CAD 2

Project: Frame Decoder Design from HDL down to Layout

Made by: Ahmed Mohamed Hussien Sabry - Computer

Masters ID: 10049

Submitted to: Dr. Dessoky

Frame Decoder Design

Introduction

This document is about the design of a frame decoder using Alliance CAD tool. This fame decoder is used to decode the input fames shown in figure 1. Two types of fames will be correctly received and output the contained address and data inside it.

SOF	ID	Address	Data	EOF
7E	80	1 Byte	1 Byte	E7

Simple Data Frame

SOF	ID	Address	Data	EOF
7E	81	1 Byte	MSB LSB	E7

Extended Data Frame

Figure 1 Types of frames that will be decoded by the frame decoder

The upper layout of the frame is shown in figure 2. The frame decoder will be designed as an FSM. Any error in the fame the error bit will be risen.

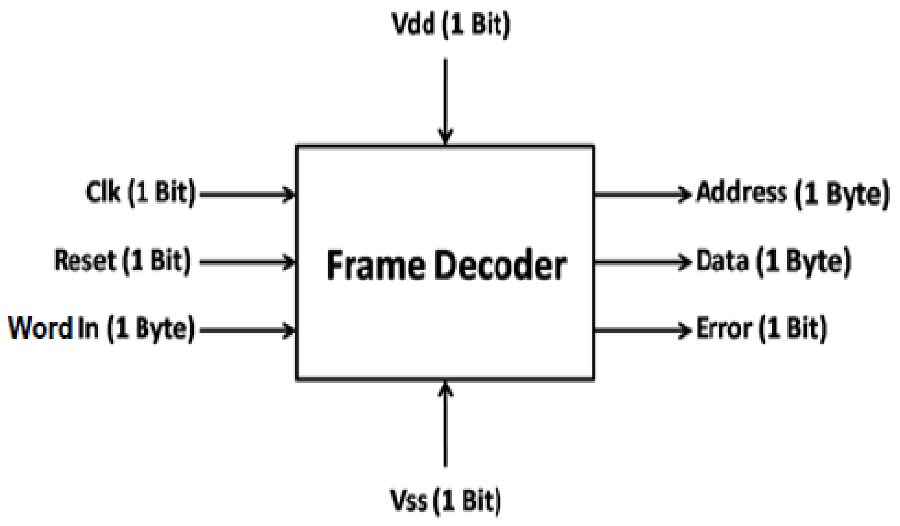


Figure 2 Frame decoder top layout

This file is divided into 3 chapters

1. Project HDL
2. Project synthesis
3. Placement and routing

All steps of the project is contained in a single **Makefile**. Another file is the **compile.sh** it contain the compilation details using the **Makefile** and outputs the simulation FSM, schematics, simulation-results and the layout of the core and the final chip.

Project HDL

The **ALLIANCE** tools used are:

- **xfsm**: Graphical FSM viewer.
- **vasy**: VHDL analyzer and convertor.
- **asimut**: VHDL Compiler and Simulator.

The frame decoder FSM is shown in figure 3. This is the FSM design for one bit of output of Data and address. For complete set of outputs this FSM will be repeated 8 times. There is some details that are not shown in the FSM figure, each FSM will have two output bits one for address and the other for the data.

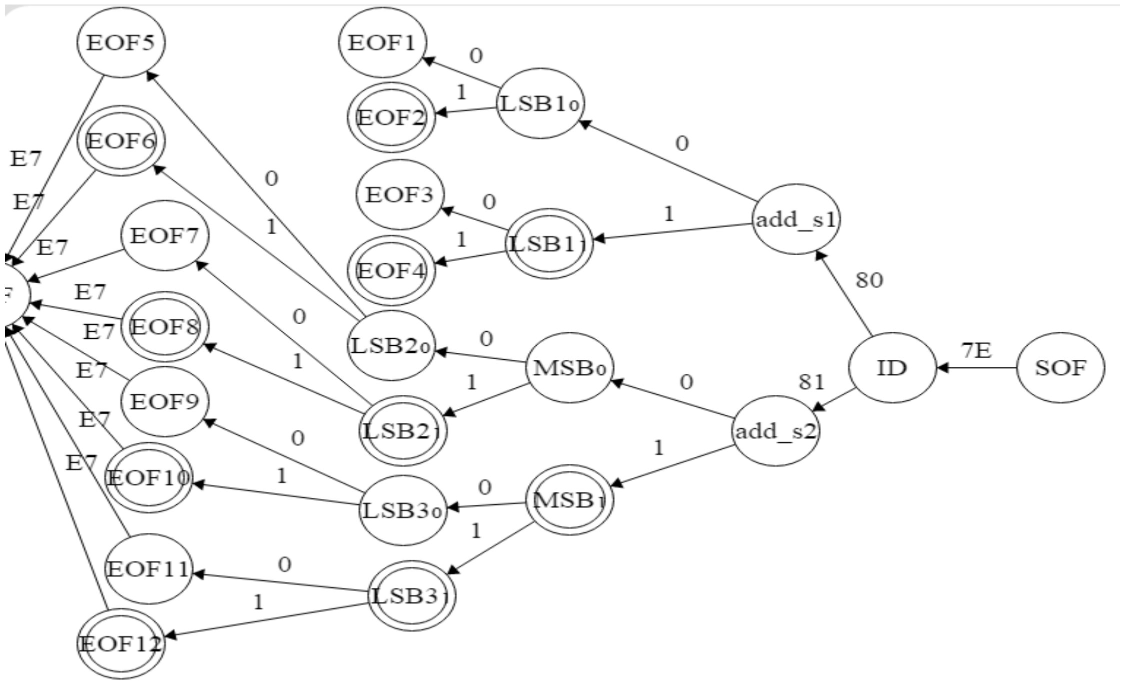
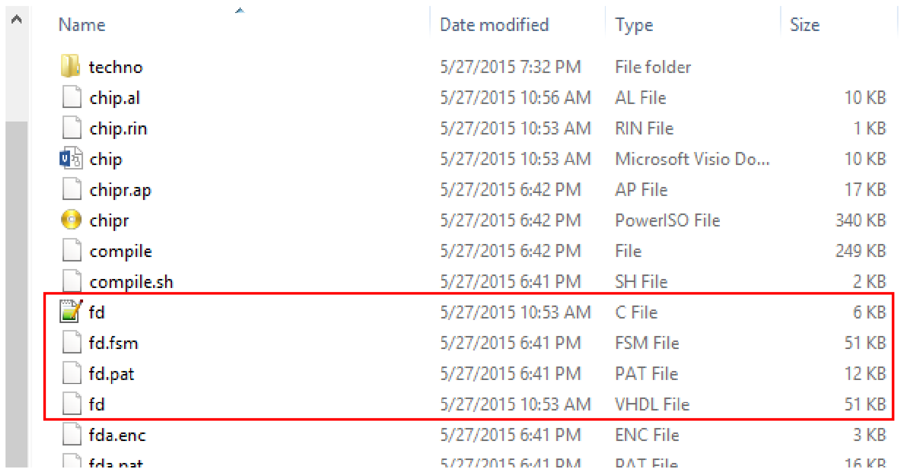


Figure 3 Frame decoder FSM

After designing the FSM, the FSM code was written by **vim** text editor in the VHDL format **(fd.vh)**. Then it is copied in to the FSM format **(fd.fsm)**. A test case pattern file **(fd.c)** is written in C format. The **Makefile** is used to produce the output of this simulation easily.



Name	Date modified	Type	Size
techno	5/27/2015 7:32 PM	File folder	
chip.al	5/27/2015 10:56 AM	AL File	10 KB
chip.rin	5/27/2015 10:53 AM	RIN File	1 KB
chip	5/27/2015 10:53 AM	Microsoft Visio Do...	10 KB
chipr.ap	5/27/2015 6:42 PM	AP File	17 KB
chipr	5/27/2015 6:42 PM	PowerISO File	340 KB
compile	5/27/2015 6:42 PM	File	249 KB
compile.sh	5/27/2015 6:41 PM	SH File	2 KB
fd	5/27/2015 10:53 AM	C File	6 KB
fd.fsm	5/27/2015 6:41 PM	FSM File	51 KB
fd.pat	5/27/2015 6:41 PM	PAT File	12 KB
fd	5/27/2015 10:53 AM	VHDL File	51 KB
fda.enc	5/27/2015 6:41 PM	ENC File	3 KB
fda.pat	5/27/2015 6:41 PM	PAT File	16 KB

Figure 4

The **compile.sh** file is used to make the simulation after any change in the **fd.vh** file easier and easier to simulate as the text editor supported the VHDL format and didn't support the (.fsm) format. The **compile.sh** is also good in trouble shooting and displaying the results. **To run the project you only have to write**

>> ./compile.sh

in the terminal only making sure that compile file is executable

>> chmod u+x complie.sh

and that all the files needed in the project is included in the same folder files that are essential for the project compilation are

- **fd.vh**
- **fd.c**
- **Makefile**
- **paramfile.lax**
- **pinorder.ioc**
- **scanpath.path**
- **spimodel.cfg**
- **techno-035.rds**
- **fdjscapin.c**

The compile script will have all the output figures displayed for you easily. The make all will do the same without displaying any figures.

The FSM using the **XFSM** is shown in figure 5

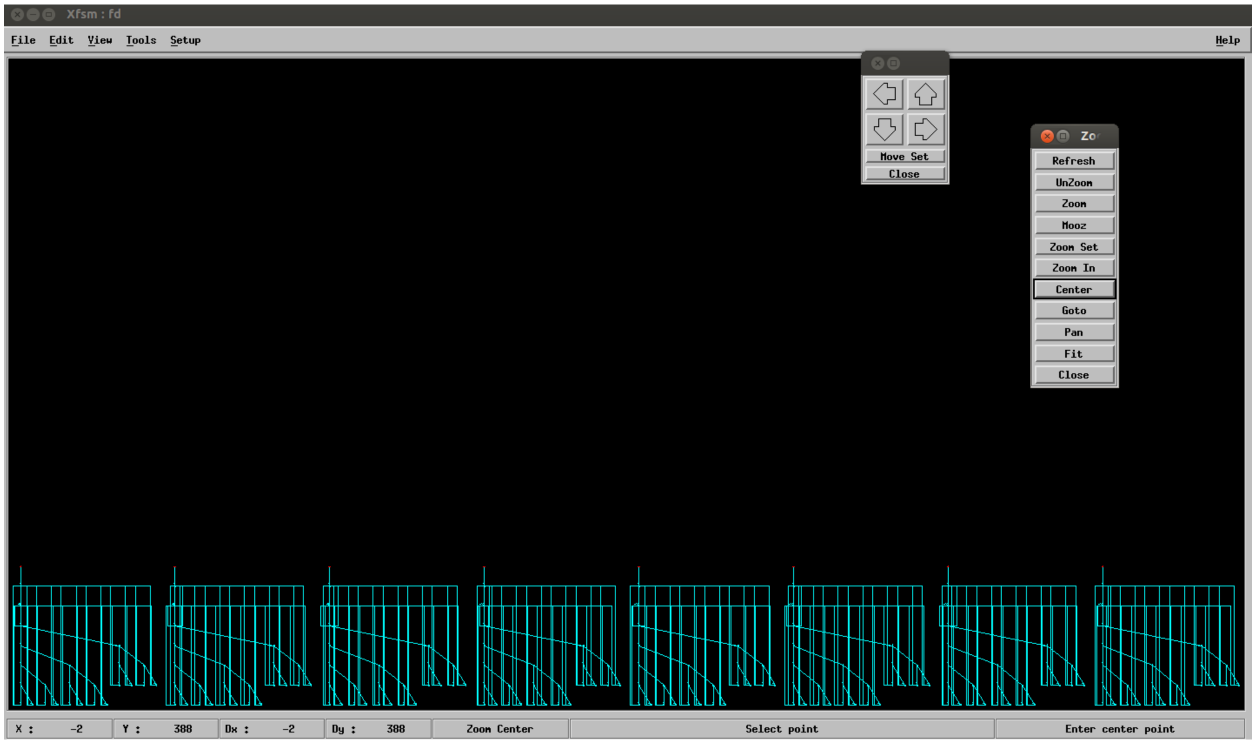


Figure 5

Patterns used are included in the spread sheet **TestVectors_CodingCompare.xlsx**
The output of the different FSM coding when simulating with **ASIMUT** is the same. In figure 6 the reset is set to one so the data and the address output is zero as expected.

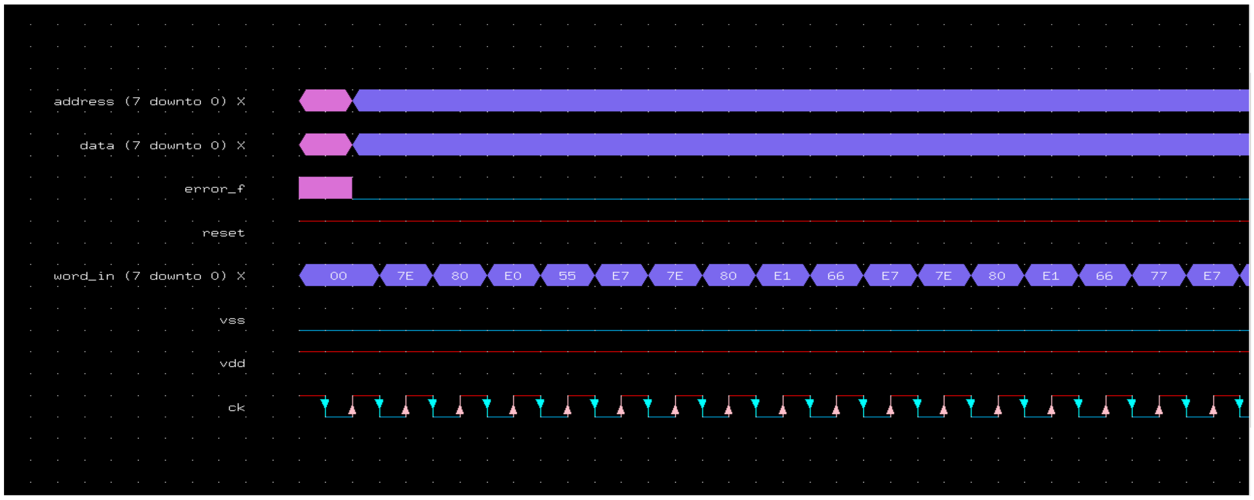


Figure 7

In figure 7 the output is as expected when the frame is right the error bit is zero when it's wrong the error bit is one. The extended and the normal frame both are shown in the test cases.

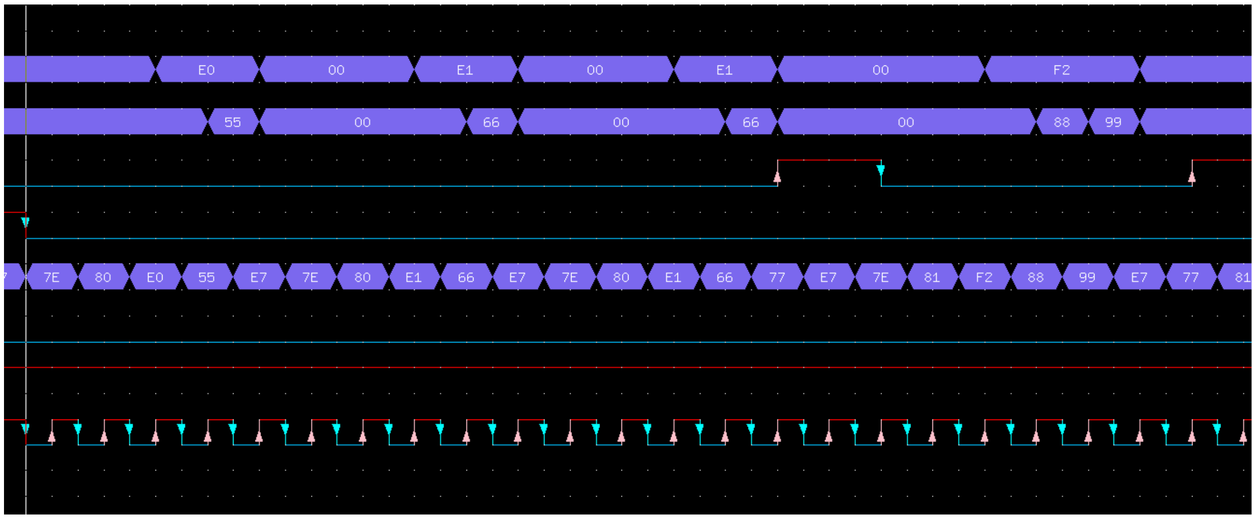


Figure 6

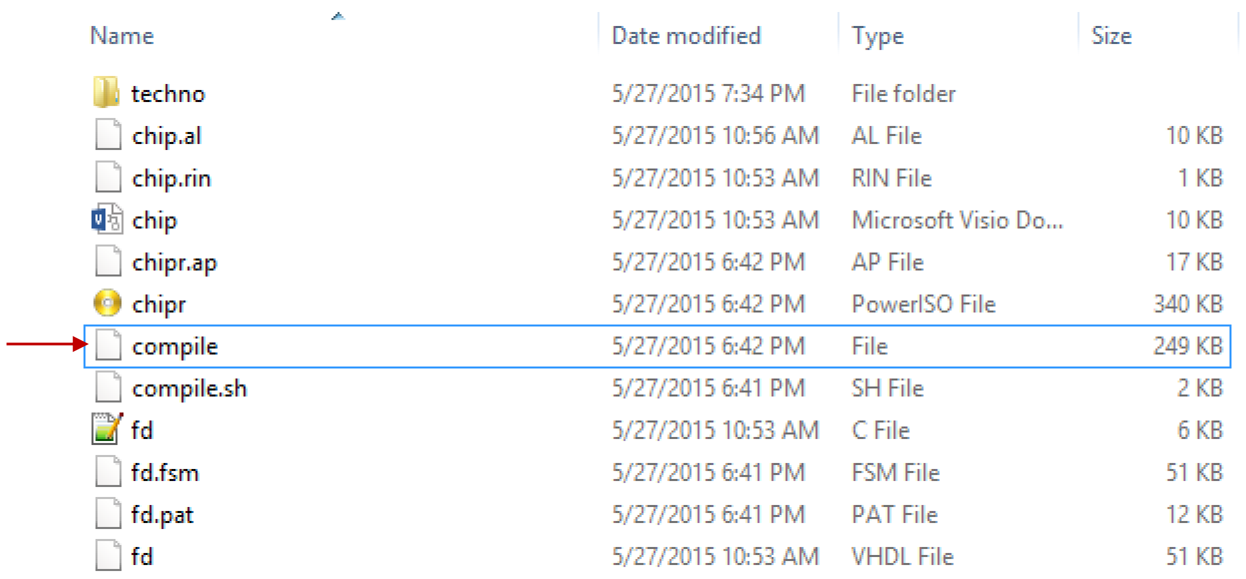
Project synthesis

In this part of the file we describe the synthesis of the frame decoder following the coming steps

2.1 Boolean network optimization

After the **boom** we simulate the different encoded finite state machines successfully with **asimut** simulator in comparison mode.

The output of the simulator + the output of all the other steps of the project is shown in the **compile** file



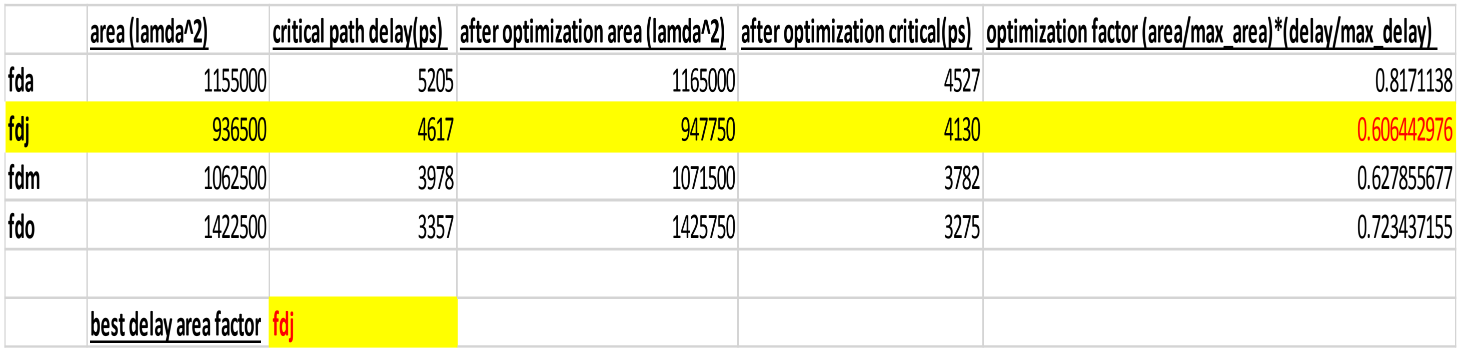
Name	Date modified	Type	Size
techno	5/27/2015 7:34 PM	File folder	
chip.al	5/27/2015 10:56 AM	AL File	10 KB
chip.rin	5/27/2015 10:53 AM	RIN File	1 KB
chip	5/27/2015 10:53 AM	Microsoft Visio Do...	10 KB
chipr.ap	5/27/2015 6:42 PM	AP File	17 KB
chipr	5/27/2015 6:42 PM	PowerISO File	340 KB
compile	5/27/2015 6:42 PM	File	249 KB
compile.sh	5/27/2015 6:41 PM	SH File	2 KB
fd	5/27/2015 10:53 AM	C File	6 KB
fd.fsm	5/27/2015 6:41 PM	FSM File	51 KB
fd.pat	5/27/2015 6:41 PM	PAT File	12 KB
fd	5/27/2015 10:53 AM	VHDL File	51 KB

2.2 Library Mapping

The output of this step is shown in the compile file along with the output **boog** file that exist in the Ahmed_Mohamed_Hussien_Sabry/ project file

2.3 Netlist optimization

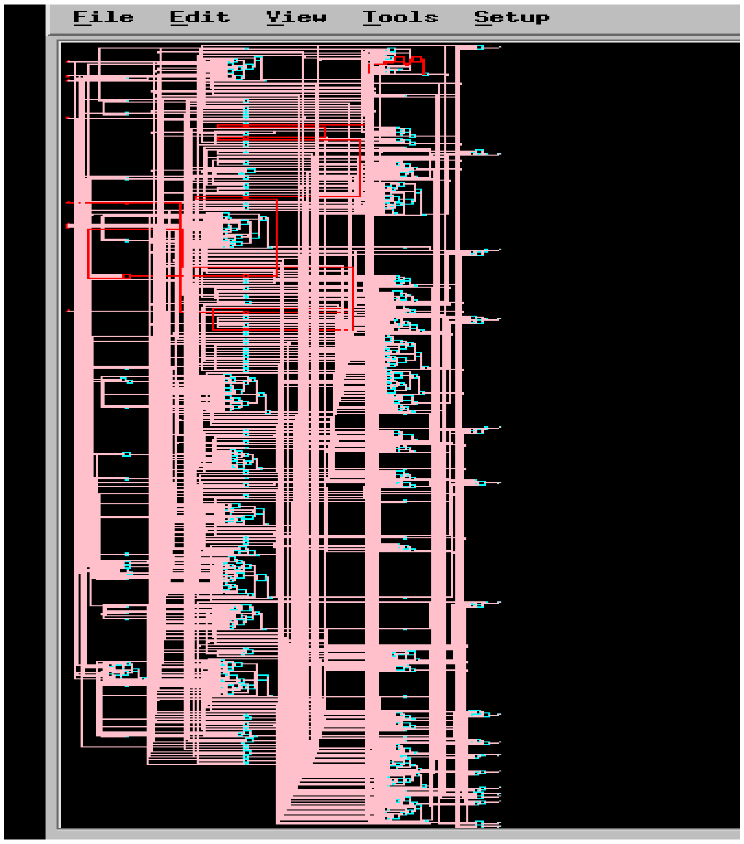
The output of this step is shown in the compile file along with the output **loon** file. The shown table contain the output of the boog delay and area along with the output form the loon delay, area and the best FSM, which will continue in the steps that follow this step.



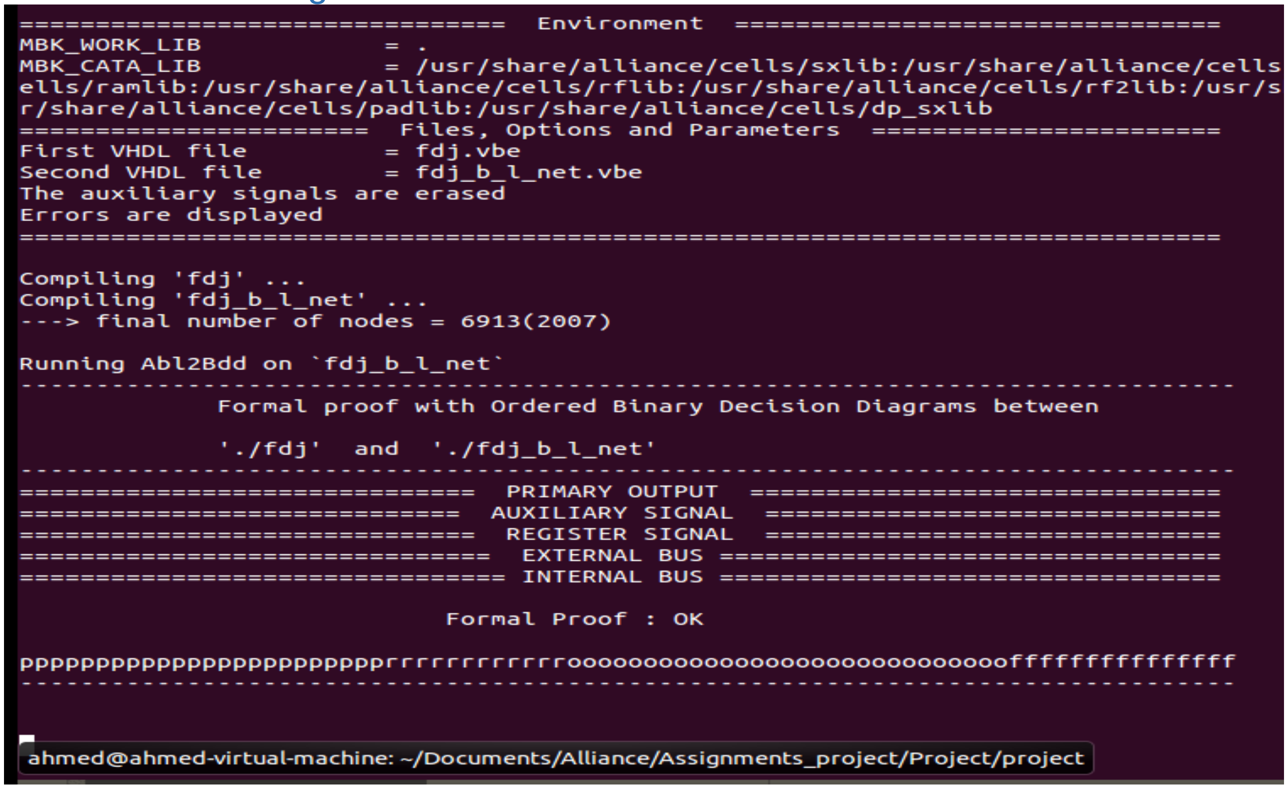
	area (lamda^2)	critical path delay(ps)	after optimization area (lamda^2)	after optimization critical(ps)	optimization factor (area/max_area)*(delay/max_delay)
fda	1155000	5205	1165000	4527	0.8171138
fdj	936500	4617	947750	4130	0.606442976
fdm	1062500	3978	1071500	3782	0.627855677
fdo	1422500	3357	1425750	3275	0.723437155
	best delay area factor	fdj			

2.4 Netlist visualization

The **XSCH** output



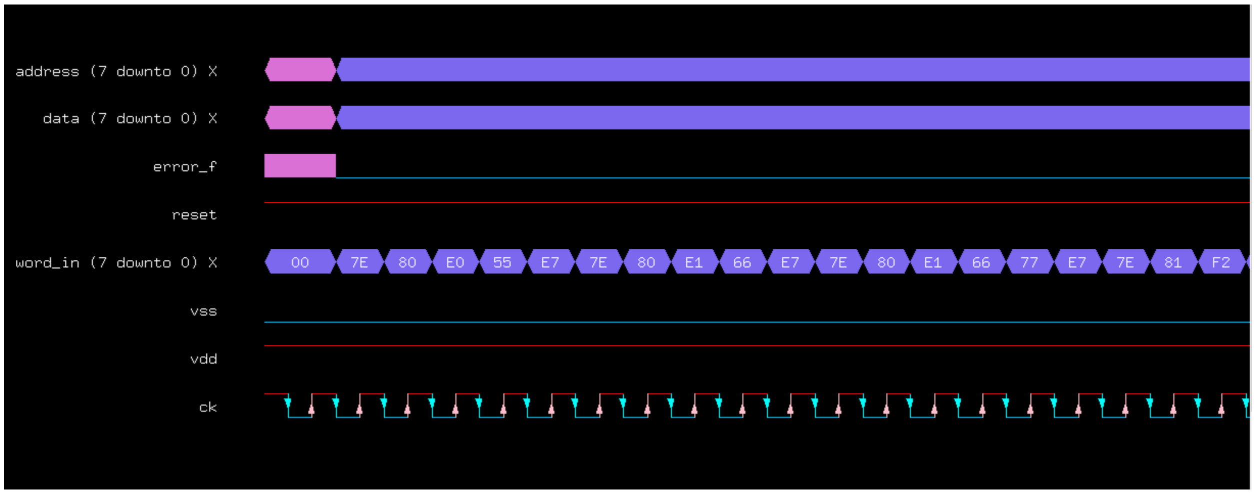
2.5 Netlist checking



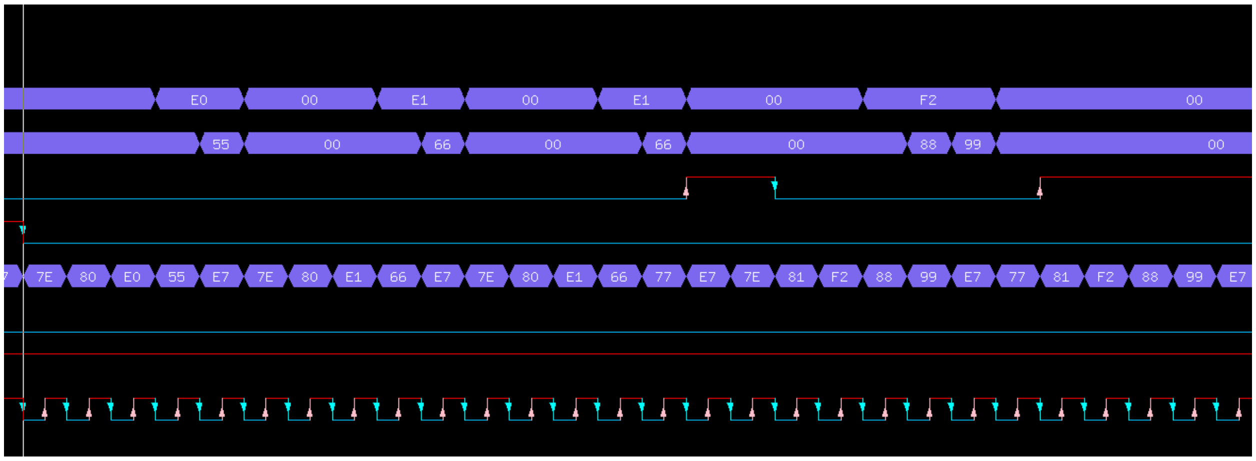
```
================================  Environment  ================================
MBK_WORK_LIB            = .
MBK_CATA_LIB            = /usr/share/alliance/cells/sxlib:/usr/share/alliance/cells
ells/ramlib:/usr/share/alliance/cells/rflib:/usr/share/alliance/cells/rf2lib:/usr/s
r/share/alliance/cells/padlib:/usr/share/alliance/cells/dp_sxlib
=======================  Files, Options and Parameters  =======================
First VHDL file         = fdj.vbe
Second VHDL file        = fdj_b_l_net.vbe
The auxiliary signals are erased
Errors are displayed
===============================================================================

Compiling 'fdj' ...
Compiling 'fdj_b_l_net' ...
---> final number of nodes = 6913(2007)

Running Abl2Bdd on `fdj_b_l_net`
--------------------------------------------------------------------------------
             Formal proof with Ordered Binary Decision Diagrams between

             './fdj'  and  './fdj_b_l_net'
--------------------------------------------------------------------------------
==============================  PRIMARY OUTPUT  ===============================
=============================  AUXILIARY SIGNAL  ==============================
==============================  REGISTER SIGNAL  ==============================
===============================  EXTERNAL BUS =================================
================================  INTERNAL BUS ================================

                            Formal Proof : OK

ppppppppppppppppppppppprrrrrrrrrrroooooooooooooooooooooooooooofffffffffffff
--------------------------------------------------------------------------------

ahmed@ahmed-virtual-machine: ~/Documents/Alliance/Assignments_project/Project/project
```

2.6 Delay Simulation

The reset signal is set to 0 at first then the data and the address signal is zero as shown in the figure below



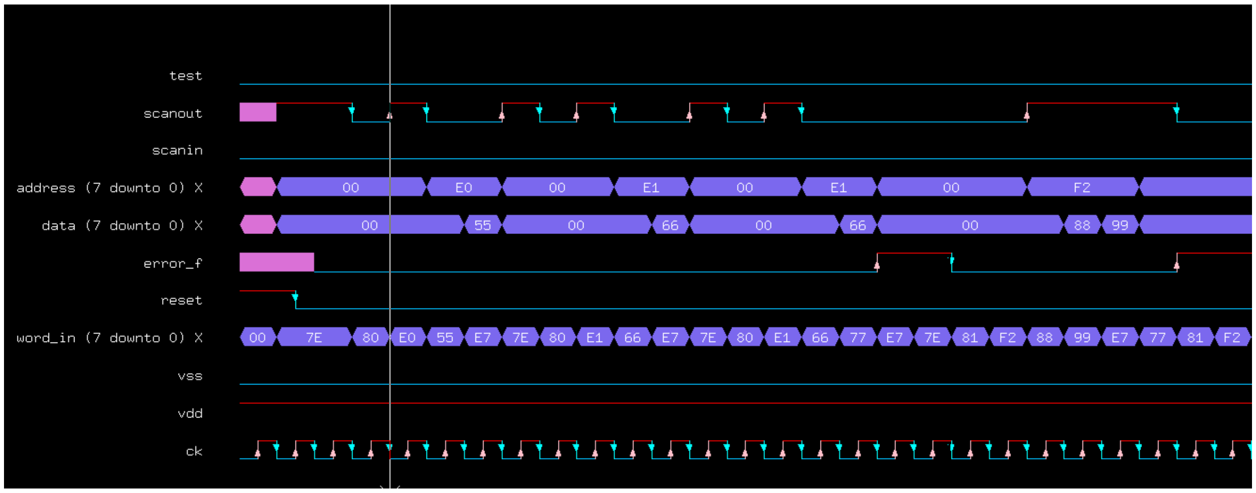
Then when its set to one the output becomes



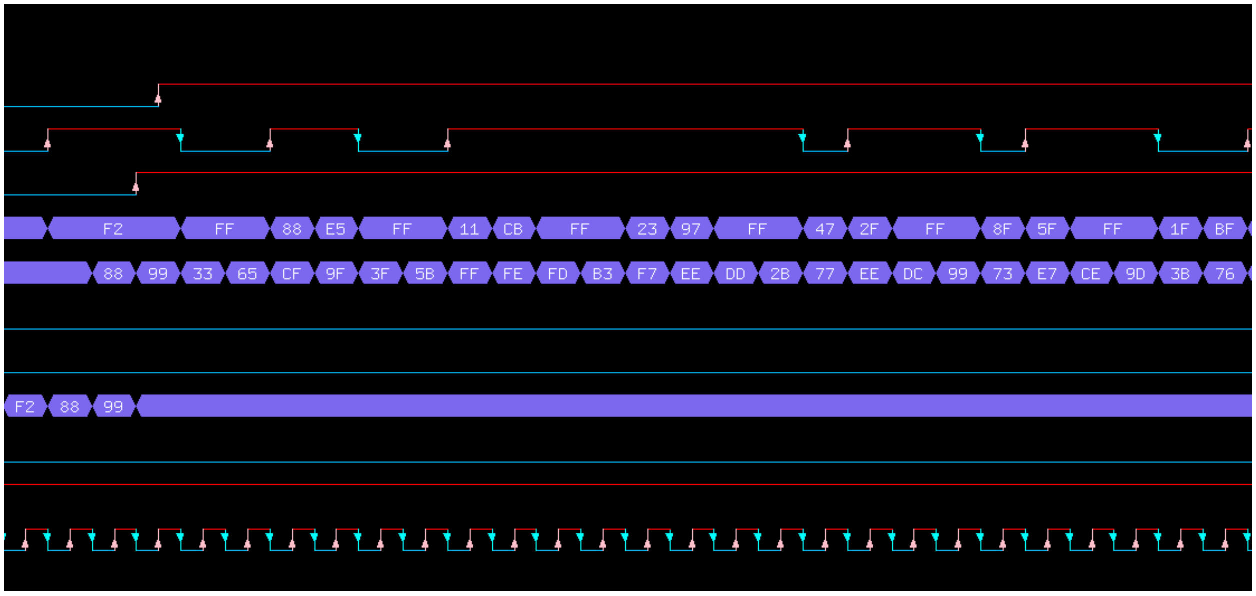
The difference between the output with delay and without is not obvious because the output if data and the address is always aligned with the clock and the critical path delay is 4.1 ns and the simulation period is 10 ns so there is no clock violation.

2.7 Scan-path insertion

The simulation results after the scan path insertion when the test = 0 and the circuit is operating normally.



When test = 1 and the scan_in =1 and the word_in is constant. The output is



Physical synthesis

The **ALLIANCE** tools used are:

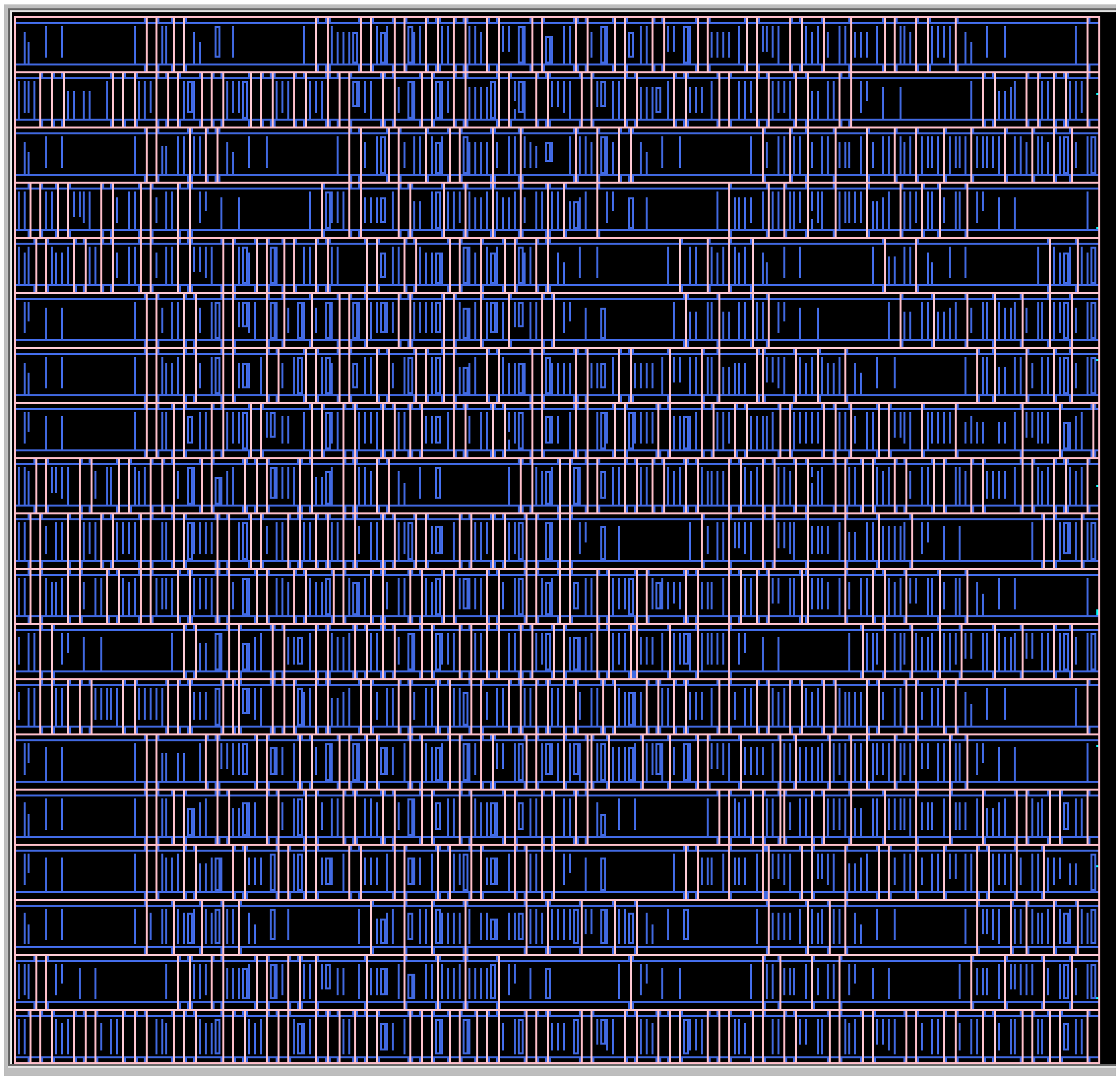
- **ocp:** Standard Cell Placer.
- **nero:** Over-Cell Router.
- **cougar:** Symbolic Netlist Extractor.
- **lvx:** Netlist comparator.
- **graal:** Symbolic layout editor.
- **druc:** Symbolic Design-rule checker.

 Ring: Pad ring router.

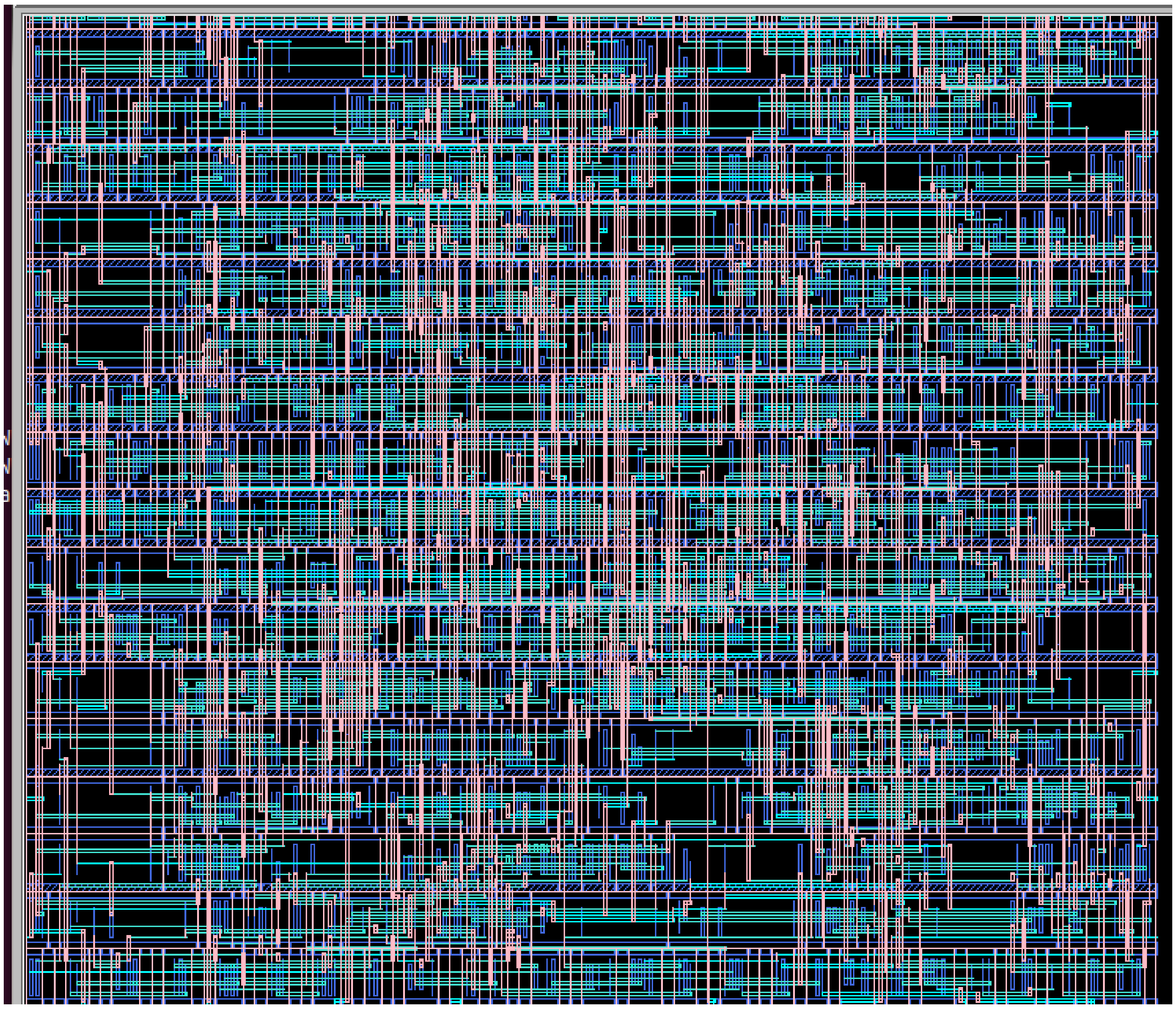
 s2r: Symbolic-to-Real layout converter.

2.1 Floor planning
Only one block is available so there is no floor planning.

2.2 Placement
The output of the placement is shown in the figure below

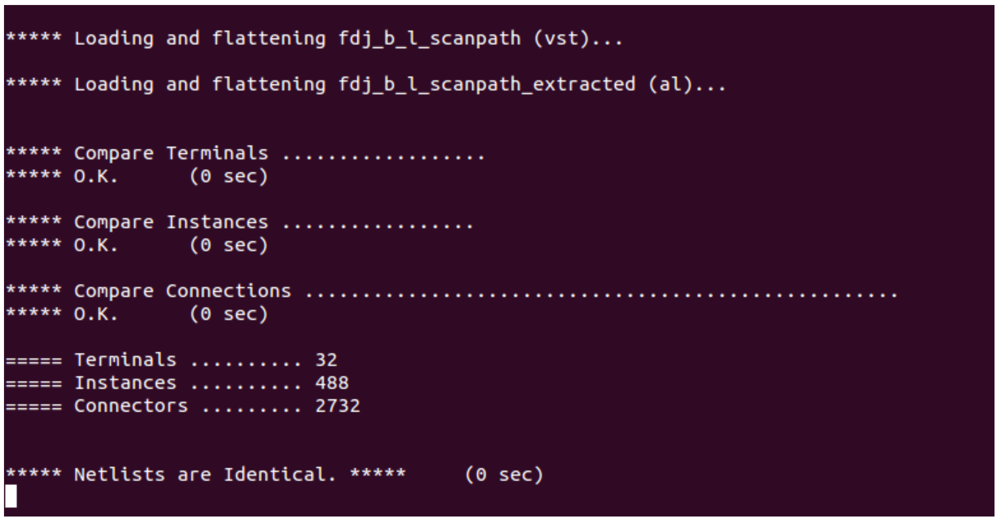


2.3 Routing
The layout after routing

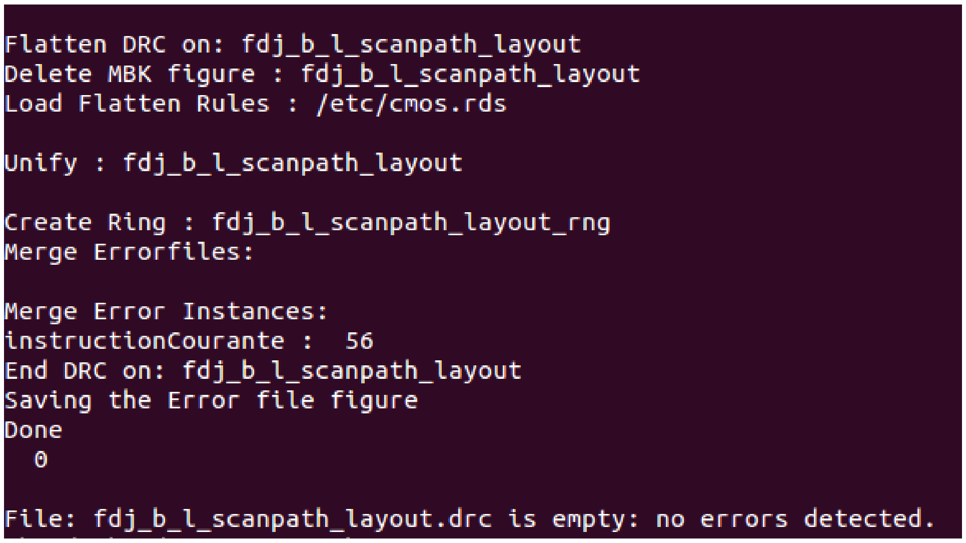


2.4 Post-Layout Verification

Layout vs schematics and DRC check using **LVX** and **DRUC**



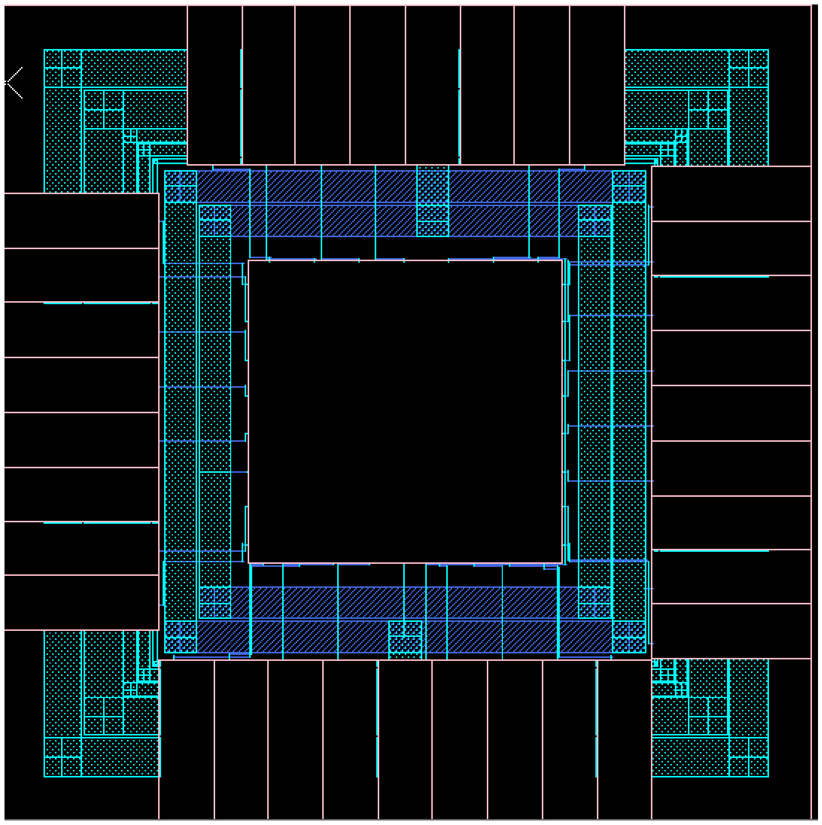
```
***** Loading and flattening fdj_b_l_scanpath (vst)...

***** Loading and flattening fdj_b_l_scanpath_extracted (al)...


***** Compare Terminals .................
***** O.K.      (0 sec)

***** Compare Instances ................
***** O.K.      (0 sec)

***** Compare Connections ...................................................
***** O.K.      (0 sec)

===== Terminals .......... 32
===== Instances .......... 488
===== Connectors ......... 2732


***** Netlists are Identical. *****     (0 sec)
```



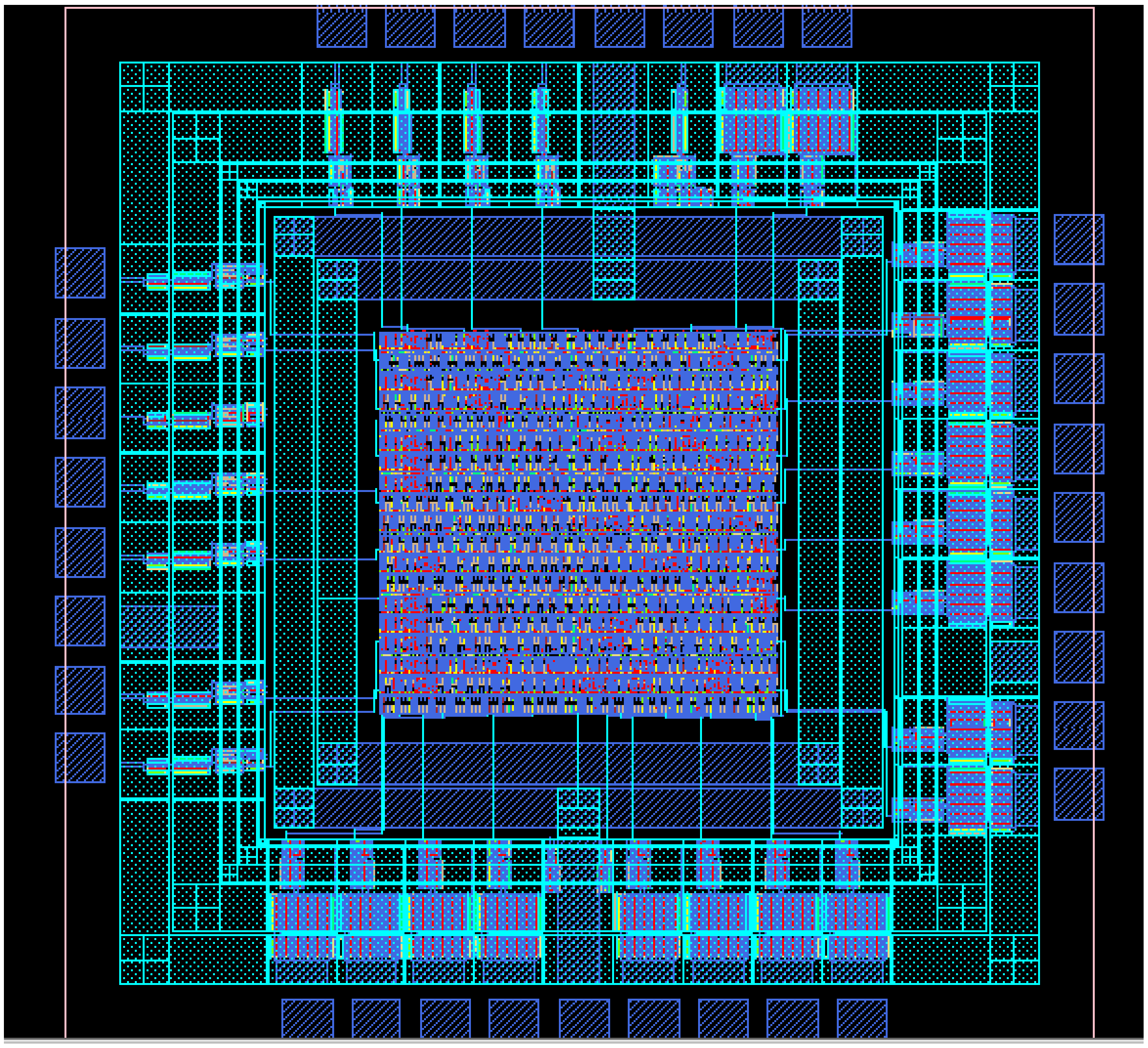
```
Flatten DRC on: fdj_b_l_scanpath_layout
Delete MBK figure : fdj_b_l_scanpath_layout
Load Flatten Rules : /etc/cmos.rds

Unify : fdj_b_l_scanpath_layout

Create Ring : fdj_b_l_scanpath_layout_rng
Merge Errorfiles:

Merge Error Instances:
instructionCourante :   56
End DRC on: fdj_b_l_scanpath_layout
Saving the Error file figure
Done
  0

File: fdj_b_l_scanpath_layout.drc is empty: no errors detected.
```

2.5 Chip assembly

The chip layout



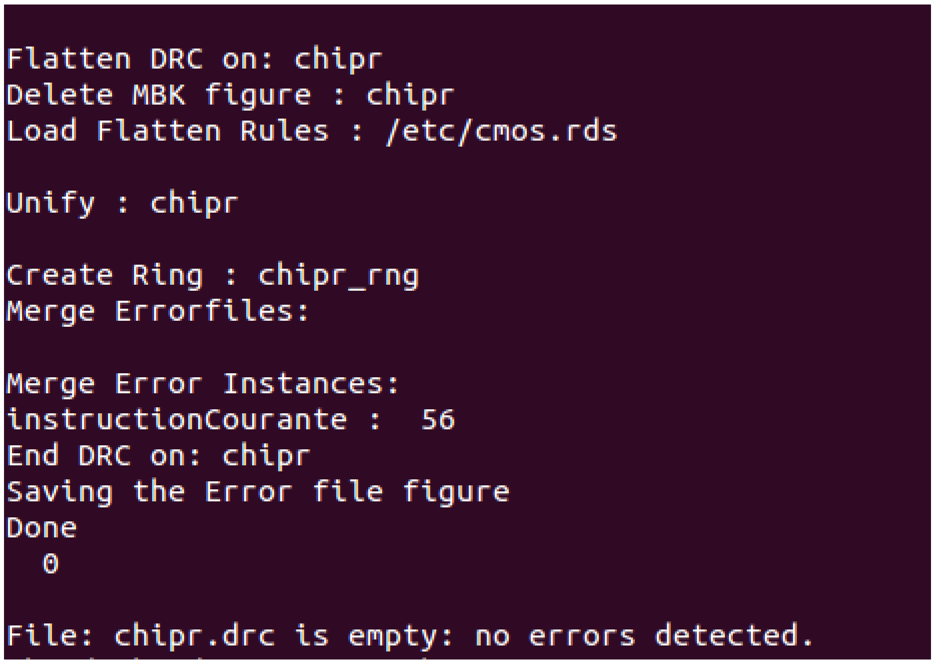
The chip layout with the flat real option



2.6 Chip verification

LVX

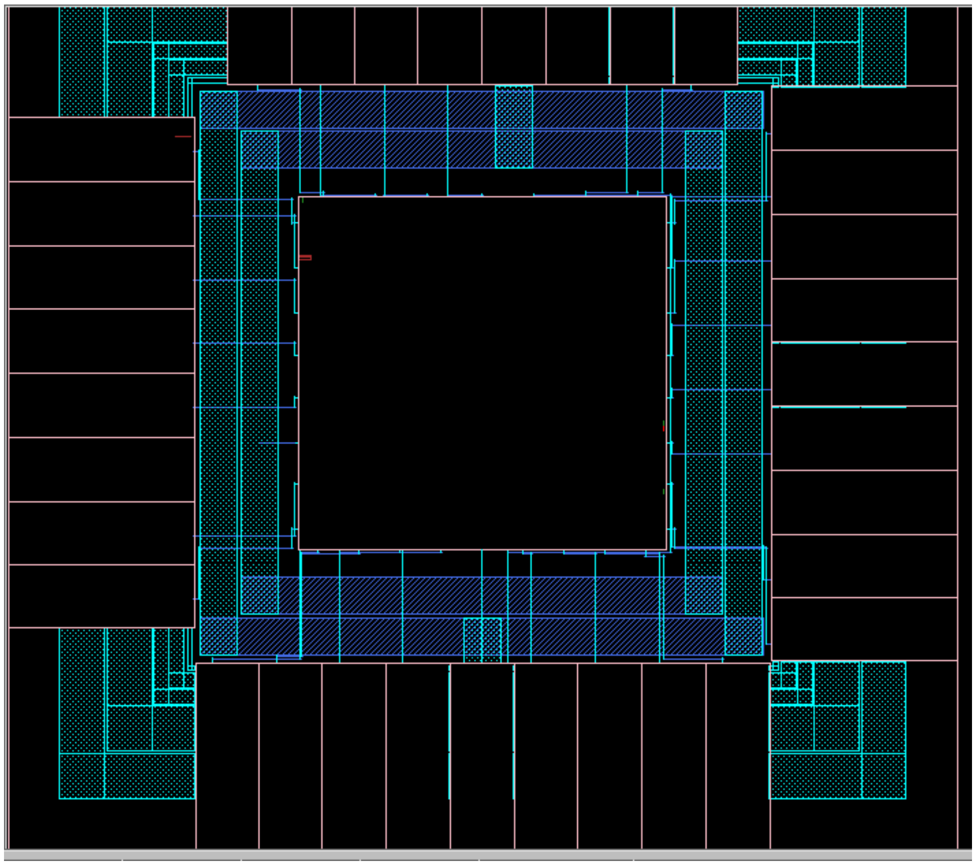
DRUC



```
Flatten DRC on: chipr
Delete MBK figure : chipr
Load Flatten Rules : /etc/cmos.rds

Unify : chipr

Create Ring : chipr_rng
Merge Errorfiles:

Merge Error Instances:
instructionCourante :   56
End DRC on: chipr
Saving the Error file figure
Done
  0

File: chipr.drc is empty: no errors detected.
```

2.7 Symbolic to real conversion

DREAL output



DREAL output with flatten option