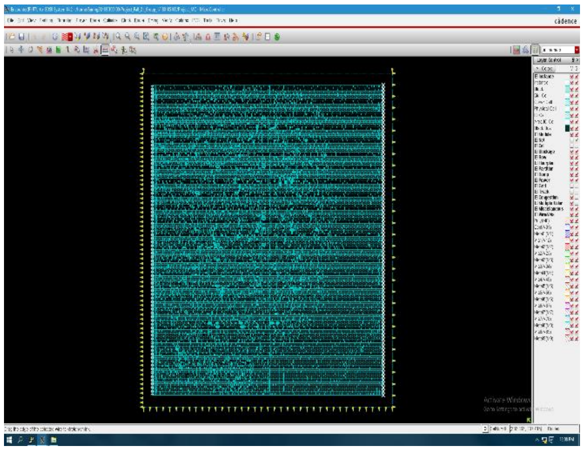
Necessary placement for the pin in all direction.

12. Placing Design



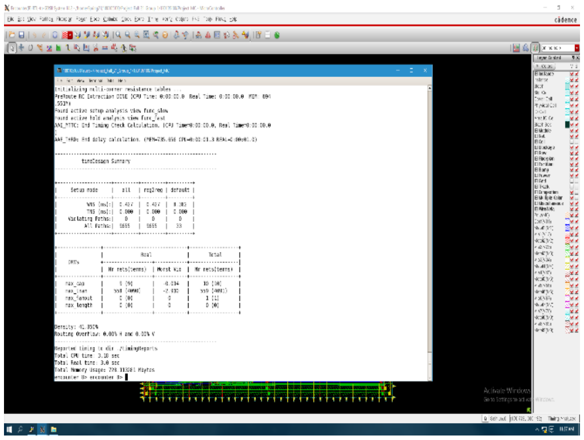
After placing the design initially.

13. Placement Without Net



Screenshot with net view turned off.

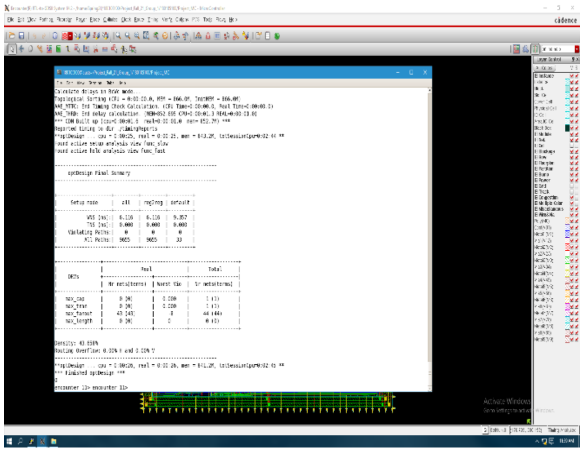
14. Time Design (Pre CTS)



```
Initializing multi-corner resistance tables ...
PreRoute RC Extraction DONE (CPU Time: 0:00:00.0  Real Time: 0:00:00.0  MEM: 694
.551M)
Found active setup analysis view func_slow
Found active hold analysis view func_fast
AAE_MTTC: End Timing Check Calculation. (CPU Time=0:00:00.0, Real Time=0:00:00.0
)
AAE_THRD: End delay calculation. (MEM=735.656 CPU=0:00:01.3 REAL=0:00:01.0)

------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.427  |  0.427  |  8.382  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  9655   |  9655   |   33    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      9 (9)       |   -0.014   |     10 (10)      |
|   max_tran     |    558 (4080)    |   -2.810   |    559 (4081)    |
|   max_fanout   |      0 (0)       |     0      |      1 (1)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 41.850%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 3.18 sec
Total Real time: 3.0 sec
Total Memory Usage: 728.113281 Mbytes
encounter 8> encounter 8>
```

STA report (detail given below) before CTS.

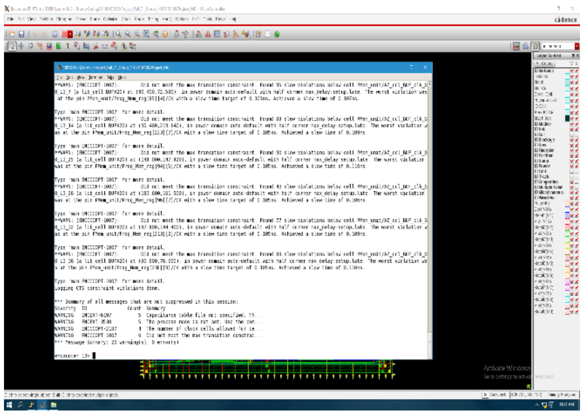
15. Optimizing Design (Pre CTS)



```
Calculate delays in BcWc mode...
Topological Sorting (CPU = 0:00:00.0, MEM = 866.0M, InitMEM = 866.0M)
AAE_MTTC: End Timing Check Calculation. (CPU Time=0:00:00.0, Real Time=0:00:00.0)
AAE_THRD: End delay calculation. (MEM=852.695 CPU=0:00:01.3 REAL=0:00:01.0)
*** CDM Built up (cpu=0:00:01.6  real=0:00:01.0  mem= 852.7M) ***
Reported timing to dir ./timingReports
**optDesign ... cpu = 0:00:25, real = 0:00:25, mem = 843.2M, totSessionCpu=0:02:44 **
Found active setup analysis view func_slow
Found active hold analysis view func_fast

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

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  6.116  |  6.116  |  9.357  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  9655   |  9655   |   33    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      1 (1)       |
|   max_tran     |      0 (0)       |   0.000    |      1 (1)       |
|   max_fanout   |     43 (43)      |     -8     |     44 (44)      |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 43.658%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
**optDesign ... cpu = 0:00:26, real = 0:00:26, mem = 841.2M, totSessionCpu=0:02:45 **
*** Finished optDesign ***
0
encounter 11> encounter 11>
```

Optimized STA report (detail given below) before CTS.

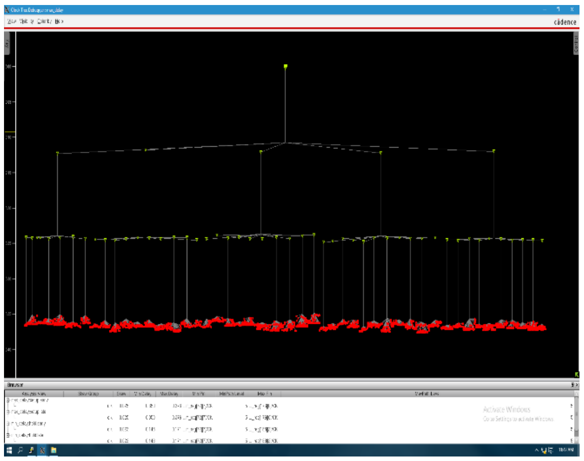
16. Creating Clock



```
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 95 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_7 (a lib_cell BUFX20) at (93.400,72.580), in power domain auto-default with half corner max_delay:setup.late. The worst violation was
 at the pin PMem_unit/Prog_Mem_reg[85][4]/CK with a slew time target of 0.105ns. Achieved a slew time of 0.107ns.

Type 'man ENCCCOPT-1007' for more detail.
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 83 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_14 (a lib_cell BUFX20) at (93.400,219.640), in power domain auto-default with half corner max_delay:setup.late. The worst violation w
as at the pin PMem_unit/Prog_Mem_reg[223][7]/CK with a slew time target of 0.106ns. Achieved a slew time of 0.109ns.

Type 'man ENCCCOPT-1007' for more detail.
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 93 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_25 (a lib_cell BUFX20) at (148.800,147.820), in power domain auto-default with half corner max_delay:setup.late. The worst violation
was at the pin PMem_unit/Prog_Mem_reg[94][5]/CK with a slew time target of 0.106ns. Achieved a slew time of 0.110ns.

Type 'man ENCCCOPT-1007' for more detail.
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 43 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_26 (a lib_cell BUFX20) at (183.600,161.500), in power domain auto-default with half corner max_delay:setup.late. The worst violation
was at the pin PMem_unit/Prog_Mem_reg[96][7]/CK with a slew time target of 0.106ns. Achieved a slew time of 0.108ns.

Type 'man ENCCCOPT-1007' for more detail.
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 77 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_32 (a lib_cell BUFX20) at (87.800,144.400), in power domain auto-default with half corner max_delay:setup.late. The worst violation w
as at the pin PMem_unit/Prog_Mem_reg[213][2]/CK with a slew time target of 0.106ns. Achieved a slew time of 0.109ns.

Type 'man ENCCCOPT-1007' for more detail.
**WARN: (ENCCCOPT-1007):     Did not meet the max transition constraint. Found 83 slew violations below cell PMem_unit/AZ_ccl_BUF_clk_G
0_L3_36 (a lib_cell BUFX20) at (93.800,76.000), in power domain auto-default with half corner max_delay:setup.late. The worst violation wa
s at the pin PMem_unit/Prog_Mem_reg[238][0]/CK with a slew time target of 0.106ns. Achieved a slew time of 0.110ns.

Type 'man ENCCCOPT-1007' for more detail.
Logging CTS constraint violations done.

*** Summary of all messages that are not suppressed in this session:
Severity  ID               Count  Summary
WARNING   ENCEXT-6197          5  Capacitance table file not specified. Th...
WARNING   ENCEXT-3530          5  The process node is not set. Use the com...
WARNING   ENCCCOPT-2107        4  The number of clock cells allowed for ce...
WARNING   ENCCCOPT-1007        6  Did not meet the max transition constrai...
*** Message Summary: 20 warning(s), 0 error(s)

encounter 13>
```

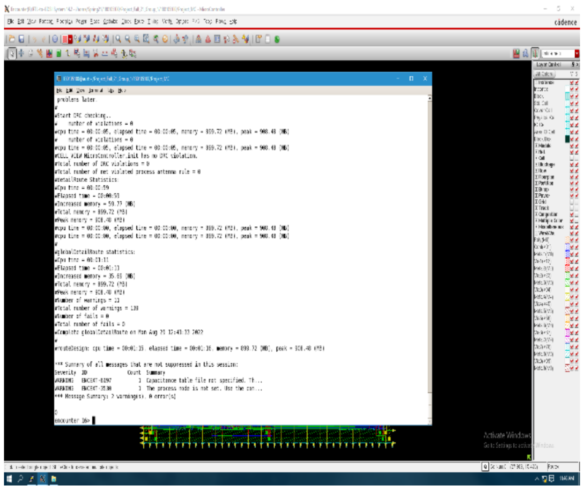
We are building clock for the design.

17. Using CTS



Here we are building a clock tree.

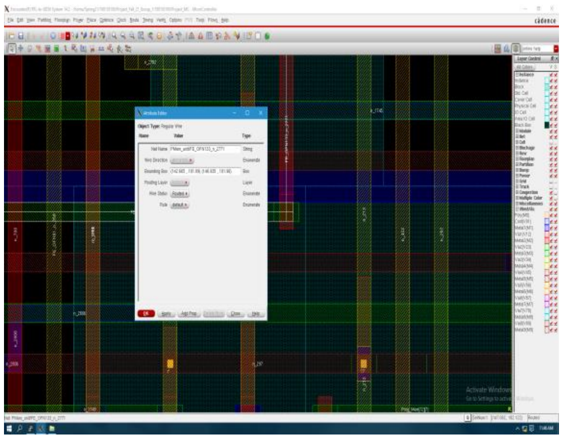
18. Routing Window



```
 problems later.
#
#Start DRC checking..
#    number of violations = 0
#cpu time = 00:00:05, elapsed time = 00:00:05, memory = 899.72 (MB), peak = 908.48 (MB)
#    number of violations = 0
#cpu time = 00:00:05, elapsed time = 00:00:05, memory = 899.72 (MB), peak = 908.48 (MB)
#CELL_VIEW MicroController,init has no DRC violation.
#Total number of DRC violations = 0
#Total number of net violated process antenna rule = 0
#detailRoute Statistics:
#Cpu time = 00:00:59
#Elapsed time = 00:00:59
#Increased memory = 59.77 (MB)
#Total memory = 899.72 (MB)
#Peak memory = 908.48 (MB)
#cpu time = 00:00:00, elapsed time = 00:00:00, memory = 899.72 (MB), peak = 908.48 (MB)
#cpu time = 00:00:00, elapsed time = 00:00:00, memory = 899.72 (MB), peak = 908.48 (MB)
#
#globalDetailRoute statistics:
#Cpu time = 00:01:11
#Elapsed time = 00:01:11
#Increased memory = 35.65 (MB)
#Total memory = 899.72 (MB)
#Peak memory = 908.48 (MB)
#Number of warnings = 11
#Total number of warnings = 139
#Number of fails = 0
#Total number of fails = 0
#Complete globalDetailRoute on Mon Aug 29 12:43:33 2022
#
#routeDesign: cpu time = 00:01:15, elapsed time = 00:01:16, memory = 893.72 (MB), peak = 908.48 (MB)

*** Summary of all messages that are not suppressed in this session:
Severity  ID               Count  Summary
WARNING   ENCEXT-6197          1  Capacitance table file not specified. Th...
WARNING   ENCEXT-3530          1  The process node is not set. Use the con...
*** Message Summary: 2 warning(s), 0 error(s)

0
encounter 16>
```

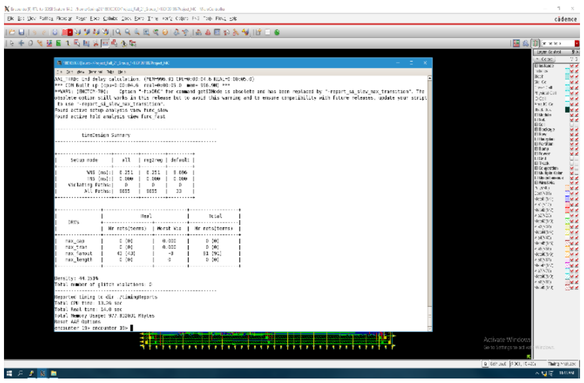
Here we are routing the design.

19. Routing Attribute



Here we see the wire status is routed.

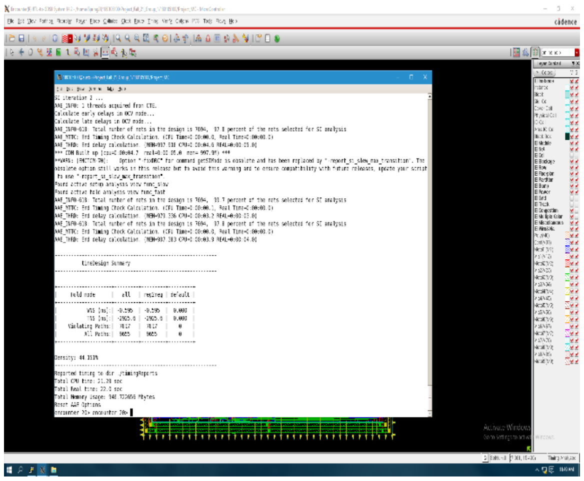
20. Time Design (Setup)



```
AAE_THRD: End delay calculation. (MEM=996.91 CPU=0:00:04.6 REAL=0:00:05.0)
*** CDM Built up (cpu=0:00:04.6  real=0:00:05.0  mem= 996.9M) ***
**WARN: (ENCTCM-70):    Option "-fixDRC" for command getSIMode is obsolete and has been replaced by "-report_si_slew_max_transition". The
obsolete option still works in this release but to avoid this warning and to ensure compatibility with future releases, update your script
 to use "-report_si_slew_max_transition".
Found active setup analysis view func_slow
Found active hold analysis view func_fast

------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  0.251  |  0.251  |  0.696  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  9855   |  9855   |   33    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              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   |     43 (43)      |     -8     |     91 (91)      |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 44.351%
Total number of glitch violations: 0
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 13.76 sec
Total Real time: 14.0 sec
Total Memory Usage: 977.832031 Mbytes
Reset AAE Options
encounter 19> encounter 19>
```

Timing design for setup mode.

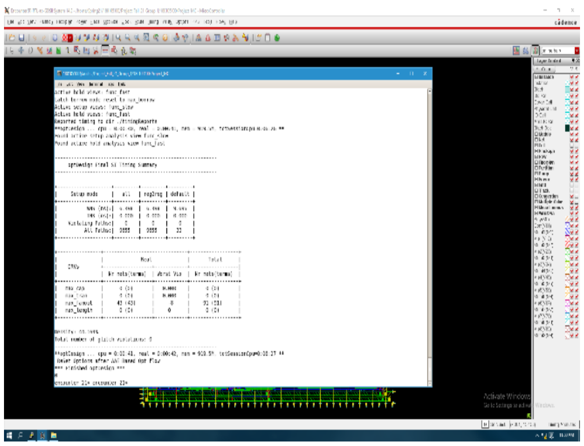
21. Time Design (Hold)



```
SI iteration 2 ...
AAE_INFO: 1 threads acquired from CTE.
Calculate early delays in OCV mode...
Calculate late delays in OCV mode...
AAE_INFO-618: Total number of nets in the design is 7694,  97.8 percent of the nets selected for SI analysis
AAE_NTTC: End Timing Check Calculation. (CPU Time=0:00:00.0, Real Time=0:00:00.0)
AAE_THRD: End delay calculation. (MEM=937.918 CPU=0:00:04.6 REAL=0:00:05.0)
*** CDM Built up (cpu=0:00:04.7  real=0:00:05.0  mem= 997.9M) ***
**WARN: (ENCTCM-70):    Option "-fixDRC" for command getSIMode is obsolete and has been replaced by "-report_si_slew_max_transition". The
obsolete option still works in this release but to avoid this warning and to ensure compatibility with future releases, update your script
 to use "-report_si_slew_max_transition".
Found active setup analysis view func_slow
Found active hold analysis view func_fast
AAE_INFO-618: Total number of nets in the design is 7694,  93.7 percent of the nets selected for SI analysis
AAE_NTTC: End Timing Check Calculation. (CPU Time=0:00:00.1, Real Time=0:00:00.0)
AAE_THRD: End delay calculation. (MEM=973.336 CPU=0:00:03.2 REAL=0:00:03.0)
AAE_INFO-618: Total number of nets in the design is 7694,  97.8 percent of the nets selected for SI analysis
AAE_NTTC: End Timing Check Calculation. (CPU Time=0:00:00.0, Real Time=0:00:00.0)
AAE_THRD: End delay calculation. (MEM=937.383 CPU=0:00:03.9 REAL=0:00:04.0)

------------------------------------------------------------
         timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| -0.595  | -0.595  |  0.000  |
|           TNS (ns):| -2925.6 | -2925.6 |  0.000  |
|    Violating Paths:|  7817   |  7817   |    0    |
|          All Paths:|  9855   |  9855   |    0    |
+--------------------+---------+---------+---------+

Density: 44.351%
------------------------------------------------------------
Reported timing to dir ./timingReports
Total CPU time: 21.28 sec
Total Real time: 22.0 sec
Total Memory Usage: 948.733656 Mbytes
Reset AAE Options
encounter 20> encounter 20>
```

Timing design for hold mode.

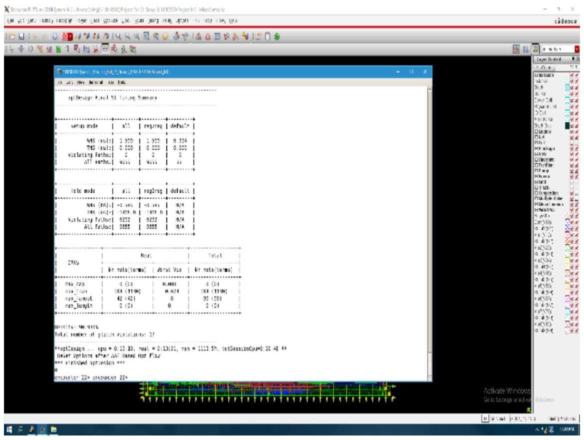
22. Optimizing Design



```
Active hold views: func_fast
Latch borrow mode reset to max_borrow
Active setup views: func_slow
Active hold views: func_fast
Reported timing to dir ./timingReports
**optDesign ... cpu = [illegible], real = [illegible], mem = [illegible], totSessionCpu=[illegible] **
Found active setup analysis view func_slow
Found active hold analysis view func_fast

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

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  [illegible]  |  [illegible]  |  [illegible]  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  9855   |  9855   |   33    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      4 (5)       |   0.000    |      4 (5)       |
|   max_tran     |      4 (5)       |   0.000    |      4 (5)       |
|   max_fanout   |     42 (42)      |     8      |     93 (93)      |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: [illegible]
Total number of glitch violations: 0
------------------------------------------------------------
**optDesign ... cpu = 0:00:41, real = 0:00:42, mem = 910.5M, totSessionCpu=0:08:37 **
 Reset Options after AAE Based Opt flow
*** Finished optDesign ***
0
encounter 21> encounter 21>
```

Optimizing design for setup mode.

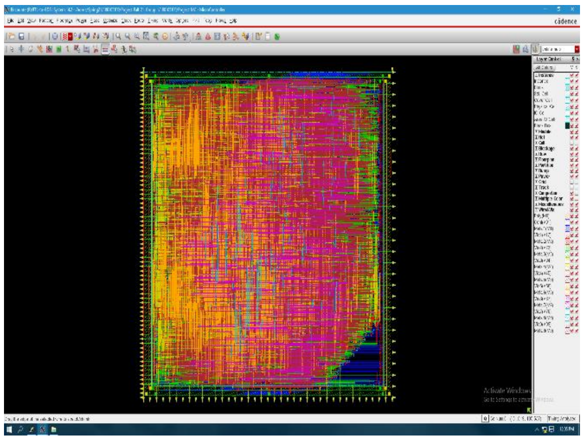
23. Optimizing Design (Hold)



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

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):|  1.959  |  1.959  |  0.334  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  9855   |  9855   |   33    |
+--------------------+---------+---------+---------+

+--------------------+---------+---------+---------+
|     Hold mode      |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| [illegible] | [illegible] |   N/A   |
|           TNS (ns):| -1405.8 | -1405.8 |   N/A   |
|    Violating Paths:|  6232   |  6232   |   N/A   |
|          All Paths:|  9855   |  9855   |   N/A   |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------|
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      4 (5)       |   0.000    |      4 (5)       |
|   max_tran     |   184 (1190)     |  -0.074    |   184 (1190)     |
|   max_fanout   |     42 (42)      |     8      |     93 (93)      |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: [illegible]
Total number of glitch violations: 37
------------------------------------------------------------
**optDesign ... cpu = 0:13:19, real = 0:13:21, mem = 1113.5M, totSessionCpu=0:30:40 **
 Reset Options after AAE Based Opt flow
*** Finished optDesign ***
0
encounter 22> encounter 22>
```

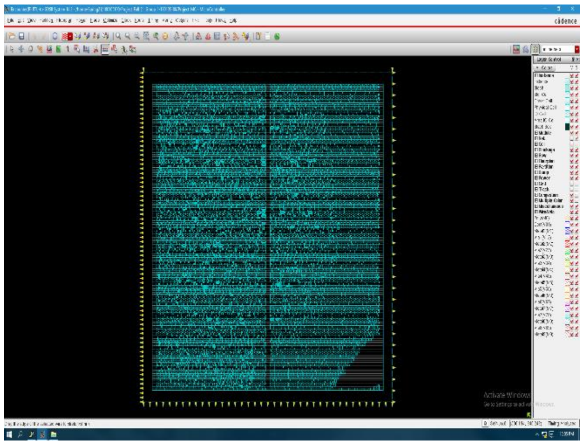
Optimizing design for hold mode.

24. Optimized Route



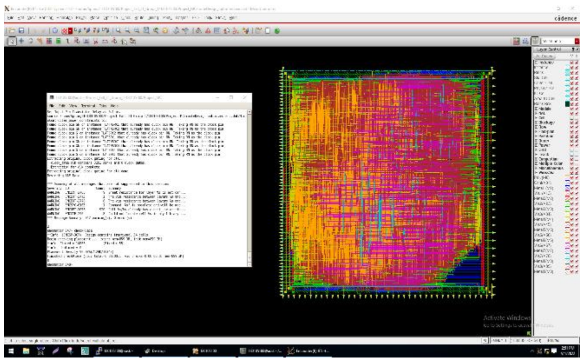
This is after optimizing. A gap is present in bottom right corner.

25. Optimized Route (Without Net)



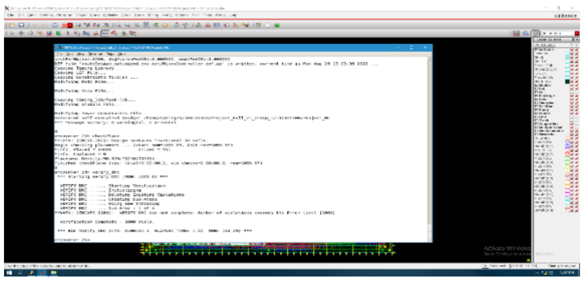
Here a screenshot without nets. Note the gap in bottom right corner.

26. Initial Check Place



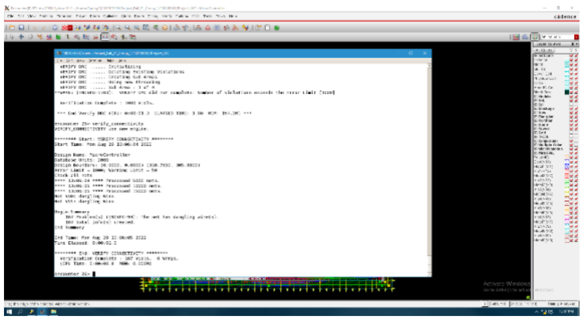
Initially, we had 90.93% density with 14886 cells.

27. Initial DRC Error



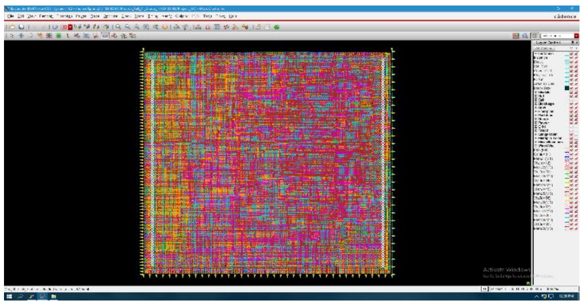
Initially, we had 1000 violations.

28. Initial Connectivity Error



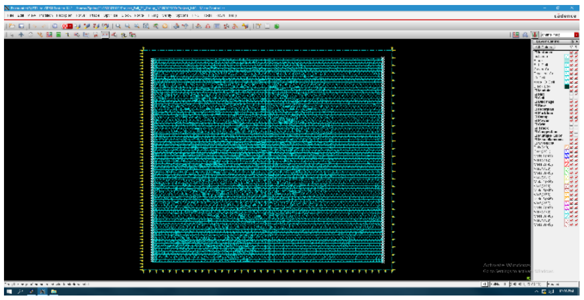
Initially. We had 167 violations and 0 Warning

29. Appling Cell & Metal Filler



In blank places we have to add cell and metal filler.

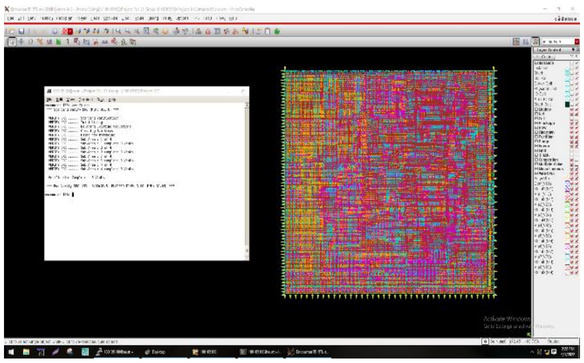
30. Cell And Filler (Without Net)



After turning net view off.

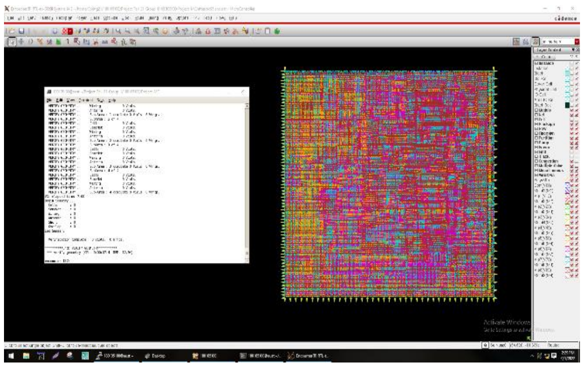
V. Physical Verification

1. DRC Check



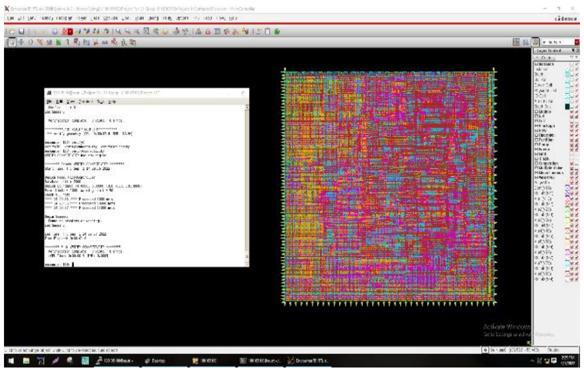
We have 0 violation and 0 warning.

2. Geometry Check



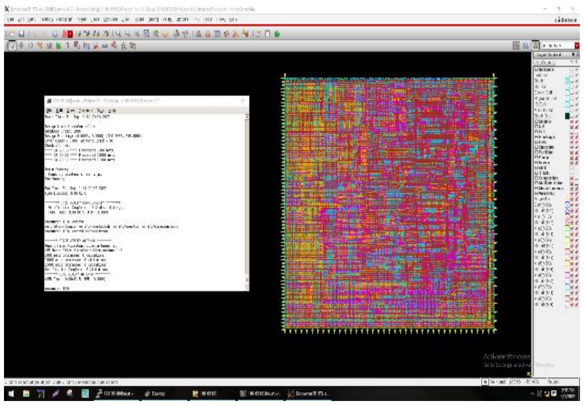
We have 0 violation and 0 warning.

3. Connectivity Check



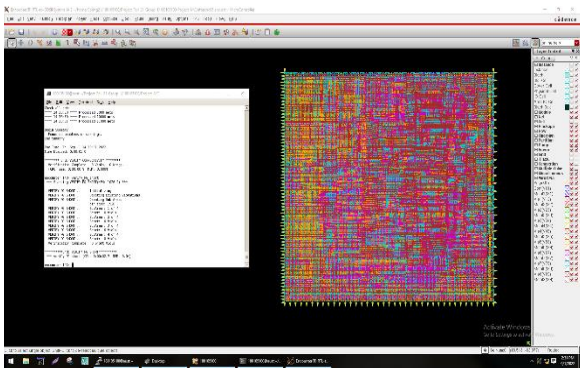
We have 0 violation and 0 warning.

4. Antenna Check



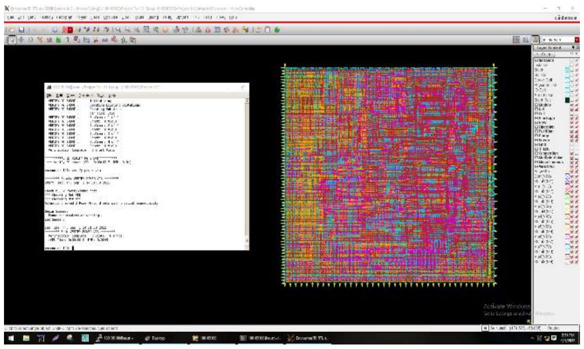
We have 0 violations and 0 warning.

5. PG Short Check



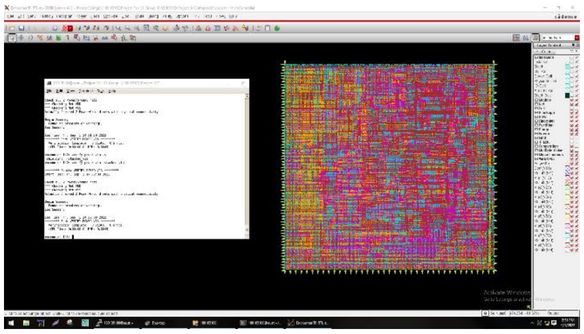
We have 0 violation and 0 warning.

6. Power Via Check



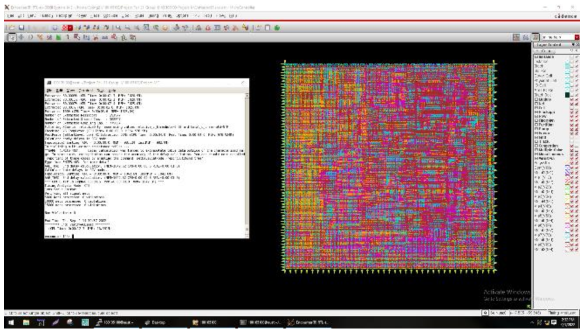
We have 0 violation and 0 warning.

7. Power Via Stacked Check



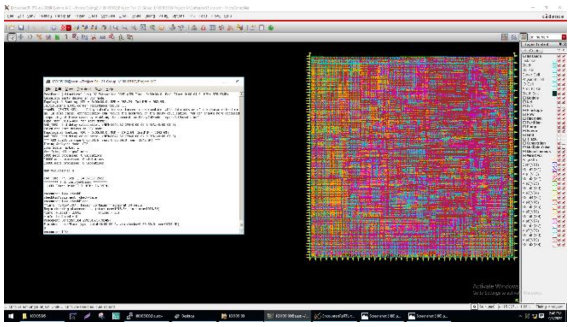
We have 0 violation and 0 warning.

8. AC Limit Check



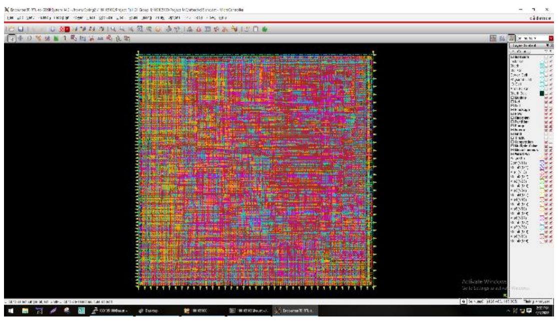
We have 0 violation and 0 warning.

9. Final Check Place



In final design we got density of 104.25% with 23591 cells.

VI. Final Design



VII. STA Report

Tables for STA report:

TABLE VI 1: Time Design (Setup Mode) (Pre CTS)

Setup Mode	All	Reg2Reg	Default
WNS (ns)	0.427	0.427	8.382
TNS (ns)	0.000	0.000	0.000
Violating Paths	0	0	0
All Paths	9655	9655	0

TABLE VI 2: Time Design DRV (Pre CTS)

DRV	Real		Total
	Nr Nets	Worst Vio	Nr Nets
max_cap	9 (9)	-0.014	10 (10)
max_tran	558 (4090)	-2.910	559 (4091)
max_fanout	0 (0)	0	1 (1)
max_lenght	0 (0)	0	0 (0)

TABLE VI 3: Optimized Time Design (Setup Mode) (Pre CTS)

Setup Mode	All	Reg2Reg	Default
WNS (ns)	6.116	6.116	9.375
TNS (ns)	0.000	0.000	0.000
Violating Paths	0	0	0
All Paths	9655	9655	0

TABLE VI 4: Optimized Time Design DRV (Pre CTS)

DRV	Real		Total
	Nr Nets	Worst Vio	Nr Nets
max_cap	0 (0)	0.000	1 (1)
max_tran	0 (0)	0.000	1 (1)
max_fanout	43(43)	-8	41 (41)
max_lenght	0 (0)	0	0 (0)

TABLE VI 5: Time Design (Setup Mode) (After CTS)

Setup Mode	All	Reg2Reg	Default
WNS (ns)	6.251	6.251	9.696
TNS (ns)	0.000	0.000	0.000
Violating Paths	0	0	0
All Paths	9655	9655	33

TABLE VI 6: Time Design (Hold Mode) (After CTS)

Hold Mode	All	Reg2Reg	Default
WNS (ns)	-0.595	-0.595	0.000
TNS (ns)	-2925.6	-2925.6	0.000
Violating Paths	7817	7817	0
All Paths	9655	9655	0

TABLE VI 7: Time Design DRV (After CTS)

DRV	Real		Total
	Nr Nets	Worst Vio	Nr Nets
max_cap	0 (0)	0.000	1 (1)
max_tran	0 (0)	0.000	1 (1)
max_fanout	43(43)	-8	41 (41)
max_lenght	0 (0)	0	0 (0)

TABLE VI 8: Optimized Time Design (Setup Mode) (After CTS)

Setup Mode	All	Reg2Reg	Default
WNS (ns)	6.498	6.498	9.696
TNS (ns)	0.000	0.000	0.000
Violating Paths	0	0	0
All Paths	9655	9655	33

TABLE VI 9: Optimized Time Design (Hold Mode) (After CTS)

Hold Mode	All	Reg2Reg	Default
WNS (ns)	-0.595	-0.595	N/A
TNS (ns)	-1406.0	-1406.0	N/A
Violating Paths	6292	6292	N/A
All Paths	9655	9655	N/A

TABLE VI 10: Optimized Time Design DRV (After CTS)

DRV	Real		Total
	Nr Nets	**Worst Vio**	**Nr Nets**
max_cap	0 (0)	0.000	0 (0)
max_tran	184(0)	-0.624	184 (1130)
max_fanout	42(42)	-8	90 (90)
max_lenght	0 (0)	0	0 (0)

VIII. Final Result Comparison Table

Here is a table provided for comparison and details.

TABLE VII 1: Table for Various Parameters and Violations

Parameter	Initial	Final
Density	90.93%	104.25%
Cells Placed	14886	23591
DRC Violations	1000 (117 After using globalnetconnect Command)	0
Geometry Violations	0	0
Connectivity Violations	167	0
Process Antenna Violations	0	0
PG Short Violations	0	0
Power Via Violations	0	0
Power Via Stacked Violations	0	0

IX. Future Aspects

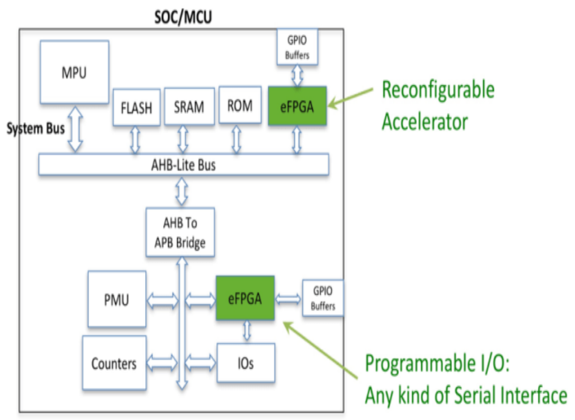
Microcontrollers today have an issue and an opportunity.

The opportunity for microcontrollers is around the “glue” FPGA chips used by designers for decades. If these FPGA chips are integrated instead of being standalone, customers can significantly improve the cost, speed and power consumption of MCUs. This is a huge value proposition.

While not a new technology, embedded FPGA (eFPGA) is finally at the stage where it is ready to go mainstream with multiple suppliers, design wins in progress and proven silicon. Customers can leverage this technology in a number of ways, such as:

A reconfigurable accelerator that can directly access on-chip buses, cache and I/O. One mask can cover multiple needs and customers can achieve higher performance.

A reconfigurable I/O that can implement any serial I/O and can push low-level processing to the I/O block. This frees up the processor and improves responsiveness and battery life.



Microcontrollers often have dozens of variations to accommodate customer requirements for different combinations of serial I/Os: UART, USART, I^{2}C, SPI and more.

With eFPGA, serial I/Os can now be programmed as needed. This enables MCU companies to save on mask costs and validation and provides customers with exactly the serial I/O they want, even variations on the standard versions.

Initially, customers may not even realize they are using eFPGA because the manufacturer can program the eFPGA differently for each SKU. The next step is to use the eFPGA to process I/O so as to offload the MPU, improve performance and even lower power.

We can see that using eFPGA to implement some simple, repetitive DSP functions can reduce power compared to implementing the same functions in the processor. The result is longer battery life. Microcontrollers today sometimes have hardwired to offload the processors to improve performance. Examples of this are crypto-engines such as Advanced Encryption Standard (AES.)

Microcontroller companies can also use embedded FPGA to implement various accelerator functions (such as AES, FFT, JPEG encode, SHA) with performance 30-130 times faster than an ARM processor. Another option is to connect the eFPGA reconfigurable accelerator directly to GPIO: 8 bits, 16 bits, 32 bits or 64 bits This enables much more internal observation of microcontroller activity when the customer is trying to understand why their c-code/RTL combination is not getting the results they were expecting. Integration of eFPGA into microcontrollers is happening today now that this technology is available from multiple suppliers in 180nm to 16nm process nodes. This will not only benefit manufacturers with lower engineering costs and faster time to market, but will also benefit microcontroller users with greater performance and flexibility in optimizing their systems.

X. Conclusion

Micro-controller circuit can be a very difficult circuit when designing. When using Genus and Encounter tool we need to be careful and save our progress every so often because these tools are prone to crashing thus resulting in losing our progress. In order to not run into such crashes, we have to follow instructions from our manual precisely.

It is absolutely necessary when designing to keep in mind that a simple miscalculation can result in a several thousand violations. In order to avoid such violations, we have to careful in in every step of the process. For example, initially we had 1000 violations. After using globalnetconnect command we had 117 violations. We removed these 117 violations manually.

Various design checks should be performed at regular intervals when removing such violations manually to avoid creating a new one or more dangerously creating one or several geometry violations.

Acknowledgment

We would like to express our eternal gratitude to our lab instructors and lab assistant for helping us with this project.

References

[1] VLSI II Lab 5 Manual (Synthesis Using Genus Synthesis Solution), EEE, AUST.
[2] VLSI II Lab 6 Manual (Physical Design Using Cadence Encounter (Part 1), EEE, AUST.
[3] VLSI II Lab 7 Manual (Physical Design Using Cadence Encounter (Part 2), EEE, AUST.
[4] VLSI II Lab 7B Manual (Static Timing Analysis Using Encounter Digital Implementation System), EEE, AUST.
[5] VLSI II Lab 8 Manual (Physical Verification and Power Analysis Using Encounter Implementation System), EEE, AUST.
[6] The Future of Microcontrollers - EETimes
[7] Verilog code for Microcontroller (Part 1 - Specification) - FPGA4student.com
[8] Verilog code for microcontroller (Part-2- Design) - FPGA4student.com
[9] Verilog code for Microcontroller (Part 3- Verilog code) - FPGA4student.com