1-1-2015

The Design And Vlsi Implementation Of Digital Arithmatic Processors - A Case Study Of A Generalized Pipeline Cellular Array

Yudi Xie
Wayne State University,

THE DESIGN AND VLSI IMPLEMENTATION OF DIGITAL ARITHMATIC PROCESSORS - A CASE STUDY OF A GENERALIZED PIPELINE CELLULAR ARRAY

by

YUDI XIE

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2015

MAJOR: COMPUTER ENGINEERING

Approved By:

____

Advisor Date

DEDICATION

To my parents.

ACKNOWLEDGMENTS

The author is thankful to Dr. Harpreet Singh for suggesting the problem and providing expertise which greatly assisted this research. The author is also thankful to committee members Dr. Le Yi Wang and Dr. Nabil Sarhan.

PREFACE

The design and implementation of arithmetic processors is taken up in this thesis. As a case study, a generalized pipeline array is discussed. A generalized pipeline array appeared in IEEE transaction in 1974. The array appeared in a few textbooks on computer arithmetic. From time to time, a number of papers appeared which reflected the modifications of this array. The objective of this thesis is to present the design and VLSI implementation of arithmetic processors. As a case study the design and VLSI implementation of a generalized pipeline cellular array is taken up in this thesis. This array can add, subtract, multiply, divide, square and square root of binary numbers. In this thesis, we suggest a step-by-step procedure by which the design can be sent to MOSIS and to get the fabricated chip back. The array has been extended from 5 rows to 7 rows so that the extended operations can be performed. In particular, a procedure is developed by which the design and the implementation methodologies are suitable for 40 pin and 500 nm technologies. An algorithm has been developed by which one can predict and meet the requirements of constrains like chip area. In order to increase data processing throughput, the extension of pipelining is conducted. It is hoped that the design and implementation done here will go a long way in the development of advanced arithmetic processors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

There has always been interest in the design of new digital systems and processors. Several authors have been proposing from time to time advanced adders, multipliers, etc. as these are basic elements of computation. In order to improve speed and accuracy, different authors have been suggesting algorithms for better computation. A generalized pipeline array first appeared in IEEE Transaction on Computers in 1974 [1]. This array can perform various arithmetic operations such as add, subtract, multiply, divide, square and square root. The array also was introduced in a number of textbooks of pipeline architectures such as [2] [3] [4] and [5]. Agrawal [6] [7] also extended this array in a number of papers. Papers such as [8] also extended this design as binary processors suitable for optical processing. Singh et al [9] presented the VLSI implementation of this array. More recent studies such as [10], [11], [12] and [13] also discuss the designs and applications of arithmetic units in processors. The objective of this thesis is to present the various design considerations and VLSI implementation of this array. In this thesis, we present a step-by-step procedure by which the array can be implemented as a VLSI chip. The thesis gives an approach by which the design can be extended to higher number of bits for various operations. The thesis also gives an algorithm by which the limits of the design such as memory, chip area, input/output pins and acceptable delays are designed. In addition, to increase data processing throughput, the array is extended using pipelining based on the original design, which is discussed in this thesis. The FPGA implementation of the array is discussed and simulation results are included.

Introduction to Arithmetic Processors

The appearance of low-cost computers on integrated circuits has revolutionized modern society. Nowadays people use general-purpose processors in computers are used for their daily computational tasks such as text processing, multimedia streaming, and online communication. In addition, many more microprocessors are now a critical part of embedded systems, providing digital enhancement for a kinds of objects ranging from appliances to automobiles to cellphones and industrial control systems.

The main use of the first microprocessors emerged in the beginning of 1970s were for digital calculators, capable of performing 4-bit arithmetic calculations. Other systems followed soon after used various kinds of microprocessors with very few bits such as 4-bit and 8-bit, in applications such as terminals, printers, and various kinds of automation systems. Figure 1 shows the micro-architecture of a typical example of general-purpose microprocessors. This figure from [44] is reproduced here as a ready reference.

In addition to general-purpose microprocessors, several specialized processing devices include special architecture are proposed with enhanced arithmetic units to support various special-purpose applications for data processing. For example digital signal processors (DSP, as shown in Figure 2 [44]) usually include specialized arithmetic unit for signal processing. Another example is the graphics processing units (GPU), which also include special arithmetic units different from general-purpose microprocessors which are used for single instruction, multiple data (SIMD) applications such as vector processing.

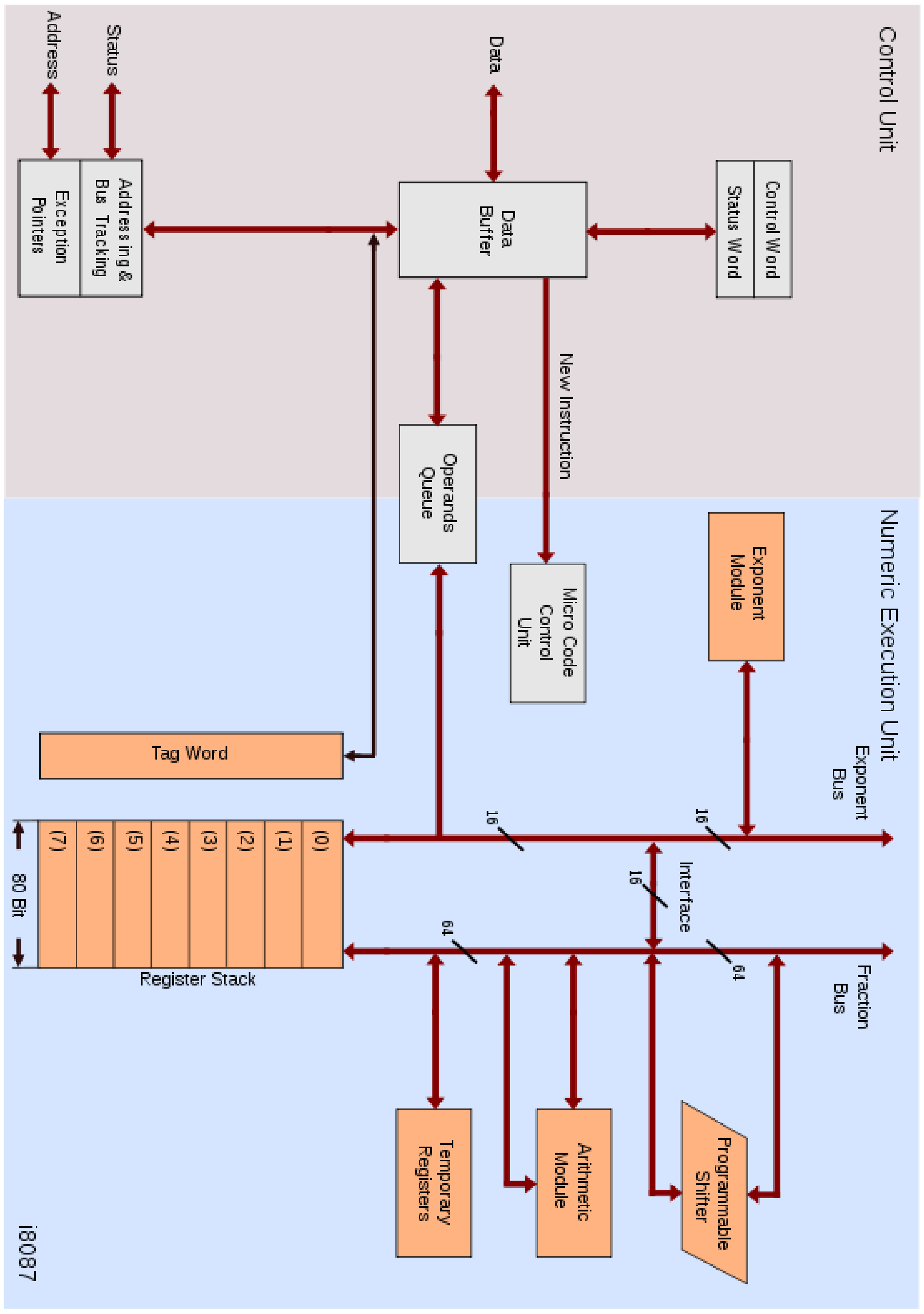


Figure 1: Intel 8087 microarchitecture [44].

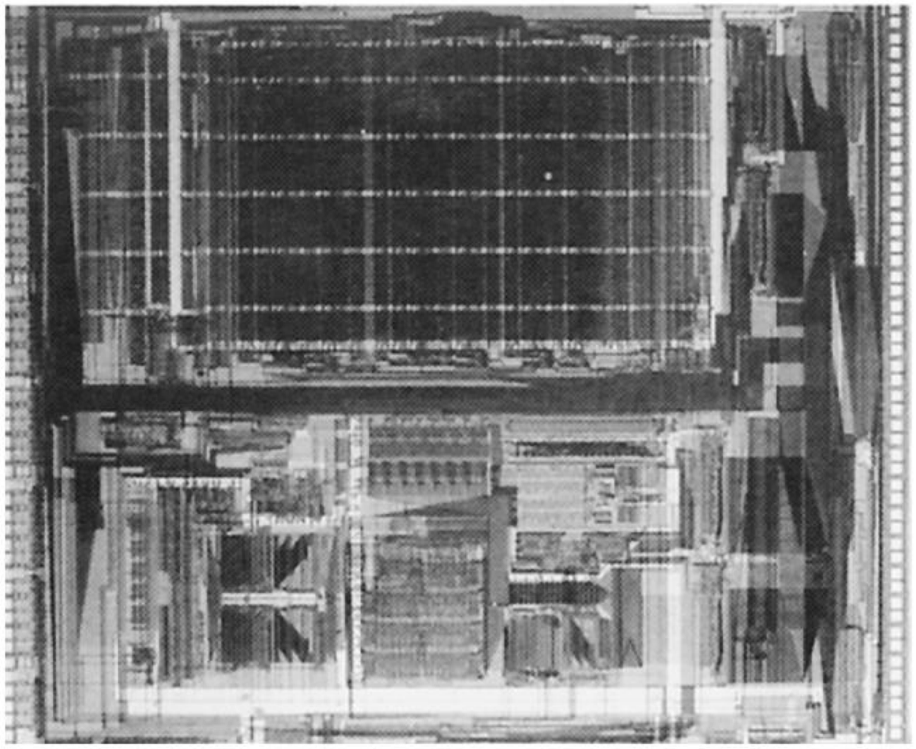
Figure 2: DSP chip from Texas Instruments [44]

Current Designs in Arithmetic Units

Arithmetic processors, or arithmetic logic units (ALU) including inside a processor, is a digital circuit that perform various arithmetic. ALUs are the central building blocks of microprocessors, including general-purpose ones and special-purpose ones, where the actual computation is performed.

Before introducing the arithmetic processor called generalized pipeline array, several typical designs of arithmetic processors or arithmetic units are introduced first as case studies. They are TI 8847, MIPS R3010 and Weitek 3364.

The TI 8847 (as shown in Figure 3), is a high-speed, double-precision floating point and integer arithmetic processor from Texas Instruments [14].



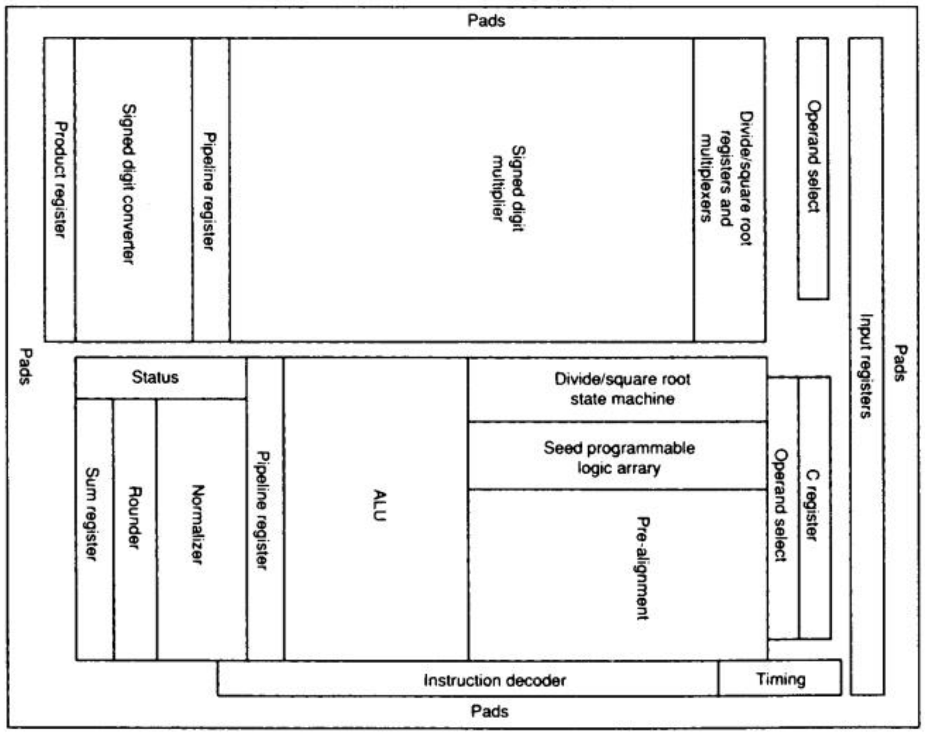


Figure 3: TI 8847 [44]

The MIPS R3010 (as shown in Figure 4) is an arithmetic co-processor for the MIPS R3000 series RISC processors. R3000 FPU can perform various arithmetic operations such as conversion with single or double-precision numbers..

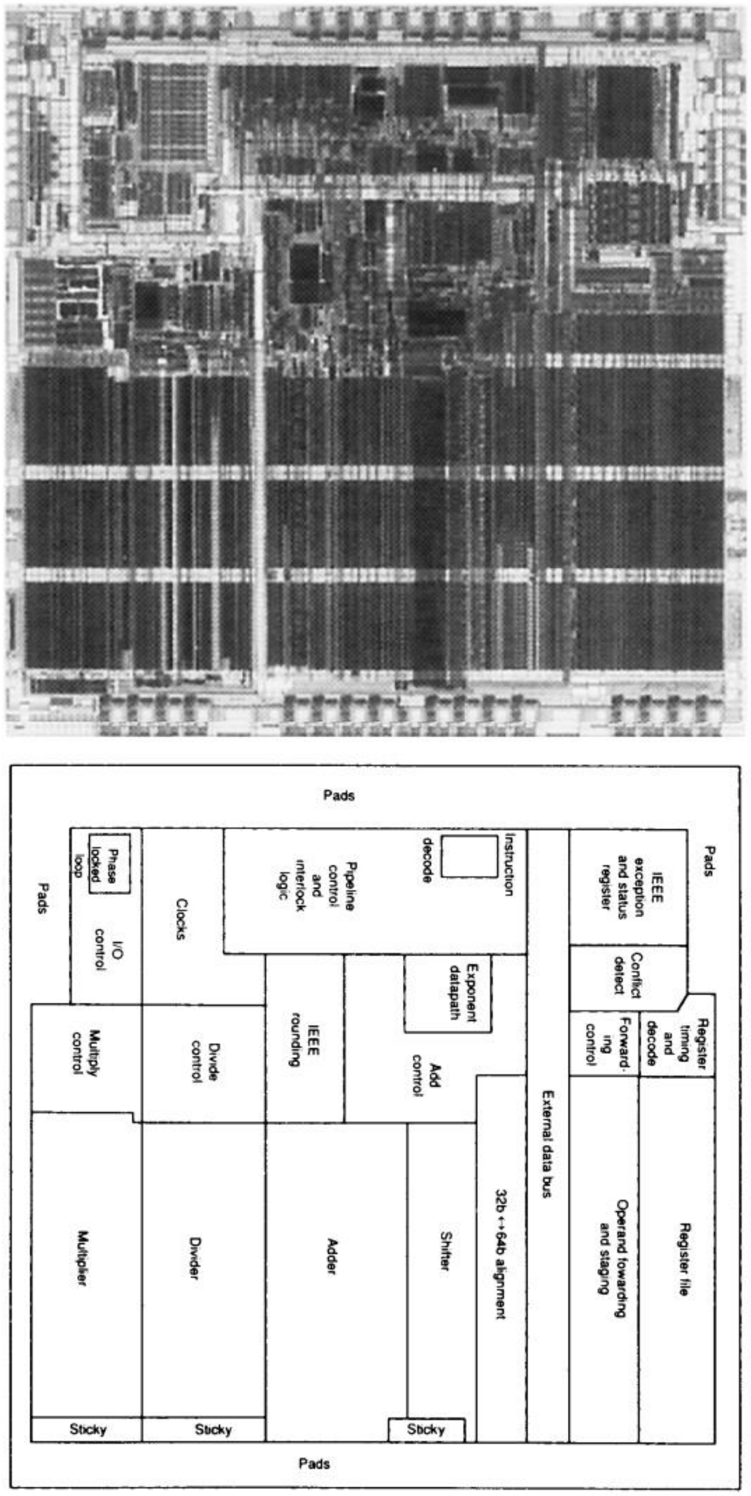


Figure 4: MIPS R3010 [44]

The Weitek 3364 (as shown in Figure 5) 64-bit arithmetic unit was designed for high-speed operations in a pipelined environment [45].

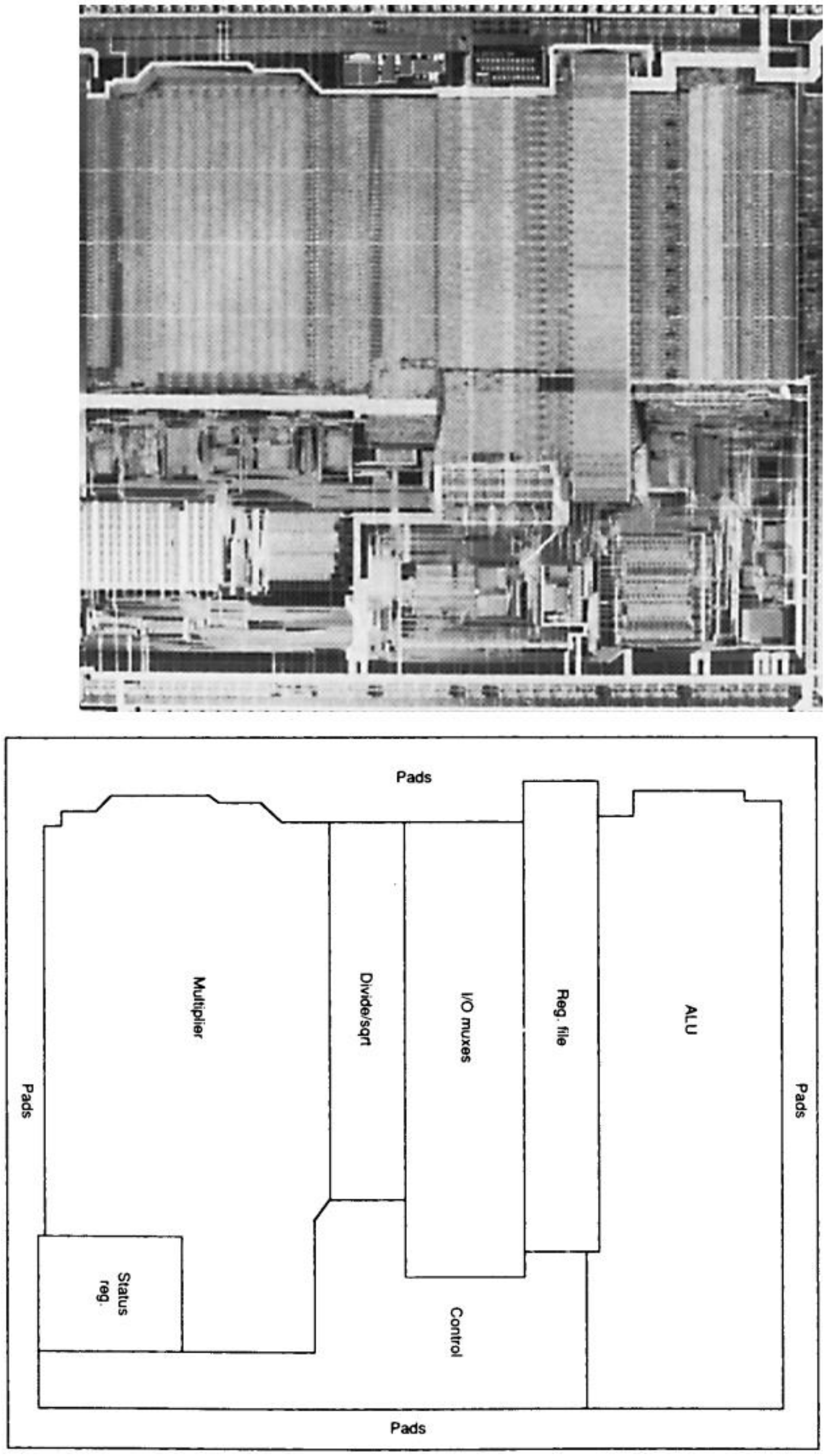


Figure 5: Weitek 3364 [44]

The arithmetic processor which is called generalized pipeline cellular array which is described in the thesis and upon which various extensions are made later is introduced here briefly. This arithmetic processor consists an array made up of 2 kinds of cells: control cells and arithmetic cells. Once connected together, it can perform various kinds of arithmetic operations such as add, subtract, multiply, divide, square and square root.

Compared to other commercial products introduced earlier, this design has several advantages. First, it can achieve relative high performance with low cost given the limited design constrains. Second, it's easily expandable, which means the design can be further expanded easily with just attaching more well-defined modules together with existing modules instead of starting over, when a better technology is available. In addition, these well-defined modules share the same level complexity, which means having almost the same delays. In addition, each stage doesn't depend on each other. These features make this design suitable for pipelining. Finally, due to its low complexity, it can easily achieve a fast throughput compared to other designs.

Table 1 shows the comparison between the generalized pipeline cellular array (extended version in this thesis) and the above other 3 arithmetic processors.

Table 1: Comparison between the generalized pipeline cellular array and Other Arithmetic Processors

Technical Parameters	**TI 8847**	**MIPS R3010**	**Weitek 3364**	**generalized pipeline cellular array (Extended)**
Clock cycle time (ns)	30	40	50	70
Size (mil^2)	156,180	114,857	147,600	1255
Transistors	180,000	75,000	165,000	5000
Pins	207	84	168	30
Cycles/add	2	2	2	1
Cycles/multiplication	3	5	2	1
Cycles/divide	11	19	17	1
Cycles/square root	14	-	30	1

The generalized pipeline cellular array is introduced and extended in this thesis. In addition a case study for a VLSI implementation procedure is discussed. This generalized pipeline cellular array provides an alternative way in arithmetic processor design, in an environment of limited resources while achieving high performance and flexibility. These characteristics make it suitable for products in such applications and also suitable for VLSI design education.

Introduction to Digital VLSI Design

The digital design is a basic subject which is taught in almost all engineering schools in the entire world. However, for the last several decades, the methodologies of design are changing. Previously in the textbooks of digital design, procedures were given in the form of dedicated gates such as NAND and NOR. With the advancement of time, the new books [15] and [16] started to introduce designs in hardware description language (HDL) such as Verilog and VHDL. The students can start digital logic designs in register transfer level (RTL) and test them on simulators and FPGA. These books usually include a number of examples written in HDL for basic components such as adders, multiplexers, etc. However, most of them don't cover the procedure for VLSI design. There are books of another type which only stress on the topic of VLSI design, such as [17] and [18]. Although the contents of these books are thorough and comprehensive, they only stress on theory and fail to provide a hands-on procedure for VLSI design. There are also reading materials provided by the software vendors such as [19] and [20] and software build-in "manpage", but these documents are more like manuals to look up than tutorials to follow, and usually assume readers to have prior technical experiences, which cause them too hard for university students to follow. Another kind of available resources for educational VLSI design courses are the documents and tutorials maintained either by online communities, such as [21] or course handouts from other educational institutes such as [22] and [23]. However, they are usually incomplete, inconsistent, outdated, or only applicable to specific environments. To the best the author's knowledge, there is only one book by Erik Brunvand [24] which could be used for the purpose of VLSI design for students, but it has not been updated for many years and many procedures in that book can no longer be applied to the new environment. There are also a number of papers in which the ways of teaching digital design in laboratories are discussed. For example, [25], [26] and [27] provide many discussions on the teaching method for VLSI design in University laboratories. Papers such as [28] and [29] also provide many insights on this topic. But they all fail to provide a complete, up-to-date and easy-to-follow procedure for VLSI digital design, by which the students in the VLSI laboratories can use any digital design in HDL languages such as Verilog and turn it into a chip with commercial software such as Cadence, and then send it for fabrication. In this thesis, a procedure is given, as a case study of one example, which will be helpful for the students to take any design in Verilog HDL language to complete the VLSI design of a chip.

This thesis also provides a step-by-step procedure for the VLSI design to facilitate digital VLSI design education, using the generalized pipeline cellular array, which is a high-performance easily-expandable arithmetic processor.

Review of Existing Implementation of generalized pipeline cellular array

For completeness, a thesis review of the pipeline array is given first [1] [9]. The generalized pipeline cellular array includes of two types of essentially cells: arithmetic cells and control cells. The diagrams of the arithmetic cells and control cells are shown in Figure 6 and Figure 7 respectively.

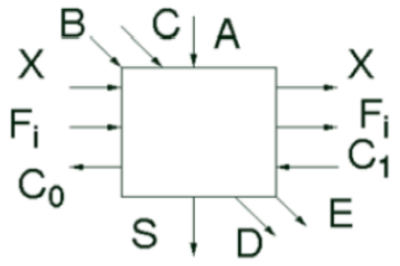


Figure 6: Arithmetic cell.

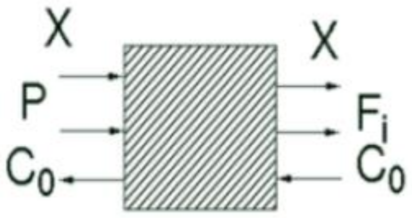


Figure 7: Control cell.

Boolean expressions of arithmetic cell are as follows.

$S = [A \oplus (B \oplus X) \oplus C] \cdot F_i + A \cdot F_i'$ (1)

$C0 = (B \oplus X) \cdot (A + C_1) + A \cdot C_1$ (2)

$D = B \cdot C + C \cdot F_i$ (3)

$E = B + C \cdot F_i$ (4)

Boolean expression of control cell is as follows.

$F_i = C_0 \cdot X + P_i \cdot X'$ (5)

After connecting those cells in the way shown in Figure 8, it becomes a system which can perform various arithmetic operations, thus gaining the name “generalized pipeline cellular array”. The operations and parameters of this pipeline array is summarized in Table 2. Based on these information, various arithmetic operations can be performed once the pipeline array is built.

Table 2: Operations of the pipeline array.

Operation	**Input**					**Output**	
	X	***P***	***C***	***B***	***A***	***F***	***S***
Square	0	operand	0100000	0011111	all 0	don't care	result
Square Root	1	all 0	0100000	0011111	operand	result	don't care
Multiplication	0	multiplier	multiplicand	B = C	all 0	don't care	result
Division	1	all 0	divisor	B = C	dividend	result	don't care

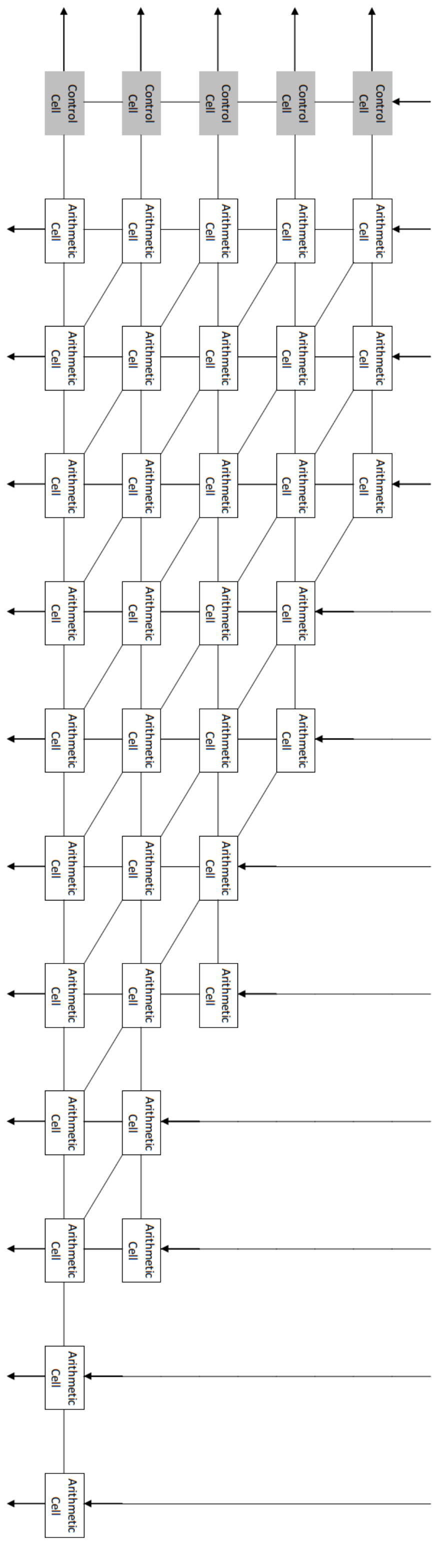


Figure 8: Organization of the pipeline array.

Arrangement of the Thesis

The organization of this thesis is as follows. Chapter 2 extends the original 5-row design of the pipeline array to a 7-row design to illustrate the way of extending the pipeline array. Chapter 3 discusses the design constrains met in the process of the VLSI implementation of the pipeline array, and an algorithm which can be incorporated in the design process to meet the design constrains. Chapter 4 discusses the extension of pipelining in order to increase the pipeline array's throughput. Chapter 5 summarizes and concludes this thesis.

Conclusion

This chapter introduces the concept of arithmetic processors, and then several current commercial designs. It introduces the generalized pipeline cellular array and comparisons are made against other arithmetic processors. This chapter introduces the concept of VLSI procedure. A brief review of the existing design of the generalized pipeline cellular array is discussed. The review of arithmetic processors and a generalized cellular array will help in the better design of advanced arithmetic processors and their VLSI implementation.

CHAPTER 2 EXTENSION OF THE PIPELINE ARRAY

Introduction

The original design of pipeline array only contains 5 rows of pipeline stages, which limits the number of input or output pins. This fact limits the accuracy of the pipeline array. Due to the advancement of VLSI technology, it's possible to put more resources on a single chip. Hence higher speed and accuracy can be achieved by extending the pipeline array.

The design and implementation of digital systems is taken up in this thesis. As a case study, the extension of the number of rows of the original generalized pipeline array is discussed. The generalized pipeline cellular array is extended such that it provides an alternative way in arithmetic processor design, in an environment of limited resources while achieving high performance and flexibility. These characteristics make it also suitable for being used as a case study in digital VLSI design education.

Design

Based on the original design of 5 rows as shown in Figure 9, more rows can be added to increase the number of bits upon which can be computed, to achieve higher speed and accuracy. For example, the pipeline array is extended to 7 rows as shown in Figure 10, as additional cells are added in the shaded area.

Once the additional cells are added and connected together, the original design of pipeline array is extended. Notice that the design here only serves as an illustration of the way of extending the pipeline array. In theory, any number of rows can be added to the pipeline array as long as the design can meet the area and timing budget.

The behavior Verilog code for this design is listed in the Appendix.

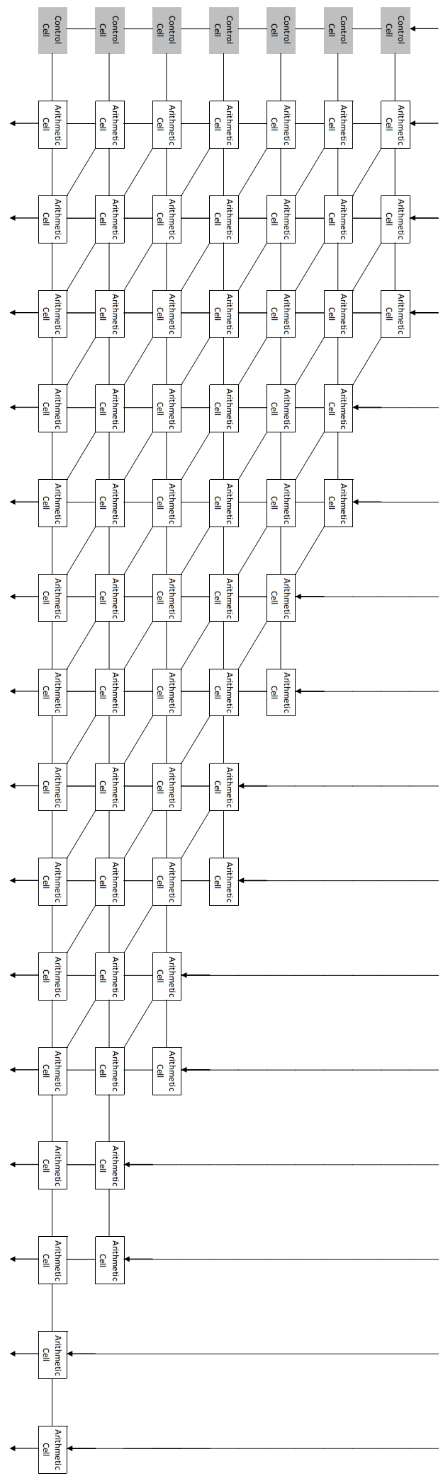
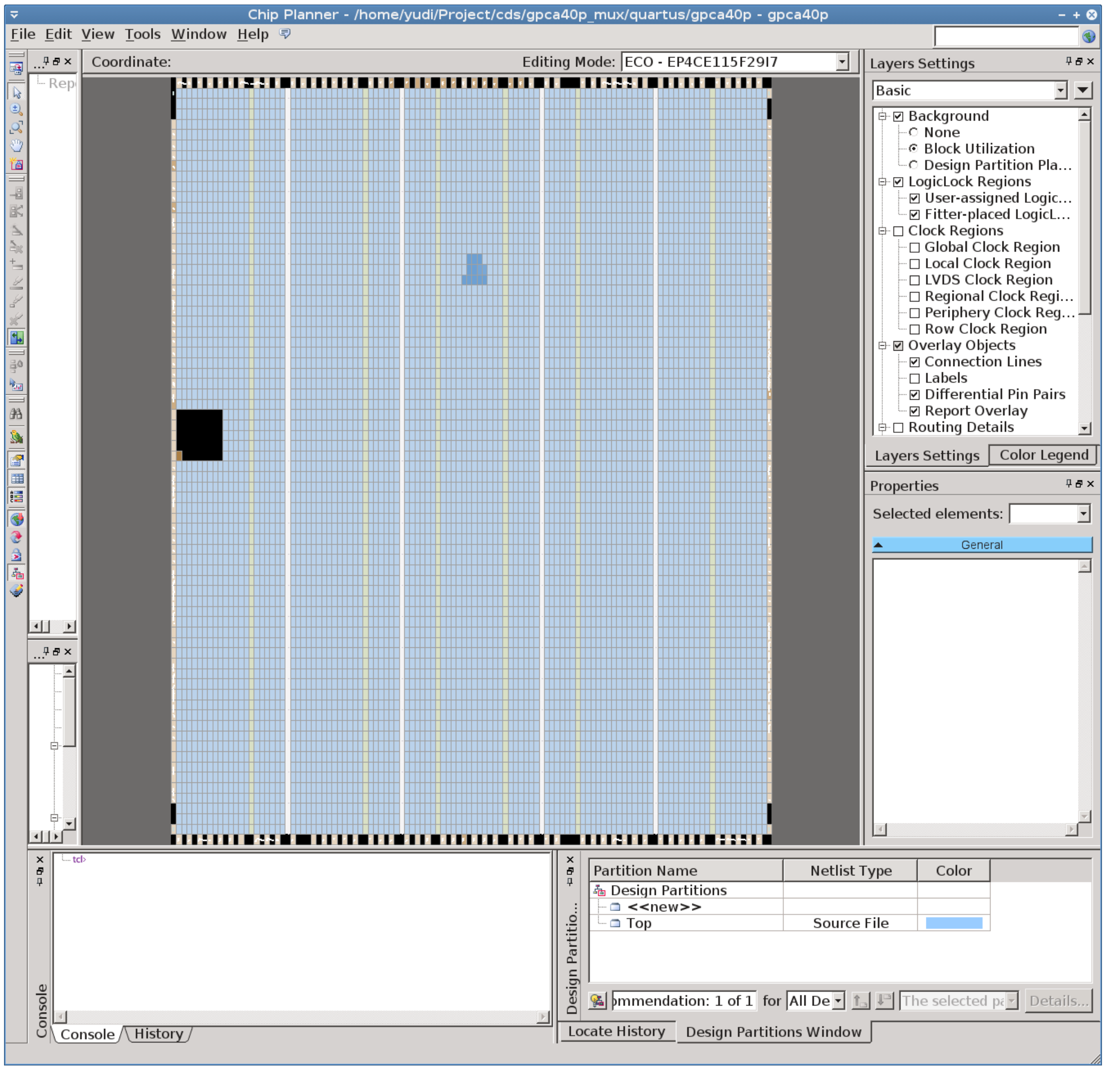


Figure 9: Pipeline array with 7 rows.



Partition Name	Netlist Type	Color
Design Partitions		
<<new>>		
Top	Source File	🟦

Figure 10: Place and route result on FPGA.

Implementation

In this section, a step-by-step procedure for VLSI digital design is given, for the 7-row implementation of the pipeline array.

There are many fabrication technologies available nowadays, from various fabrication facilities such as GlobalFoundries and TSMC, in technologies such as 14 nm, 28 nm, 40 nm, 65 nm, 0.13 µm and 0.18 µm and so on [30]. Since we use the MOSIS Educational Program (MEP) for free fabrication service, which limits us to use the ON Semi 0.50 µm CMOS (C5N) technology [30], C5N is used in this procedure. However, the general procedure is the same for other technologies. There are many computer aided design (CAD) software available for VLSI design, such as Cadence and Synopsys. This procedure uses Cadence Encounter Digital Implementation Systems 14.00, Cadence Virtuoso Design Environment 6.15 along with NCSU CDK 1.6.0 [21], UofU Technology Library and UofU Standard Cell Library [24].

This procedure consists of 3 parts, each of which represents one major step for the VLSI design. They are logic synthesis, place and route and chip assembly respectively, which are introduced for the rest of this section. In the end, the procedure for MOSIS submission is given.

Logic Synthesis

In digital logic design, logic synthesis is a procedure by which a behavior-level HDL code describing the function of a circuit, is turned into a gate-level netlist which describes the implementation of a design in terms of logic gates, typically using a computer program called a synthesis tool.

In this subsection, a procedure for synthesis is given. For the concision of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [31]. There are other alternative RTL synthesizers available as well, such as Design Compiler by Synopsys. If tools other than what's described here are used, it's advised to refer to their respective manuals. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL Compiler

2) Prerequisites before This Step:

Behavior Verilog Code (e.g. "simple.v")

Tcl Script for RC Compiler ("rc.cmd", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Netlist Verilog Code ("nl.v")

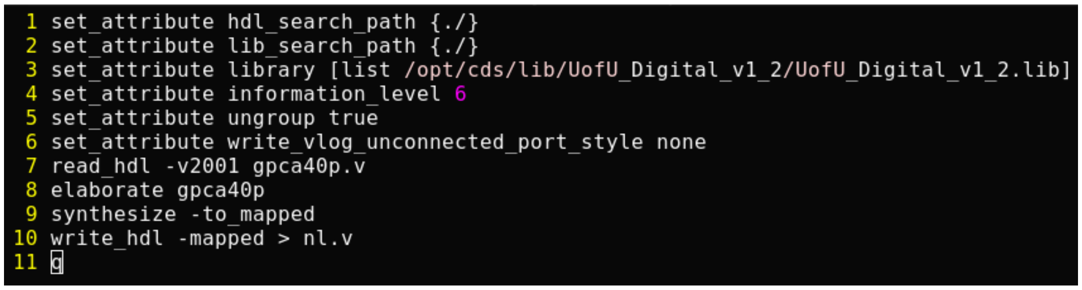
4) Steps:

a) Modify rc.cmd based on the requirement (as shown in Figure 11).

Line 3: Change UofU standard cell library path to where it's installed.

Line 7: Change "gpca40p.v" to the file name of Verilog code (e.g. "simple.v").

Line 8: Change "gpca40p" to the top-level entity name (e.g. "simple").



```
 1 set_attribute hdl_search_path {./}
 2 set_attribute lib_search_path {./}
 3 set_attribute library [list /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib]
 4 set_attribute information_level 6
 5 set_attribute ungroup true
 6 set_attribute write_vlog_unconnected_port_style none
 7 read_hdl -v2001 gpca40p.v
 8 elaborate gpca40p
 9 synthesize -to_mapped
10 write_hdl -mapped > nl.v
11 q
```

Figure 11: rc.cmd.

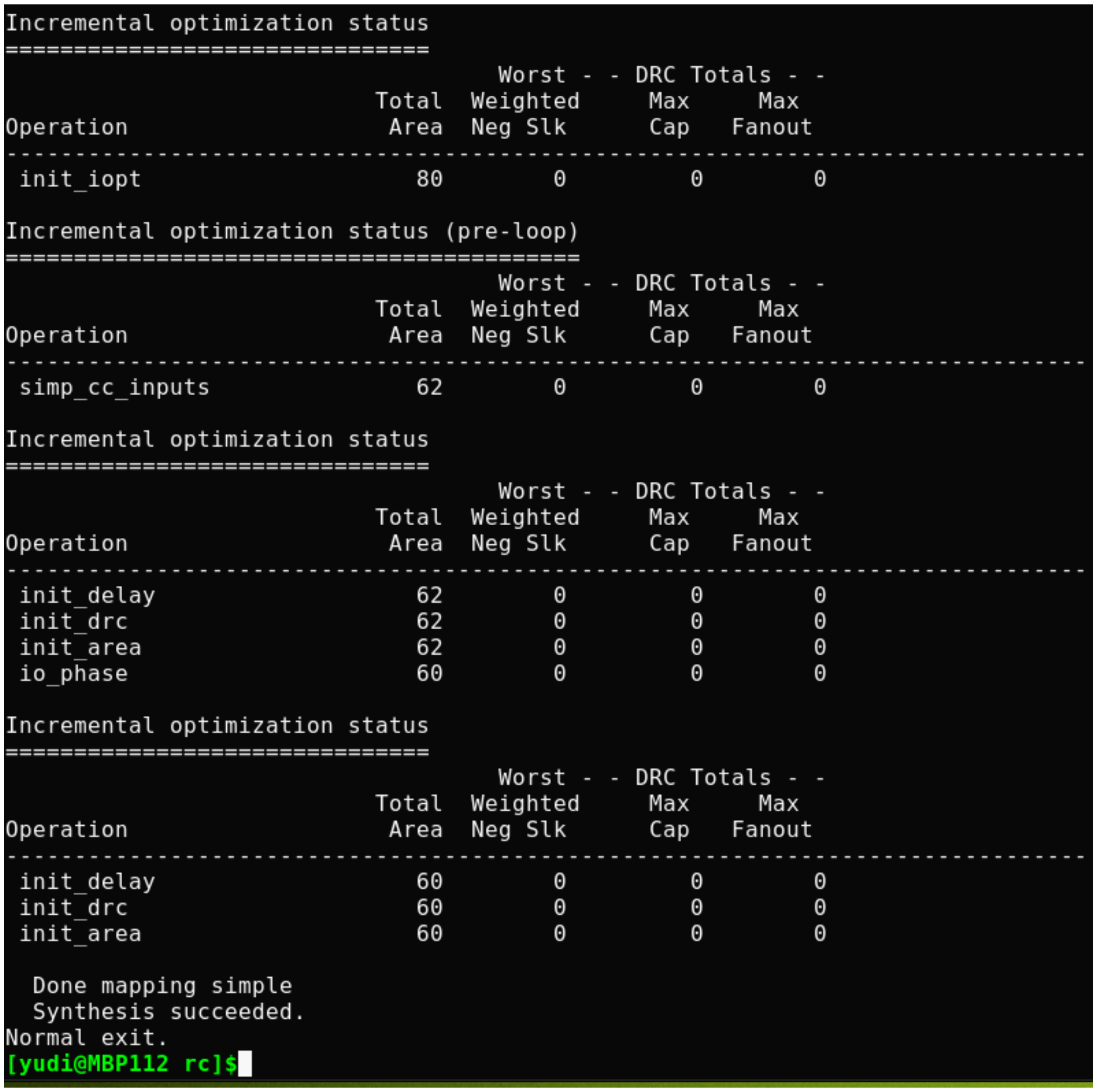
b) Run

$ rc -files rc.cmd

c) Check the result (Figure 12)

In the end, a netlist file called "nl.v" is generated containing information which will be used later for place and route.

If everything goes smooth as above, continue to the next step. If anything goes wrong, fix it first before continuing further.



```
Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_iopt                    80        0        0        0

Incremental optimization status (pre-loop)
==========================================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 simp_cc_inputs               62        0        0        0

Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   62        0        0        0
 init_drc                     62        0        0        0
 init_area                    62        0        0        0
 io_phase                     60        0        0        0

Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   60        0        0        0
 init_drc                     60        0        0        0
 init_area                    60        0        0        0

  Done mapping simple
  Synthesis succeeded.
Normal exit.
[yudi@MBP112 rc]$
```

Figure 12: Synthesis summary generated by RC Compiler.

Place and Route

"Place and route" is a stage in the process of VLSI design, in which the location to place all the logic elements within a generally limited amount of space and the way of all the wires needed to connect the logic elements are decided.

In this section, a procedure for "place and route" is given. For the brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [33]. If the reader is interested in using GUI commands instead of TCL scripts, please refer to EDI System Menu Reference [34] and textbooks [24] for more information. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL-to-GDSII System

2) Prerequisites before This Step:

Netlist Verilog Code ("nl.v" from the last step)

Tcl Script for RC Compiler ("encounter.cmd", given in Appendix)

Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", given in Appendix)

Synopsys Design Constraints ("typical.sdc", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Optimized Netlist Verilog Code ("nlopt.v")

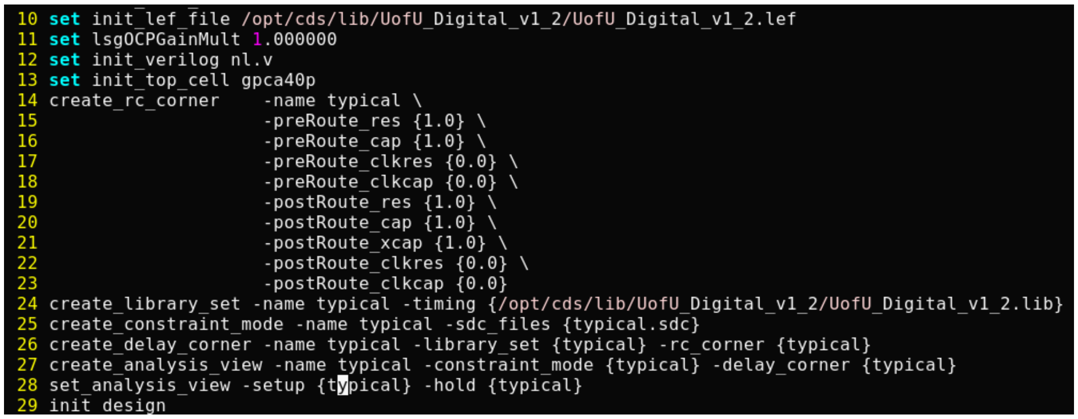
Design Exchange Format (DEF) File (e.g. "simple.def")

4) Steps:

a) Modify Tcl Script for RC Compiler ("encounter.cmd", Figure 13)

Line 10, 24: May change UofU standard cell library path to where it's installed

Line 13, 99: Change "gpca40p" to the top-level entity name (e.g. "simple")

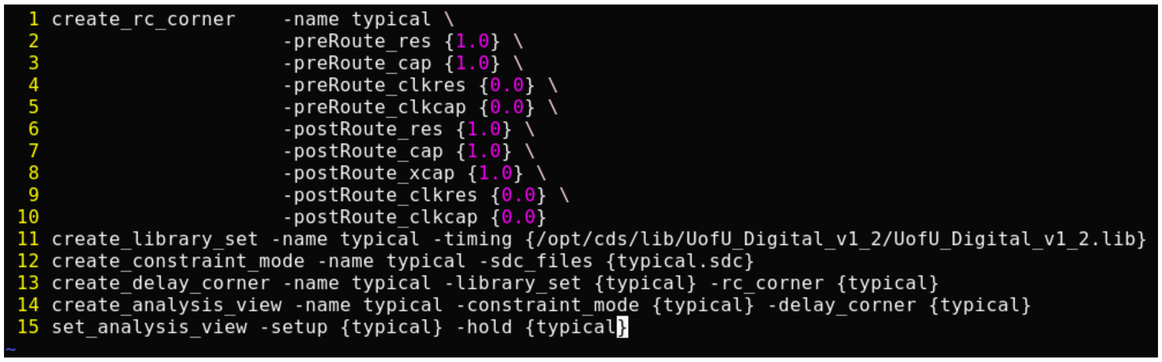


```
10 set init_lef_file /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lef
11 set lsgOCPGainMult 1.000000
12 set init_verilog nl.v
13 set init_top_cell gpca40p
14 create_rc_corner    -name typical \
15                     -preRoute_res {1.0} \
16                     -preRoute_cap {1.0} \
17                     -preRoute_clkres {0.0} \
18                     -preRoute_clkcap {0.0} \
19                     -postRoute_res {1.0} \
20                     -postRoute_cap {1.0} \
21                     -postRoute_xcap {1.0} \
22                     -postRoute_clkres {0.0} \
23                     -postRoute_clkcap {0.0}
24 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
25 create_constraint_mode -name typical -sdc_files {typical.sdc}
26 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
27 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
28 set_analysis_view -setup {typical} -hold {typical}
29 init_design
```

Figure 13: encounter.cmd.

b) Modify Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", Figure 14)

Line 11: May change UofU standard cell library path to where it's installed



```
 1 create_rc_corner    -name typical \
 2                     -preRoute_res {1.0} \
 3                     -preRoute_cap {1.0} \
 4                     -preRoute_clkres {0.0} \
 5                     -preRoute_clkcap {0.0} \
 6                     -postRoute_res {1.0} \
 7                     -postRoute_cap {1.0} \
 8                     -postRoute_xcap {1.0} \
 9                     -postRoute_clkres {0.0} \
10                     -postRoute_clkcap {0.0}
11 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
12 create_constraint_mode -name typical -sdc_files {typical.sdc}
13 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
14 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
15 set_analysis_view -setup {typical} -hold {typical}
```

Figure 14: mmmc.tcl.

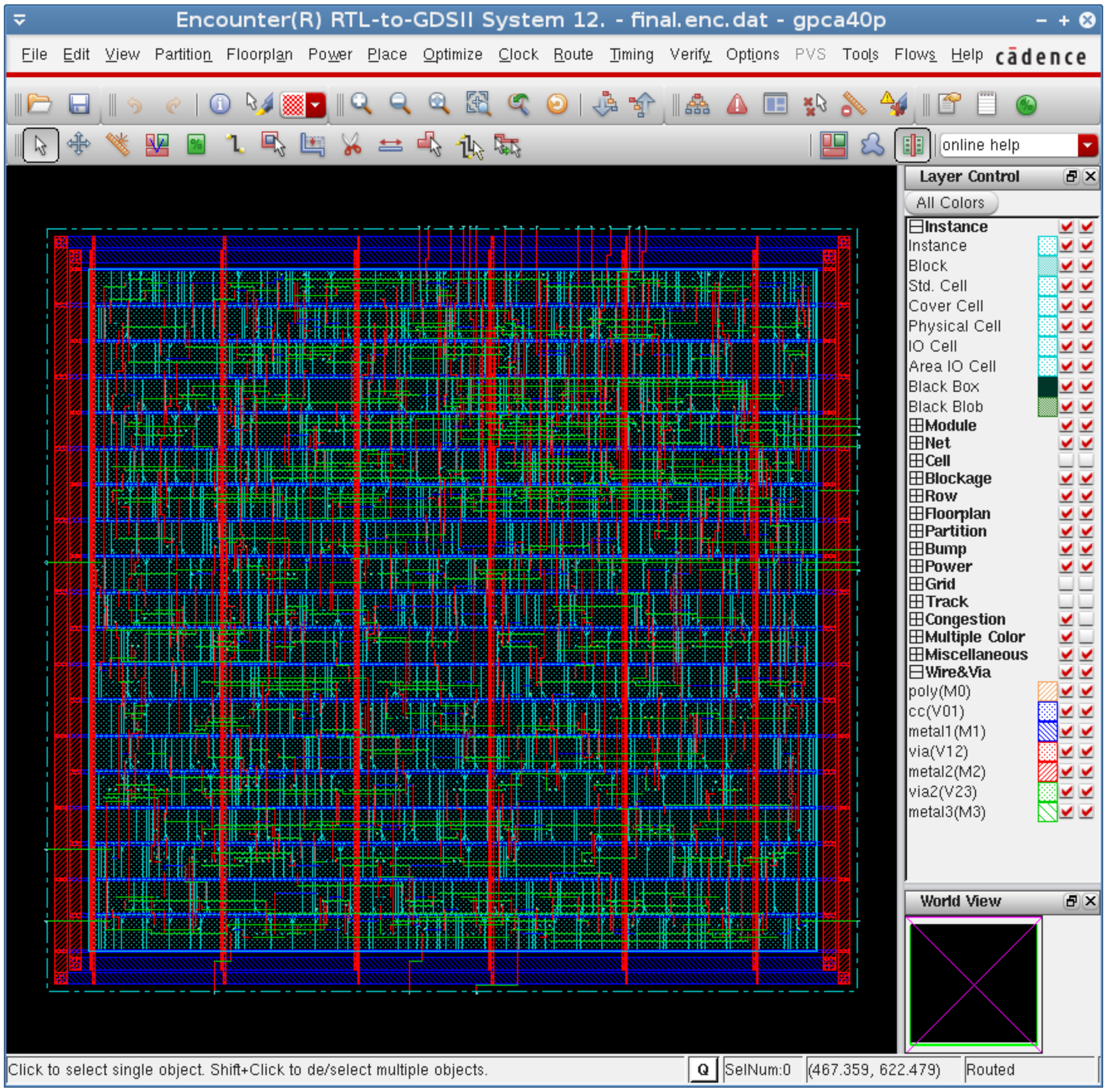
c) Run

$ encounter -init encounter.cmd

d) Check the result

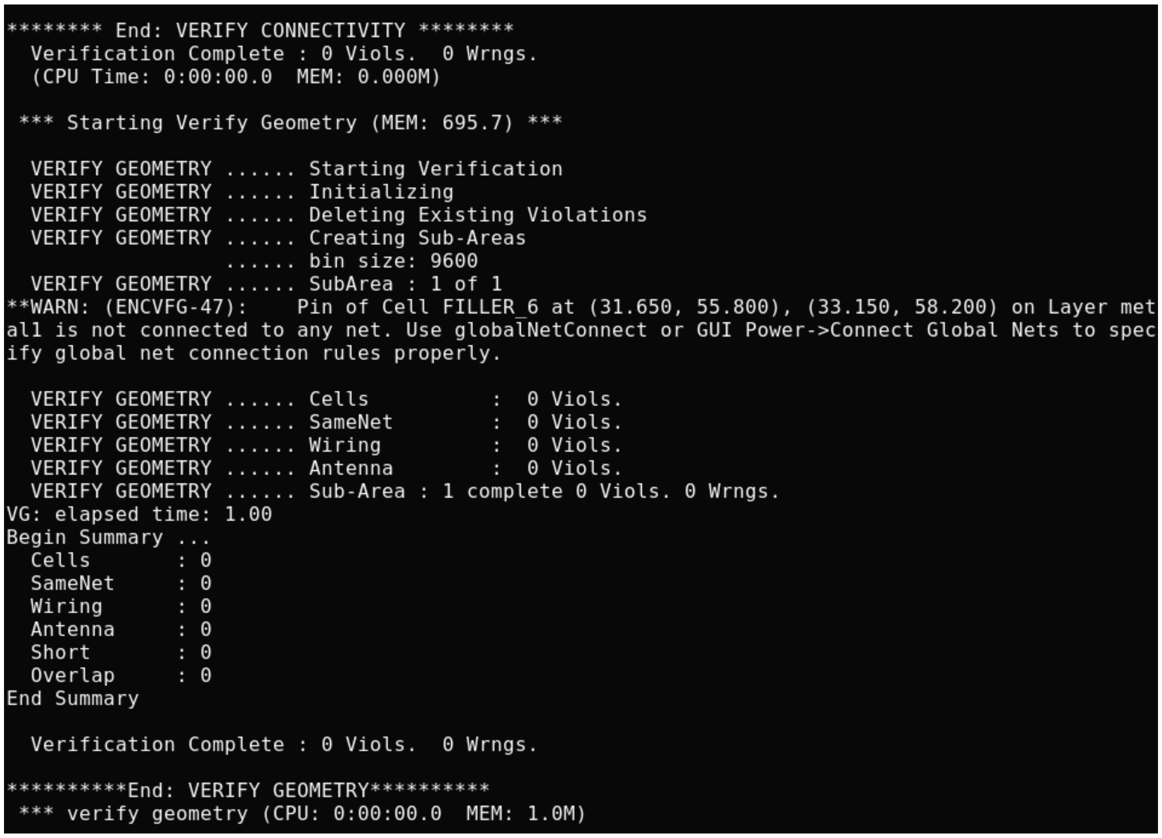
It should run all the way to the final step without any errors if the above steps are

followed correctly, as shown in Figure 15 and Figure 16. In the end, a DEF file (e.g. "simple.def") for the chip layout (without pad frame) as well as a netlist file called "nlopt.v" is generated.



Click to select single object. Shift+Click to de/select multiple objects. SelNum:0 (467.359, 622.479) Routed

Figure 15: Final view in Encounter.



```
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)

 *** Starting Verify Geometry (MEM: 695.7) ***

  VERIFY GEOMETRY ...... Starting Verification
  VERIFY GEOMETRY ...... Initializing
  VERIFY GEOMETRY ...... Deleting Existing Violations
  VERIFY GEOMETRY ...... Creating Sub-Areas
                  ...... bin size: 9600
  VERIFY GEOMETRY ...... SubArea : 1 of 1
**WARN: (ENCVFG-47):    Pin of Cell FILLER_6 at (31.650, 55.800), (33.150, 58.200) on Layer met
al1 is not connected to any net. Use globalNetConnect or GUI Power->Connect Global Nets to spec
ify global net connection rules properly.

  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: 1.00
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: 0:00:00.0  MEM: 1.0M)
```

Figure 16: DRC report generated by Encounter.

Chip Assembly

As a requirement by AMI05 technology [35], design should be submitted along with pad frame. This section gives the procedure to assemble the pad frame with the chip. For brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [36]. Materials such as textbooks [24] also have many useful information for reference. The detailed procedure and codes used are included in [32].

1) Tools:

Cadence Virtuoso Design Environment

2) Prerequisites before this step:

Optimized Netlist Verilog Code ("nlopt.v" from last step)

DEF File (e.g. "simple.def" from last step)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

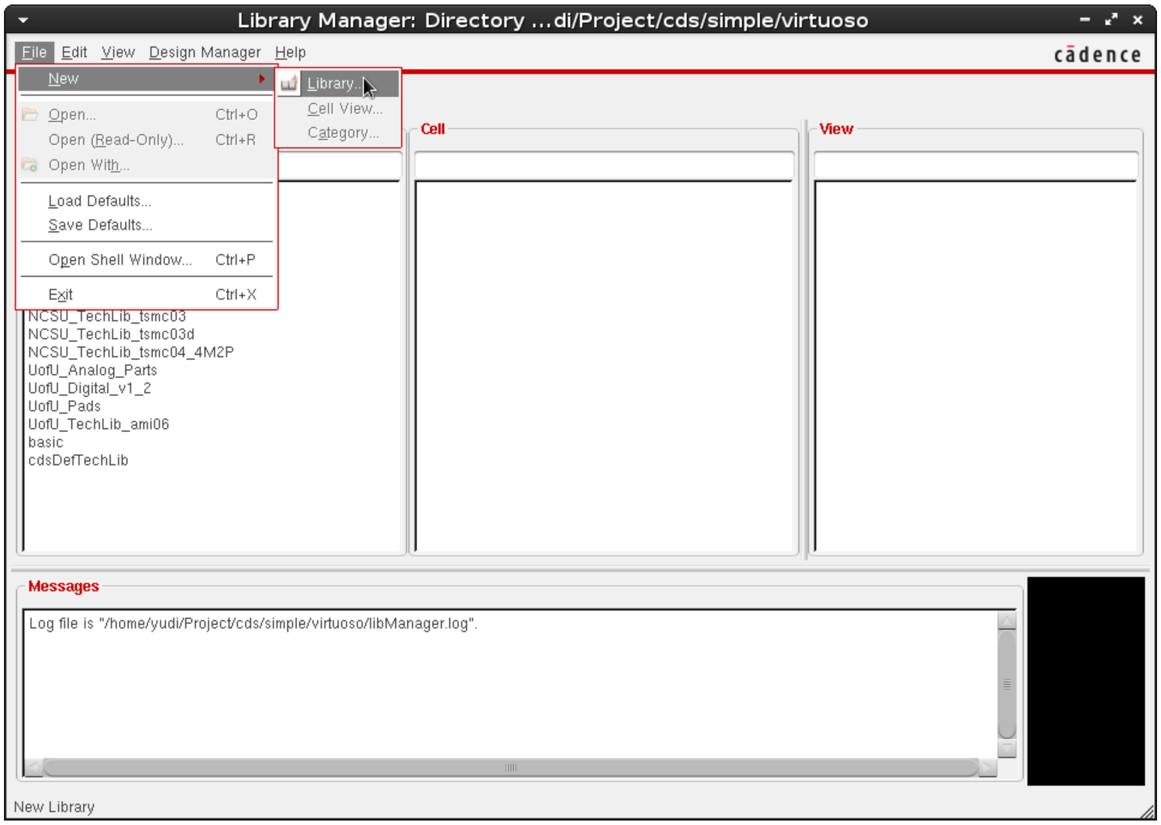
GDSII Stream File (e.g. "simple_final.gds")

4) Steps:

a) Launch Cadence Virtuoso Design Environment

```
$ virtuoso
```

b) Create New Library (Figure 17 to Figure 20)



NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts
UofU_Digital_v1_2
UofU_Pads
UofU_TechLib_ami06
basic
cdsDefTechLib

```
Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".
```

Figure 17: Create New Library with Virtuoso.

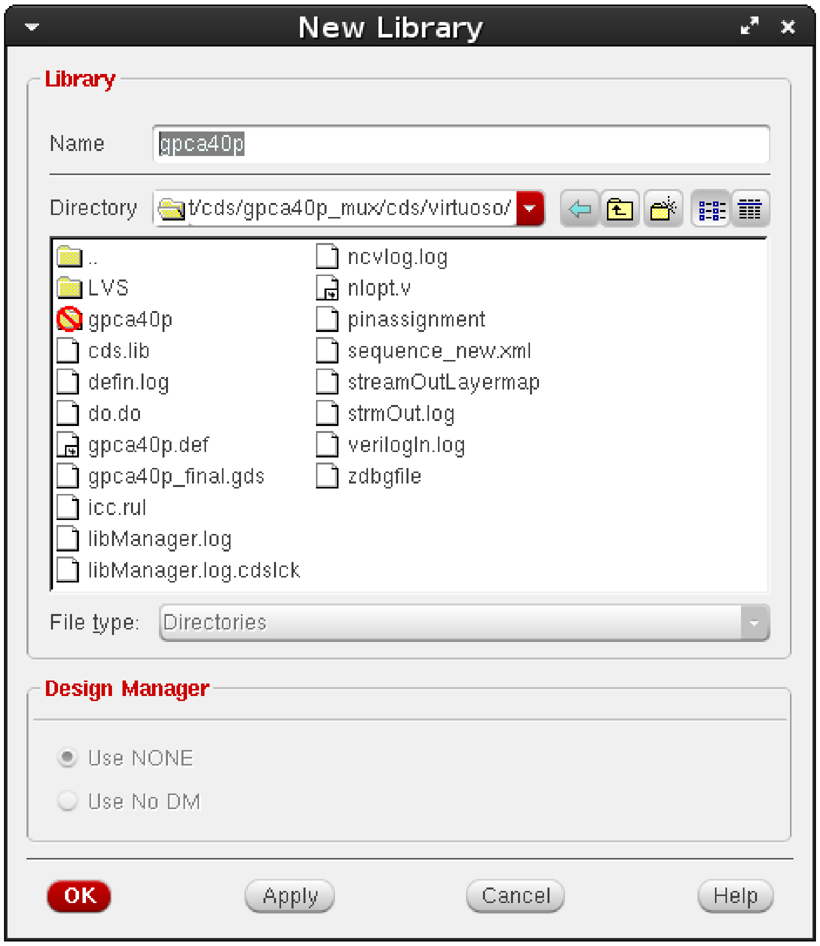
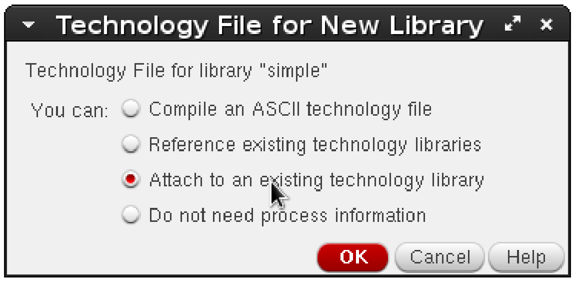


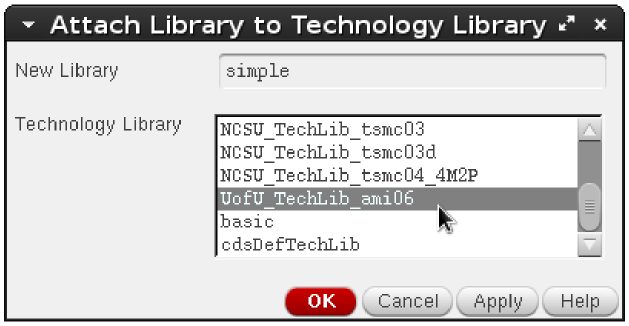
Figure 18: "New Library" window of Virtuoso.


Technology File for New Library

Technology File for library "simple"

You can: Compile an ASCII technology file [ ] Reference existing technology libraries [ ] [✓ Attach to an existing technology library] Do not need process information [ ]

Figure 19: "Technology File for New Library" window of Virtuoso.


Attach Library to Technology Library

New Library: simple

Technology Library:

```
NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_TechLib_ami06
basic
cdsDefTechLib
```

Figure 20: "Technology File for New Library" window of Virtuoso.

c) Import Optimized Netlist Verilog Code ("nlopt.v" from last step) and DEF File (e.g. "simple.def" from last step). As shown in Figure 21 to Figure 25.

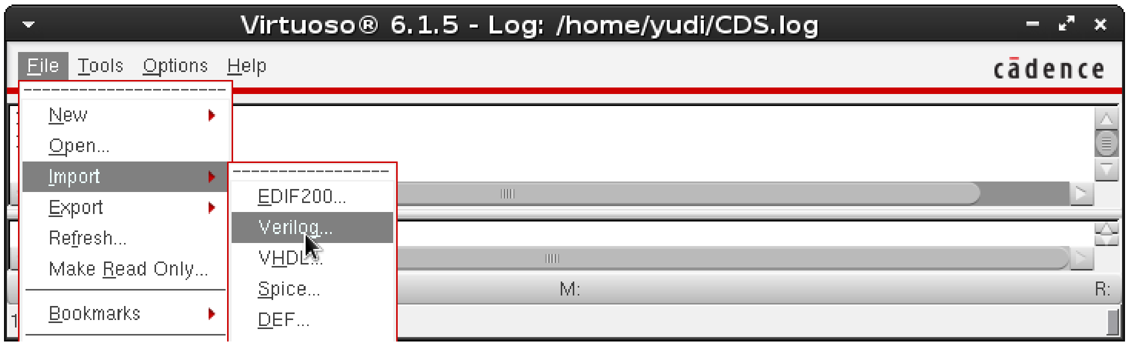
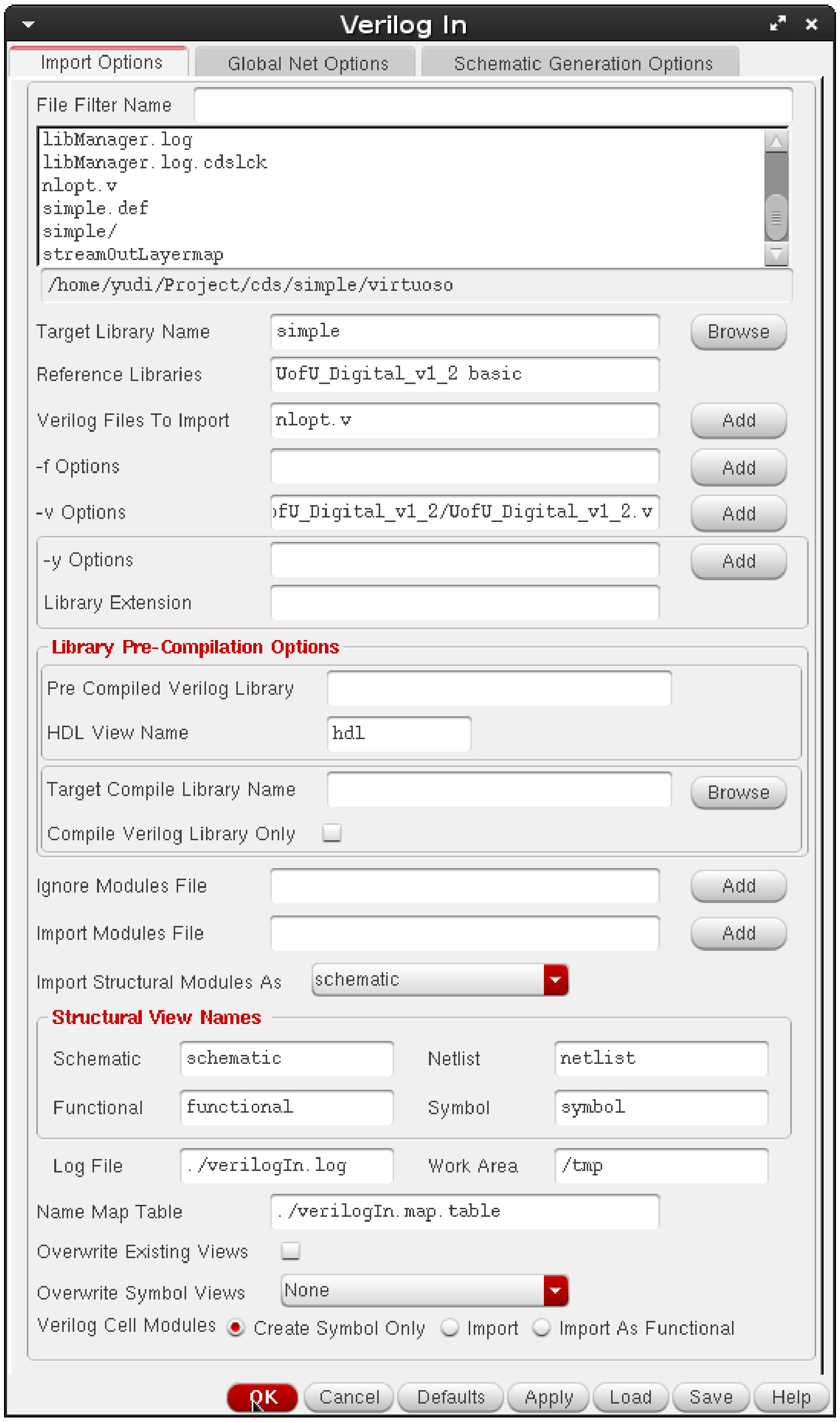


Figure 21: Import netlist with Virtuoso.



File Filter Name

```
libManager.log
libManager.log.cdslck
nlopt.v
simple.def
simple/
streamOutLayermap
/home/yudi/Project/cds/simple/virtuoso
```

Target Library Name	simple
Reference Libraries	UofU_Digital_v1_2 basic
Verilog Files To Import	nlopt.v
-f Options	
-v Options	›fU_Digital_v1_2/UofU_Digital_v1_2.v
-y Options	
Library Extension	

Library Pre-Compilation Options

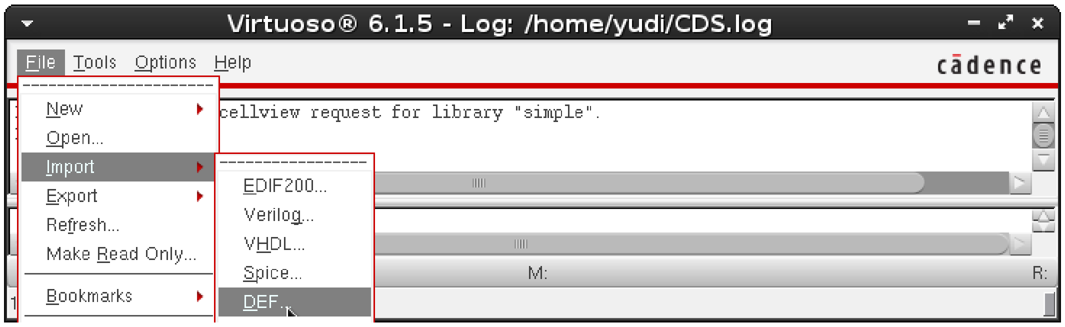
Pre Compiled Verilog Library	
HDL View Name	hdl
Target Compile Library Name	
Compile Verilog Library Only	☐
Ignore Modules File	
Import Modules File	
Import Structural Modules As	schematic

Structural View Names

Schematic	schematic	Netlist	netlist
Functional	functional	Symbol	symbol
Log File	./verilogIn.log	Work Area	/tmp

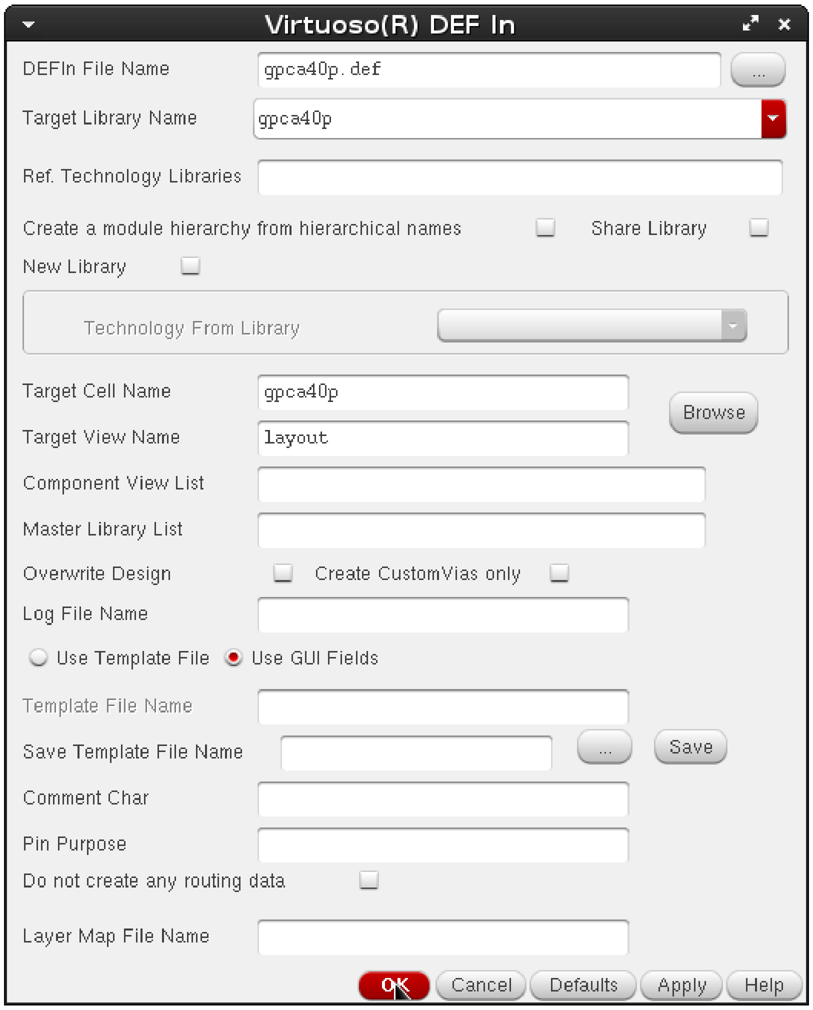
Name Map Table	./verilogIn.map.table
Overwrite Existing Views	☐
Overwrite Symbol Views	None
Verilog Cell Modules	[✓ Create Symbol Only] Import Import As Functional

Figure 22: "Verilog In" window of Virtuoso.



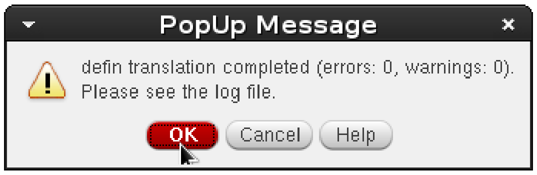
```
cellview request for library "simple".
```

Figure 23: Import DEF with Virtuoso.



DEFIn File Name	gpca40p.def
Target Library Name	gpca40p
Ref. Technology Libraries	
Create a module hierarchy from hierarchical names	☐
Share Library	☐
New Library	☐
Technology From Library	
Target Cell Name	gpca40p
Target View Name	layout
Component View List	
Master Library List	
Overwrite Design	☐
Create CustomVias only	☐
Log File Name	
Use Template File [✓ Use GUI Fields]	
Template File Name	
Save Template File Name	
Comment Char	
Pin Purpose	
Do not create any routing data	☐
Layer Map File Name	

Figure 24: "DEF In" window of Virtuoso.


PopUp Message

defin translation completed (errors: 0, warnings: 0).
Please see the log file.

Figure 25: "DEF In" successful translation prompt.

Once both layouts (from DEF) and schematic (from netlist in this section) are generated, DRC, Extract and LVS should be performed, as shown in Figure 26 to 34.

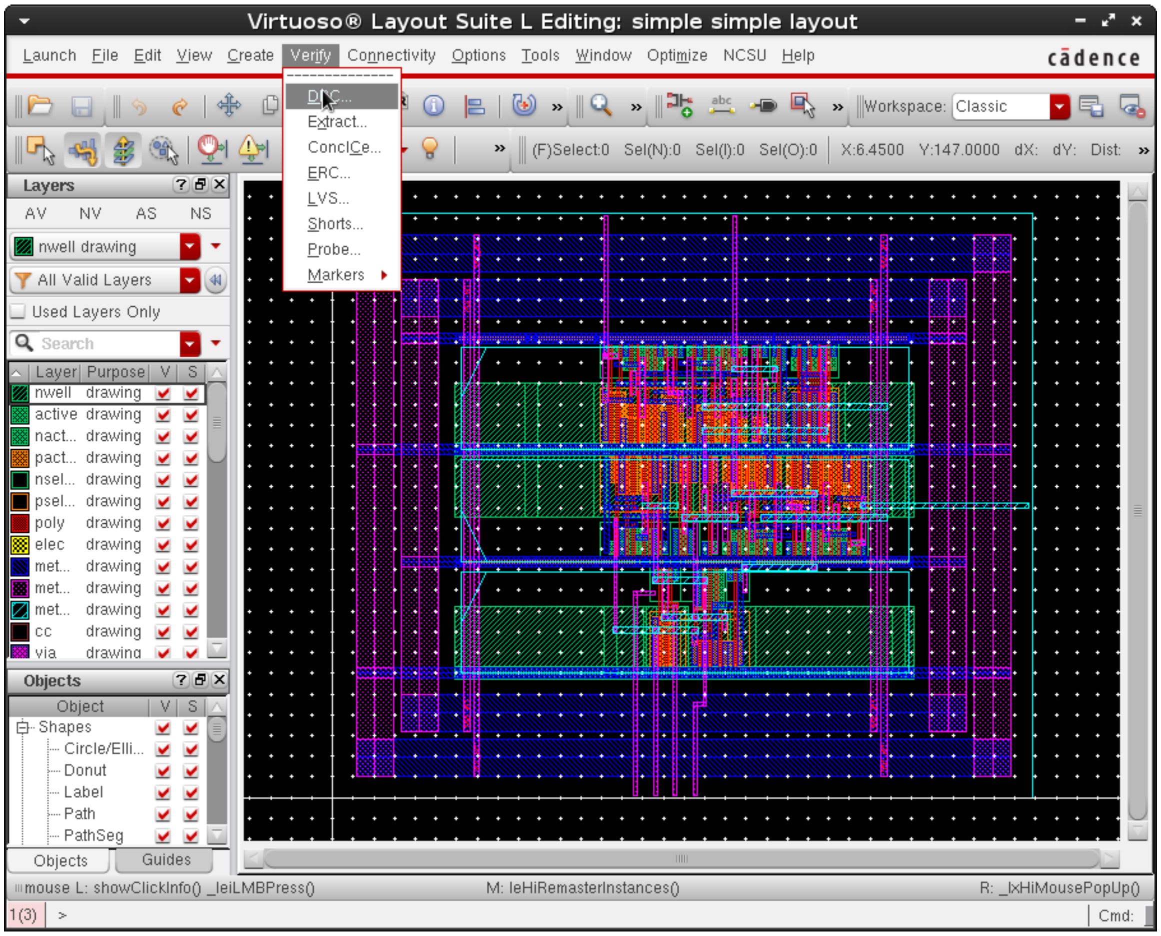
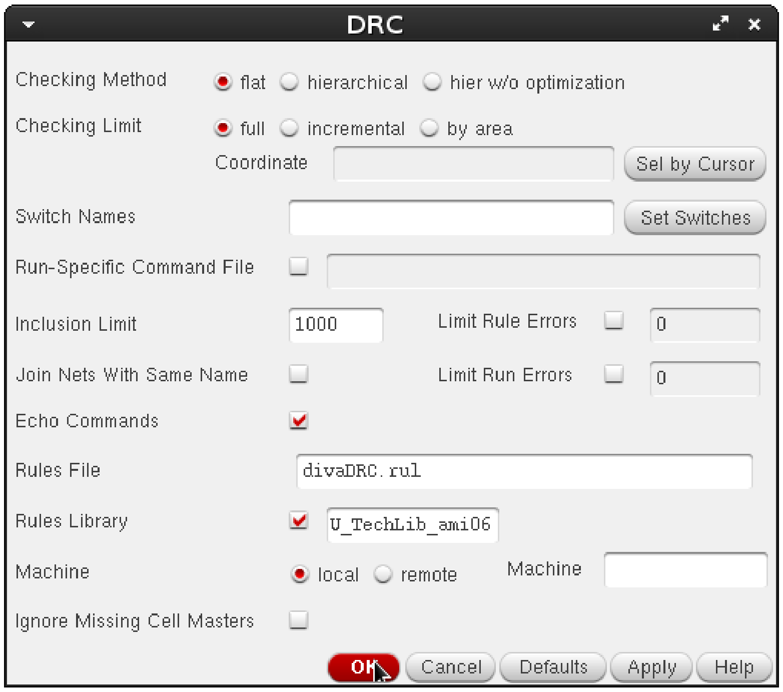
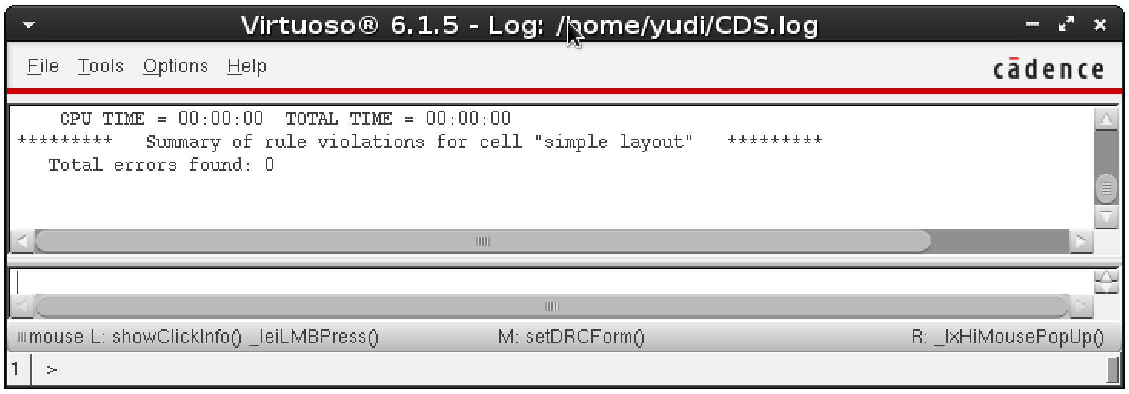

Virtuoso® Layout Suite L Editing: simple simple layout

Figure 26: Start DRC from Virtuoso Layout Suite.



Checking Method [✓ flat] hierarchical hier w/o optimization
Checking Limit [✓ full] incremental by area
Coordinate
Switch Names
Run-Specific Command File ☐
Inclusion Limit 1000
Limit Rule Errors ☐ 0
Join Nets With Same Name ☐
Limit Run Errors ☐ 0
Echo Commands ✓
Rules File divaDRC.rul
Rules Library ✓ U_TechLib_ami06
Machine [✓ local] remote
Machine
Ignore Missing Cell Masters ☐

Figure 27: "DRC" window of Virtuoso.



```
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violations for cell "simple layout"   *********
   Total errors found: 0
```

mouse L: showClickInfo() _leiLMBPress() M: setDRCForm() R: _lxHiMousePopUp()

1 >

Figure 28: Report of successful DRC.

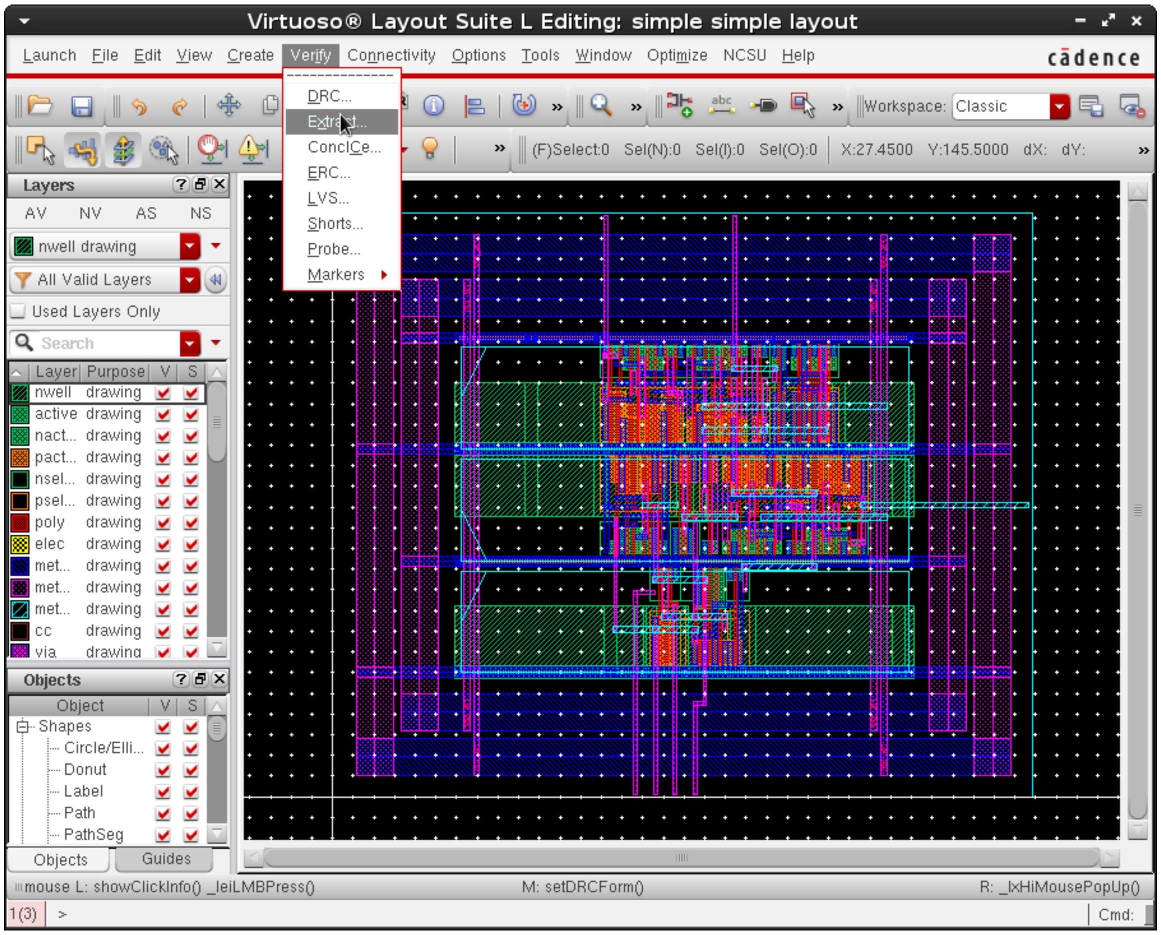
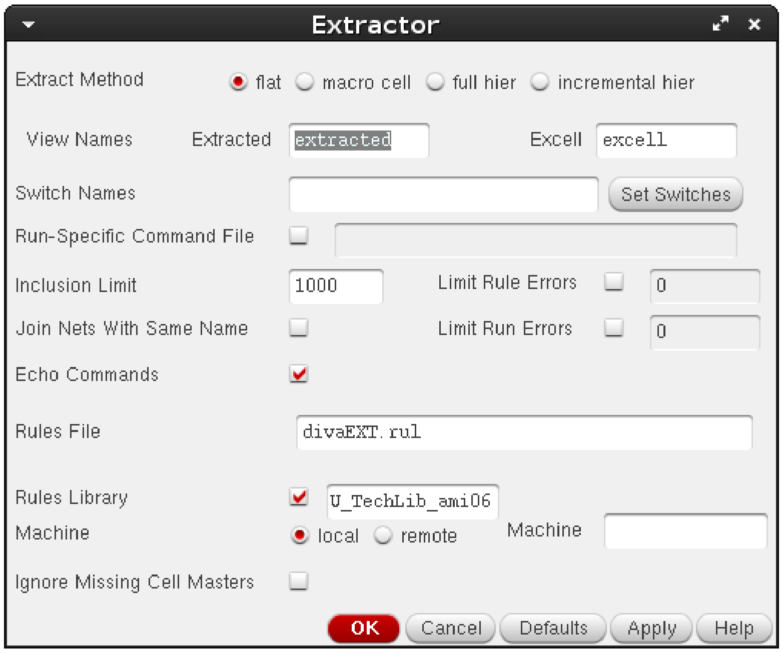


Figure 29: Start Extract from Virtuoso Layout Suite.



Extract Method [✓ flat] macro cell full hier incremental hier

View Names Extracted extracted Excell excell

Switch Names

Run-Specific Command File ☐

Inclusion Limit 1000 Limit Rule Errors ☐ 0

Join Nets With Same Name ☐ Limit Run Errors ☐ 0

Echo Commands ✓

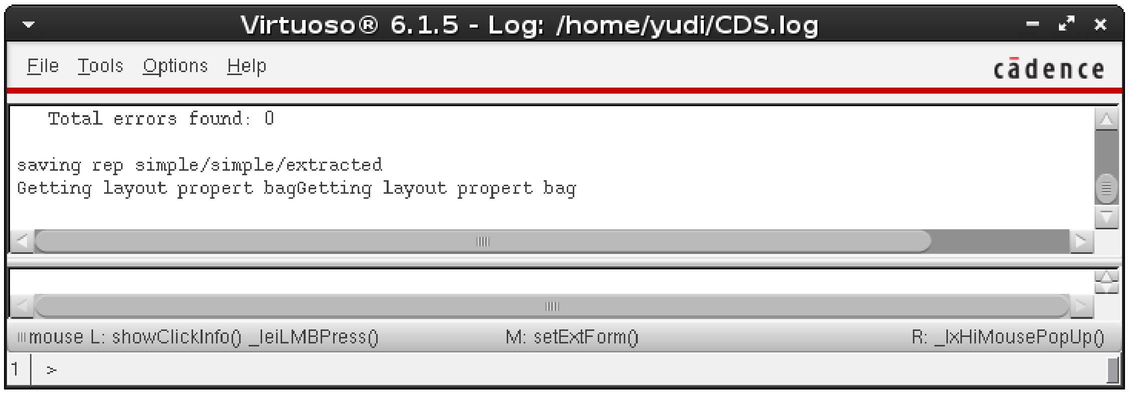
Rules File divaEXT.rul

Rules Library ✓ U_TechLib_ami06

Machine [✓ local] remote Machine

Ignore Missing Cell Masters ☐

Figure 30: "Extractor" window of Virtuoso.



```
   Total errors found: 0

saving rep simple/simple/extracted
Getting layout propert bagGetting layout propert bag
```

mouse L: showClickInfo() _leiLMBPress() M: setExtForm() R: _lxHiMousePopUp()

1 >

Figure 31: Report of successful Extract.

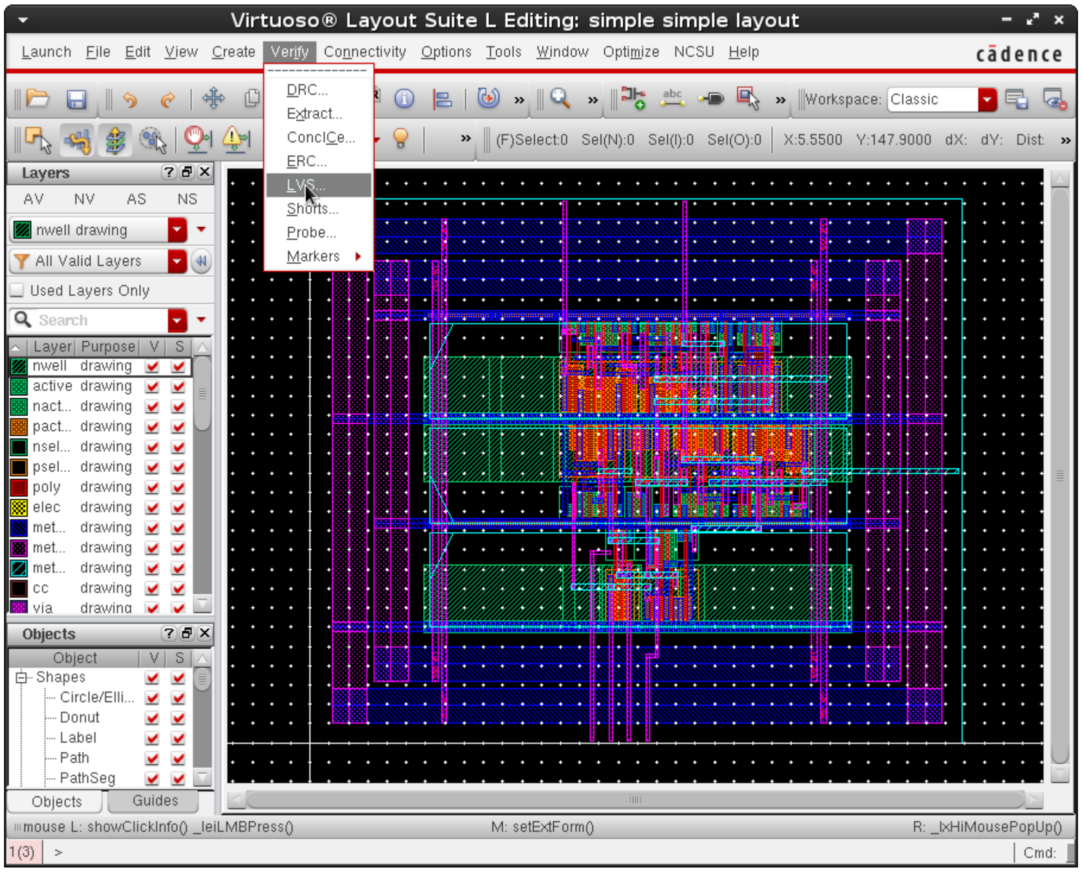
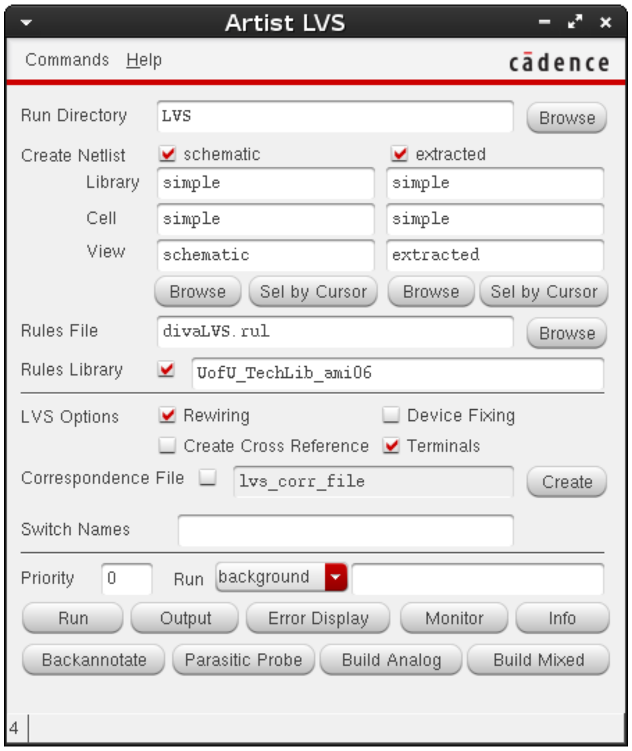


Figure 32: Start LVS from Virtuoso Layout Suite.


Artist LVS
Commands Help
cādence
Browse
Browse Sel by Cursor Browse Sel by Cursor
Browse
Create
Run Output Error Display Monitor Info
Backannotate Parasitic Probe Build Analog Build Mixed
4

Run Directory: LVS

Create Netlist: [✓ schematic] [✓ extracted]

	schematic	extracted
Library	simple	simple
Cell	simple	simple
View	schematic	extracted

Rules File: divaLVS.rul

Rules Library: [✓] UofU_TechLib_ami06

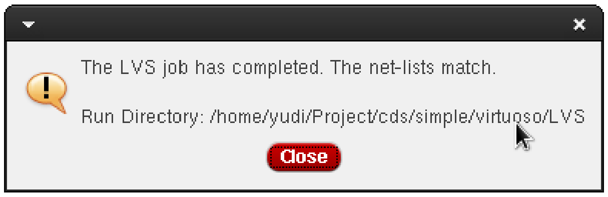
LVS Options: [✓ Rewiring] Device Fixing, Create Cross Reference [✓ Terminals]

Correspondence File: lvs_corr_file

Switch Names:

Priority: 0 Run: background

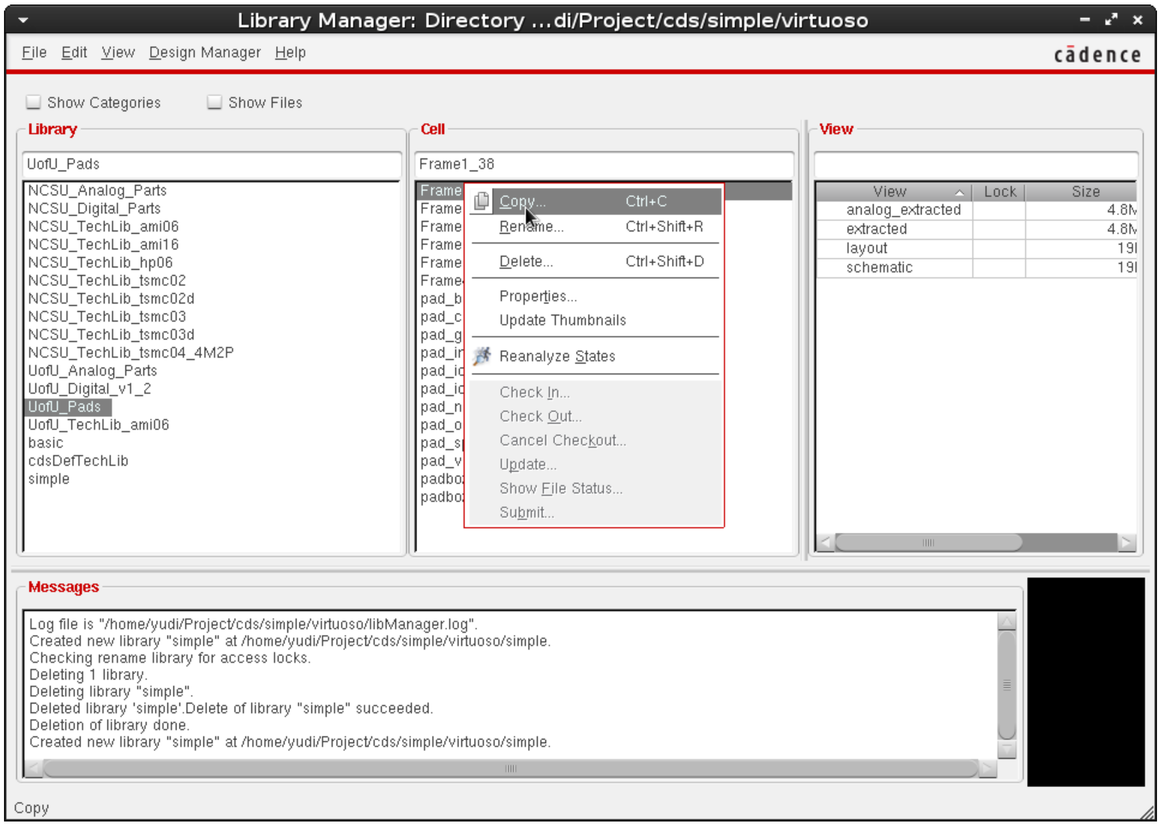
Figure 33: "Artist LVS" window of Virtuoso.



```
The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS
```

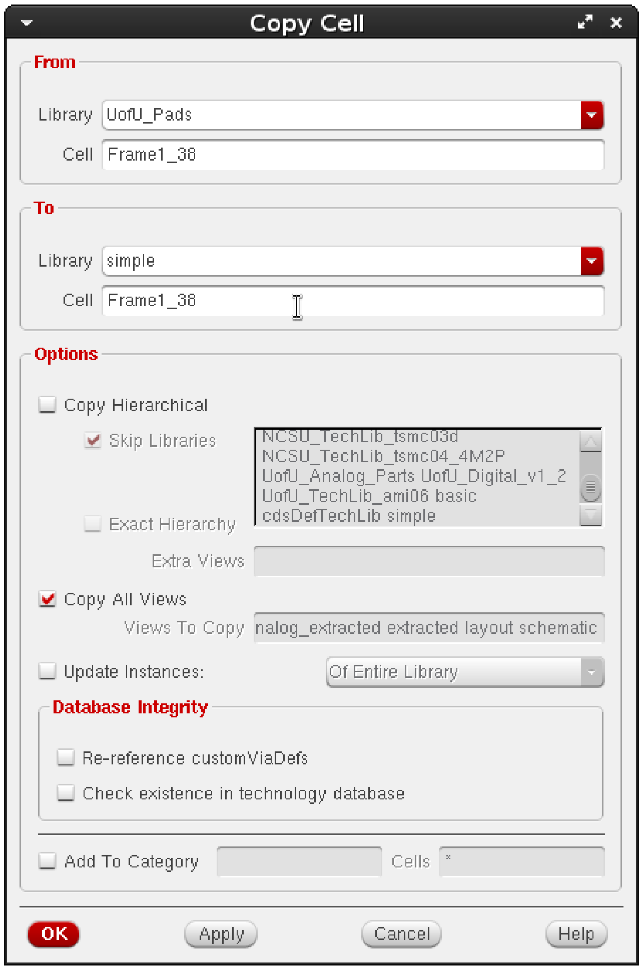
Figure 34: "Artist LVS" successful LVS prompt.

d) Customize Pad Frame (Figure 35 to 41)



```
Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
Checking rename library for access locks.
Deleting 1 library.
Deleting library "simple".
Deleted library 'simple'.Delete of library "simple" succeeded.
Deletion of library done.
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
```

Figure 35: Copy pad frame in Virtuoso.


Copy Cell

From

Library UofU_Pads

Cell Frame1_38

To

Library simple

Cell Frame1_38

Options

[ ] Copy Hierarchical

[✓] Skip Libraries

NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts UofU_Digital_v1_2
UofU_TechLib_ami06 basic
cdsDefTechLib simple

[ ] Exact Hierarchy

Extra Views

[✓] Copy All Views

Views To Copy nalog_extracted extracted layout schematic

[ ] Update Instances: Of Entire Library

Database Integrity

[ ] Re-reference customViaDefs

[ ] Check existence in technology database

[ ] Add To Category Cells *

OK Apply Cancel Help

Figure 36: "Copy Cell" window of Virtuoso.

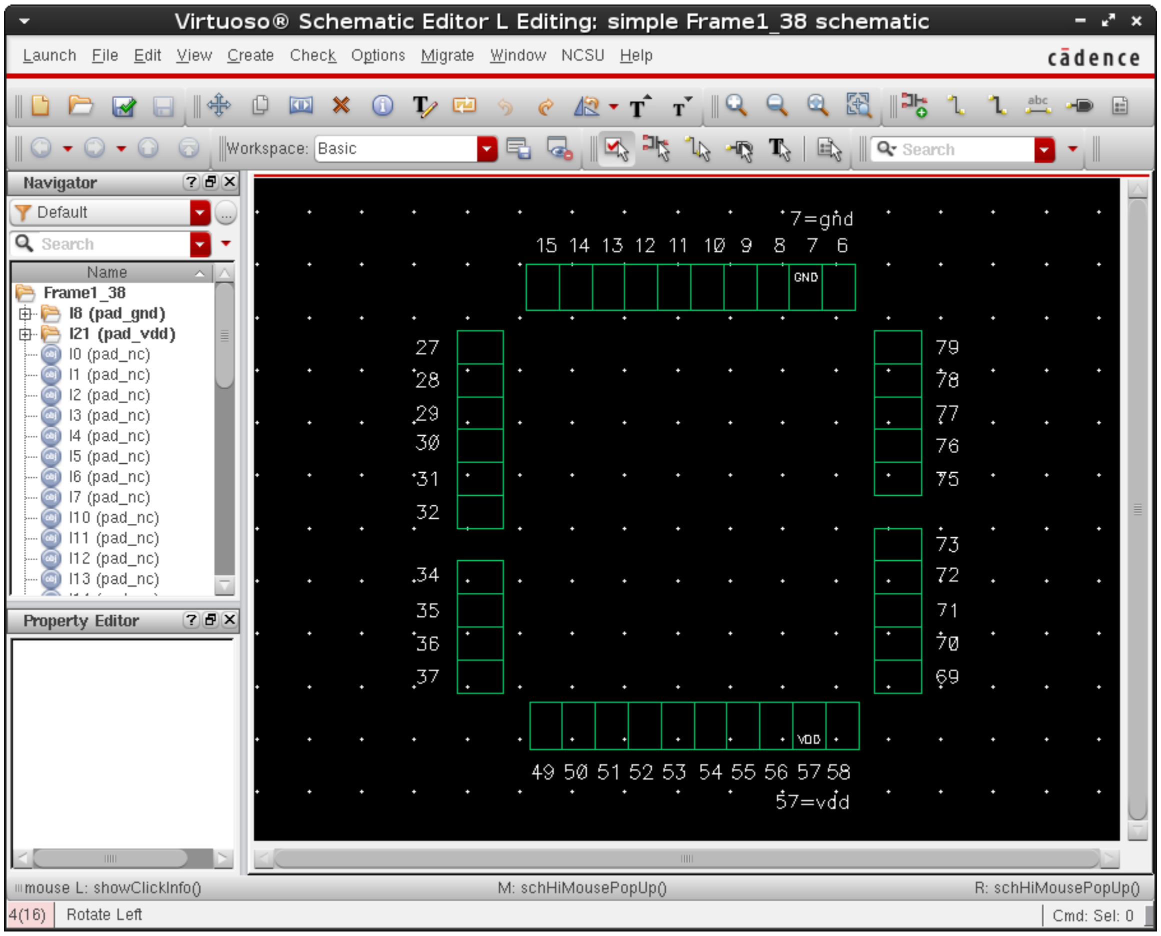


Figure 37: Initial schematic view of the pad frame.

Modify the used pads in the pad frame (both schematic and layout), from "pad_nc" to "pad_in" or "pad_out".

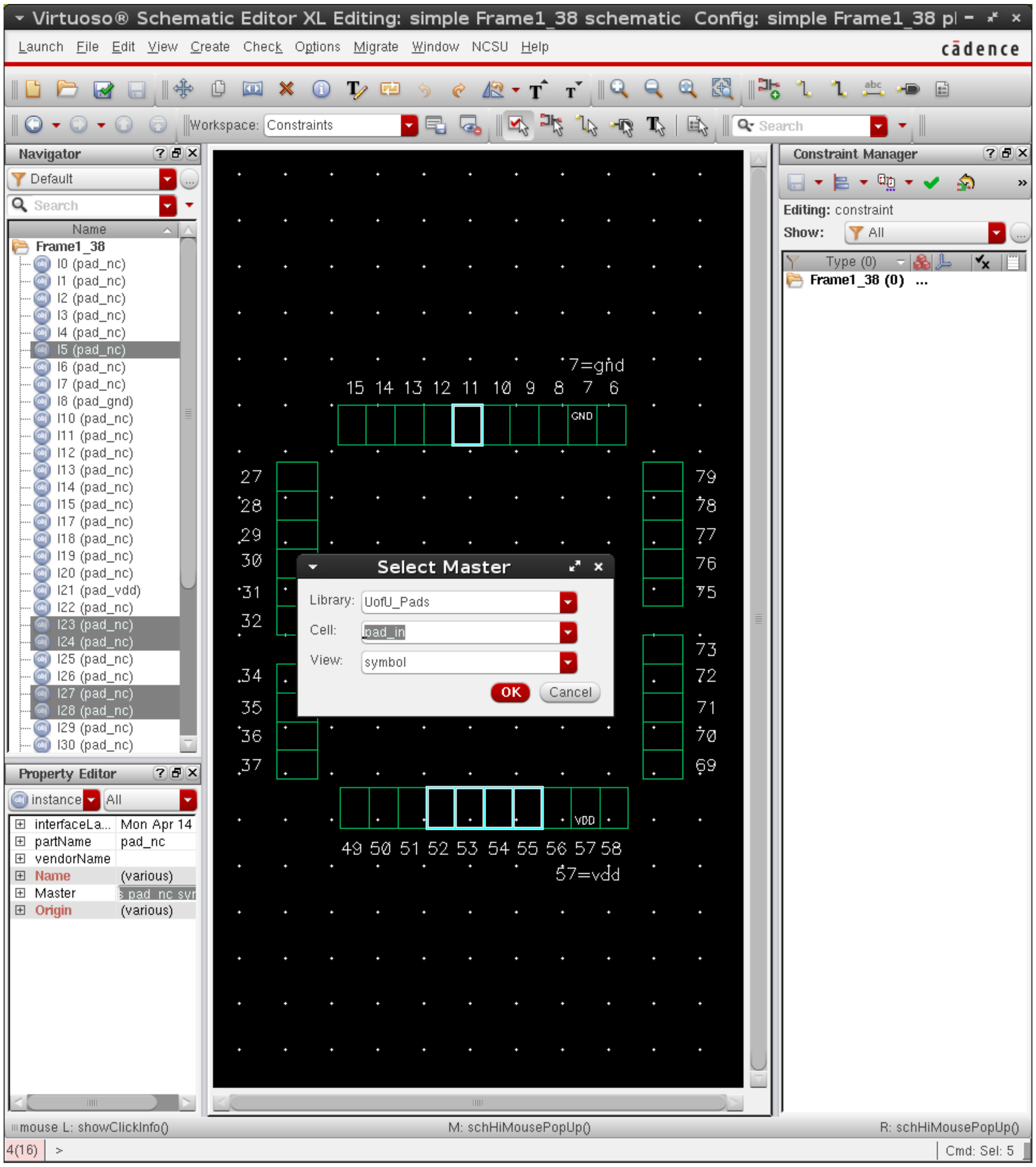
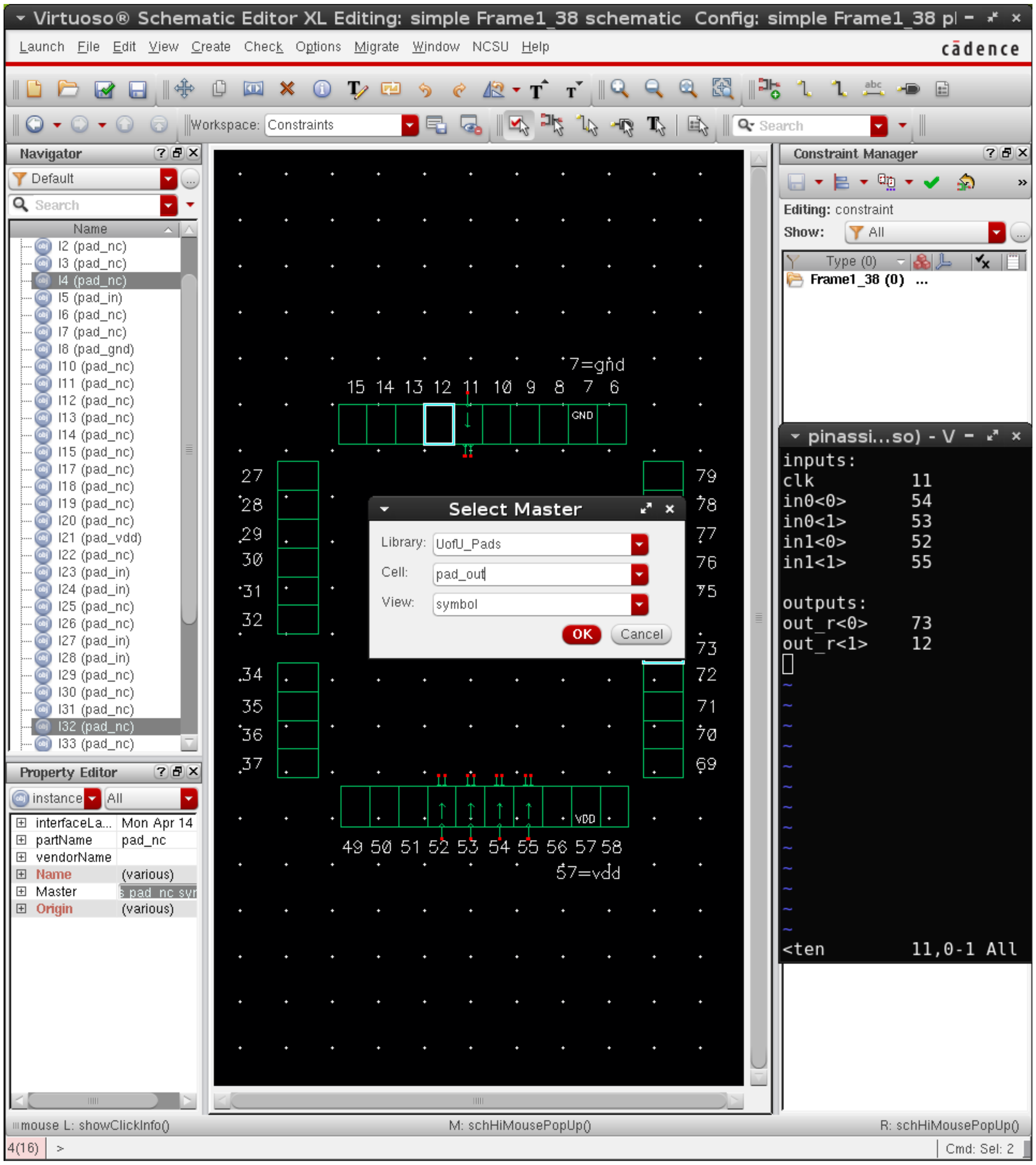


Figure 38: Add input pads in schematic.



I2 (pad_nc)
I3 (pad_nc)
I4 (pad_nc)
I5 (pad_in)
I6 (pad_nc)
I7 (pad_nc)
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)
I18 (pad_nc)
I19 (pad_nc)
I20 (pad_nc)
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_in)
I24 (pad_in)
I25 (pad_nc)
I26 (pad_nc)
I27 (pad_in)
I28 (pad_in)
I29 (pad_nc)
I30 (pad_nc)
I31 (pad_nc)
I32 (pad_nc)
I33 (pad_nc)

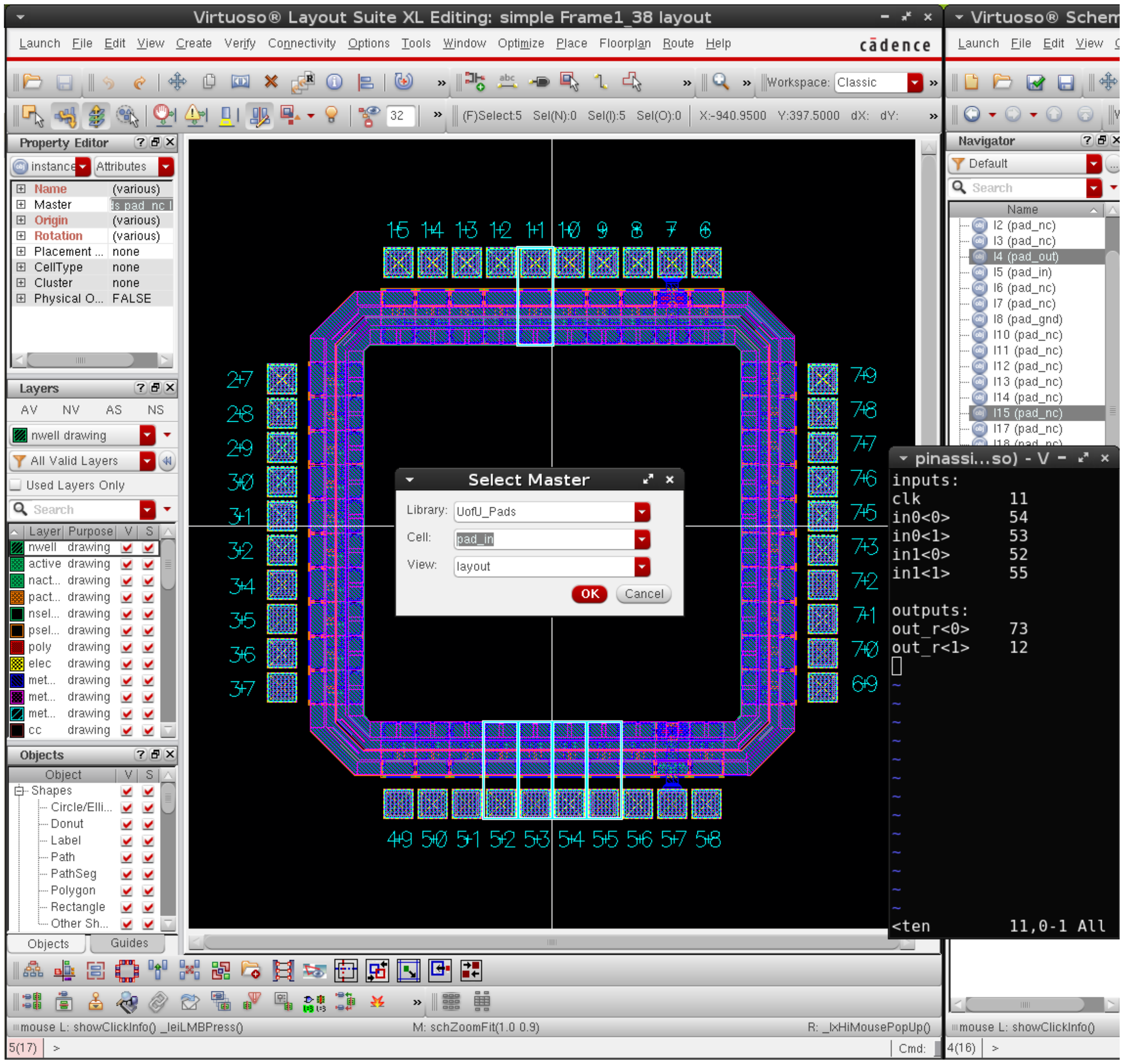
instance All

interfaceLa...	Mon Apr 14
partName	pad_nc
vendorName	
Name	(various)
Master	s pad_nc sym
Origin	(various)

Library: UofU_Pads
Cell: pad_out
View: symbol

```
inputs:
clk         11
in0<0>      54
in0<1>      53
in1<0>      52
in1<1>      55

outputs:
out_r<0>    73
out_r<1>    12

<ten        11,0-1 All
```

Figure 39: Add output pads in schematic.


Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cadence
Workspace: Classic
Property Editor
Layers
Objects
Select Master
OK
Cancel
Navigator
Virtuoso® Schem
pinassi...so) - V

(F)Select:5 Sel(N):0 Sel(I):5 Sel(O):0 X:-940.9500 Y:397.5000 dX: dY:

Name	(various)
Master	s pad_nc
Origin	(various)
Rotation	(various)
Placement ...	none
CellType	none
Cluster	none
Physical O...	FALSE

Library: UofU_Pads
Cell: pad_in
View: layout

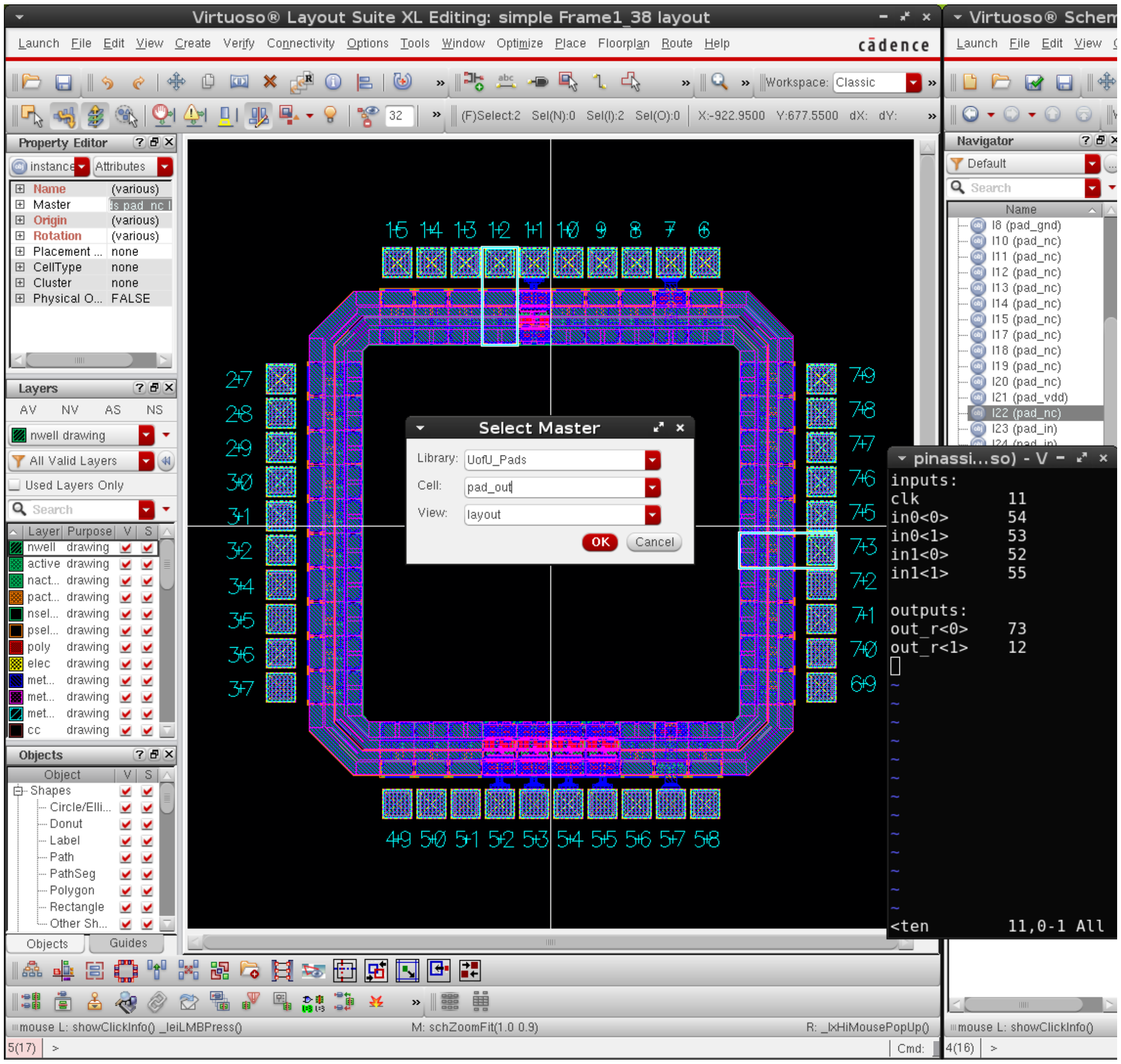
Layer	Purpose	V	S
nwell	drawing	✓	✓
active	drawing	✓	✓
nact...	drawing	✓	✓
pact...	drawing	✓	✓
nsel...	drawing	✓	✓
psel...	drawing	✓	✓
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓

I2 (pad_nc)
I3 (pad_nc)
I4 (pad_out)
I5 (pad_in)
I6 (pad_nc)
I7 (pad_nc)
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12

<ten          11,0-1 All
```

mouse L: showClickInfo() _leiLMBPress()   M: schZoomFit(1.0 0.9)   R: _lxHiMousePopUp()

Figure 40: Add input pads in layout.



(F)Select:2 Sel(N):0 Sel(I):2 Sel(O):0 X:-922.9500 Y:677.5500 dX: dY:

Property	Value
Name	(various)
Master	s pad_nc l
Origin	(various)
Rotation	(various)
Placement ...	none
CellType	none
Cluster	none
Physical O...	FALSE

Select Master
Library: UofU_Pads
Cell: pad_out
View: layout
OK Cancel

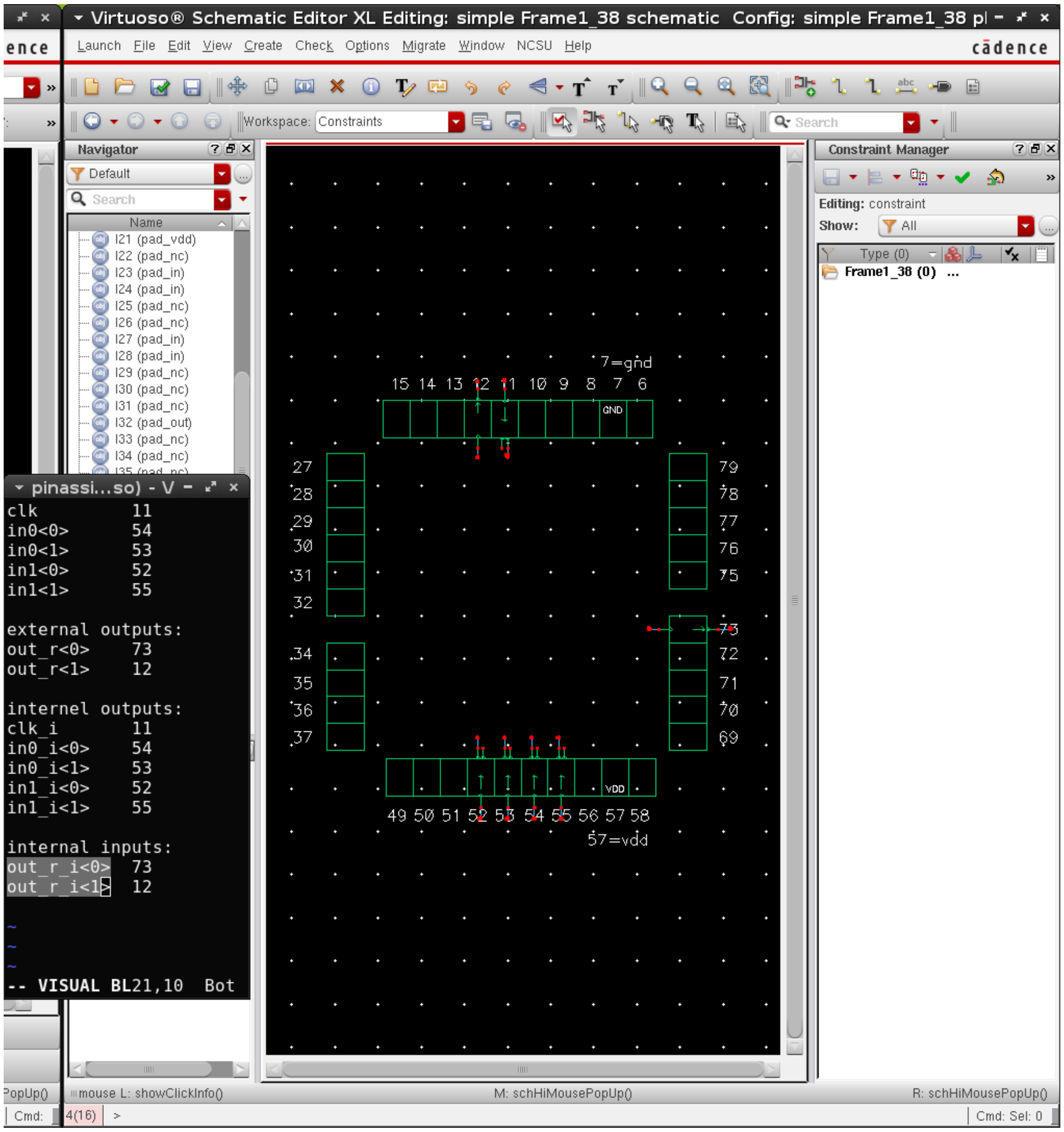
Navigator (Default):
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)
I18 (pad_nc)
I19 (pad_nc)
I20 (pad_nc)
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_in)

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12

<ten                11,0-1 All
```

mouse L: showClickInfo() _leiLMBPress()   M: schZoomFit(1.0 0.9)   R: _lxHiMousePopUp()
5(17) >   Cmd:

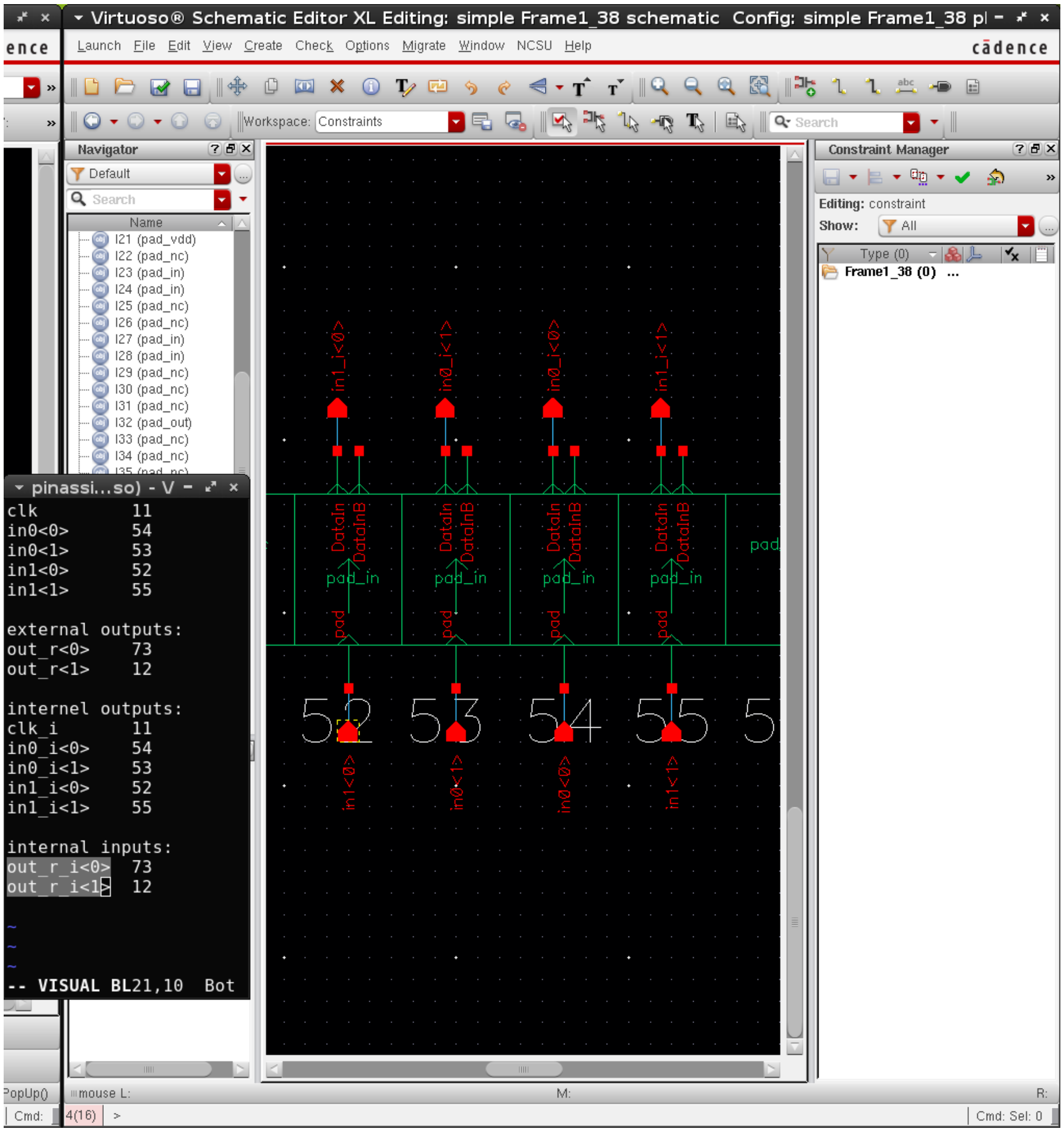
Figure 41: Add output pads in layout.

Add pins to pad frame (both schematic and layout). The pad frame works like a wraparound. The external pins of the pad frame (e.g. "Frame1_38") have the same name as the chip layout (e.g. "gpca40p"), and the internal pins of the pad frame connecting the chip layout use original names affixed by "_i" (Hence e.g. "clk" becomes "clk_i.) This procedure is shown in Figure 42 to Figure 55.



```
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

external outputs:
out_r<0>     73
out_r<1>     12

internel outputs:
clk_i        11
in0_i<0>     54
in0_i<1>     53
in1_i<0>     52
in1_i<1>     55

internal inputs:
out_r_i<0>   73
out_r_i<1>   12

~
~
~
-- VISUAL BL21,10   Bot
```

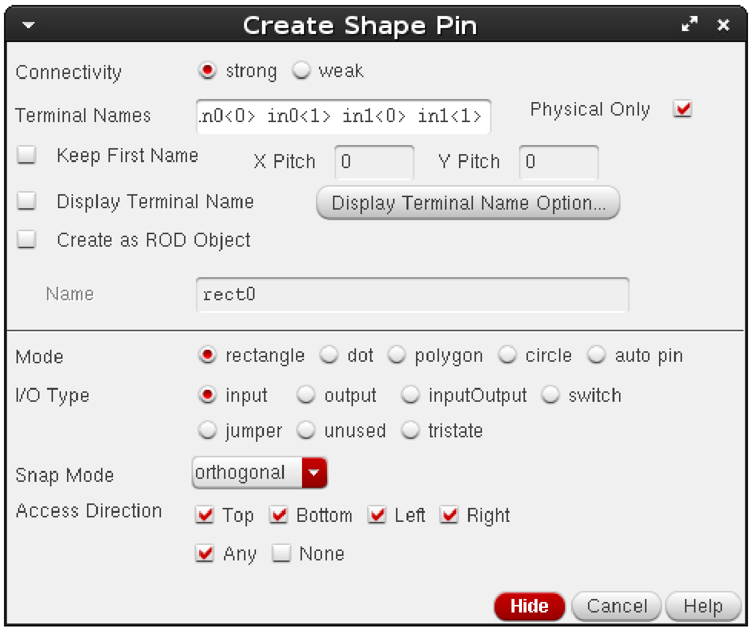
Figure 42: Add pins to the pad frame in schematic.



```
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

external outputs:
out_r<0>     73
out_r<1>     12

internel outputs:
clk_i        11
in0_i<0>     54
in0_i<1>     53
in1_i<0>     52
in1_i<1>     55

internal inputs:
out_r_i<0>   73
out_r_i<1>   12

~
~
~
-- VISUAL BL21,10   Bot
```

Figure 43: Add pins to the pad frame schematic (detailed view).

Same for the layout by tapping the ports with pins


Create Shape Pin

Connectivity [✓ strong] weak

Terminal Names .n0<0> in0<1> in1<0> in1<1>

Physical Only ✓

☐ Keep First Name X Pitch 0 Y Pitch 0

☐ Display Terminal Name

☐ Create as ROD Object

Name rect0

Mode [✓ rectangle] dot polygon circle auto pin

I/O Type [✓ input] output inputOutput switch jumper unused tristate

Snap Mode orthogonal

Access Direction ✓ Top ✓ Bottom ✓ Left ✓ Right ✓ Any ☐ None

Figure 44: "Create Shape Pin" Window of Virtuoso.

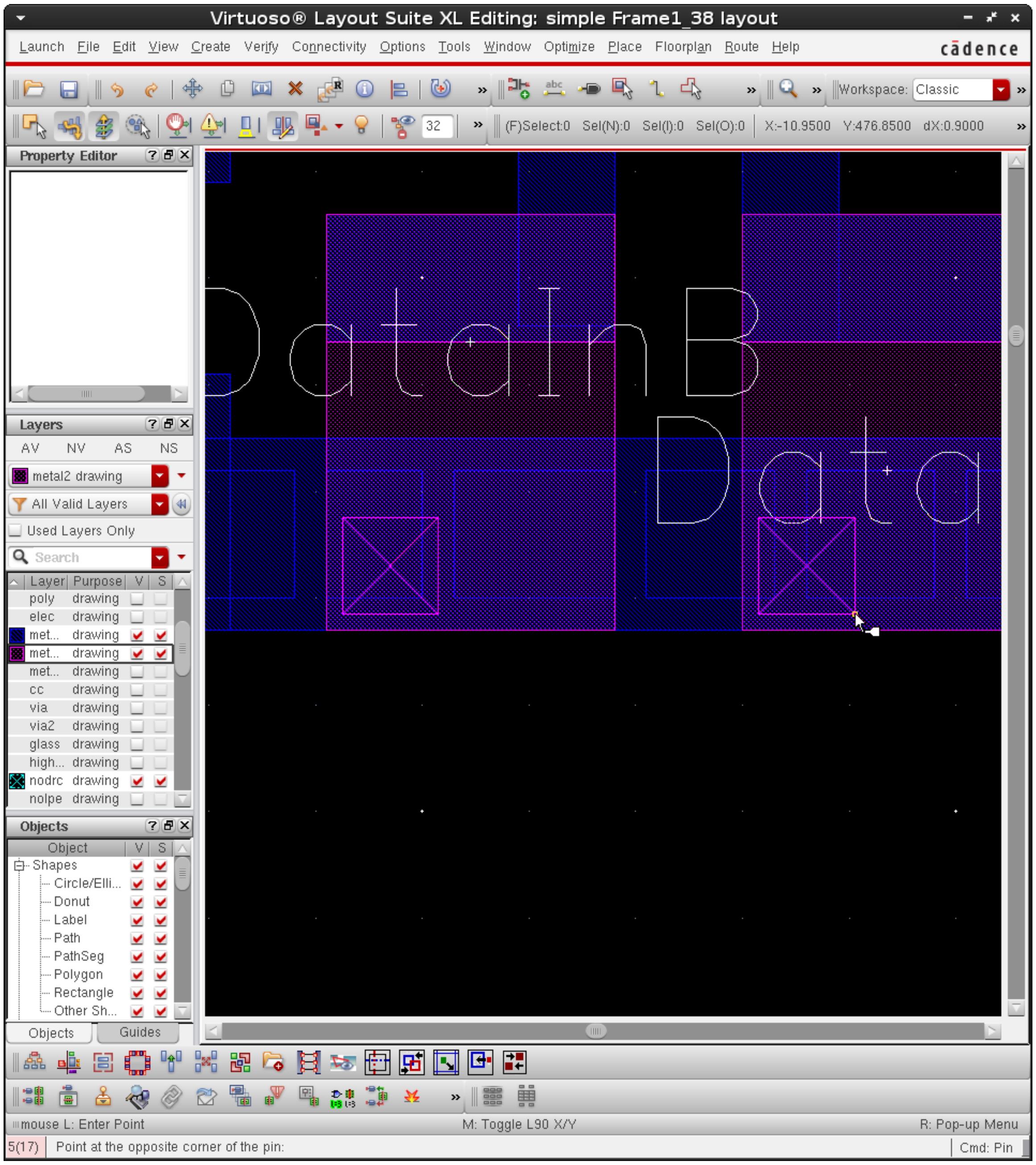
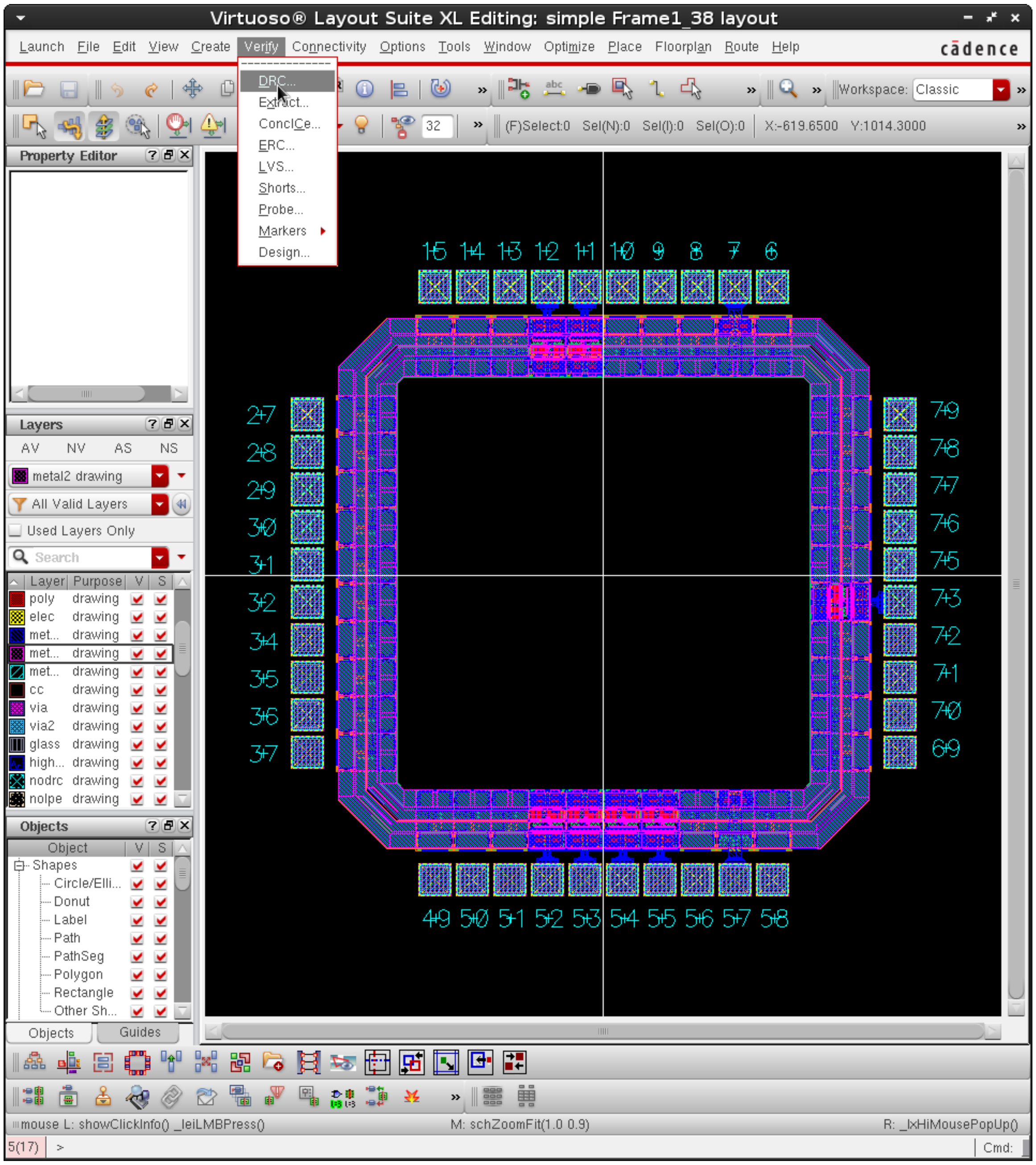


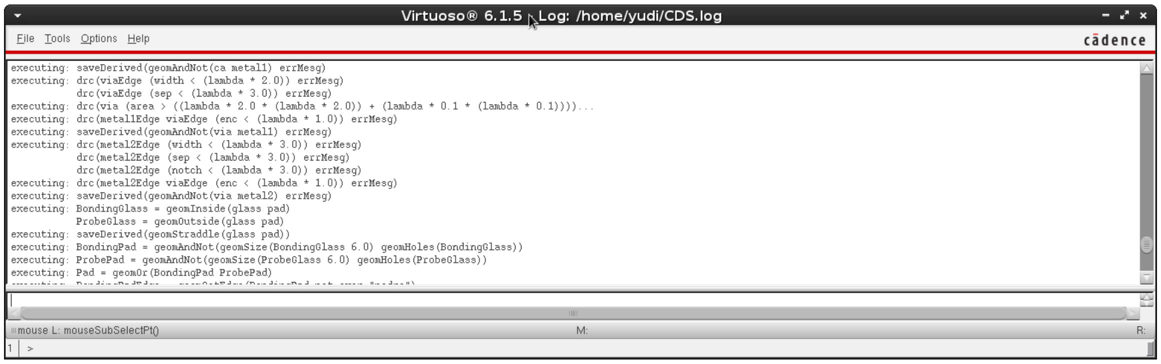
Figure 45: Add pins to the pad frame layout (detailed view).

DRC, Extract, LVS to make sure no rule is violated.



Layer	Purpose	V	S
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓
via	drawing	✓	✓
via2	drawing	✓	✓
glass	drawing	✓	✓
high...	drawing	✓	✓
nodrc	drawing	✓	✓
nolpe	drawing	✓	✓

Figure 46: Start DRC from Virtuoso Layout Suite.



```
executing: saveDerived(geomAndNot(ca metal1) errMesg)
executing: drc(viaEdge (width < (lambda * 2.0)) errMesg)
           drc(viaEdge (sep < (lambda * 3.0)) errMesg)
executing: drc(via (area > ((lambda * 2.0 * (lambda * 2.0)) + (lambda * 0.1 * (lambda * 0.1))))...
executing: drc(metal1Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal1) errMesg)
executing: drc(metal2Edge (width < (lambda * 3.0)) errMesg)
           drc(metal2Edge (sep < (lambda * 3.0)) errMesg)
           drc(metal2Edge (notch < (lambda * 3.0)) errMesg)
executing: drc(metal2Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal2) errMesg)
executing: BondingGlass = geomInside(glass pad)
           ProbeGlass = geomOutside(glass pad)
executing: saveDerived(geomStraddle(glass pad))
executing: BondingPad = geomAndNot(geomSize(BondingGlass 6.0) geomHoles(BondingGlass))
executing: ProbePad = geomAndNot(geomSize(ProbeGlass 6.0) geomHoles(ProbeGlass))
executing: Pad = geomOr(BondingPad ProbePad)
```

Virtuoso® 6.1.5 Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: mouseSubSelectPt()
M:
R:

Figure 47: Report of successful DRC.

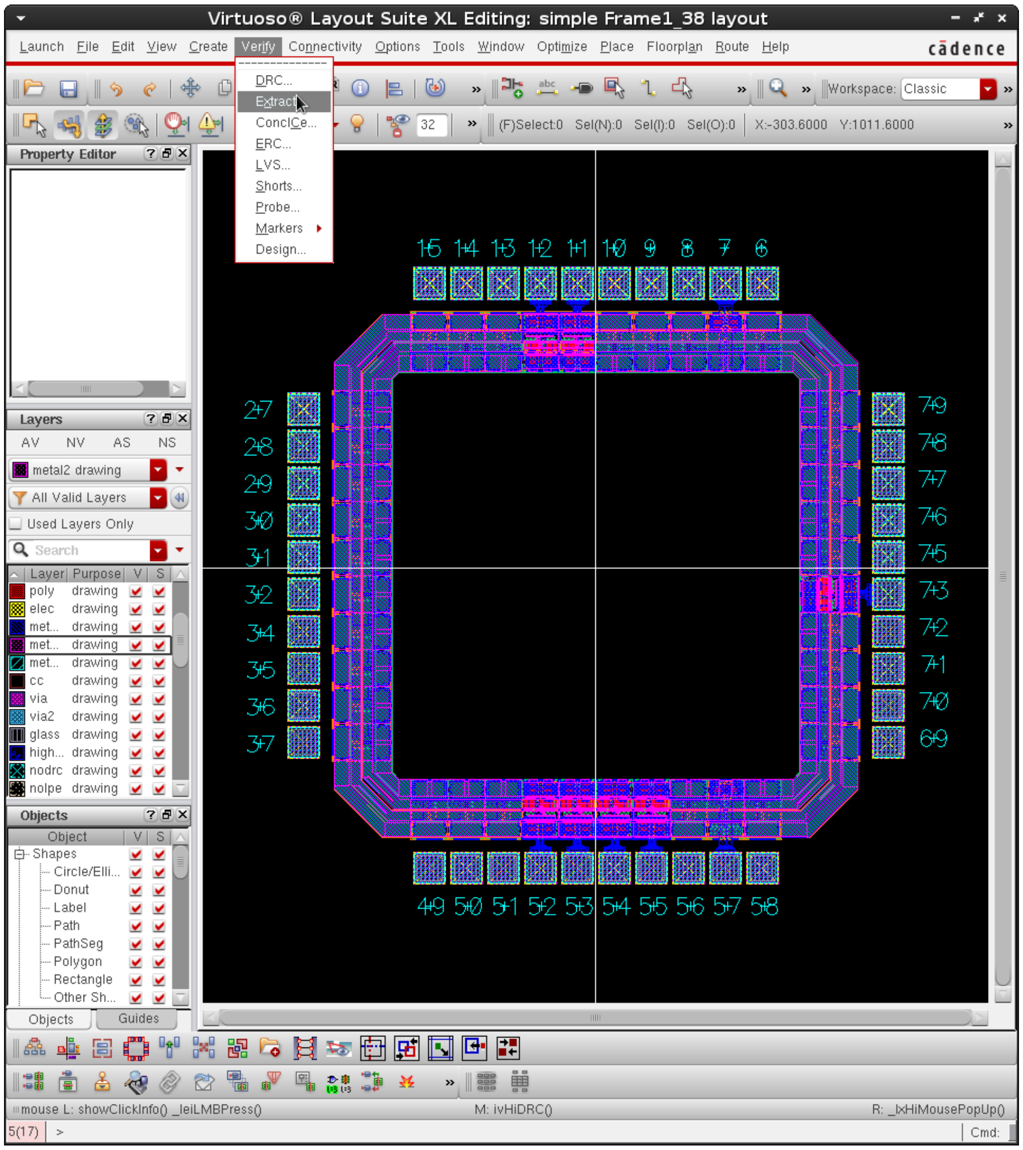
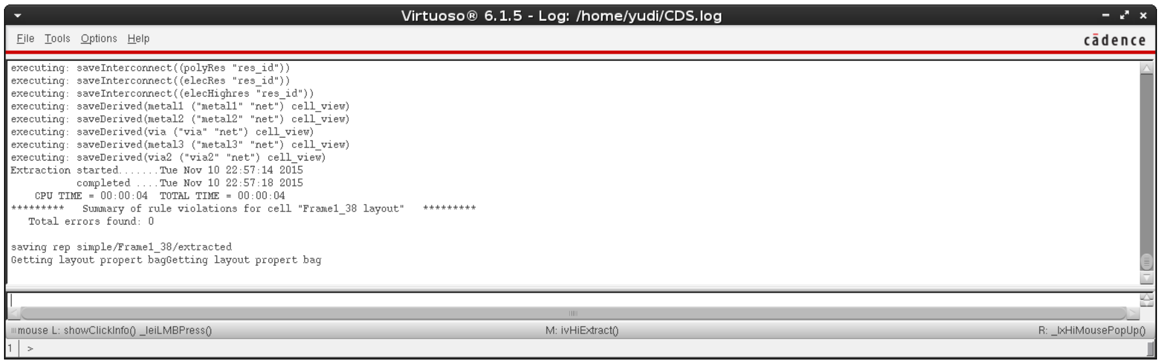


Figure 48: Start Extract from Virtuoso Layout Suite.



```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 22:57:14 2015
           completed ....Tue Nov 10 22:57:18 2015
    CPU TIME = 00:00:04  TOTAL TIME = 00:00:04
*********   Summary of rule violations for cell "Frame1_38 layout"   *********
   Total errors found: 0

saving rep simple/Frame1_38/extracted
Getting layout propert bagGetting layout propert bag
```

Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cadence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiExtract()
R: _lxHiMousePopUp()

Figure 49: Report of successful Extract.

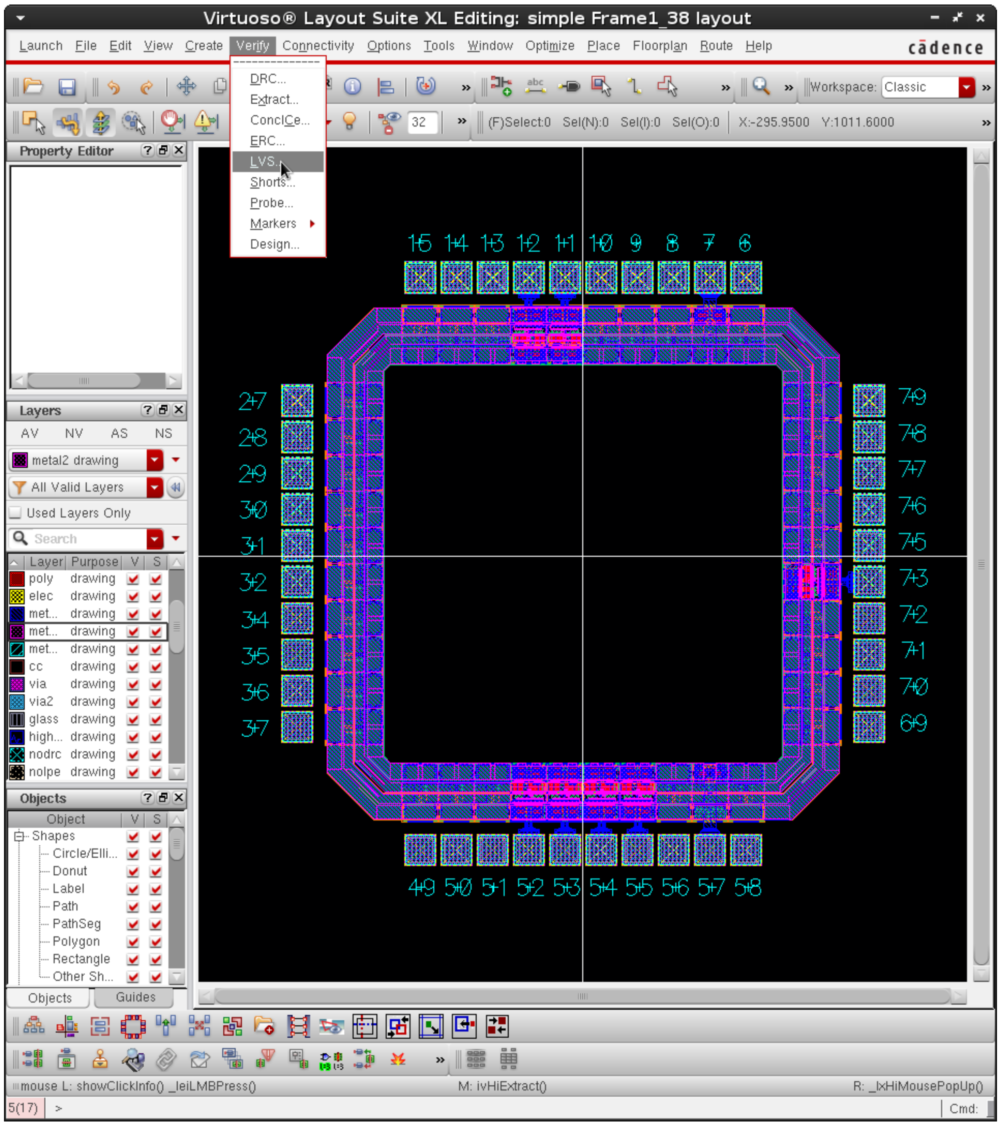
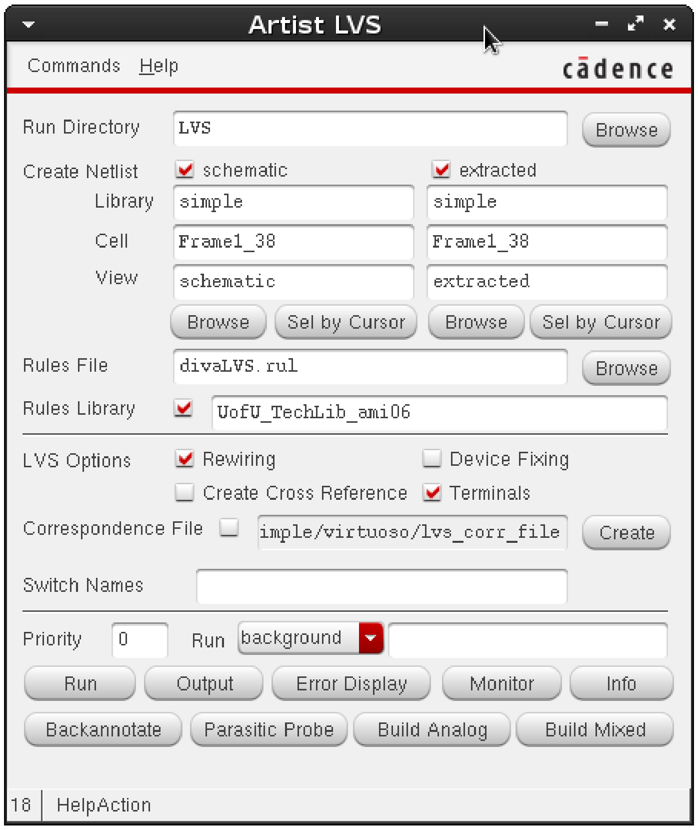
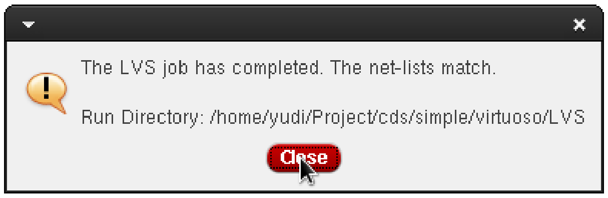


Figure 50: Start LVS from Virtuoso Layout Suite.



Run Directory: LVS
Create Netlist: [✓ schematic] [✓ extracted]
Library: simple | simple
Cell: Frame1_38 | Frame1_38
View: schematic | extracted
Rules File: divaLVS.rul
Rules Library: [✓] UofU_TechLib_ami06
LVS Options: [✓ Rewiring] Device Fixing Create Cross Reference [✓ Terminals]
Correspondence File: imple/virtuoso/lvs_corr_file
Switch Names:
Priority: 0 Run: background
18 HelpAction

Figure 51: "Artist LVS" window of Virtuoso.



The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

Figure 52: "Artist LVS" successful LVS prompt.

Generate symbol for the pad frame.

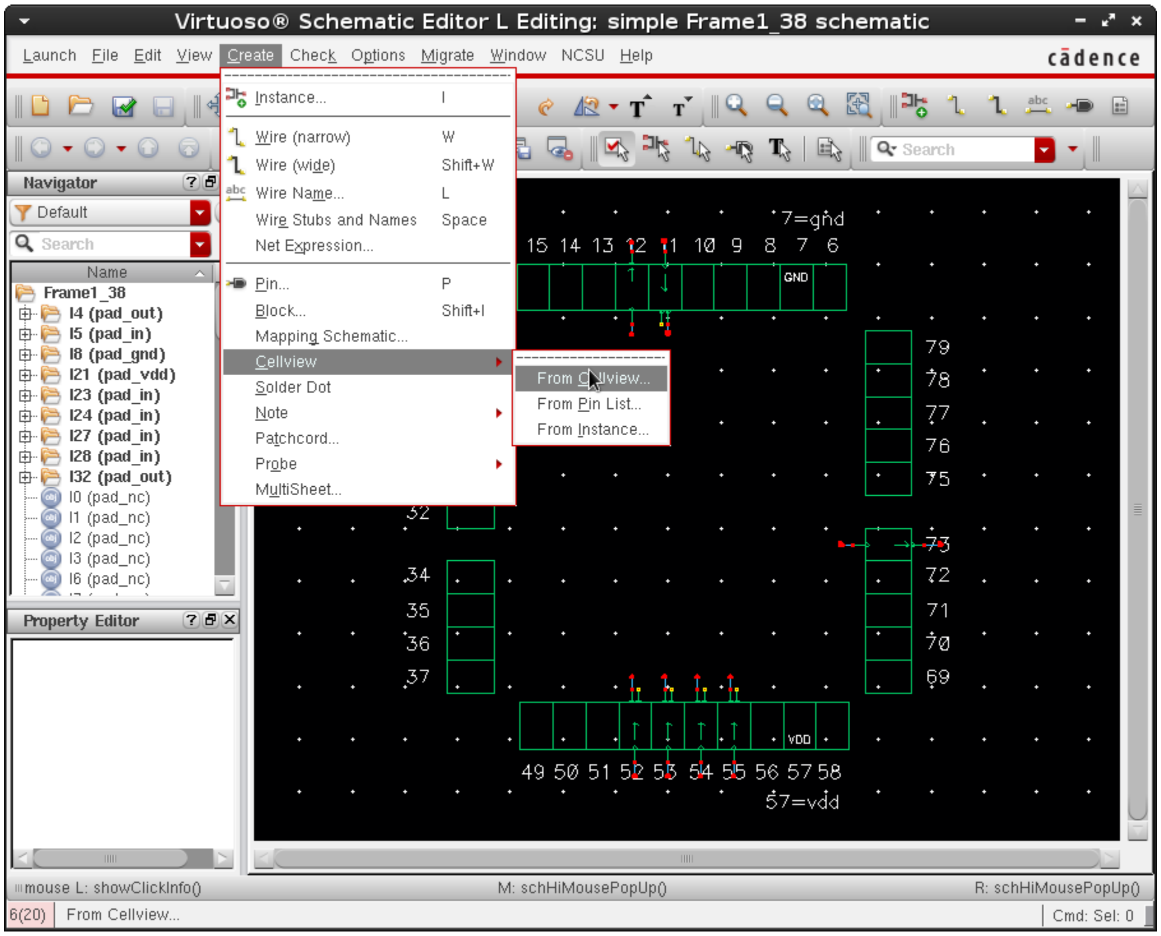
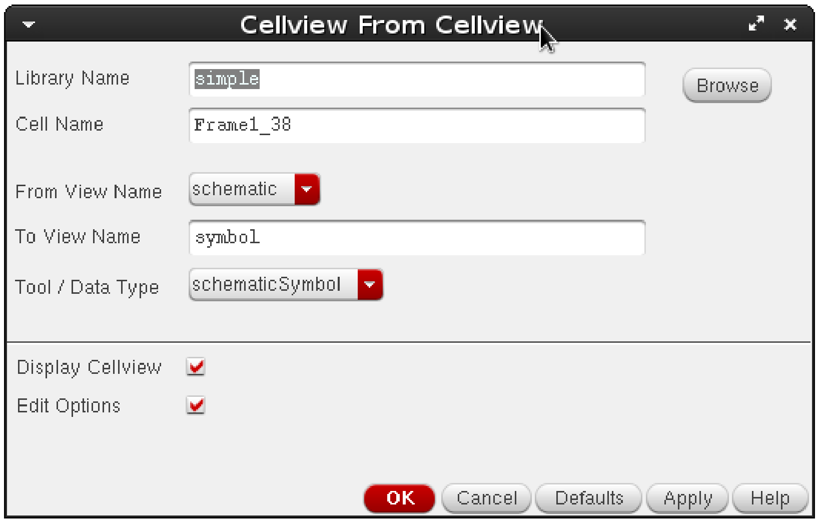
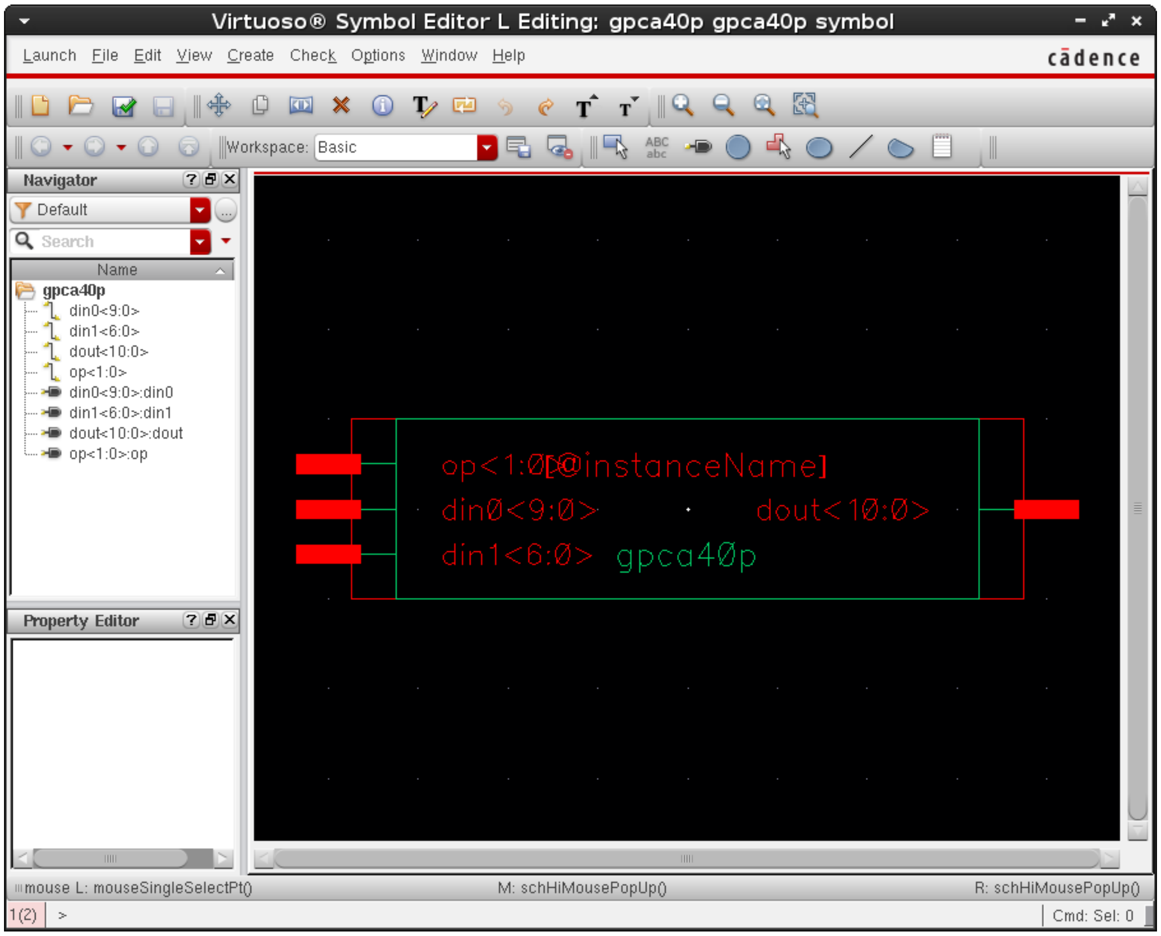


Figure 53: Create symbol view from schematic view.



Library Name simple
Cell Name Frame1_38
From View Name schematic
To View Name symbol
Tool / Data Type schematicSymbol
Display Cellview ✓
Edit Options ✓

Figure 54: "CellView from CellView" window of Virtuoso.



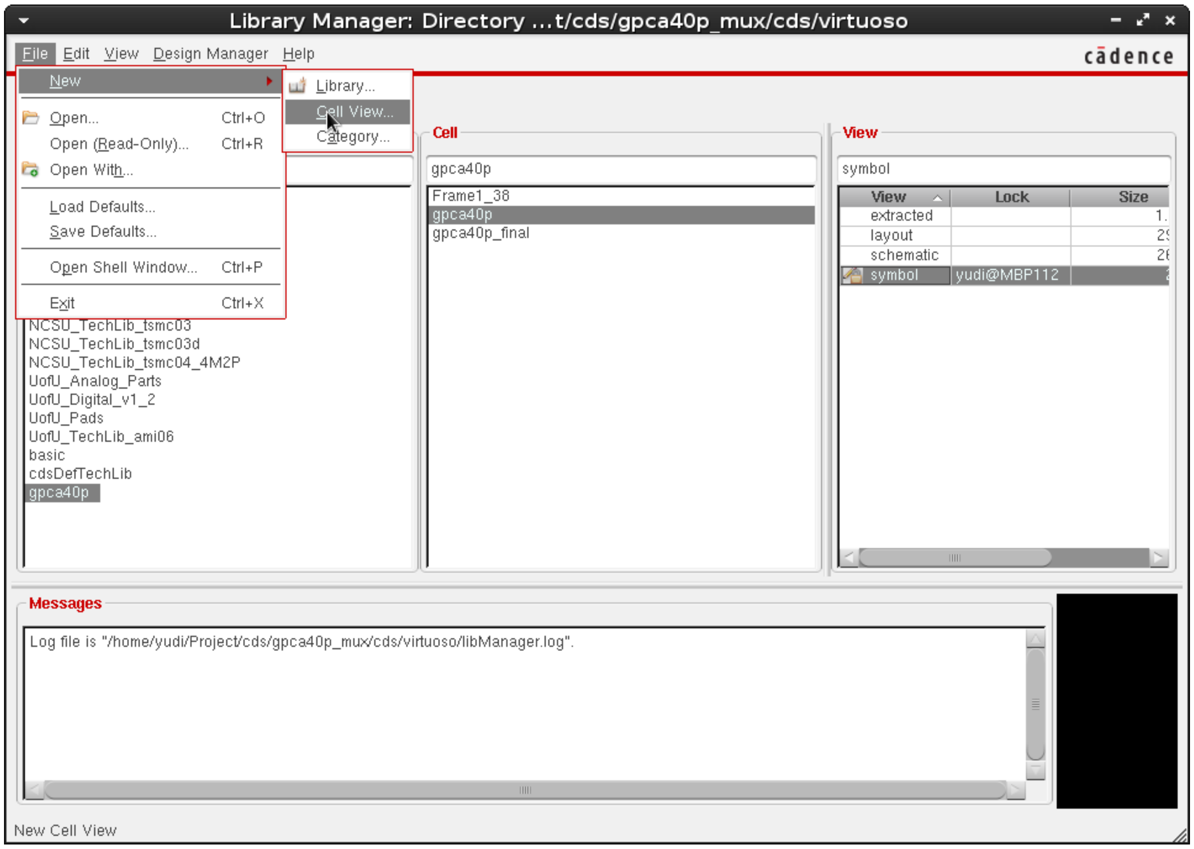
mouse L: mouseSingleSelectPt()
M: schHiMousePopUp()
R: schHiMousePopUp()
1(2) >
Cmd: Sel: 0

Figure 55: Final symbol view of the pad frame.

e) Final Chip Assembly (Add Pad Frame)

With both the core of the chip ("simple") and the pad frame (modified "Frame1_38") are ready, creating a new cell to put them together. This procedure is shown in Figure 56 to 72.

Create a new Cell View (I call it "gpca40p_final").



NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts
UofU_Digital_v1_2
UofU_Pads
UofU_TechLib_ami06
basic
cdsDefTechLib
gpca40p

Cell: gpca40p

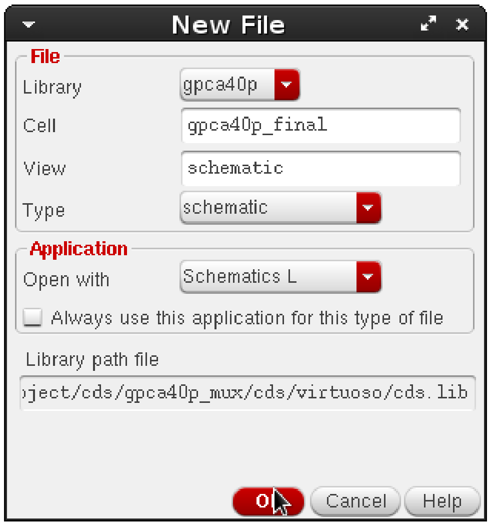
Frame1_38
gpca40p
gpca40p_final

View: symbol

View	Lock	Size
extracted		1.
layout		2
schematic		2
symbol	yudi@MBP112	

Log file is "/home/yudi/Project/cds/gpca40p_mux/cds/virtuoso/libManager.log".

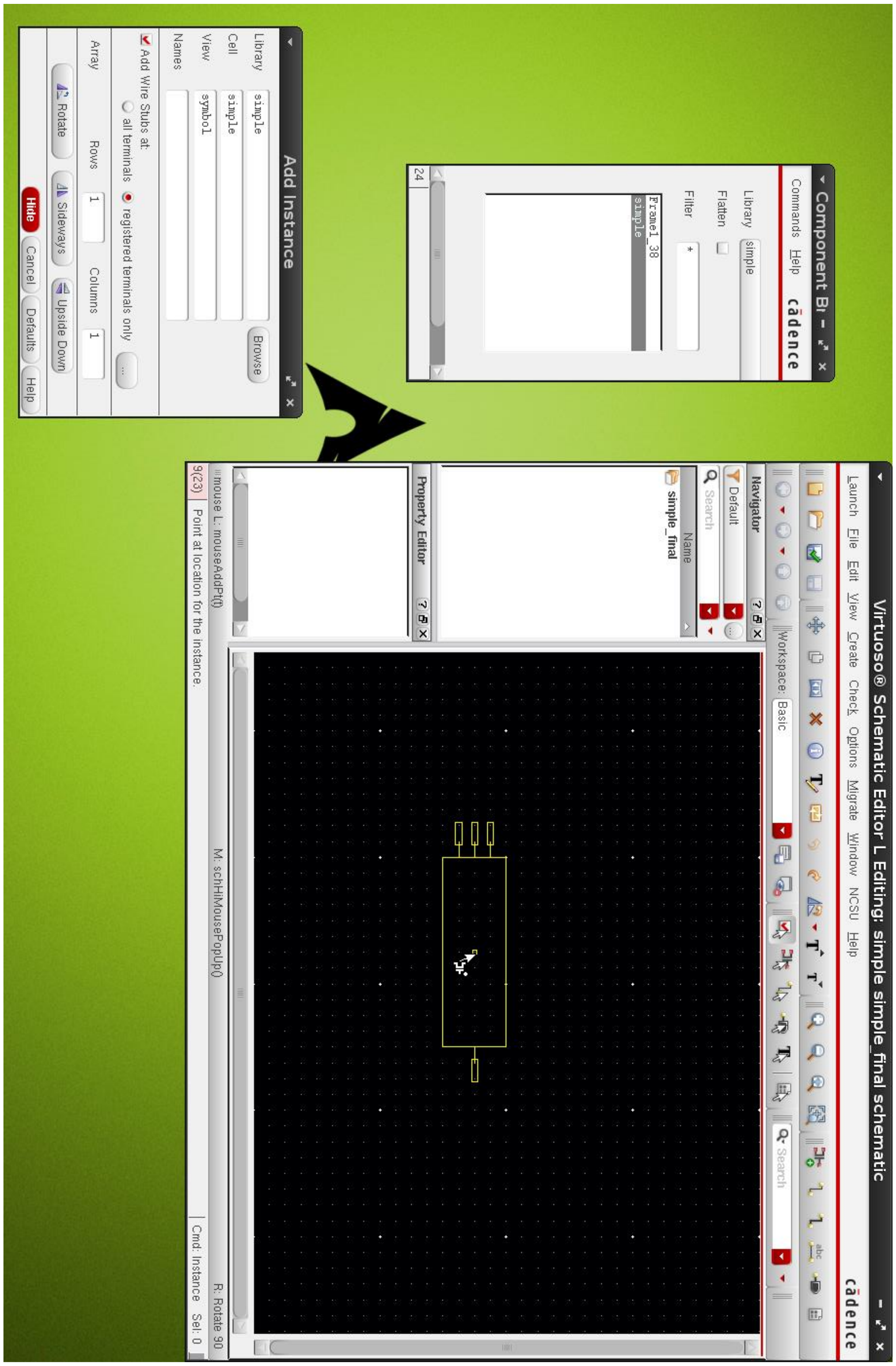
Figure 56: Create new "Cell View" from Virtuoso Layout Suite



Library gpca40p
Cell gpca40p_final
View schematic
Type schematic
Open with Schematics L
Always use this application for this type of file
Library path file
›ject/cds/gpca40p_mux/cds/virtuoso/cds.lib

Figure 57: "New File" window of Virtuoso.

Instantiate both the core of the chip ("gpca40p") and the pad frame (modified "Frame1_38") (by pressing "I" to instantiate instances).



Library simple
Cell simple
View symbol
Names
Add Wire Stubs at: all terminals [✓ registered terminals only]
Array Rows 1 Columns 1

Library simple
Flatten
Filter *
Frame1_38
simple

mouse L: mouseAddPt()
M: schHiMousePopUp()
R: Rotate 90
9(23) Point at location for the instance.
Cmd: Instance Sel: 0

Figure 58: Create the final schematic view.

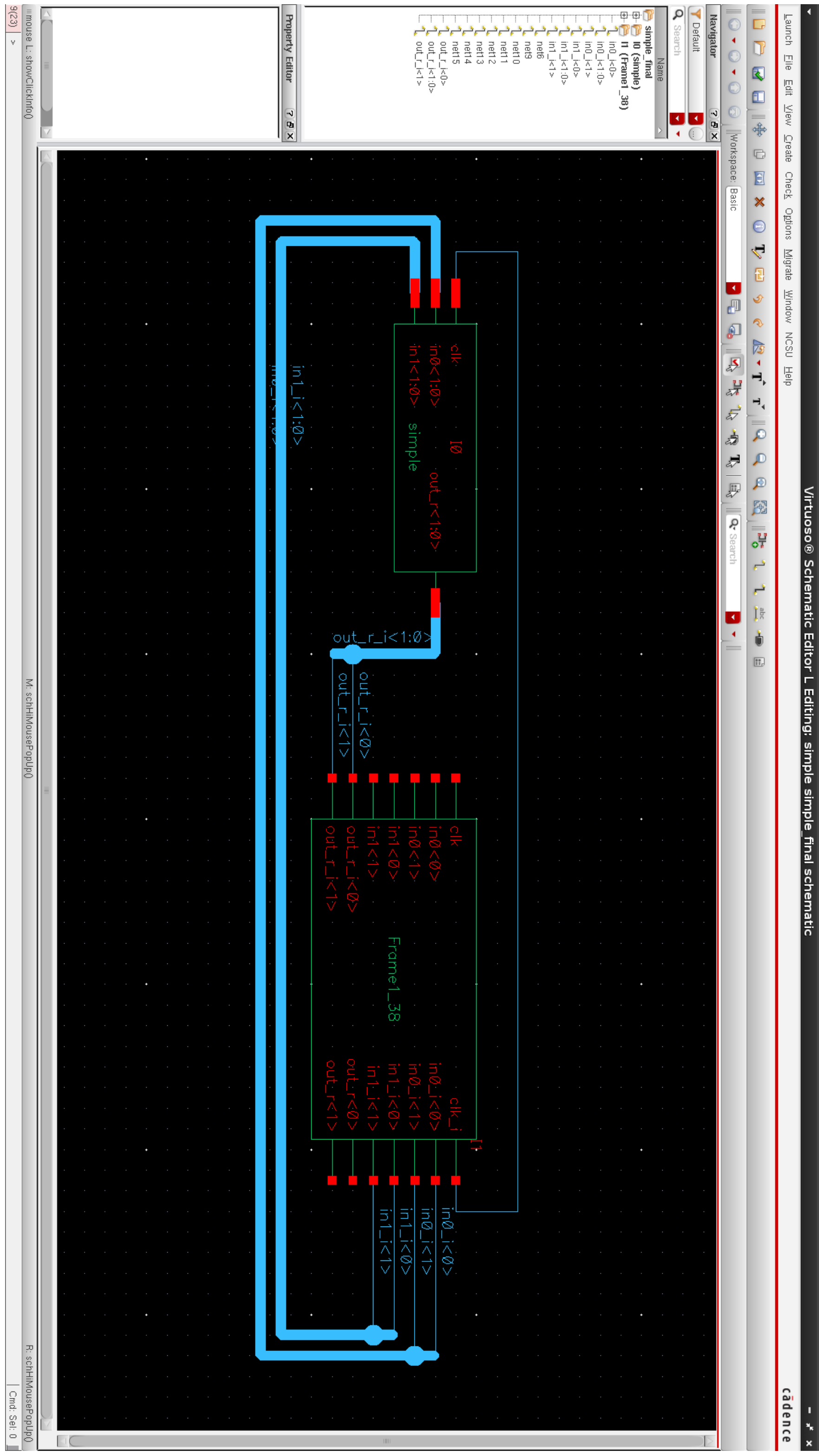
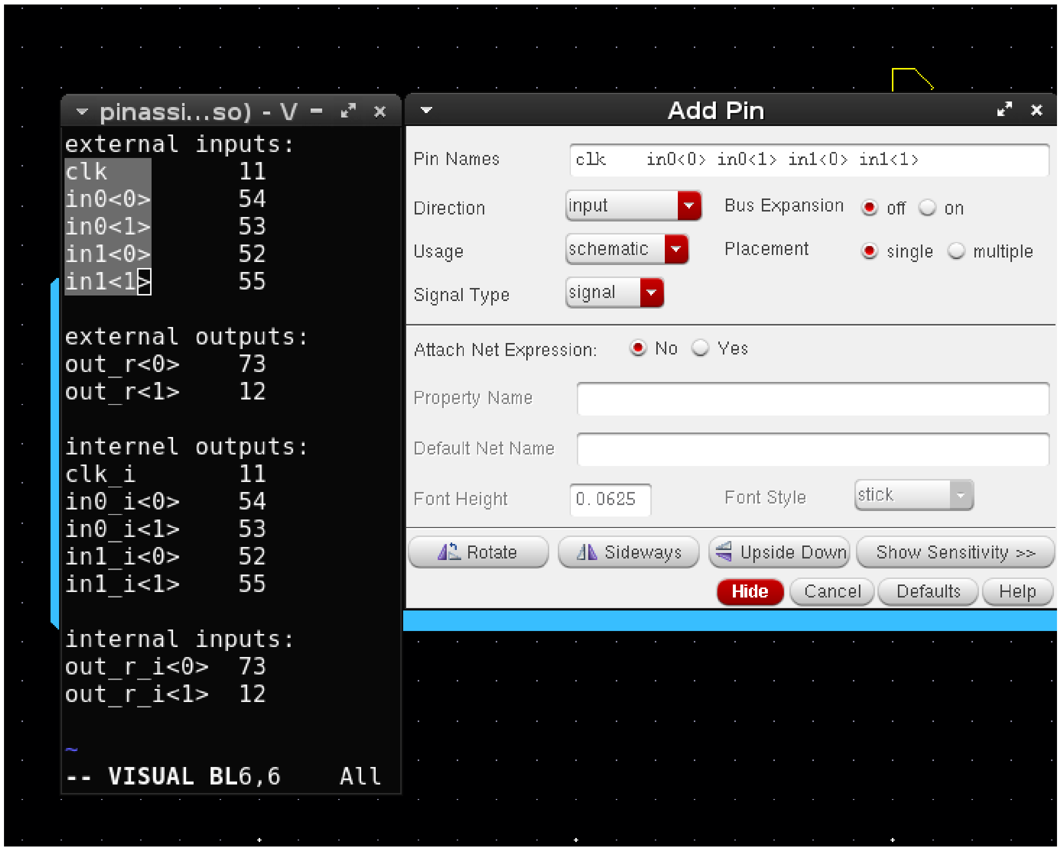


Figure 59: Routed schematic view.

Then add the pins.



```
external inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

external outputs:
out_r<0>     73
out_r<1>     12

internel outputs:
clk_i        11
in0_i<0>     54
in0_i<1>     53
in1_i<0>     52
in1_i<1>     55

internal inputs:
out_r_i<0>   73
out_r_i<1>   12

~
-- VISUAL BL6,6      All
```

Add Pin

Pin Names: clk in0<0> in0<1> in1<0> in1<1>

Direction: input — Bus Expansion: [✓ off] on

Usage: schematic — Placement: [✓ single] multiple

Signal Type: signal

Attach Net Expression: [✓ No] Yes

Property Name:

Default Net Name:

Font Height: 0.0625 — Font Style: stick

Figure 60: "Add Pins" window of Virtuoso.

Final schematic view is shown below.

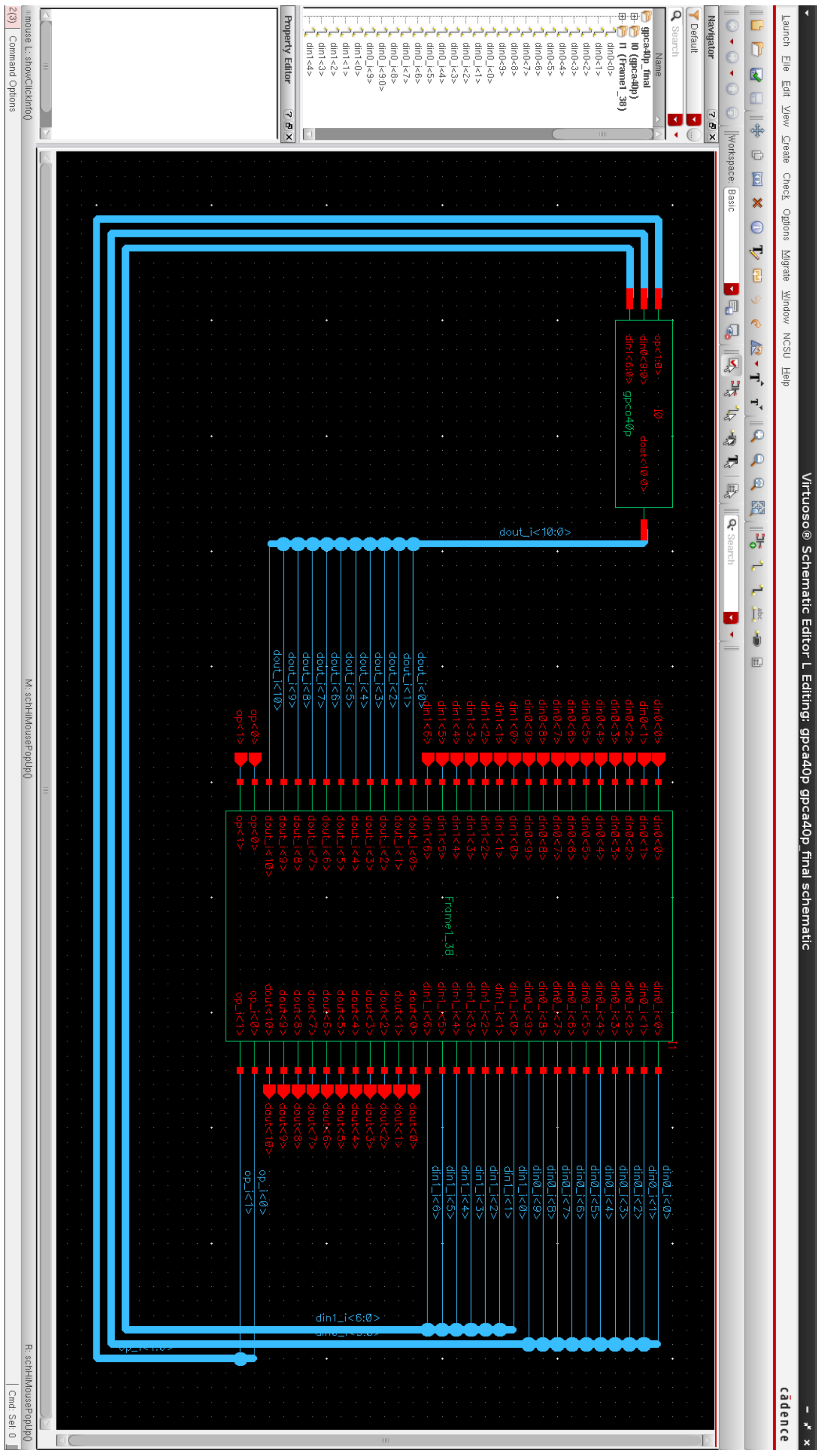
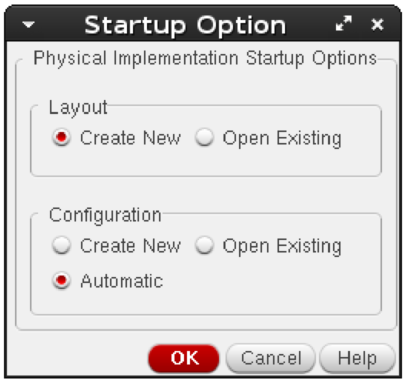


Figure 61: Final schematic view.

Then create the final layout view from the schematic.

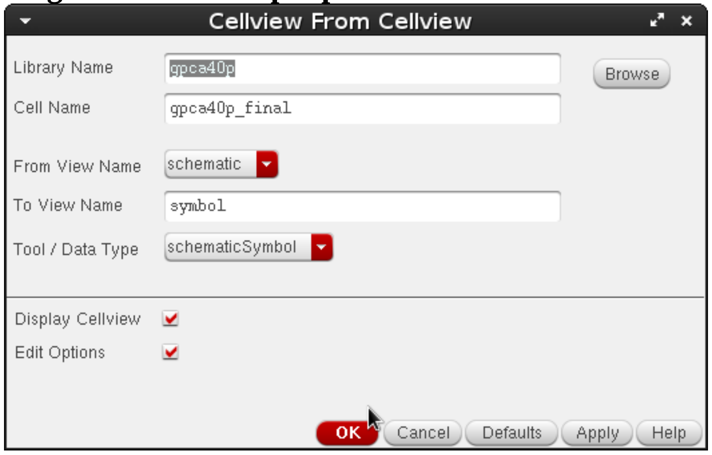

Startup Option

Physical Implementation Startup Options

Layout: [✓ Create New] Open Existing

Configuration: Create New, Open Existing, [✓ Automatic]

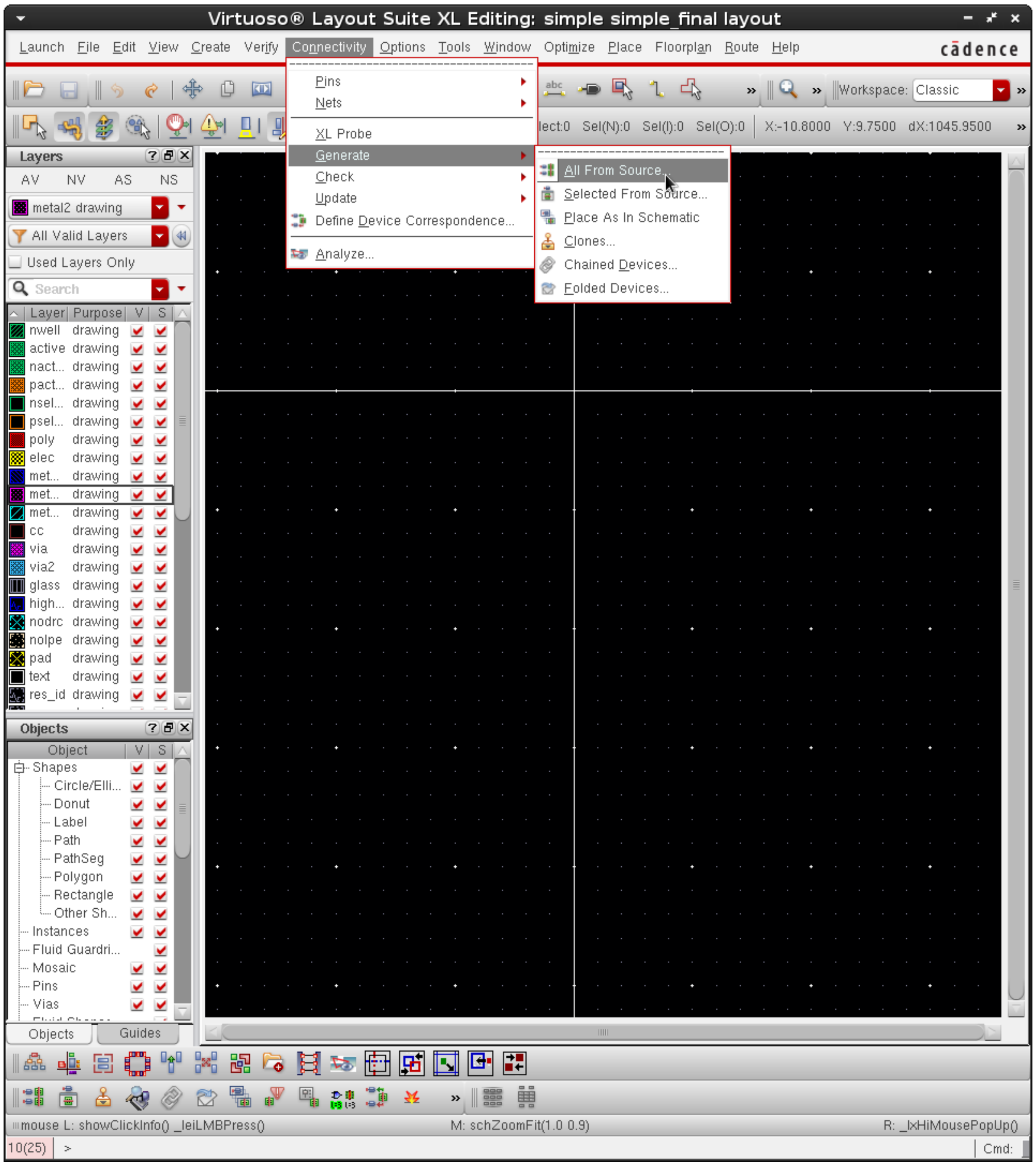
Figure 62: "Startup Option" window of Virtuoso.


Cellview From Cellview

Library Name	gpca40p
Cell Name	gpca40p_final
From View Name	schematic
To View Name	symbol
Tool / Data Type	schematicSymbol
Display Cellview	✓
Edit Options	✓

Figure 63: "Startup Option" window of Virtuoso.

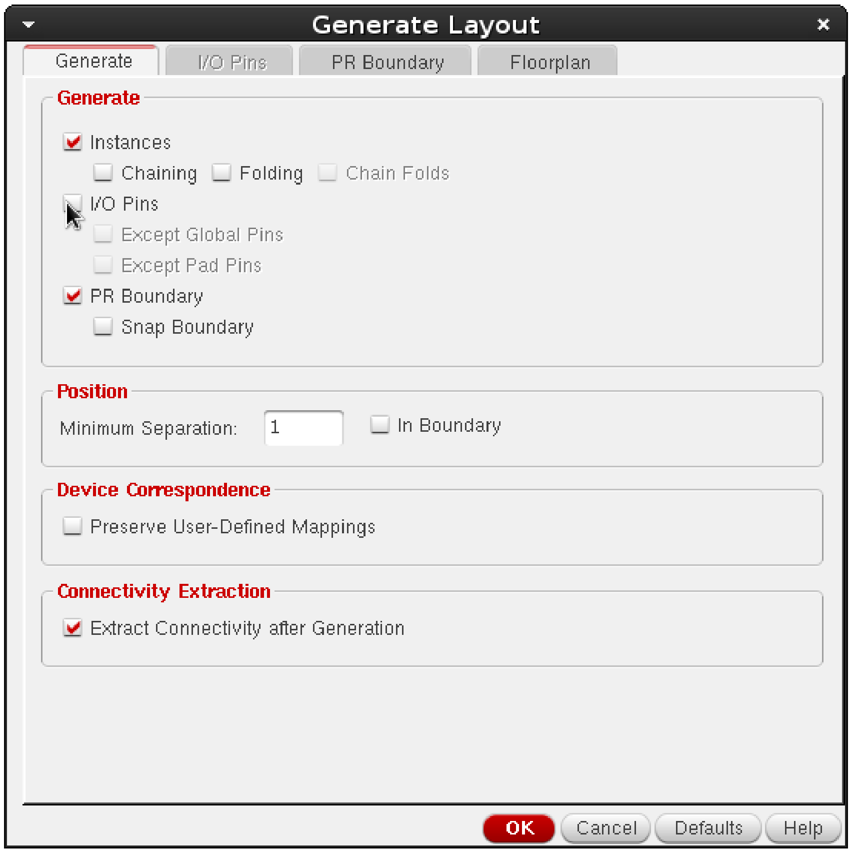
In the new empty layout view



Layer	Purpose	V	S
nwell	drawing	✓	✓
active	drawing	✓	✓
nact...	drawing	✓	✓
pact...	drawing	✓	✓
nsel...	drawing	✓	✓
psel...	drawing	✓	✓
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓
via	drawing	✓	✓
via2	drawing	✓	✓
glass	drawing	✓	✓
high...	drawing	✓	✓
nodrc	drawing	✓	✓
nolpe	drawing	✓	✓
pad	drawing	✓	✓
text	drawing	✓	✓
res_id	drawing	✓	✓

Object	V	S
Shapes	✓	✓
Circle/Elli...	✓	✓
Donut	✓	✓
Label	✓	✓
Path	✓	✓
PathSeg	✓	✓
Polygon	✓	✓
Rectangle	✓	✓
Other Sh...	✓	✓
Instances	✓	✓
Fluid Guardri...		✓
Mosaic	✓	✓
Pins	✓	✓
Vias	✓	✓

Figure 64: Start "Generate Layout" of Virtuoso.


Generate Layout

Generate | I/O Pins | PR Boundary | Floorplan

Generate

- [✓] Instances
  - [ ] Chaining [ ] Folding [ ] Chain Folds
- [ ] I/O Pins
  - [ ] Except Global Pins
  - [ ] Except Pad Pins
- [✓] PR Boundary
  - [ ] Snap Boundary

Position

Minimum Separation: 1 [ ] In Boundary

Device Correspondence

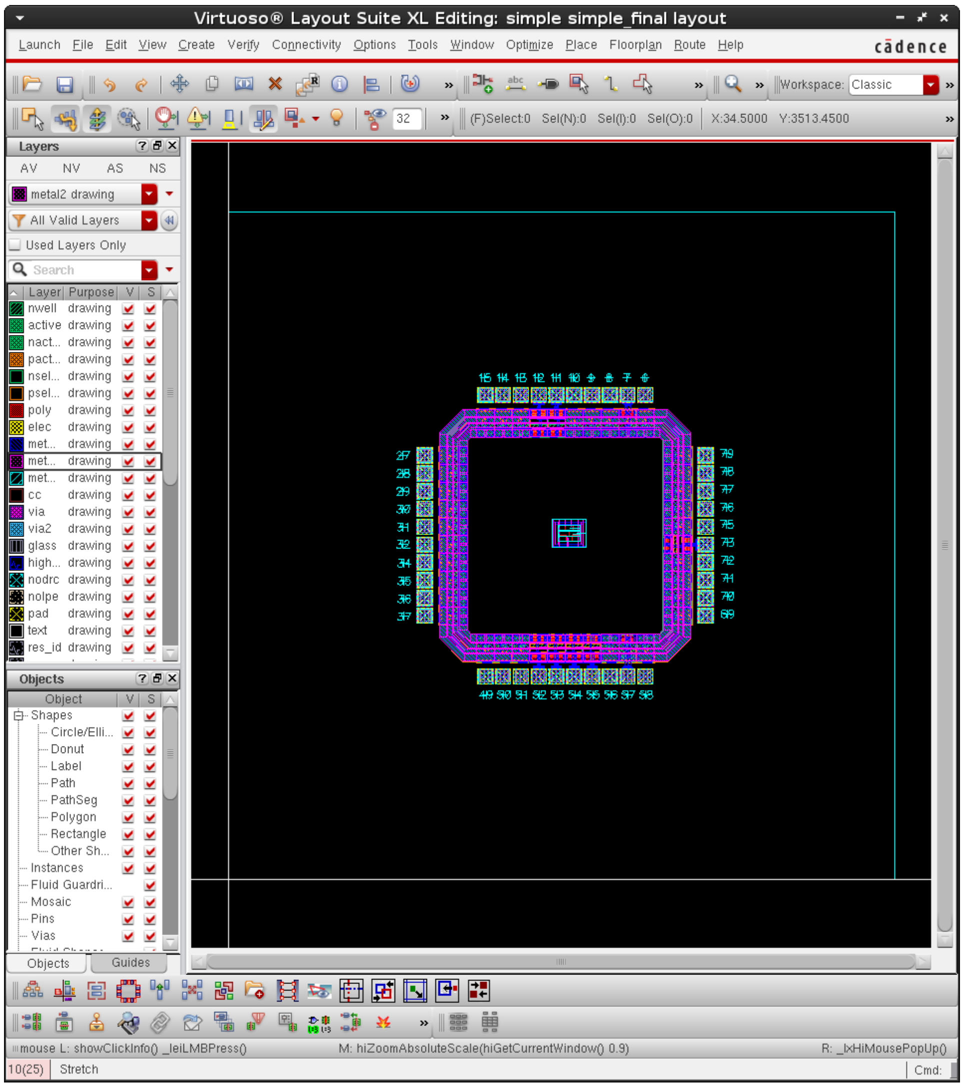
[ ] Preserve User-Defined Mappings

Connectivity Extraction

[✓] Extract Connectivity after Generation

Figure 65: "Generate Layout" window of Virtuoso.

Then place the pad frame ("Frame1_38") inside the "PR Boundary", and the core of the chip ("gpca40p") inside the pad frame.


Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
Layers
metal2 drawing
All Valid Layers
Used Layers Only
Search
Objects
Guides

```
(F)Select:0   Sel(N):0   Sel(I):0   Sel(O):0   X:34.5000   Y:3513.4500
mouse L: showClickInfo() _leiLMBPress()   M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)   R: _lxHiMousePopUp()
10(25)   Stretch   Cmd:
```

Figure 66: Component placement in Virtuoso.

Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.

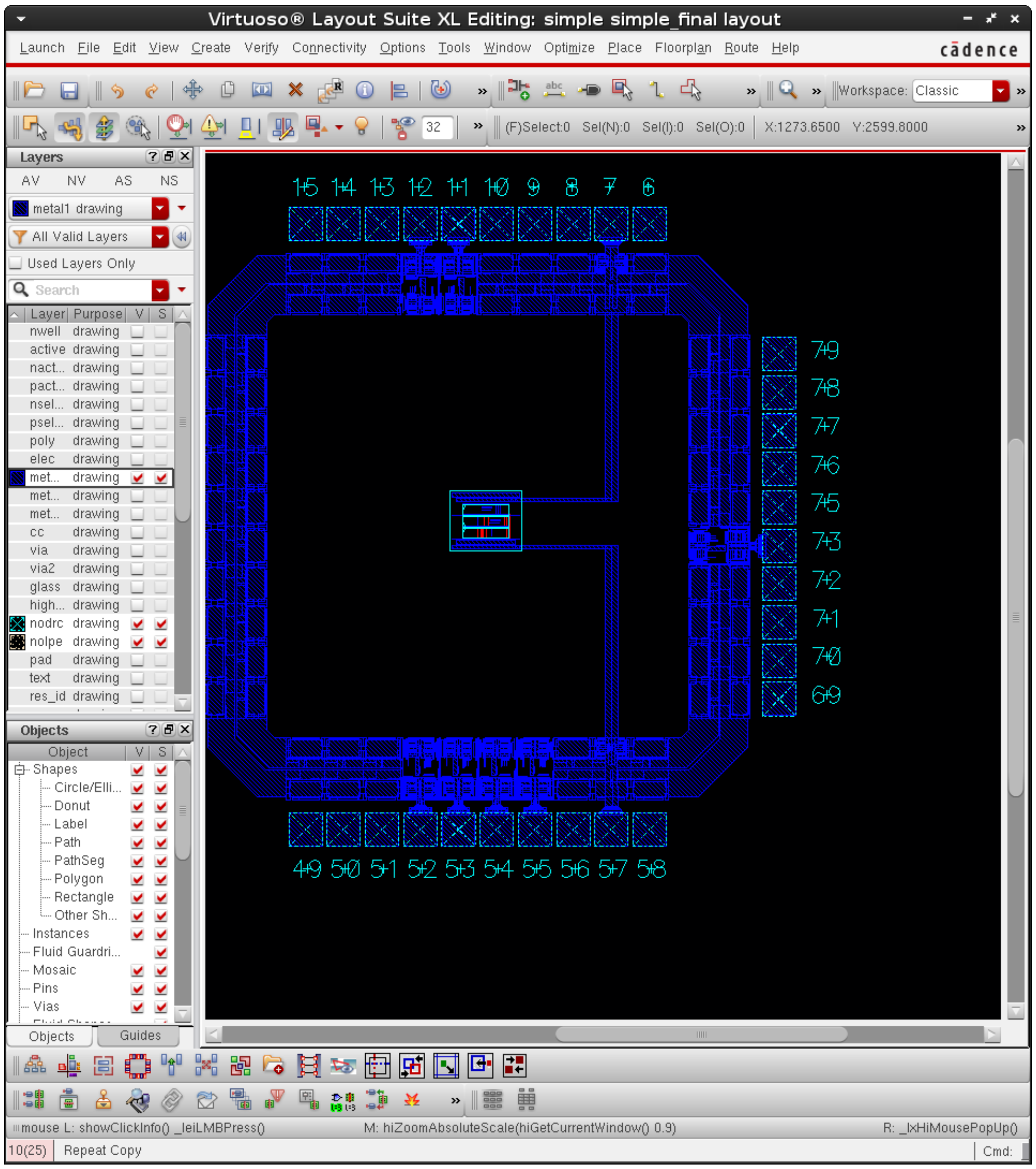
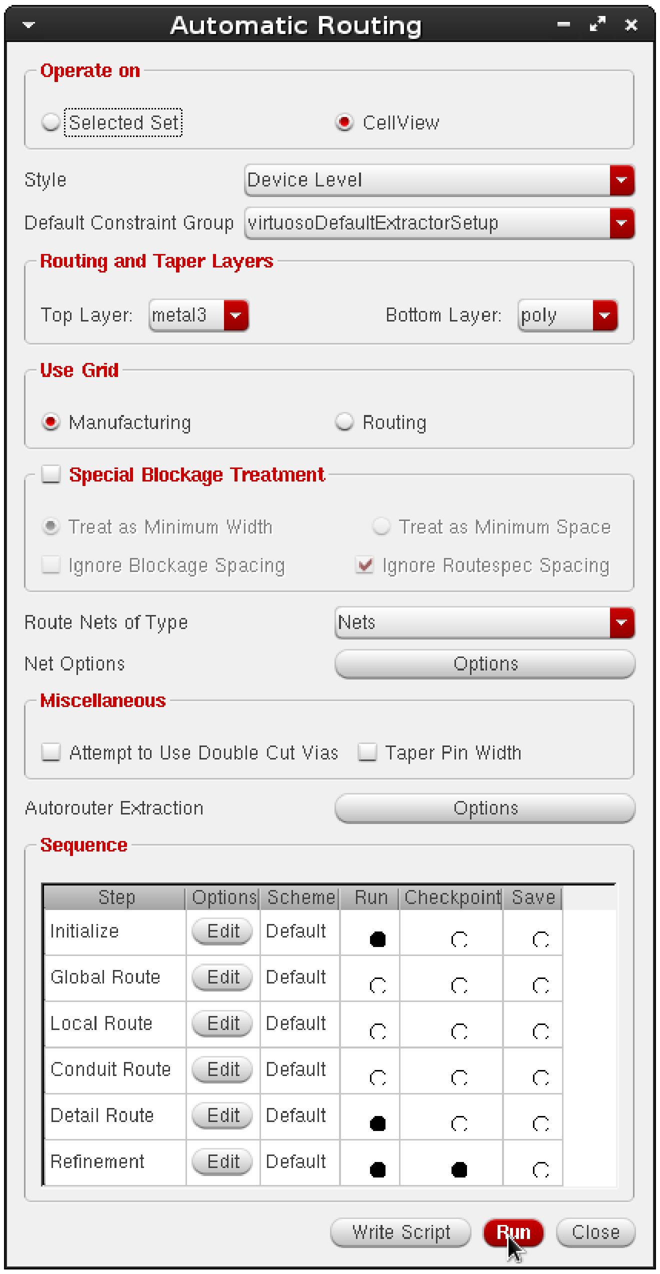

Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:1273.6500 Y:2599.8000
Layers
metal1 drawing
All Valid Layers
Used Layers Only
Objects
Guides
15 14 13 12 11 10 9 8 7 6
79 78 77 76 75 73 72 71 70 69
49 50 51 52 53 54 55 56 57 58
mouse L: showClickInfo() _leiLMBPress()
M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)
R: _lxHiMousePopUp()
10(25) Repeat Copy
Cmd:

Figure 67: Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.

Then invoke Automatic Routing.



Automatic Routing

Operate on: ( ) Selected Set [✓ CellView]

Style: Device Level

Default Constraint Group: virtuosoDefaultExtractorSetup

Routing and Taper Layers: Top Layer: metal3 Bottom Layer: poly

Use Grid: [✓ Manufacturing] ( ) Routing

☐ Special Blockage Treatment

Treat as Minimum Width / Treat as Minimum Space

☐ Ignore Blockage Spacing ☑ Ignore Routespec Spacing

Route Nets of Type: Nets

Net Options: Options

Miscellaneous: ☐ Attempt to Use Double Cut Vias ☐ Taper Pin Width

Autorouter Extraction: Options

Sequence

Step	Options	Scheme	Run	Checkpoint	Save
Initialize	Edit	Default	●	○	○
Global Route	Edit	Default	○	○	○
Local Route	Edit	Default	○	○	○
Conduit Route	Edit	Default	○	○	○
Detail Route	Edit	Default	●	○	○
Refinement	Edit	Default	●	●	○

Write Script · Run · Close

Figure 68: "Automatic Routing" window of Virtuoso.

The following is the final layout. Fill the empty space with poly fills to meet minimum polysilicon density required by AMI05 [35].

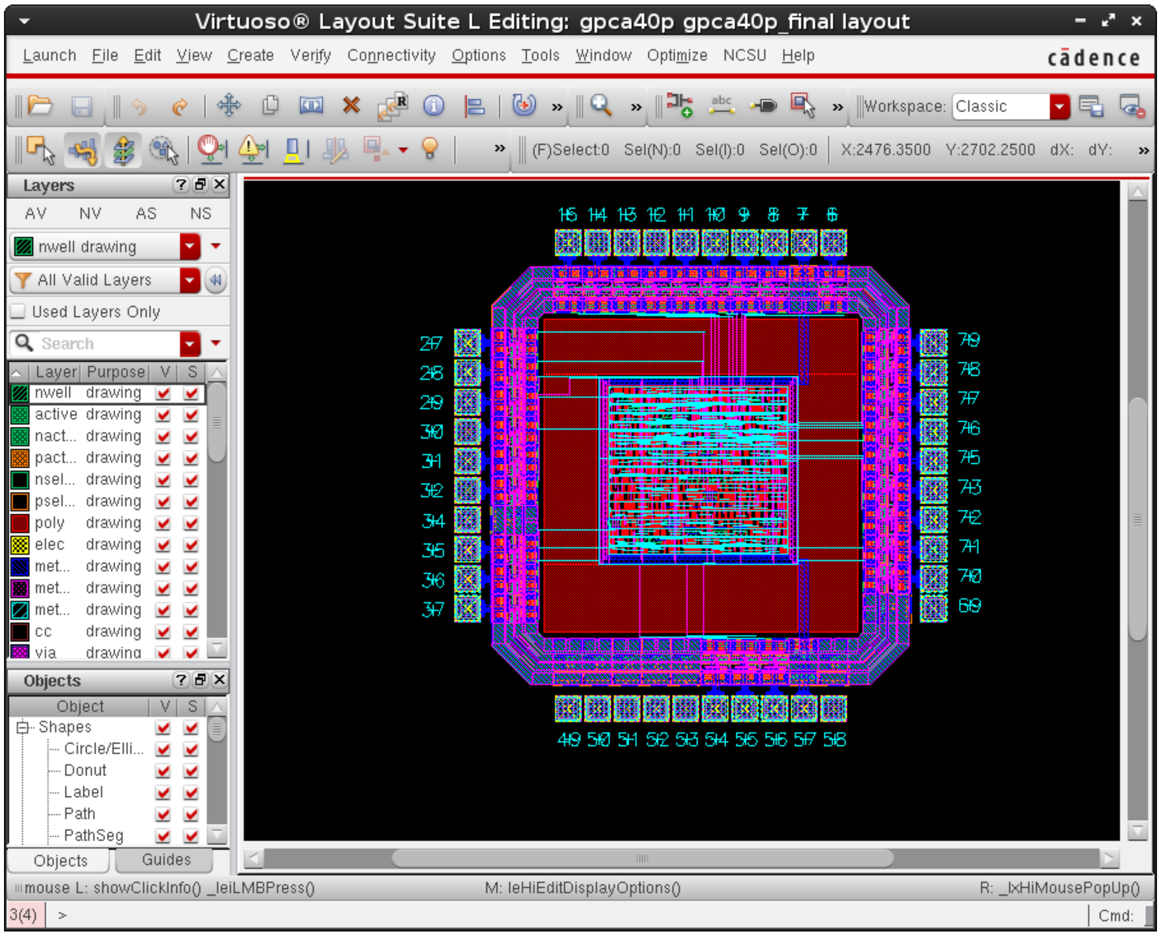
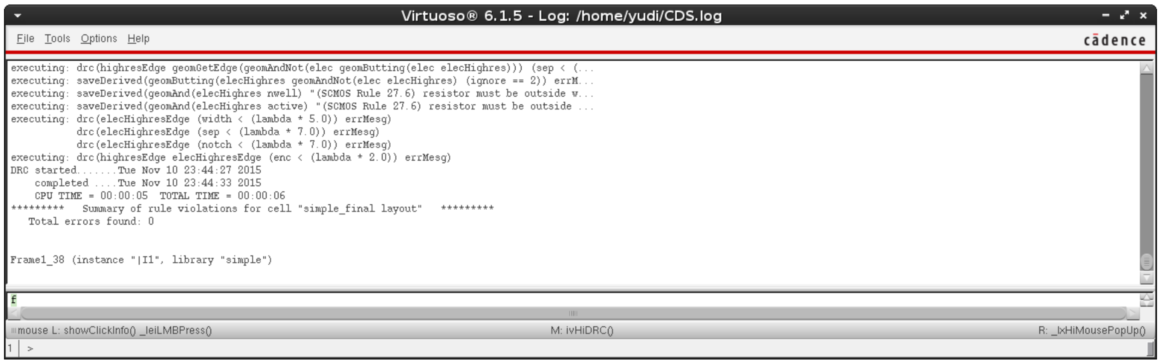


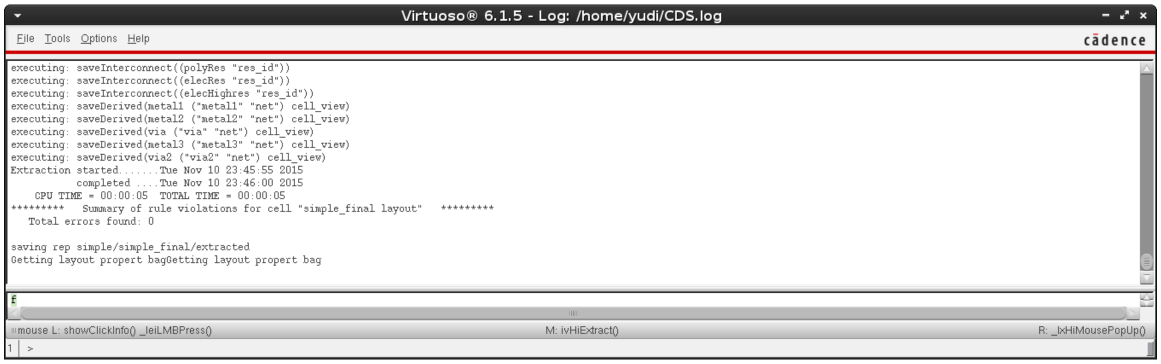
Figure 69: Final layout in Virtuoso.

DRC, Extract, LVS. They should give no error or warning.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiDRC()
R: _lxHiMousePopUp()

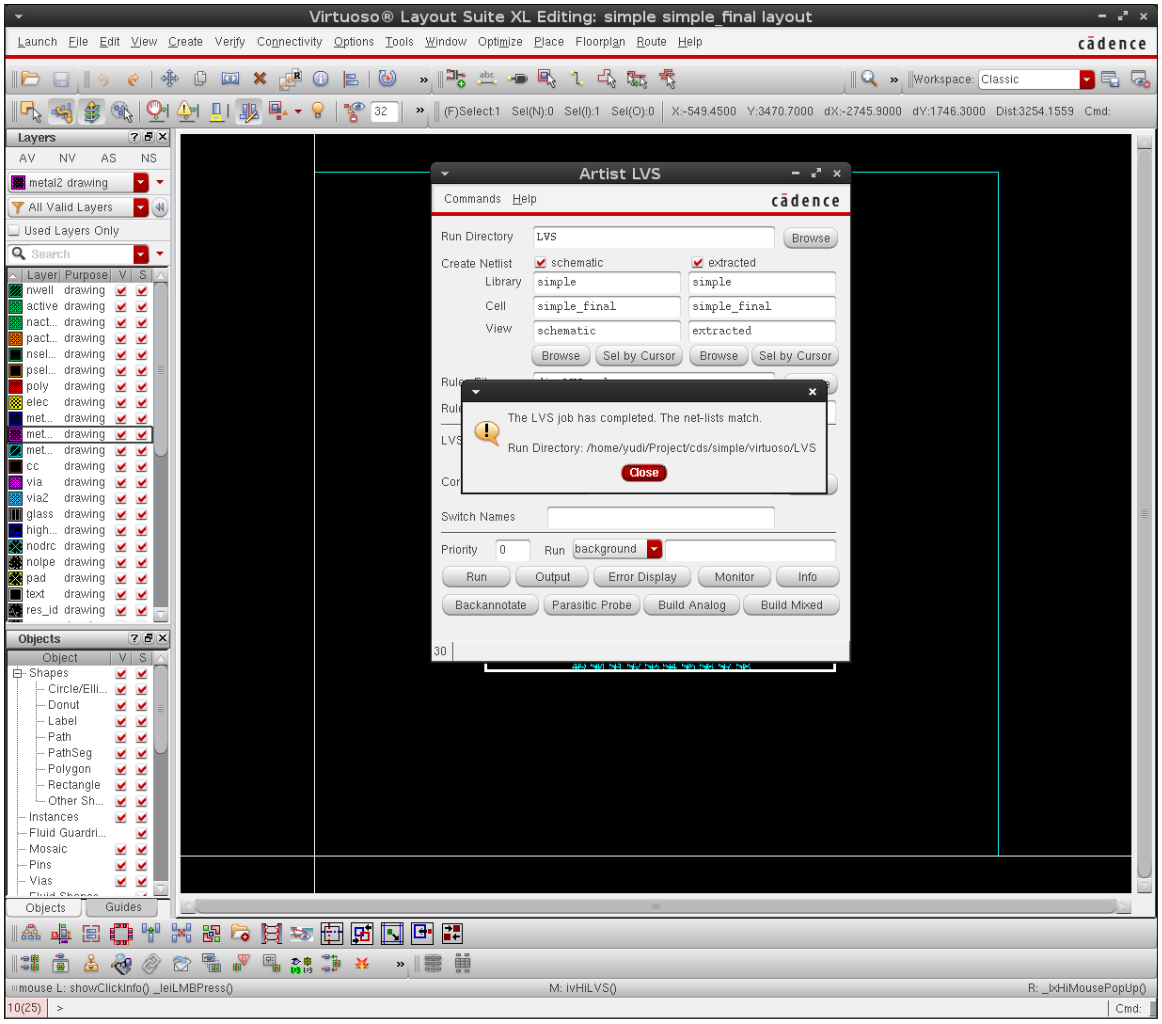
```
executing: drc(highresEdge geomGetEdge(geomAndNot(elec geomButting(elec elecHighres))) (sep < (...
executing: saveDerived(geomButting(elecHighres geomAndNot(elec elecHighres) (ignore == 2)) errM...
executing: saveDerived(geomAnd(elecHighres nwell) "(SCMOS Rule 27.6) resistor must be outside w...
executing: saveDerived(geomAnd(elecHighres active) "(SCMOS Rule 27.6) resistor must be outside ...
executing: drc(elecHighresEdge (width < (lambda * 5.0)) errMesg)
           drc(elecHighresEdge (sep < (lambda * 7.0)) errMesg)
           drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.......Tue Nov 10 23:44:27 2015
    completed ....Tue Nov 10 23:44:33 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:06
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0


Frame1_38 (instance "|I1", library "simple")
```

Figure 70: Report of successful DRC.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiExtract()
R: _lxHiMousePopUp()

```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 23:45:55 2015
           completed ....Tue Nov 10 23:46:00 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:05
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0

saving rep simple/simple_final/extracted
Getting layout propert bagGetting layout propert bag
```

Figure 71: Report of successful Extract.


Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Artist LVS
Commands Help
Browse
Sel by Cursor
Close
Run
Output
Error Display
Monitor
Info
Backannotate
Parasitic Probe
Build Analog
Build Mixed

Workspace: Classic

(F)Select:1 Sel(N):0 Sel(I):1 Sel(O):0 X:-549.4500 Y:3470.7000 dX:-2745.9000 dY:1746.3000 Dist:3254.1559 Cmd:

Run Directory: LVS

Create Netlist: [✓] schematic [✓] extracted

	schematic	extracted
Library	simple	simple
Cell	simple_final	simple_final
View	schematic	extracted

The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

Switch Names:

Priority: 0 Run: background

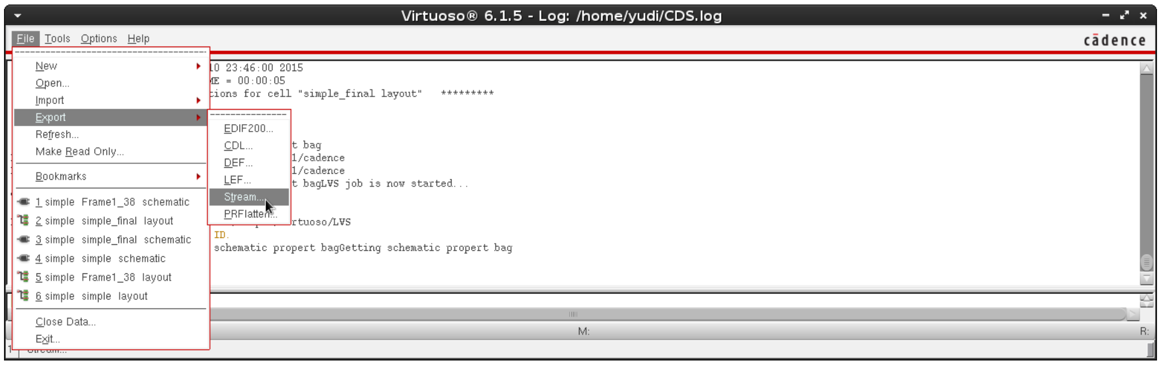
mouse L: showClickInfo() _leiLMBPress() M: ivHiLVS() R: _lxHiMousePopUp()

Figure 72: "Artist LVS" successful LVS prompt.

f) Export GDSII Stream File ("simple_final.gds")

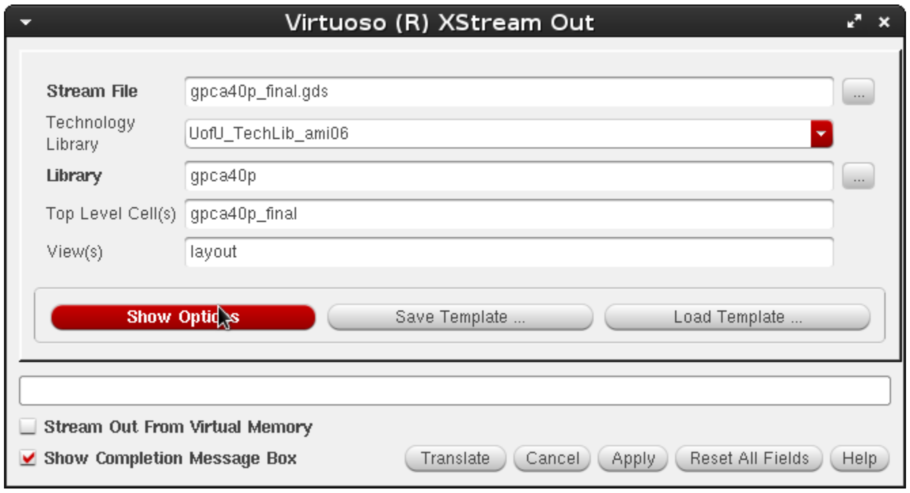
This procedure is shown in Figure 73 to Figure 78. This is the file to be sent for fabrication.

This is the file to be sent for fabrication.



```
10 23:46:00 2015
ME = 00:00:05
tions for cell "simple_final layout"   *********
t bag
1/cadence
1/cadence
t bagLVS job is now started...
rtuoso/LVS
ID.
schematic propert bagGetting schematic propert bag
```

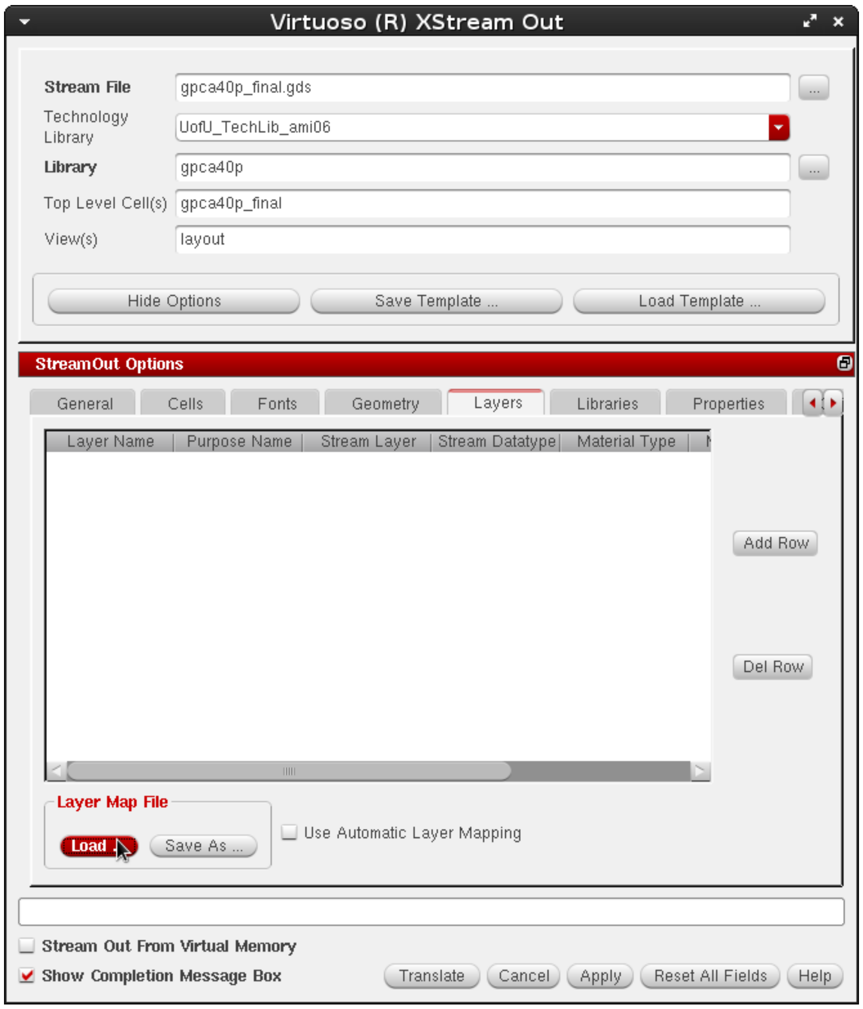
Figure 73: Start "Stream Out" from Virtuoso Layout Suite.



Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout
☐ Stream Out From Virtual Memory
☑ Show Completion Message Box

Figure 74: "Show Options" in "XStream Out" window of Virtuoso.

"Show Options", then tab "Layers", "Load ..."



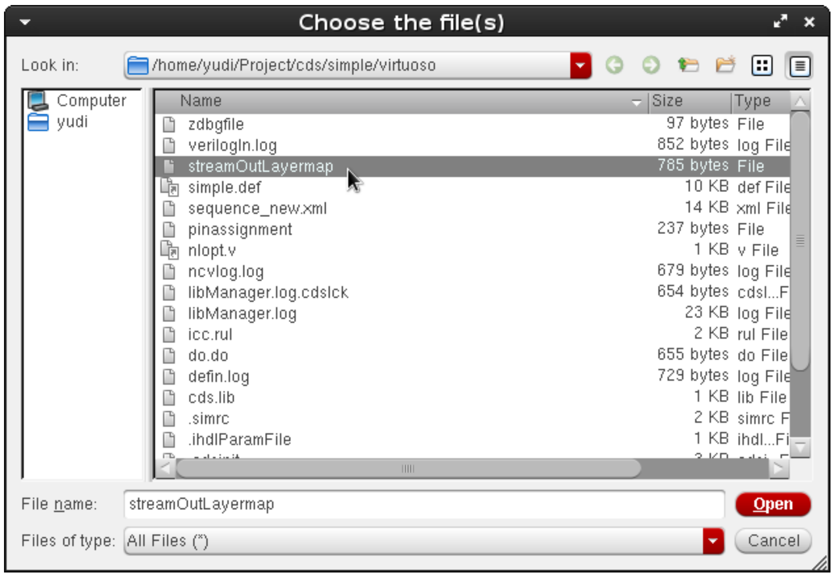
Virtuoso (R) XStream Out

Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

☐ Use Automatic Layer Mapping
☐ Stream Out From Virtual Memory
☑ Show Completion Message Box

Figure 75: "Load..." in "XStream Out" window of Virtuoso.

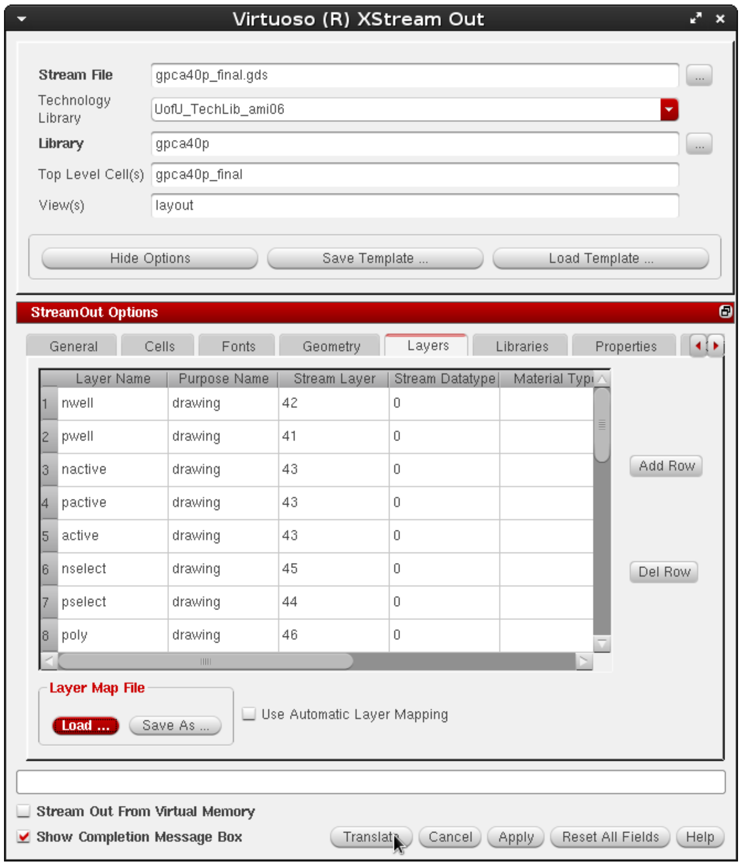
Select "streamOutLayermap" then hit "Open"



Name	Size	Type
zdbgfile	97 bytes	File
verilogIn.log	852 bytes	log File
streamOutLayermap	785 bytes	File
simple.def	10 KB	def File
sequence_new.xml	14 KB	xml File
pinassignment	237 bytes	File
nlopt.v	1 KB	v File
ncvlog.log	679 bytes	log File
libManager.log.cdslck	654 bytes	cdsl...F
libManager.log	23 KB	log File
icc.rul	2 KB	rul File
do.do	655 bytes	do File
defin.log	729 bytes	log File
cds.lib	1 KB	lib File
.simrc	2 KB	simrc F
.ihdlParamFile	1 KB	ihdl...Fi

Figure 76: Choose file for the "XStream Out".

Click "Translate".



Stream File	gpca40p_final.gds
Technology Library	UofU_TechLib_ami06
Library	gpca40p
Top Level Cell(s)	gpca40p_final
View(s)	layout

	Layer Name	Purpose Name	Stream Layer	Stream Datatype	Material Typ
1	nwell	drawing	42	0	
2	pwell	drawing	41	0	
3	nactive	drawing	43	0	
4	pactive	drawing	43	0	
5	active	drawing	43	0	
6	nselect	drawing	45	0	
7	pselect	drawing	44	0	
8	poly	drawing	46	0	

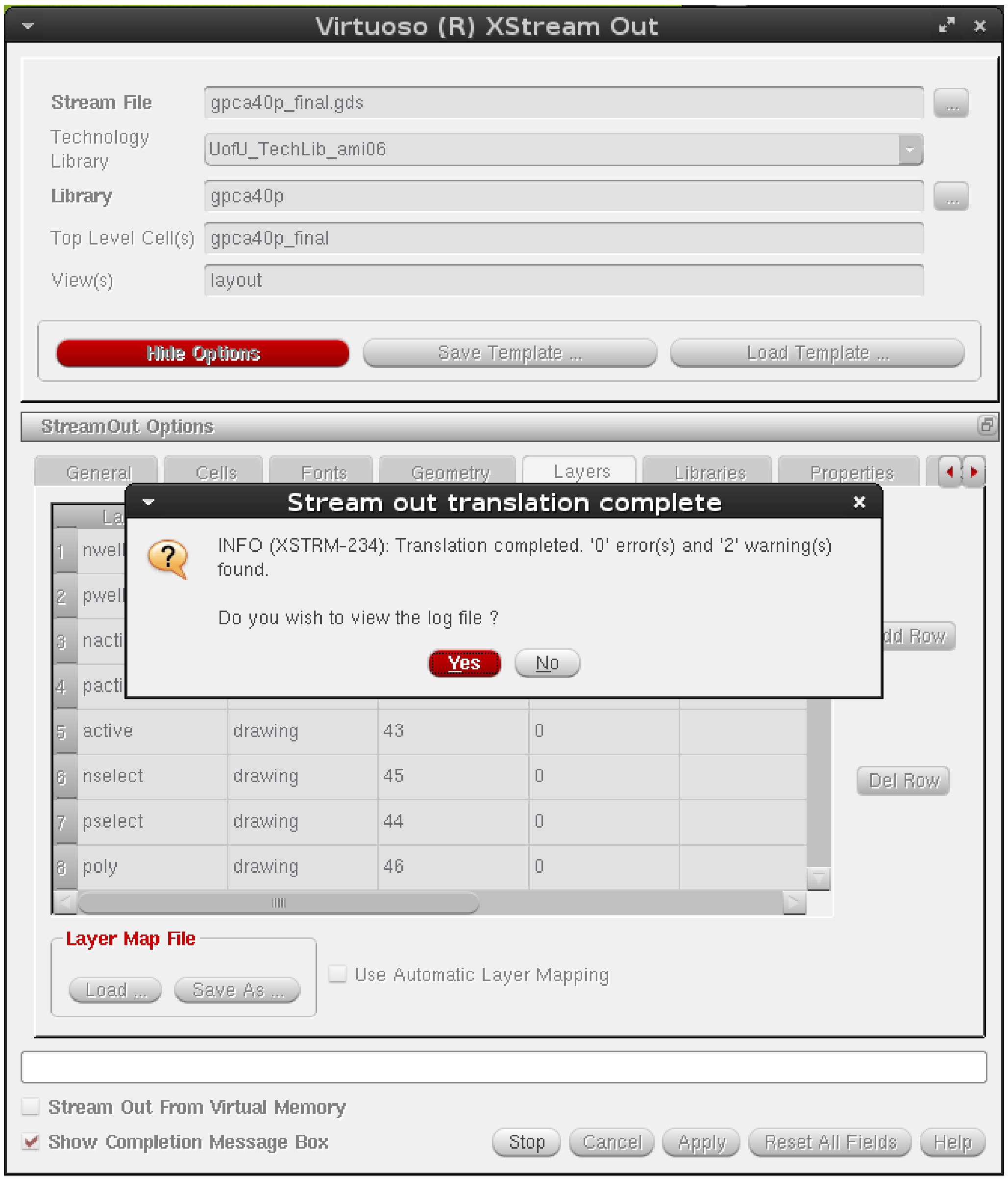
☐ Use Automatic Layer Mapping

☐ Stream Out From Virtual Memory

☑ Show Completion Message Box

Figure 77: "Translate" in "XStream Out" window of Virtuoso.

It should produce no error and the only warning is about "nodrc:drawing" (and a possible overwriting existing file warning).


Virtuoso (R) XStream Out

Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

StreamOut Options

5	active	drawing	43	0
6	nselect	drawing	45	0
7	pselect	drawing	44	0
8	poly	drawing	46	0

Stream out translation complete

INFO (XSTRM-234): Translation completed. '0' error(s) and '2' warning(s) found.

Do you wish to view the log file ?

Use Automatic Layer Mapping
Stream Out From Virtual Memory
Show Completion Message Box ✓

Figure 78: "Stream out translation complete" successful prompt.

Design Submission

In this section, the procedure for MOSIS submission with the MOSIS Educational Program (MEP) is given as a reference. More information is available in documents from MOSIS [37], and adapted from [38].

1) Fill in "MOSIS New Project Request Form":

Run Type: Shared IC Fabrication Run

Design Rules: Scalable CMOS

Technology: SCN3M_SUBM (if the second layer of poly is not used); SCN3ME_SUBM (if the second layer of poly is used).

Design Name and Password

Export Control: Standard

Substrate: none

Needs Library Installation: No

IP Included: none

Fill Authorized: Yes

Foundry: On Semi

Intended Disposition: Research

Design Size X and Y: Size including pads

Pad Count: How many pads used in the design (including signals and power)

Quantity Packaged: 5

Package Name: Depends on the design

Rotation in Package: None

Bonding Diagram Supplier: MOSIS

Downbond Locations: None

Quantity Unpackaged: 0

2) Fill in "Fabricate Form":

Go to Project Request -> Fabricate

Layout Transfer Method: I will upload layout via secure web form (HTTPS)

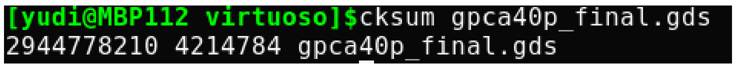
Compression/Encryption: Uncompressed

Generate the checksum and Count for the GDS file (Figure 79)

Layout Status: Final

Layout Format: GDS

Top Structure: the name of the top-level (e.g. "simple")



```
[yudi@MBP112 virtuoso]$cksum gpca40p_final.gds
2944778210 4214784 gpca40p_final.gds
```

Figure 79: Generating checksum with GNU cksum

Results

The array has been extended to 7 rows, which can achieve operations of bigger number. The result is summarized in Table 3. The design has been sent to MOSIS for fabrication.

Table 3. Summary of the 7-row extension

Operation	**Original**	**Extended**
Multiply	7 by 5	9 by 7
Divide	7 by 4	9 by 6
Square	5	7
Square root	10	14

Conclusion

In this chapter, the original design of pipeline array with 5 rows is extended to 7 rows as a case study. By extending this array to 7 rows, one can have the arithmetic operations of increased number of bits. Although the extension up to 7 rows is discussed in this chapter, the procedure can be applied to any number of rows. We have to limit the number of rows so as to take into account the size of the chip to be developed. In addition, the VLSI implementation of such a design has been discussed and detailed procedure for the implementation is included. The design has been sent to MOSIS for fabrication. The chip will be tested once we get the fabricated chip back. The behavior Verilog code for this design is listed in the Appendix.

CHAPTER 3 DESIGN TO MEET TECHNICAL CONSTRAINS

Introduction

The VLSI work flow is an iterative process, which introduces problem. For example, if a design itself turns out requiring too much resource such as area at the end of the physical design phase, the entire design are to be reset and start over again, which is time consuming. There is no way to mitigate this issue because the actual dimension of the design is only known after physical design. If a method is found, such that the resource requirement of the design is estimated based on the contains first, then starting the design with this estimated information in mind, the problem incurred by this design iteration is much mitigated or even eliminated if the estimation algorithm is accurate enough.

For the design of the pipeline array, traditionally at the specification phase, the designer picks up a number as the number of rows of the pipeline array, then continues the design process. Once the physical design phase is finished, the layout is checked to make sure that it's within constrains such as pin count and area. If the design cannot meet these contains, the designer then modifies the specification and start over again. For this VLSI implementation of the pipeline array, ideally the design can go infinitely large with infinite accuracy, but here are two technical constrains posed by MOSIS Educational Program: maximum pin count and chip area. According to the documents from MOSIS [30], the design should be within 1 tiny chip unit (TCU), which means the design should only contains 40 pins maximum and the area should be within 1500 µm by 1500 µm including pad frame (approximately 810000 µm2 of usable core area [24]).

In this chapter, an algorithm is devised, such that the pin count and the chip area of the pipeline can be estimated based on the number of rows, which is introduced in the following section. The procedure for VLSI implementation of this design is also given.

Design

This algorithm is derived in the following way. 8 experimental designs of the pipeline array are conducted, with 1 to 8 rows respectively. The number of rows in these experimental designs is specifically chosen to be small enough to keep the complexity of the circuit low, so that they don't require much effort to design. Then data of the cost of these designs such as pin count and chip area are collected for further analysis to derive the estimation algorithm. These sample data are summarized Table 4 and illustrated in Figure 80 and Figure 81 respectively.

Table 4: Summary of resource requirement of pipeline array designs.

Number of rows	**Pin count**	**chip area (µm^2)**	**gate count (standard cell)**
1	14	16330	22
2	22	48730	64
3	30	95515	127
4	38	154548	207
5	46	229781	304
6	54	314345	418
7	62	407462	558
8	70	524880	697

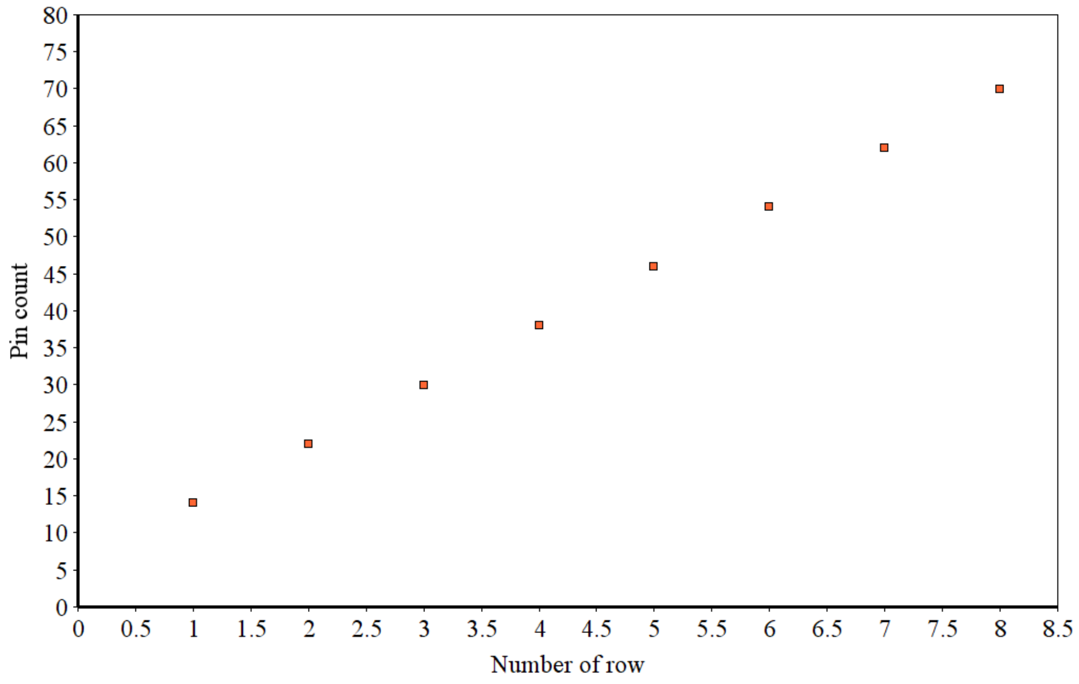


Figure 80: Number of row vs. pin count.

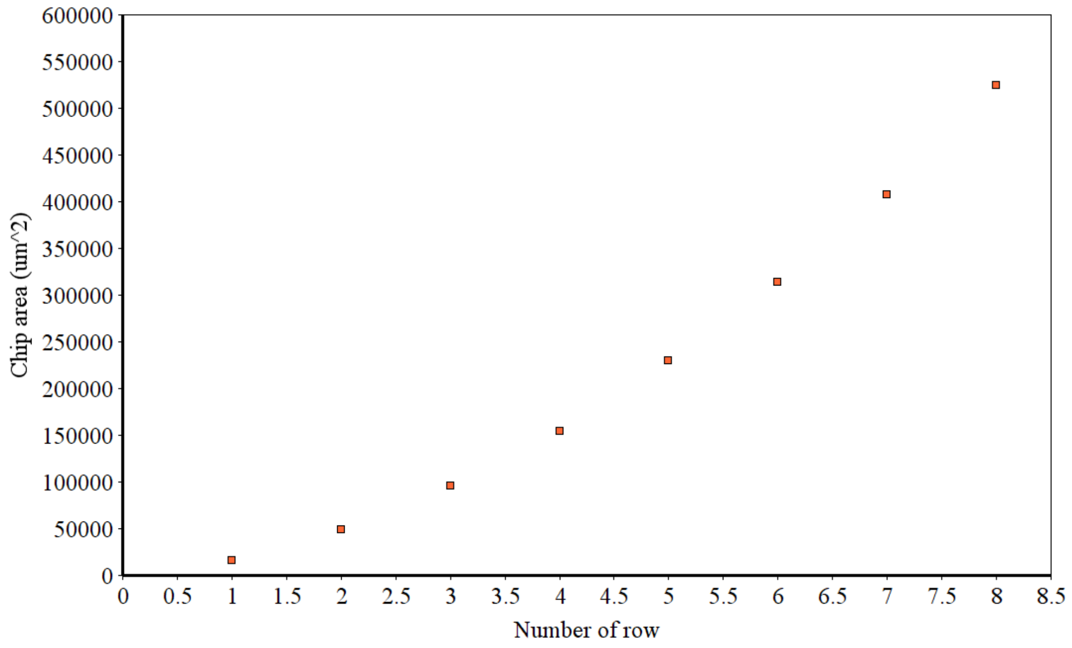


Figure 81: Number of row vs. chip area.

Based on this information, a statistic method called least squares [39] is used as the approach of cost estimation. The method of least squares is a way in regression analysis to the estimate solution of systems, so that the overall solution minimizes the sum of the squares of individual errors. The objective of least square is to adjust the parameters of a model function to best fit a data set.

For pin count estimation, linear least square are used to estimate pin counts. Given the sample data sets, the goal is to find the relationship of the pin count (y) and the number of rows (x) $y = a_0 + a_1 x$ such that it can best fit the sample data sets. The calculation begins by solving the linear equations below.

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix} \tag{6}$$

After substituting the variables with the sample data sets shown in Table 4, the result is as follows.

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix} \tag{7}$$

Hence the estimation function of the relationship of the pin count (y) and the number of rows (x) is given below.

$$y = 6 + 8x \tag{8}$$

For chip area estimation, quadratic least square are used to estimate chip area because area is a quadratic function of the number of rows. Given the sample data sets, the goal is to find the relationship of the chip area (y) and number of rows (x) $y = a_0 + a_1 x + a_2 x^2$ such that it can best fit the sample data sets. The calculation begins by solving the linear equations below.

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix} \tag{9}$$

Then substitute the variables with the sample data sets shown in Table 4.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \frac{-247377}{56} \\ \frac{776511}{56} \\ \frac{364479}{56} \end{bmatrix} \tag{10}$$

Hence the estimation function of the relationship of the chip area (y) and the number of rows (x) is given below.

$$y = \frac{1}{56}(-247377 + 776511x + 364479x^2) \tag{11}$$

Due to the limitations of the MOSIS Educational Program (MEP) [30] and the fabrication budget allotted, the VLSI implementation should be within 1 tiny chip unit (TCU), which means the design should only contains 40 pins maximum and the area should be within 1500 µm by 1500 µm including pad frame (approximately 900 µm by 900 µm of usable core area [24]). According to the estimation formula 11 substituting y equals 900 * 900:

$$900 * 900 = \frac{1}{56}(-247377 + 776511x + 364479x^2) \tag{12}$$

The positive root of the above equation equals 10.17, which estimates that the maximum number of rows allowed for this given area is 10.

To solve the excessive pin count issue, multiplexers are used in the design. As illustrated in Table 2, for each operation, not all inputs and outputs are used. Some inputs and outputs are either constant or "don't care", which are not needed to be connected externally. Only inputs and outputs for variable signals of data are needed to be connected. Based on this observation, the solution of designing to fit within 40 pins is devised, as illustrated in Figure 82.

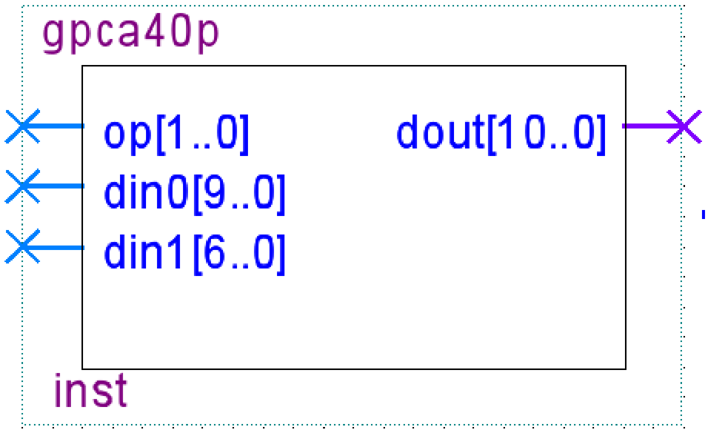


Figure 82: Overview of the pipeline array implementation within 40 pins.

To achieve such an implementation, multiplexers are used as a wraparound between the original design and the pad frame, as illustrated in Figure 83. Based on what operation is to be performed indicated by signal "op", the multiplexers drives the original circuit to either constant signals or the desired signals of data inputs, as indicated in Table 2. These signals are ready to let the pipeline array circuit to operate correctly. Once the operation of the pipeline array is done, the results are selected by another set of multiplexers, to drive the data output pins. The operations of the new design are illustrated in Table 5.

The behavior Verilog code for this design is listed in the Appendix.

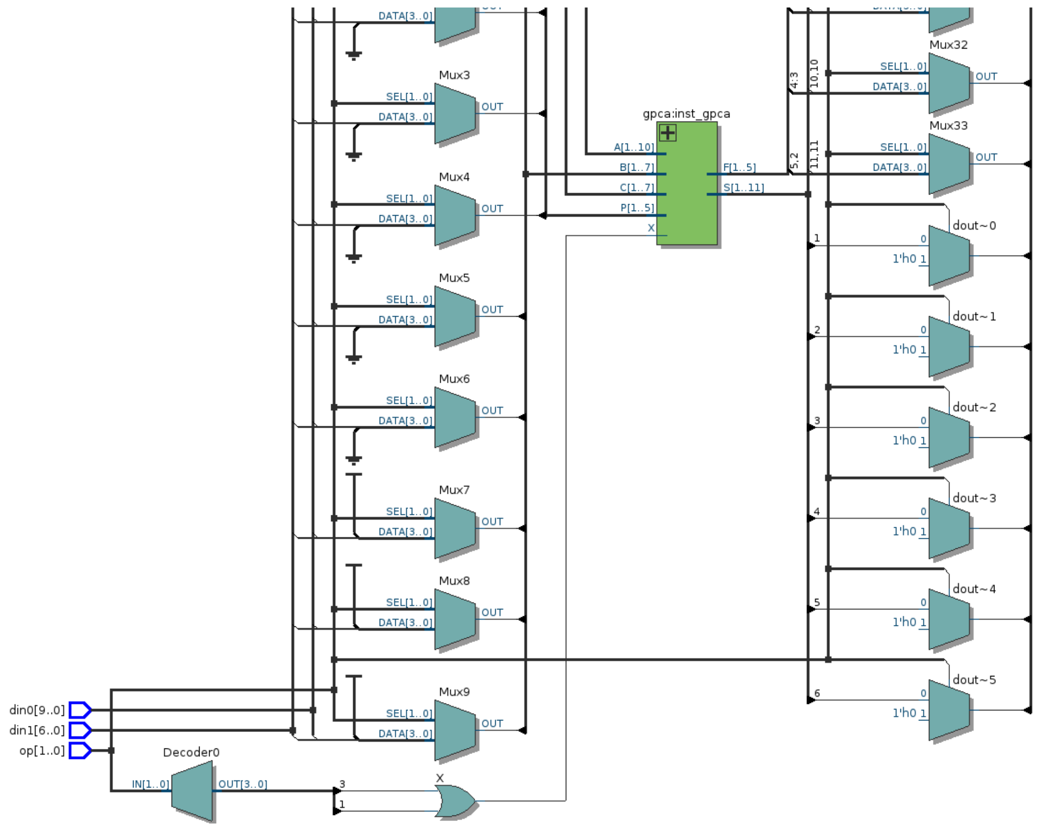


Figure 83: Logic diagram (partial) of the implementation within 40 pins.

Table 5: Operations of the new implementation within 40 pins.

Operation	**Input**			**Output**
	op	***din0***	***din1***	***dout***
Square	00	P	don't care	S
Square Root	01	A	don't care	F
Multiplication	10	B = C	P	S
Division	11	A	B=C	F

In order to make sure the circuit operates correctly as required, it's simulated in functional simulator and implemented on FPGA, to see if it ready meets the requirement. The functional simulator used here is called ModelSim by Mentor Graphics [40]. The functional simulation of this circuit is shown in Figure 84. In this simulation, all 4 operations (namely square, square root, multiplication and division) are tested with random data sets. According to the result shown in Figure 84, all the results of calculations done by this circuit are mathematically correct, hence the circuit functions exactly the same as required.

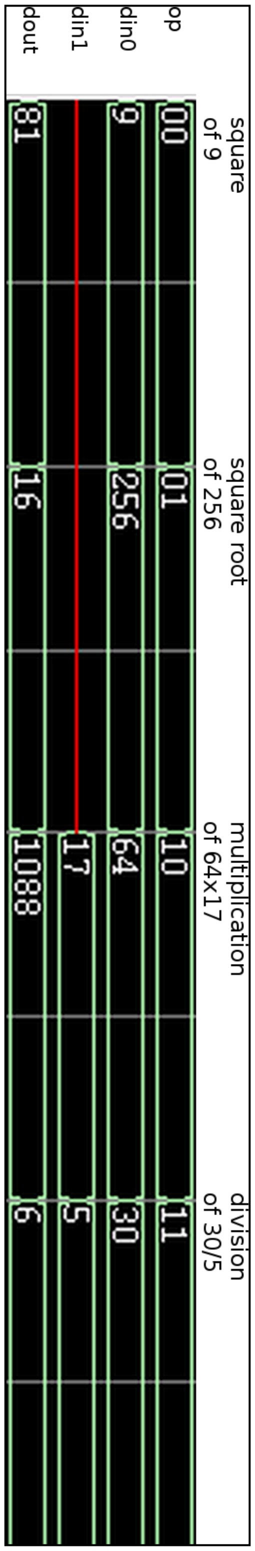
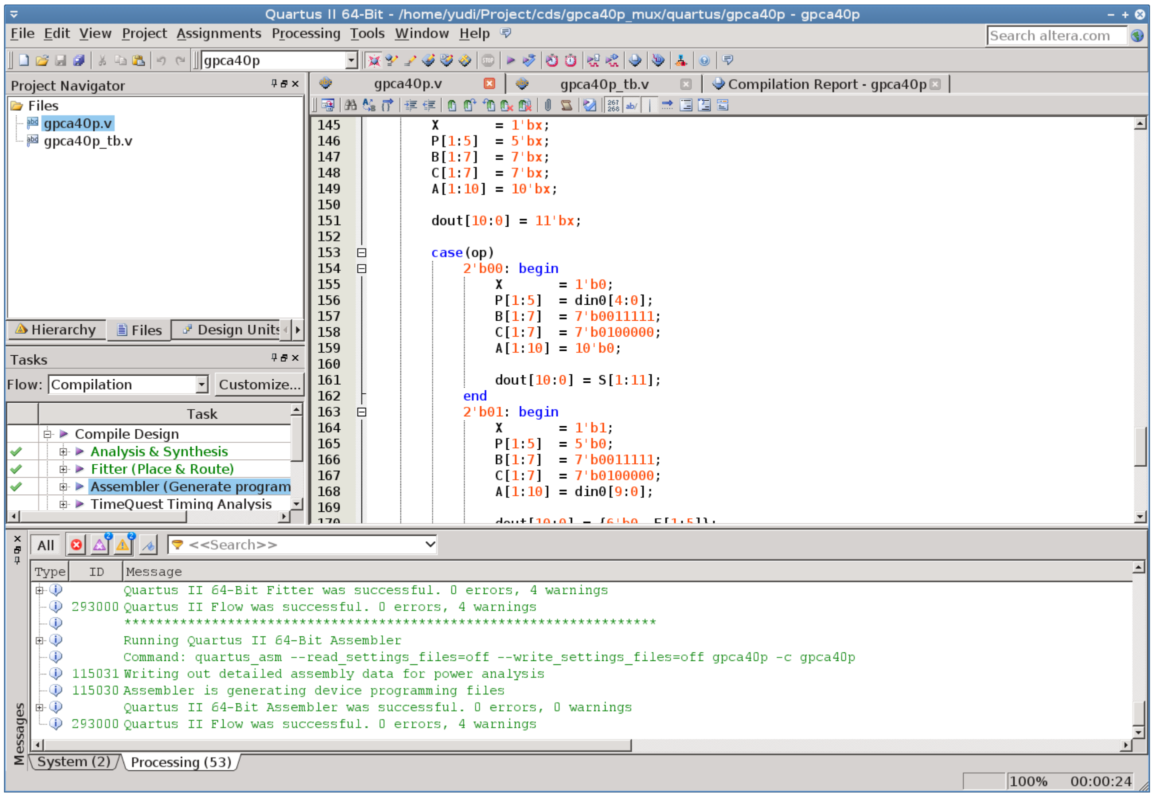


Figure 84: Simulation result of the implementation within 40 pins.

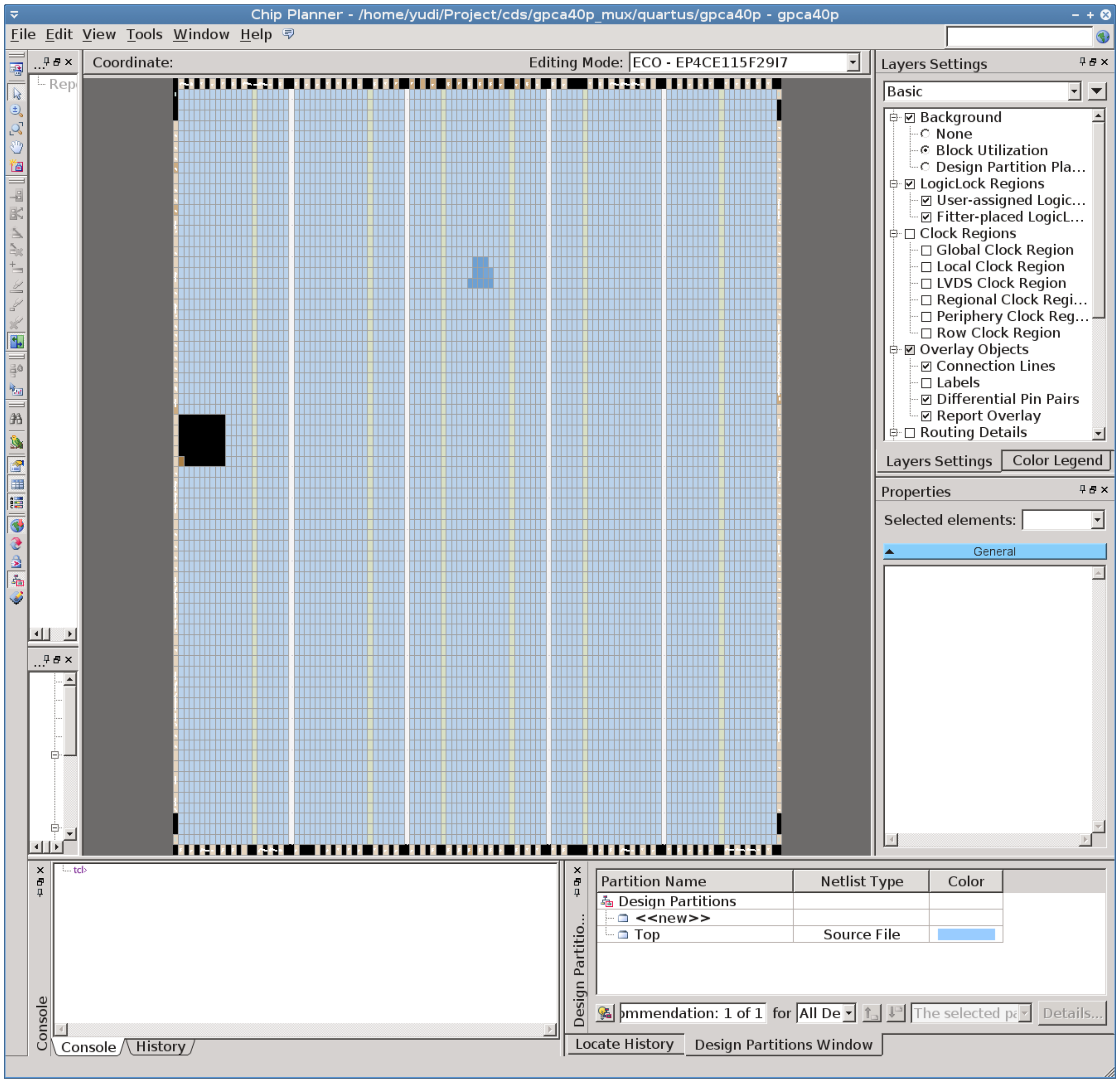
A field-programmable gate array (FPGA) is an integrated circuit which can be configured flexibly by the designer, hence gaining the name "field-programmable" [41]. This functionality makes FPGA useful in the process of VLSI design. The circuit described here is implemented on FPGA for verification and validation before implementing in VLSI. Result of synthesis, place and route on FPGA with Altera Quartus II software [42] is shown in Figure 85 and Figure 86 respectively. Then the circuit is tested on the FPGA. Once the circuit is fully validated, it is ready to be implemented in VLSI.



```
        X       = 1'bx;
        P[1:5]  = 5'bx;
        B[1:7]  = 7'bx;
        C[1:7]  = 7'bx;
        A[1:10] = 10'bx;

        dout[10:0] = 11'bx;

        case(op)
            2'b00: begin
                X       = 1'b0;
                P[1:5]  = din0[4:0];
                B[1:7]  = 7'b0011111;
                C[1:7]  = 7'b0100000;
                A[1:10] = 10'b0;

                dout[10:0] = S[1:11];
            end
            2'b01: begin
                X       = 1'b1;
                P[1:5]  = 5'b0;
                B[1:7]  = 7'b0011111;
                C[1:7]  = 7'b0100000;
                A[1:10] = din0[9:0];
```

```
       Quartus II 64-Bit Fitter was successful. 0 errors, 4 warnings
293000 Quartus II Flow was successful. 0 errors, 4 warnings
       *******************************************************************
       Running Quartus II 64-Bit Assembler
       Command: quartus_asm --read_settings_files=off --write_settings_files=off gpca40p -c gpca40p
115031 Writing out detailed assembly data for power analysis
115030 Assembler is generating device programming files
       Quartus II 64-Bit Assembler was successful. 0 errors, 0 warnings
293000 Quartus II Flow was successful. 0 errors, 4 warnings
```

Figure 85: Synthesis result of FPGA.



Chip Planner - /home/yudi/Project/cds/gpca40p_mux/quartus/gpca40p - gpca40p

Coordinate:

Editing Mode: ECO - EP4CE115F29I7

Layers Settings: Basic

- [✓] Background
  - ○ None
  - ● Block Utilization
  - ○ Design Partition Pla...
- [✓] LogicLock Regions
  - [✓] User-assigned Logic...
  - [✓] Fitter-placed LogicL...
- [ ] Clock Regions
  - [ ] Global Clock Region
  - [ ] Local Clock Region
  - [ ] LVDS Clock Region
  - [ ] Regional Clock Regi...
  - [ ] Periphery Clock Reg...
  - [ ] Row Clock Region
- [✓] Overlay Objects
  - [✓] Connection Lines
  - [ ] Labels
  - [✓] Differential Pin Pairs
  - [✓] Report Overlay
- [ ] Routing Details

Selected elements:

Partition Name	Netlist Type	Color
Design Partitions		
<<new>>		
Top	Source File	🟦

ommendation: 1 of 1 for All De

The selected p

Figure 86: Place and route result (partial) on FPGA.

Implementation

In this section, a step-by-step procedure for VLSI digital design is given, for the 40-pin implementation of the pipeline array.

There are many fabrication technologies available nowadays, from various fabrication facilities such as GlobalFoundries and TSMC, in technologies such as 14 nm, 28 nm, 40 nm, 65 nm, 0.13 µm and 0.18 µm and so on [30]. Since we use the MOSIS Educational Program (MEP) for free fabrication service, which limits us to use the ON Semi 0.50 µm CMOS (C5N) technology [30], C5N is used in this procedure. However, the general procedure is the same for other technologies. There are many computer aided design (CAD) software available for VLSI design, such as Cadence and Synopsys. This procedure uses Cadence Encounter Digital Implementation Systems 14.00, Cadence Virtuoso Design Environment 6.15 along with NCSU CDK 1.6.0 [21], UofU Technology Library and UofU Standard Cell Library [24].

This procedure consists of 3 parts, each of which represents one major step for the VLSI design. They are logic synthesis, place and route and chip assembly respectively, which are introduced for the rest of this section. In the end, the procedure for MOSIS submission is given.

Logic Synthesis

In digital logic design, logic synthesis is a procedure by which a behavior-level HDL code describing the function of a circuit, is turned into a gate-level netlist which describes the implementation of a design in terms of logic gates, typically using a computer program called a synthesis tool.

In this subsection, a procedure for synthesis is given. For the concision of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [31]. There are other alternative RTL synthesizers available as well, such as Design Compile7r by Synopsys. If tools other than what's described here are used, it's advised to refer to their respective manuals. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL Compiler

2) Prerequisites before This Step:

Behavior Verilog Code (e.g. "simple.v")

Tcl Script for RC Compiler ("rc.cmd", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Netlist Verilog Code ("nl.v")

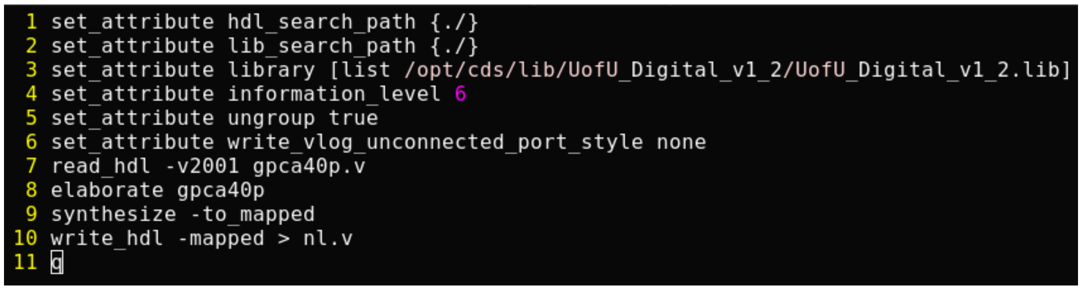
4) Steps:

a) Modify rc.cmd based on the requirement (as shown in Figure 87).

Line 3: Change UofU standard cell library path to where it's installed.

Line 7: Change "gpca40p.v" to the file name of Verilog code (e.g. "simple.v").

Line 8: Change "gpca40p" to the top-level entity name (e.g. "simple").



```
 1 set_attribute hdl_search_path {./}
 2 set_attribute lib_search_path {./}
 3 set_attribute library [list /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib]
 4 set_attribute information_level 6
 5 set_attribute ungroup true
 6 set_attribute write_vlog_unconnected_port_style none
 7 read_hdl -v2001 gpca40p.v
 8 elaborate gpca40p
 9 synthesize -to_mapped
10 write_hdl -mapped > nl.v
11 q
```

Figure 87: rc.cmd.

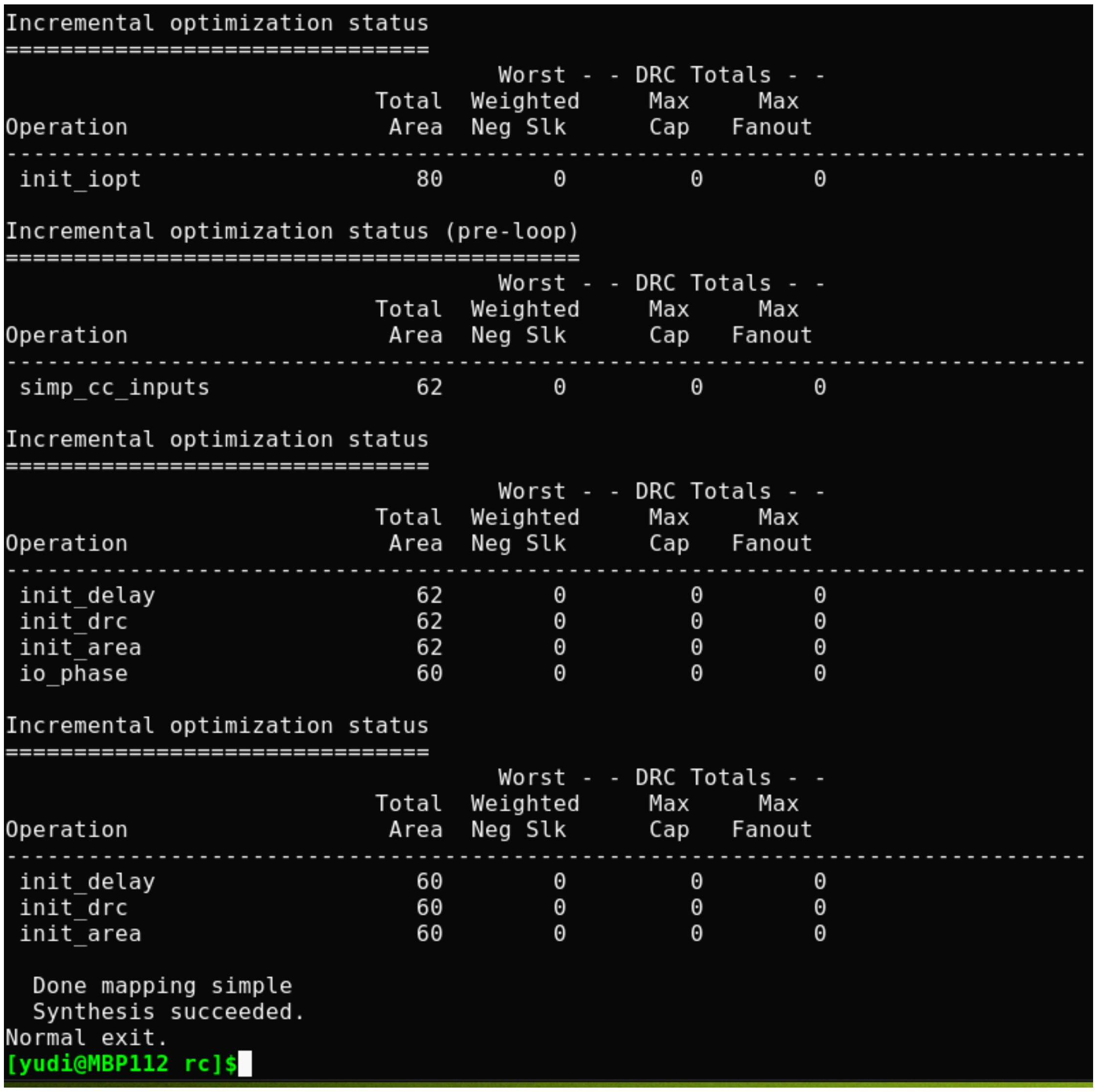
b) Run

$ rc -files rc.cmd

c) Check the result (Figure 88)

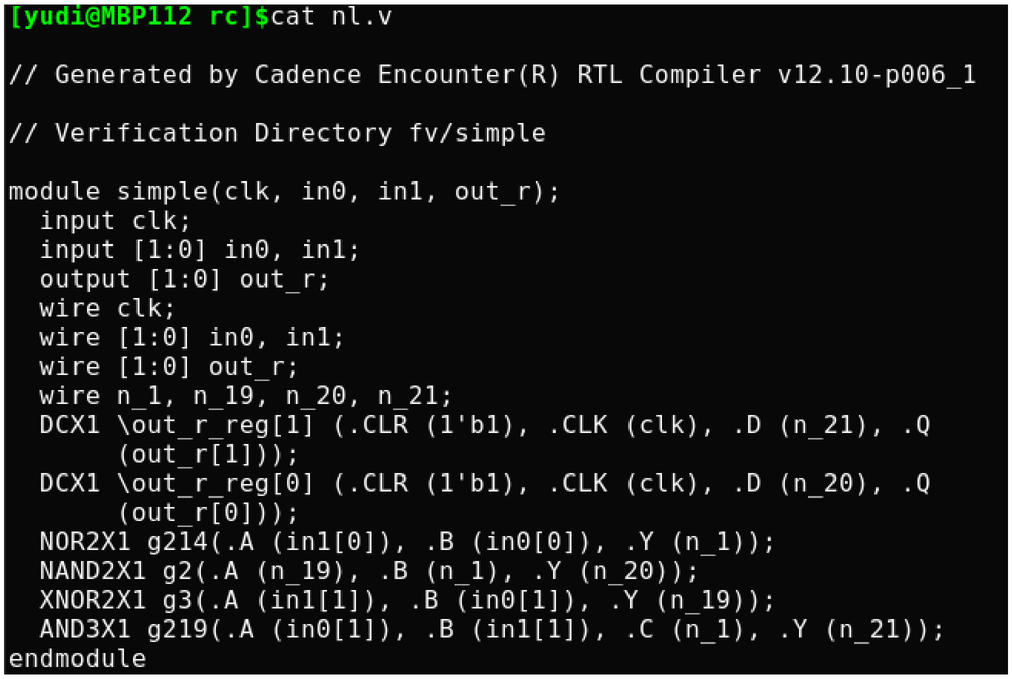
In the end, a netlist file called "nl.v" (Figure 89) is generated containing information which will be used later for place and route.

If everything goes smooth as above, continue to the next step. If anything goes wrong, fix it first before continuing further.



```
Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_iopt                    80        0        0        0

Incremental optimization status (pre-loop)
==========================================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 simp_cc_inputs               62        0        0        0

Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   62        0        0        0
 init_drc                     62        0        0        0
 init_area                    62        0        0        0
 io_phase                     60        0        0        0

Incremental optimization status
===============================
                                    Worst - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   60        0        0        0
 init_drc                     60        0        0        0
 init_area                    60        0        0        0

  Done mapping simple
  Synthesis succeeded.
Normal exit.
[yudi@MBP112 rc]$
```

Figure 88: Synthesis summary generated by RC Compiler.



```
[yudi@MBP112 rc]$cat nl.v

// Generated by Cadence Encounter(R) RTL Compiler v12.10-p006_1

// Verification Directory fv/simple

module simple(clk, in0, in1, out_r);
  input clk;
  input [1:0] in0, in1;
  output [1:0] out_r;
  wire clk;
  wire [1:0] in0, in1;
  wire [1:0] out_r;
  wire n_1, n_19, n_20, n_21;
  DCX1 \out_r_reg[1] (.CLR (1'b1), .CLK (clk), .D (n_21), .Q
       (out_r[1]));
  DCX1 \out_r_reg[0] (.CLR (1'b1), .CLK (clk), .D (n_20), .Q
       (out_r[0]));
  NOR2X1 g214(.A (in1[0]), .B (in0[0]), .Y (n_1));
  NAND2X1 g2(.A (n_19), .B (n_1), .Y (n_20));
  XNOR2X1 g3(.A (in1[1]), .B (in0[1]), .Y (n_19));
  AND3X1 g219(.A (in0[1]), .B (in1[1]), .C (n_1), .Y (n_21));
endmodule
```

Figure 89: Synthesis summary generated by RC Compiler.

Place and Route

"Place and route" is a stage in the process of VLSI design, in which the location to place all the logic elements within a generally limited amount of space and the way of all the wires needed to connect the logic elements are decided.

In this section, a procedure for "place and route" is given. For the brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [33]. If the reader is interested in using GUI commands instead of TCL scripts, please refer to EDI System Menu Reference [34] and textbooks [24] for more information. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL-to-GDSII System

2) Prerequisites before This Step:

Netlist Verilog Code ("nl.v" from the last step)

Tcl Script for RC Compiler ("encounter.cmd", given in Appendix)

Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", given in Appendix)

Synopsys Design Constraints ("typical.sdc", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Optimized Netlist Verilog Code ("nlopt.v")

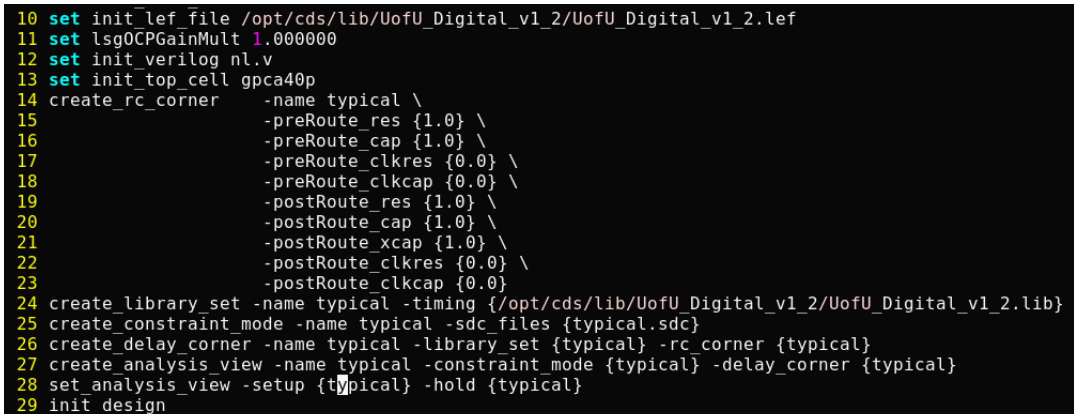
Design Exchange Format (DEF) File (e.g. "simple.def")

4) Steps:

a) Modify Tcl Script for RC Compiler ("encounter.cmd", Figure 90)

Line 10, 24: May change UofU standard cell library path to where it's installed

Line 13, 99: Change "gpca40p" to the top-level entity name (e.g. "simple")

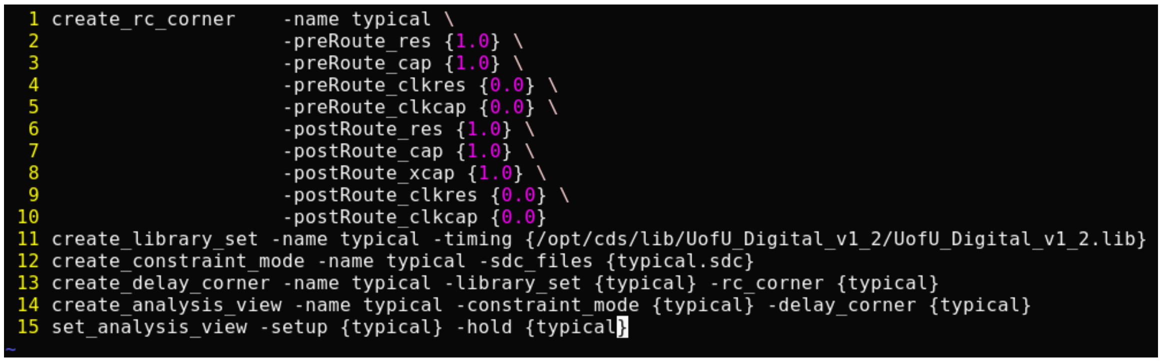


```
10 set init_lef_file /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lef
11 set lsgOCPGainMult 1.000000
12 set init_verilog nl.v
13 set init_top_cell gpca40p
14 create_rc_corner    -name typical \
15                     -preRoute_res {1.0} \
16                     -preRoute_cap {1.0} \
17                     -preRoute_clkres {0.0} \
18                     -preRoute_clkcap {0.0} \
19                     -postRoute_res {1.0} \
20                     -postRoute_cap {1.0} \
21                     -postRoute_xcap {1.0} \
22                     -postRoute_clkres {0.0} \
23                     -postRoute_clkcap {0.0}
24 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
25 create_constraint_mode -name typical -sdc_files {typical.sdc}
26 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
27 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
28 set_analysis_view -setup {typical} -hold {typical}
29 init_design
```

Figure 90: encounter.cmd.

b) Modify Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", Figure 91)

Line 11: May change UofU standard cell library path to where it's installed



```
 1 create_rc_corner    -name typical \
 2                     -preRoute_res {1.0} \
 3                     -preRoute_cap {1.0} \
 4                     -preRoute_clkres {0.0} \
 5                     -preRoute_clkcap {0.0} \
 6                     -postRoute_res {1.0} \
 7                     -postRoute_cap {1.0} \
 8                     -postRoute_xcap {1.0} \
 9                     -postRoute_clkres {0.0} \
10                     -postRoute_clkcap {0.0}
11 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
12 create_constraint_mode -name typical -sdc_files {typical.sdc}
13 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
14 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
15 set_analysis_view -setup {typical} -hold {typical}
~
```

Figure 91: mmmc.tcl.

c) Run

$ encounter -init encounter.cmd

d) Check the result

It should run all the way to the final step without any errors if the above steps are

followed correctly, as shown in Figure 92 and Figure 93. In the end, a DEF file (e.g. "simple.def") for the chip layout (without pad frame) as well as a netlist file called "nlopt.v" is generated.

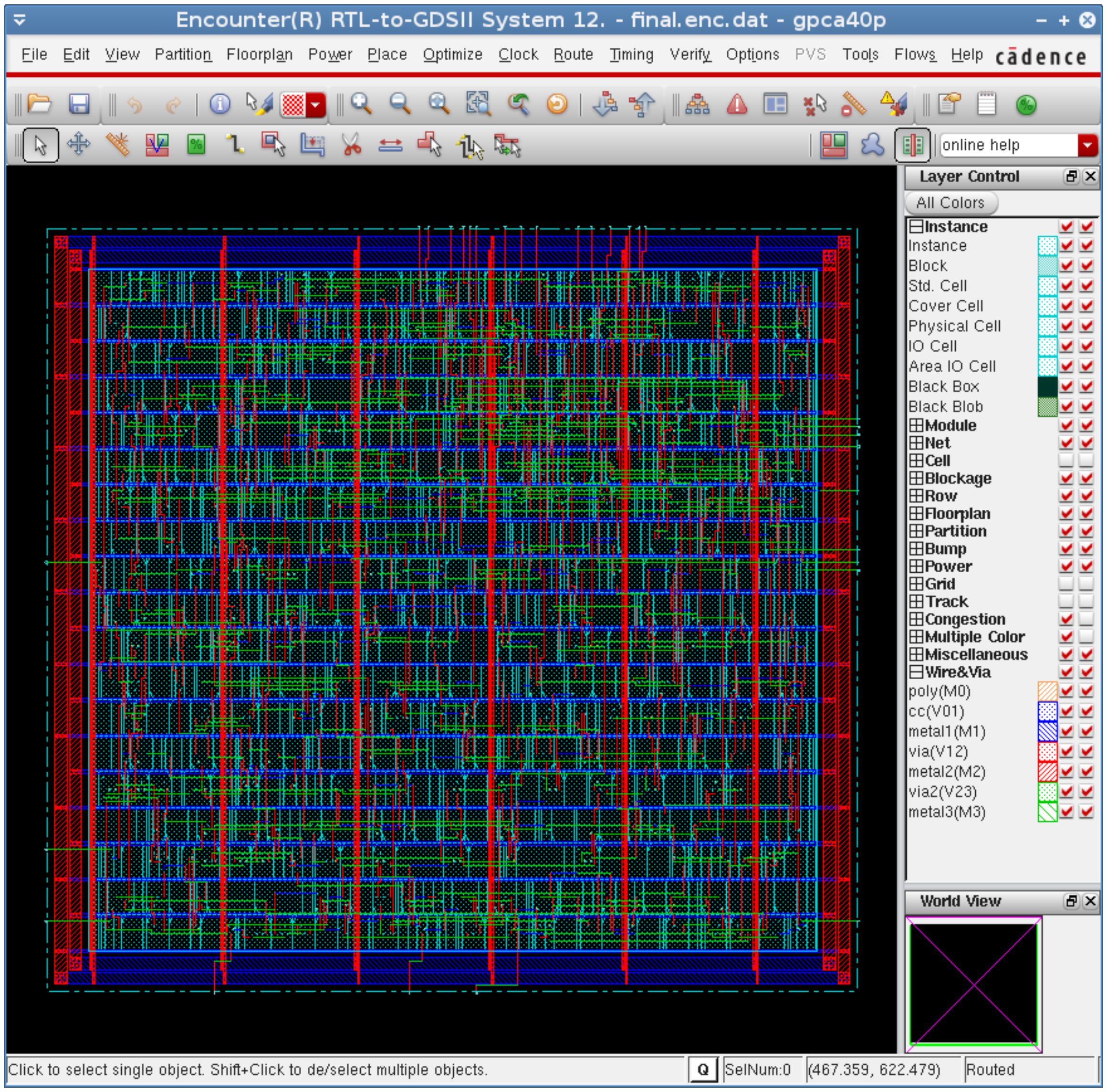
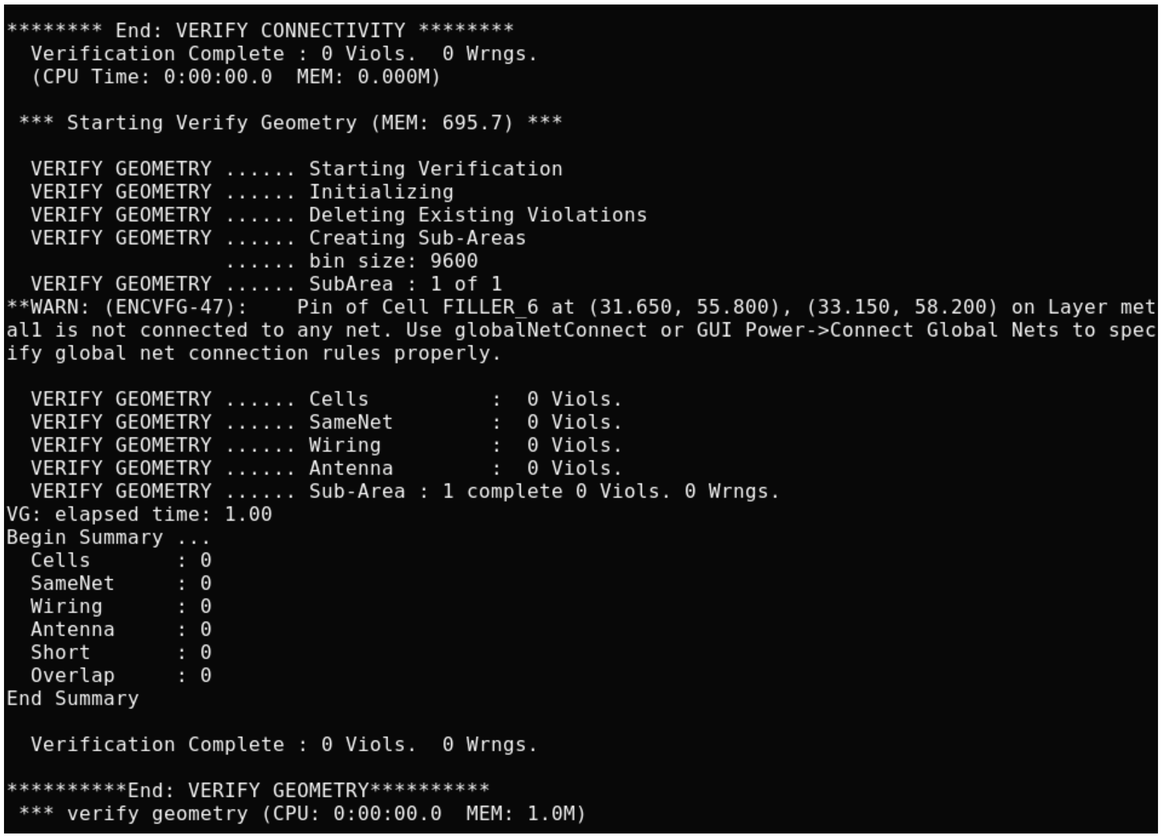


Figure 92: Final view in Encounter.



```
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)

 *** Starting Verify Geometry (MEM: 695.7) ***

  VERIFY GEOMETRY ...... Starting Verification
  VERIFY GEOMETRY ...... Initializing
  VERIFY GEOMETRY ...... Deleting Existing Violations
  VERIFY GEOMETRY ...... Creating Sub-Areas
                  ...... bin size: 9600
  VERIFY GEOMETRY ...... SubArea : 1 of 1
**WARN: (ENCVFG-47):    Pin of Cell FILLER_6 at (31.650, 55.800), (33.150, 58.200) on Layer met
al1 is not connected to any net. Use globalNetConnect or GUI Power->Connect Global Nets to spec
ify global net connection rules properly.

  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: 1.00
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: 0:00:00.0  MEM: 1.0M)
```

Figure 93: DRC report generated by Encounter.

Chip Assembly

As a requirement by AMI05 technology [35], design should be submitted along with pad frame. This section gives the procedure to assemble the pad frame with the chip. For the brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [36]. Materials such as textbooks [24] also have many useful information for reference. The detailed procedure and codes used are included in [32].

1) Tools:

Cadence Virtuoso Design Environment

2) Prerequisites before this step:

Optimized Netlist Verilog Code ("nlopt.v" from last step)

DEF File (e.g. "simple.def" from last step)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

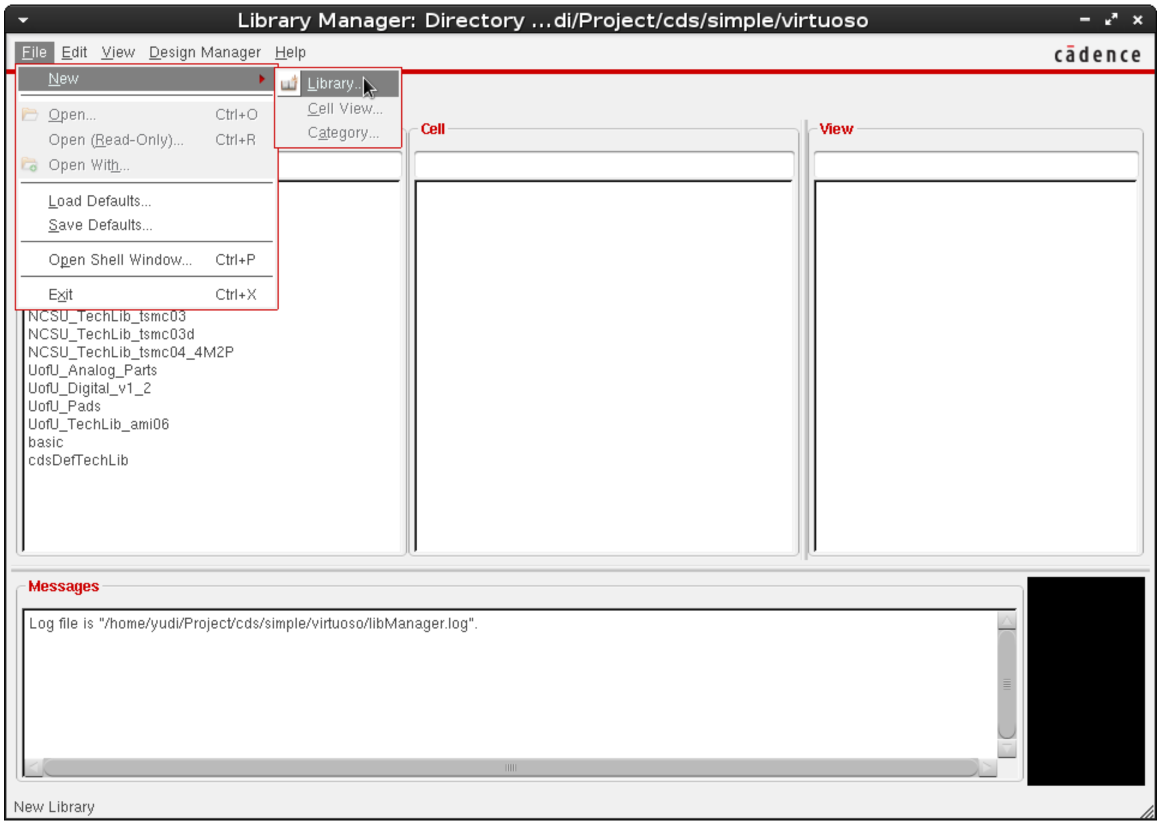
GDSII Stream File (e.g. "simple_final.gds")

4) Steps:

a) Launch Cadence Virtuoso Design Environment

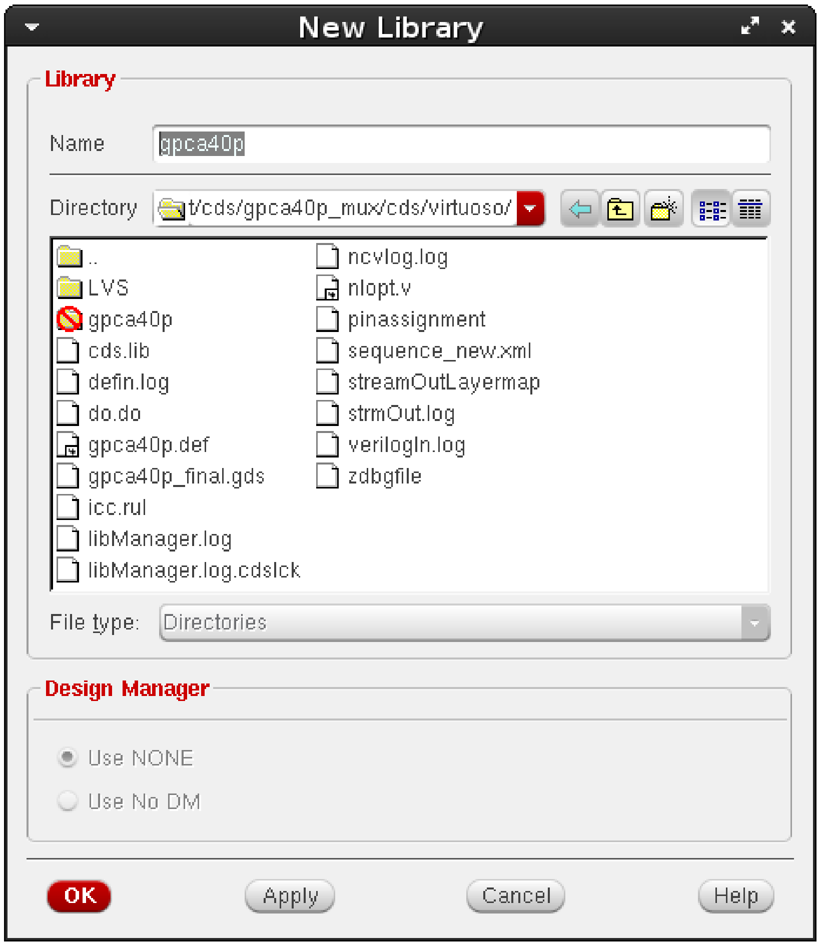
```
$ virtuoso
```

b) Create New Library (Figure 94 to Figure 97)



```
Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".
```

Figure 94: Create New Library with Virtuoso.


New Library

Library

Name: gpca40p

Directory: t/cds/gpca40p_mux/cds/virtuoso/

```
..                      ncvlog.log
LVS                     nlopt.v
gpca40p                 pinassignment
cds.lib                 sequence_new.xml
defin.log               streamOutLayermap
do.do                   strmOut.log
gpca40p.def             verilogIn.log
gpca40p_final.gds       zdbgfile
icc.rul
libManager.log
libManager.log.cdslck
```

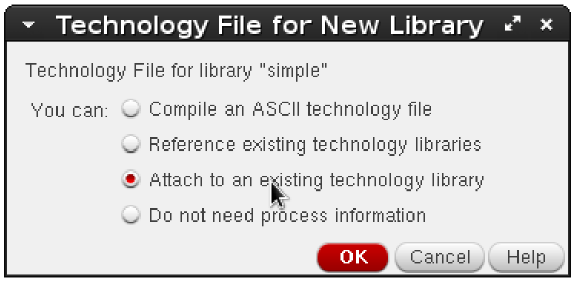
File type: Directories

Design Manager

[✓ Use NONE]

Use No DM

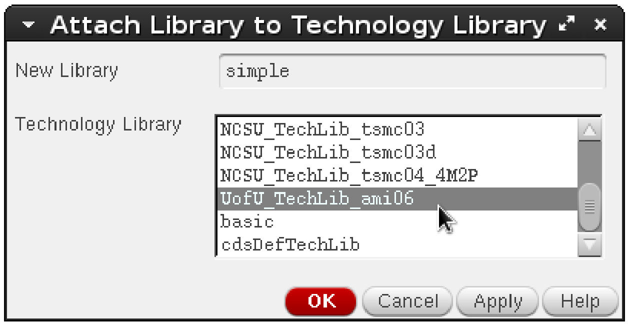
Figure 95: "New Library" window of Virtuoso.


Technology File for New Library

Technology File for library "simple"

You can: Compile an ASCII technology file [ ] Reference existing technology libraries [ ] [✓ Attach to an existing technology library] Do not need process information [ ]

Figure 96: "Technology File for New Library" window of Virtuoso.


Attach Library to Technology Library

New Library: simple

Technology Library:

```
NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_TechLib_ami06
basic
cdsDefTechLib
```

Figure 97: "Technology File for New Library" window of Virtuoso.

c) Import Optimized Netlist Verilog Code ("nlopt.v" from last step) and DEF File (e.g. "simple.def" from last step). As shown in Figure 98 to Figure 102.

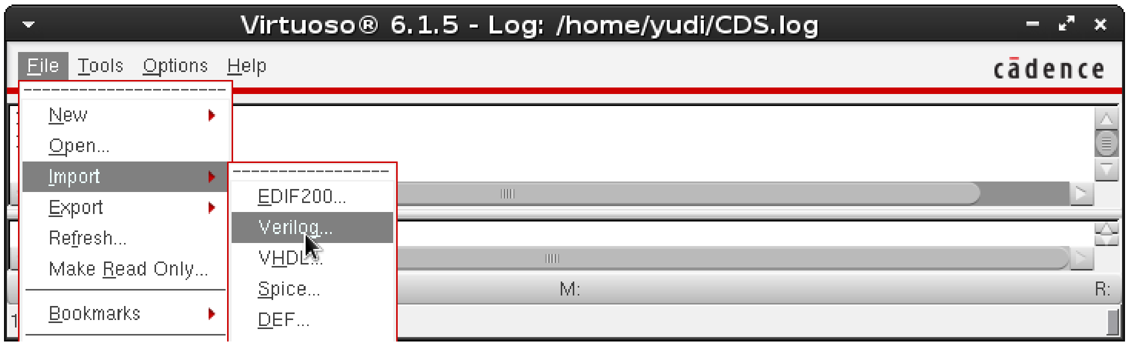
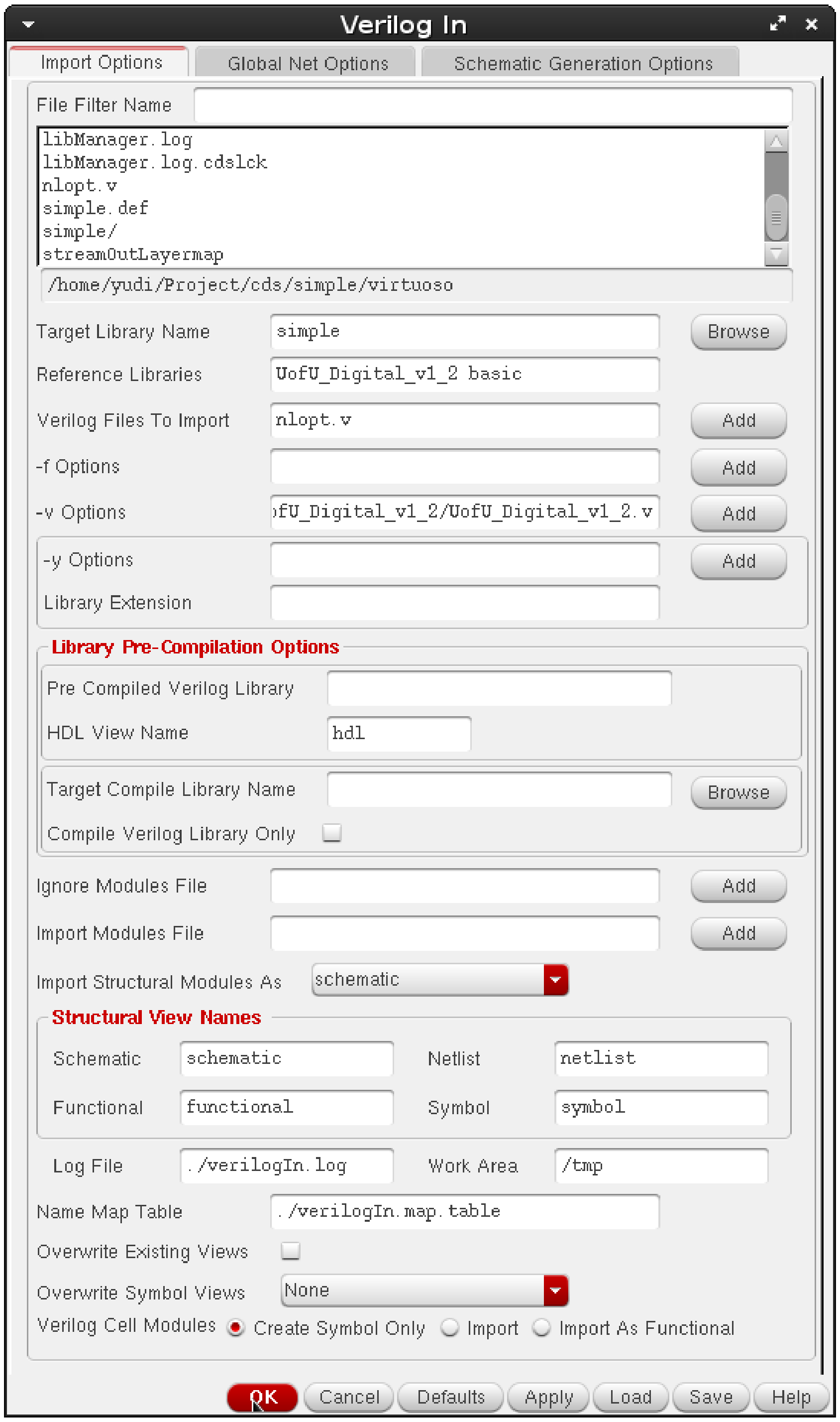


Figure 98: Import netlist with Virtuoso.



File Filter Name

```
libManager.log
libManager.log.cdslck
nlopt.v
simple.def
simple/
streamOutLayermap
/home/yudi/Project/cds/simple/virtuoso
```

Target Library Name: simple
Reference Libraries: UofU_Digital_v1_2 basic
Verilog Files To Import: nlopt.v
-f Options:
-v Options: ofU_Digital_v1_2/UofU_Digital_v1_2.v
-y Options:
Library Extension:

Library Pre-Compilation Options

Pre Compiled Verilog Library:
HDL View Name: hdl
Target Compile Library Name:
Compile Verilog Library Only ☐

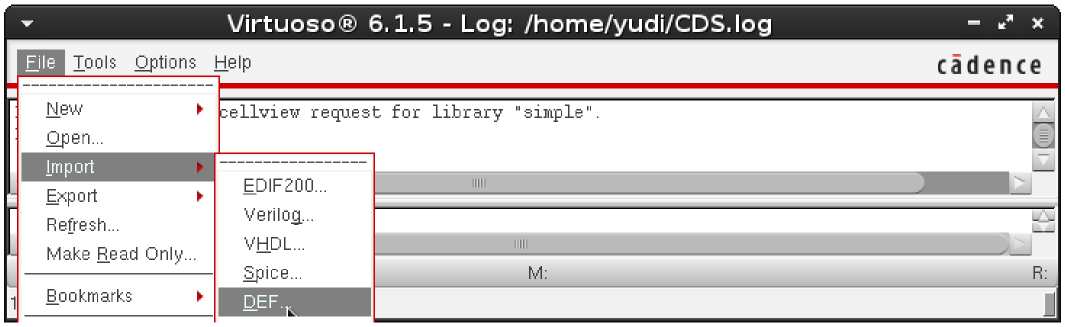
Ignore Modules File:
Import Modules File:
Import Structural Modules As: schematic

Structural View Names

Schematic: schematic
Netlist: netlist
Functional: functional
Symbol: symbol

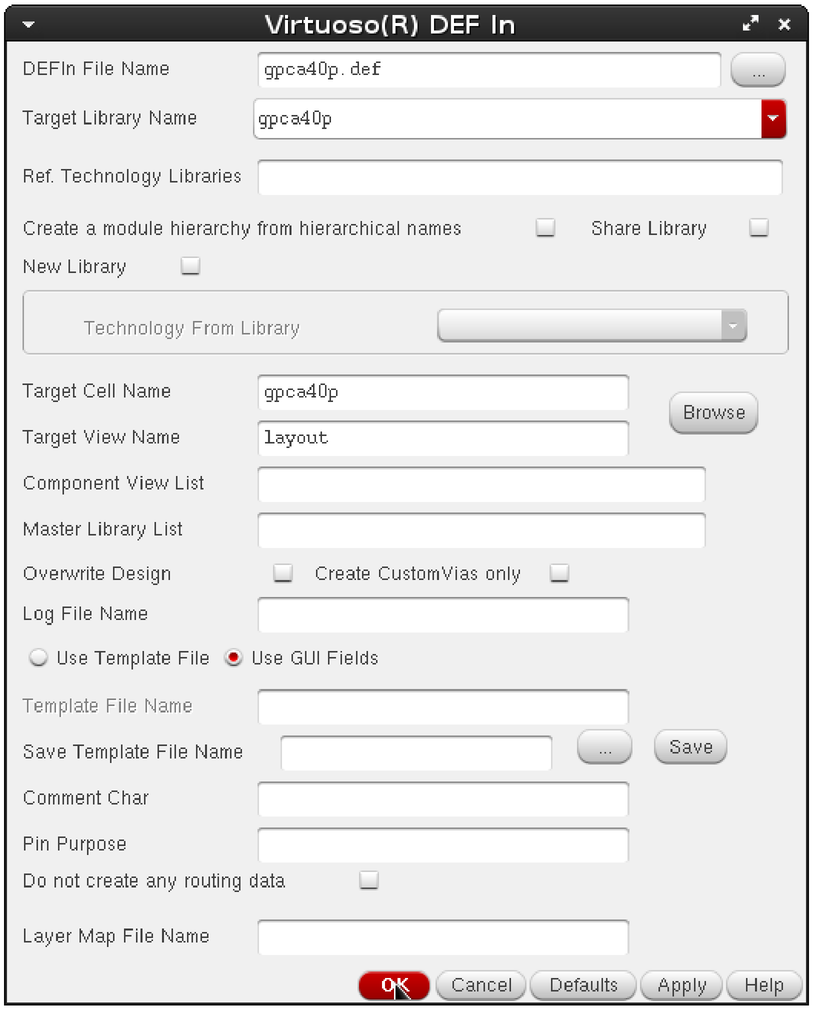
Log File: ./verilogIn.log
Work Area: /tmp
Name Map Table: ./verilogIn.map.table
Overwrite Existing Views ☐
Overwrite Symbol Views: None
Verilog Cell Modules: [✓ Create Symbol Only] Import Import As Functional

Figure 99: "Verilog In" window of Virtuoso.



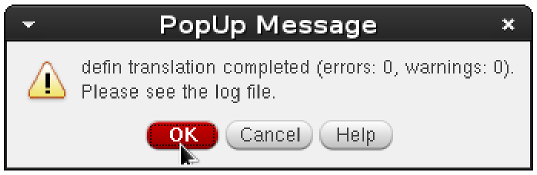
```
cellview request for library "simple".
```

Figure 100: Import DEF with Virtuoso.



DEFIn File Name	gpca40p.def
Target Library Name	gpca40p
Ref. Technology Libraries	
Create a module hierarchy from hierarchical names	☐
Share Library	☐
New Library	☐
Technology From Library	
Target Cell Name	gpca40p
Target View Name	layout
Component View List	
Master Library List	
Overwrite Design	☐
Create CustomVias only	☐
Log File Name	
Use Template File [✓ Use GUI Fields]	
Template File Name	
Save Template File Name	
Comment Char	
Pin Purpose	
Do not create any routing data	☐
Layer Map File Name	

Figure 101: "DEF In" window of Virtuoso.



PopUp Message

defin translation completed (errors: 0, warnings: 0).
Please see the log file.

Figure 102: "DEF In" successful translation prompt.

Once both layouts (from DEF) and schematic (from netlist in this section) are generated, DRC, Extract and LVS should be performed, as shown in Figure 103 to 111.

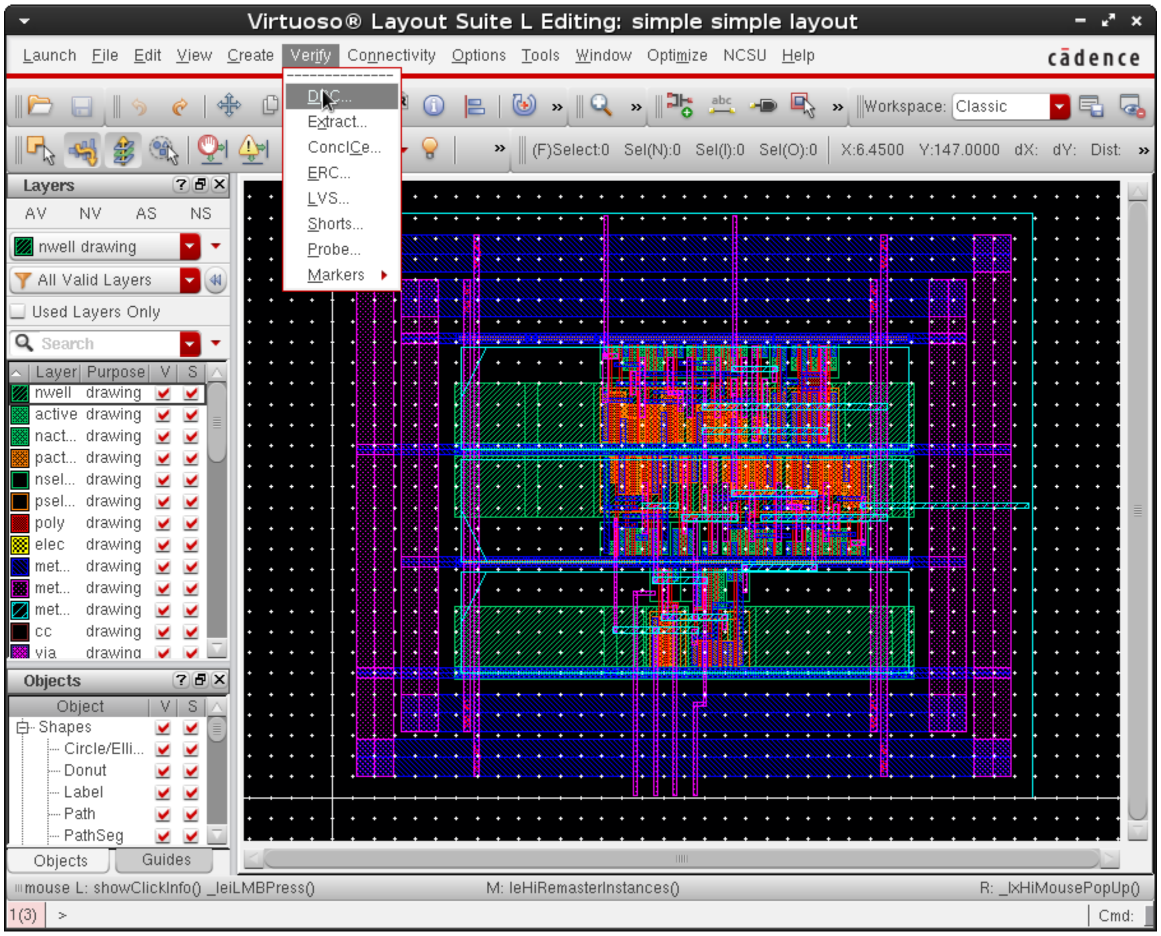
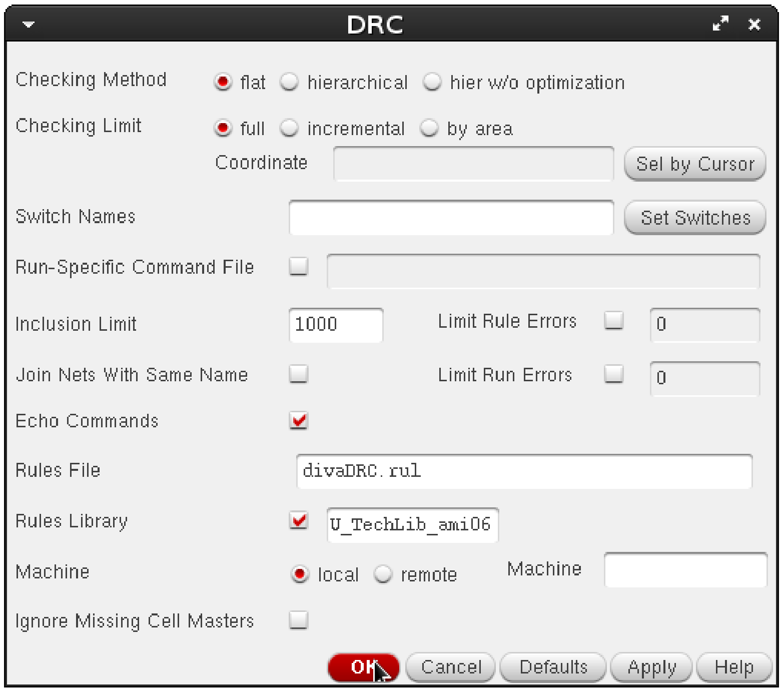
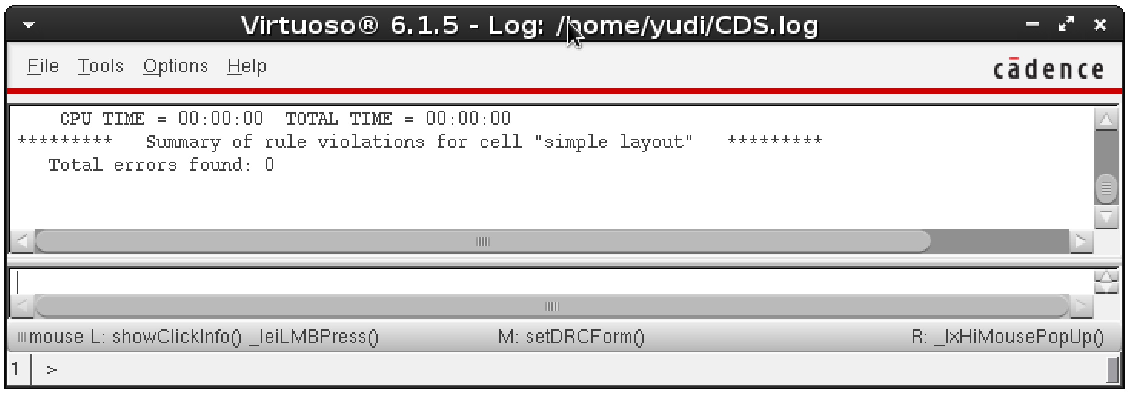


Figure 103: Start DRC from Virtuoso Layout Suite.



Checking Method [✓ flat] hierarchical hier w/o optimization
Checking Limit [✓ full] incremental by area
Coordinate
Switch Names
Run-Specific Command File ☐
Inclusion Limit 1000
Limit Rule Errors ☐ 0
Join Nets With Same Name ☐
Limit Run Errors ☐ 0
Echo Commands ✓
Rules File divaDRC.rul
Rules Library ✓ U_TechLib_ami06
Machine [✓ local] remote
Machine
Ignore Missing Cell Masters ☐

Figure 104: "DRC" window of Virtuoso.



```
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violations for cell "simple layout"   *********
   Total errors found: 0
```

mouse L: showClickInfo() _leiLMBPress()   M: setDRCForm()   R: _lxHiMousePopUp()

1 >

Figure 105: Report of successful DRC.

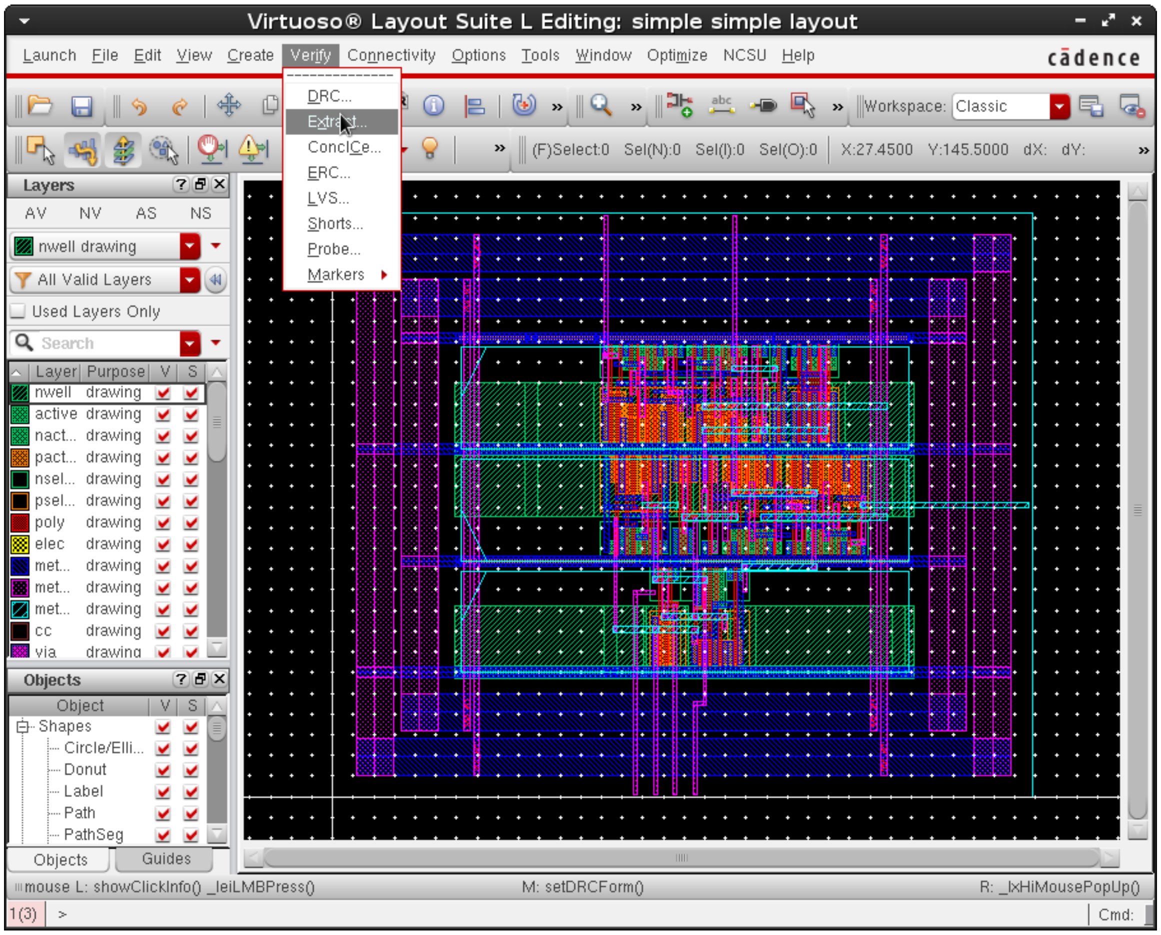
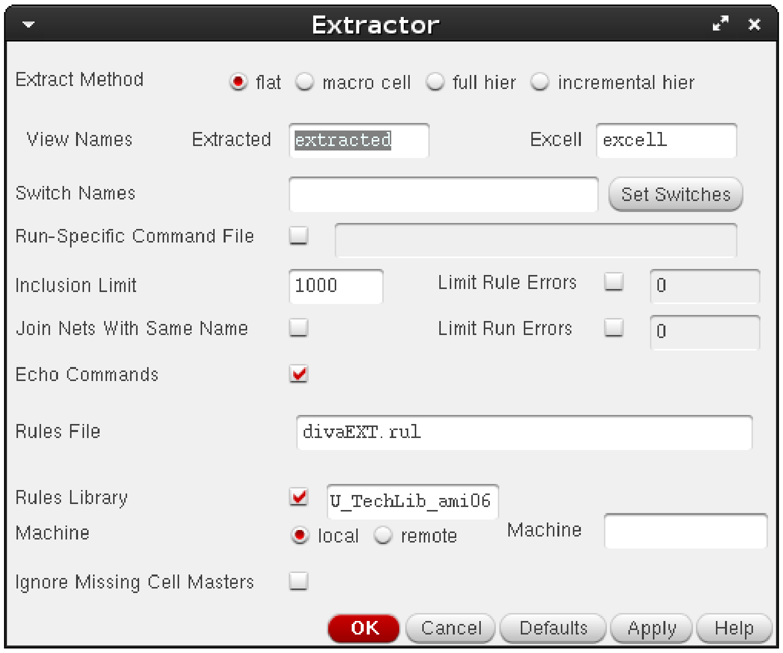


Figure 106: Start Extract from Virtuoso Layout Suite.



Extract Method [✓ flat] macro cell full hier incremental hier

View Names Extracted: extracted Excell: excell

Switch Names:

Run-Specific Command File ☐

Inclusion Limit: 1000

Limit Rule Errors ☐ 0

Join Nets With Same Name ☐

Limit Run Errors ☐ 0

Echo Commands ✓

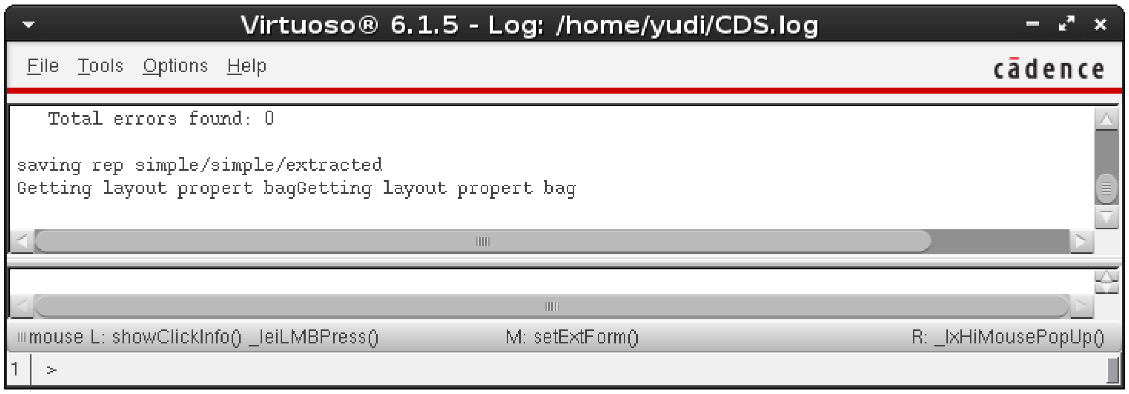
Rules File: divaEXT.rul

Rules Library ✓ U_TechLib_ami06

Machine [✓ local] remote Machine:

Ignore Missing Cell Masters ☐

Figure 107: "Extractor" window of Virtuoso.



```
    Total errors found: 0

saving rep simple/simple/extracted
Getting layout propert bagGetting layout propert bag
```

```
mouse L: showClickInfo() _leiLMBPress()          M: setExtForm()          R: _lxHiMousePopUp()
1  >
```

Figure 108: Report of successful Extract.

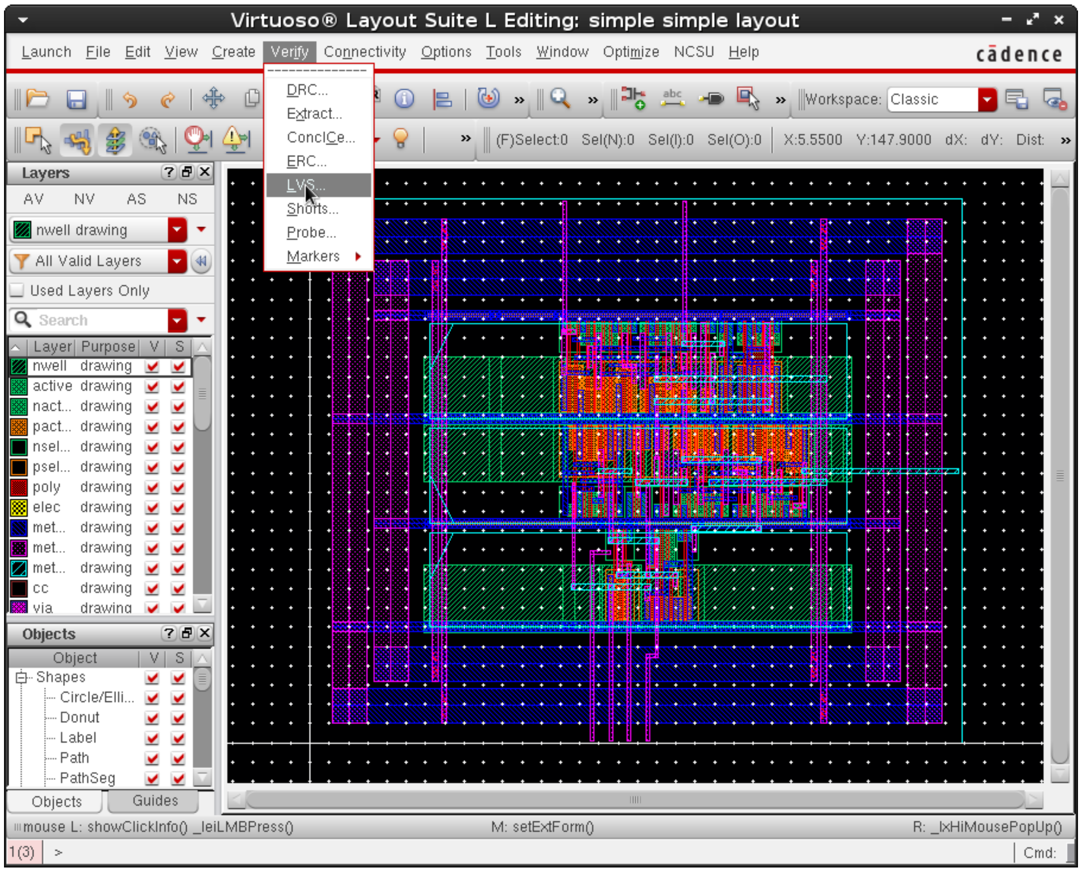
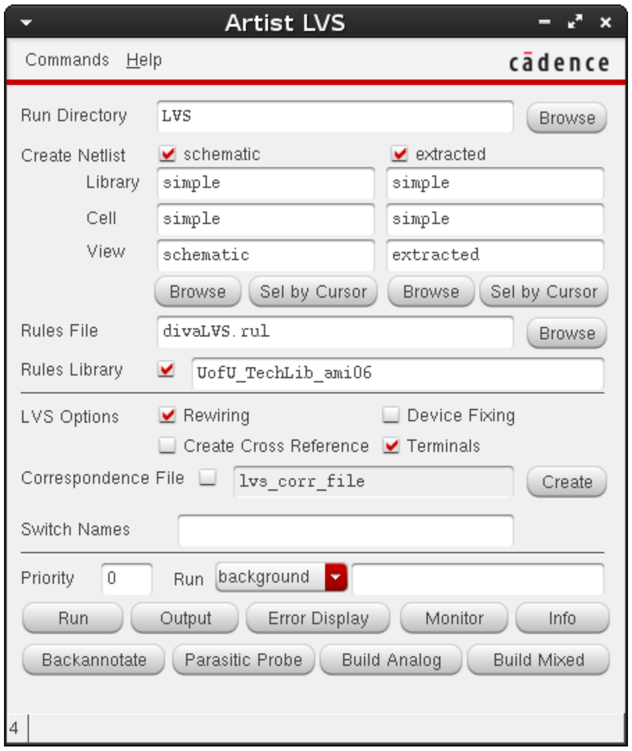


Figure 109: Start LVS from Virtuoso Layout Suite.



Run Directory: LVS

Create Netlist: ✓ schematic | ✓ extracted

	schematic	extracted
Library	simple	simple
Cell	simple	simple
View	schematic	extracted

Rules File: divaLVS.rul

Rules Library: ✓ UofU_TechLib_ami06

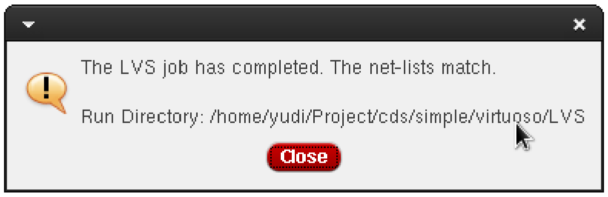
LVS Options: ✓ Rewiring, ☐ Device Fixing, ☐ Create Cross Reference, ✓ Terminals

Correspondence File: ☐ lvs_corr_file

Switch Names:

Priority: 0 Run: background

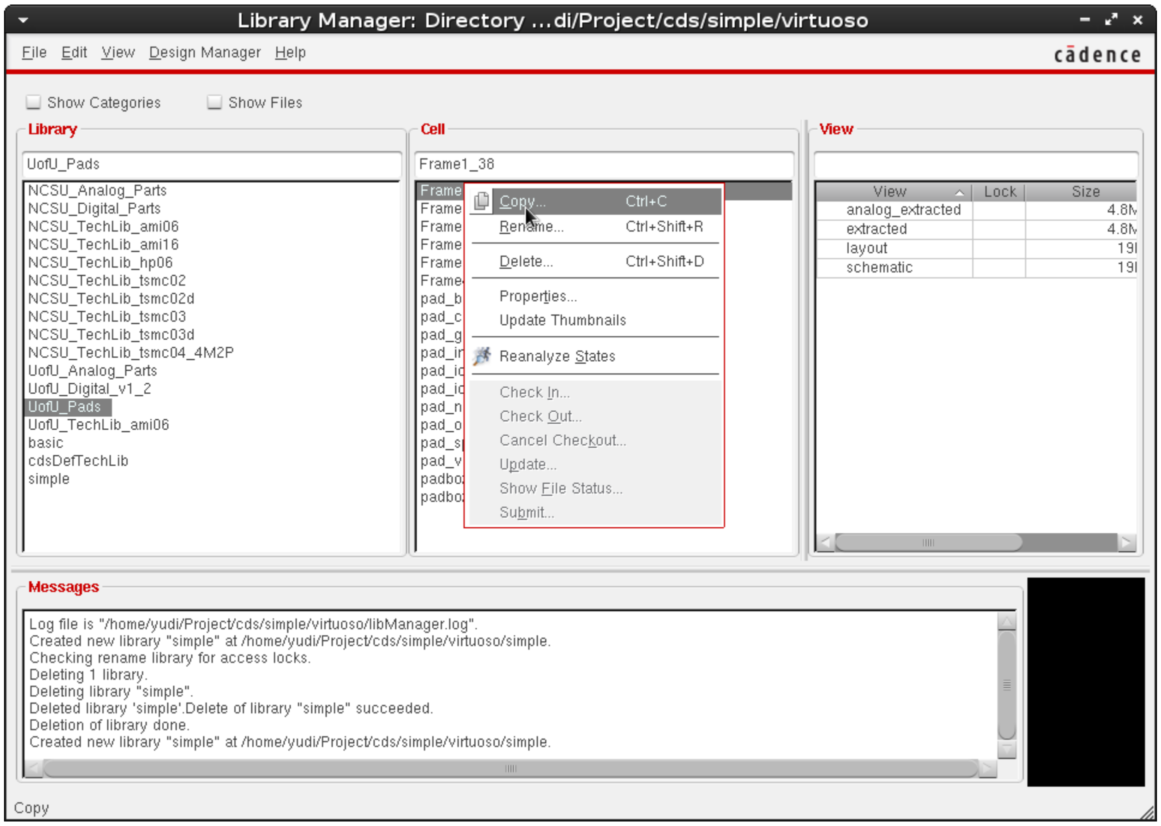
Figure 110: "Artist LVS" window of Virtuoso.


The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

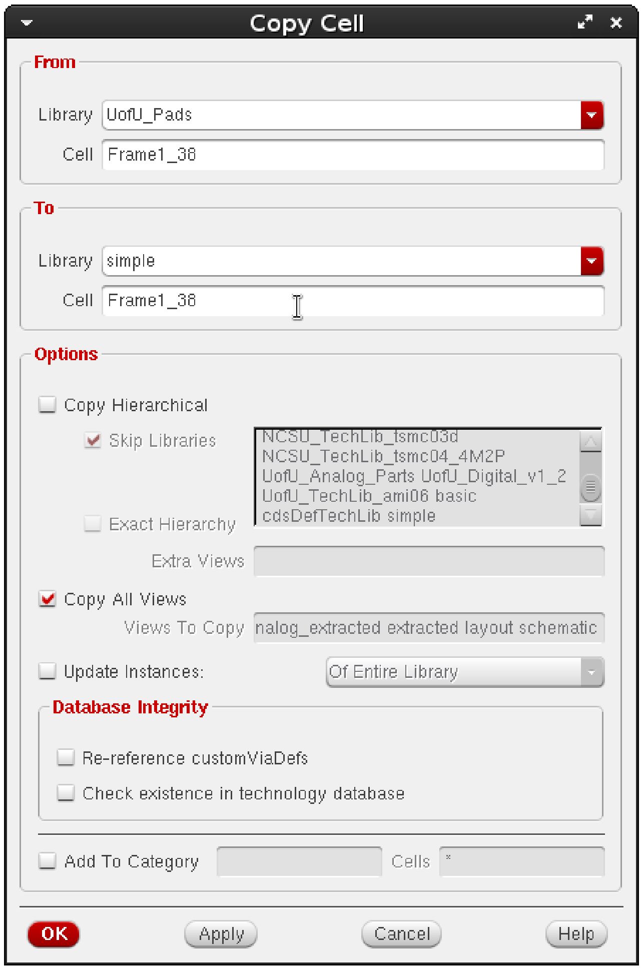
Figure 111: "Artist LVS" successful LVS prompt.

d) Customize Pad Frame (Figure 112 to 132)



```
Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
Checking rename library for access locks.
Deleting 1 library.
Deleting library "simple".
Deleted library 'simple'.Delete of library "simple" succeeded.
Deletion of library done.
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
```

Figure 112: Copy pad frame in Virtuoso.



Copy Cell

From
Library UofU_Pads
Cell Frame1_38

To
Library simple
Cell Frame1_38

Options
☐ Copy Hierarchical
✓ Skip Libraries
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts UofU_Digital_v1_2
UofU_TechLib_ami06 basic
cdsDefTechLib simple
☐ Exact Hierarchy
Extra Views
✓ Copy All Views
Views To Copy nalog_extracted extracted layout schematic
☐ Update Instances: Of Entire Library

Database Integrity
☐ Re-reference customViaDefs
☐ Check existence in technology database

☐ Add To Category Cells *

Figure 113: "Copy Cell" window of Virtuoso.

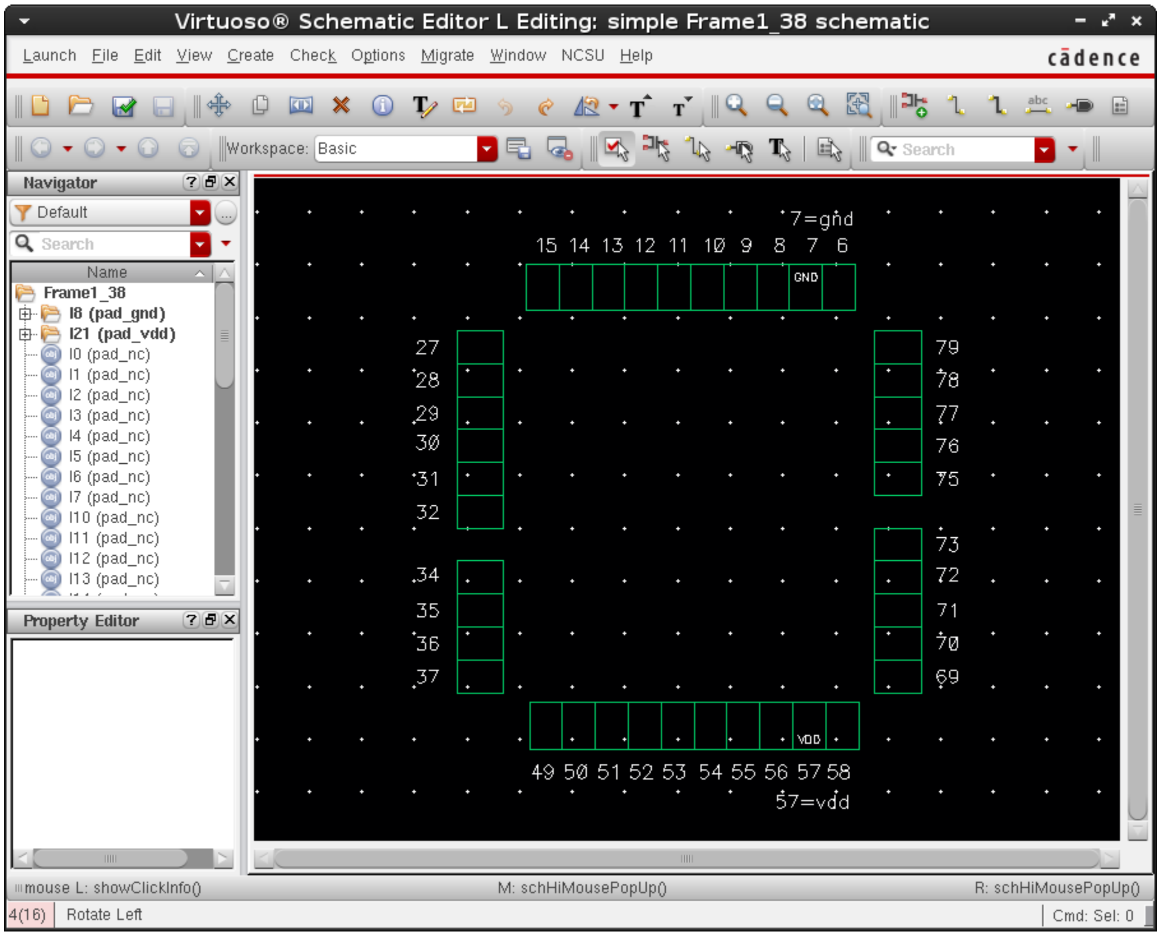
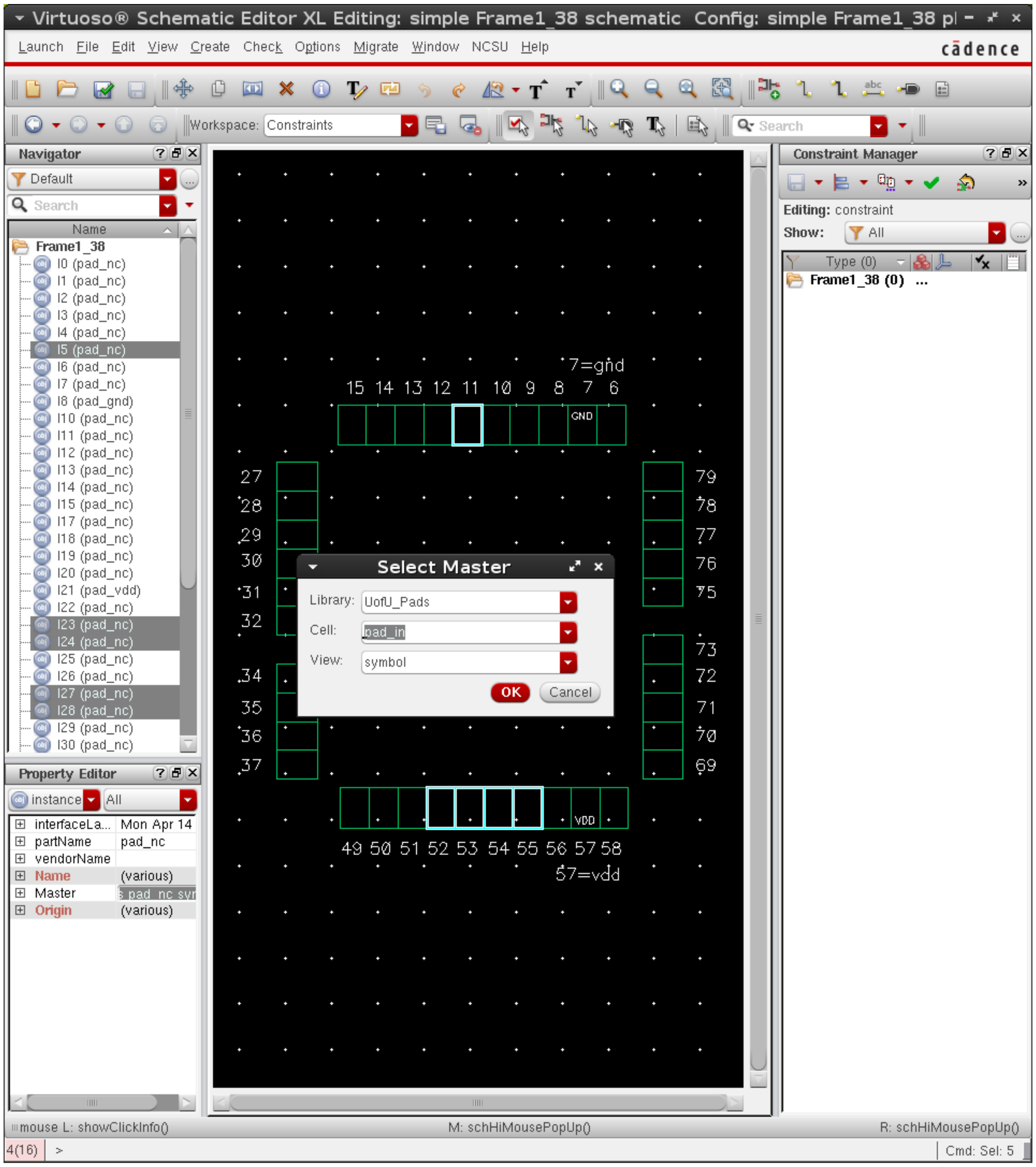


Figure 114: Initial schematic view of the pad frame.

Modify the used pads in the pad frame (both schematic and layout), from "pad_nc" to "pad_in" or "pad_out".


Virtuoso® Schematic Editor XL Editing: simple Frame1_38 schematic Config: simple Frame1_38 pl

Launch File Edit View Create Check Options Migrate Window NCSU Help
cādence
Workspace: Constraints
Search
Navigator
Default
Search
Name

Frame1_38
I0 (pad_nc)
I1 (pad_nc)
I2 (pad_nc)
I3 (pad_nc)
I4 (pad_nc)
I5 (pad_nc)
I6 (pad_nc)
I7 (pad_nc)
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)
I18 (pad_nc)
I19 (pad_nc)
I20 (pad_nc)
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_nc)
I24 (pad_nc)
I25 (pad_nc)
I26 (pad_nc)
I27 (pad_nc)
I28 (pad_nc)
I29 (pad_nc)
I30 (pad_nc)

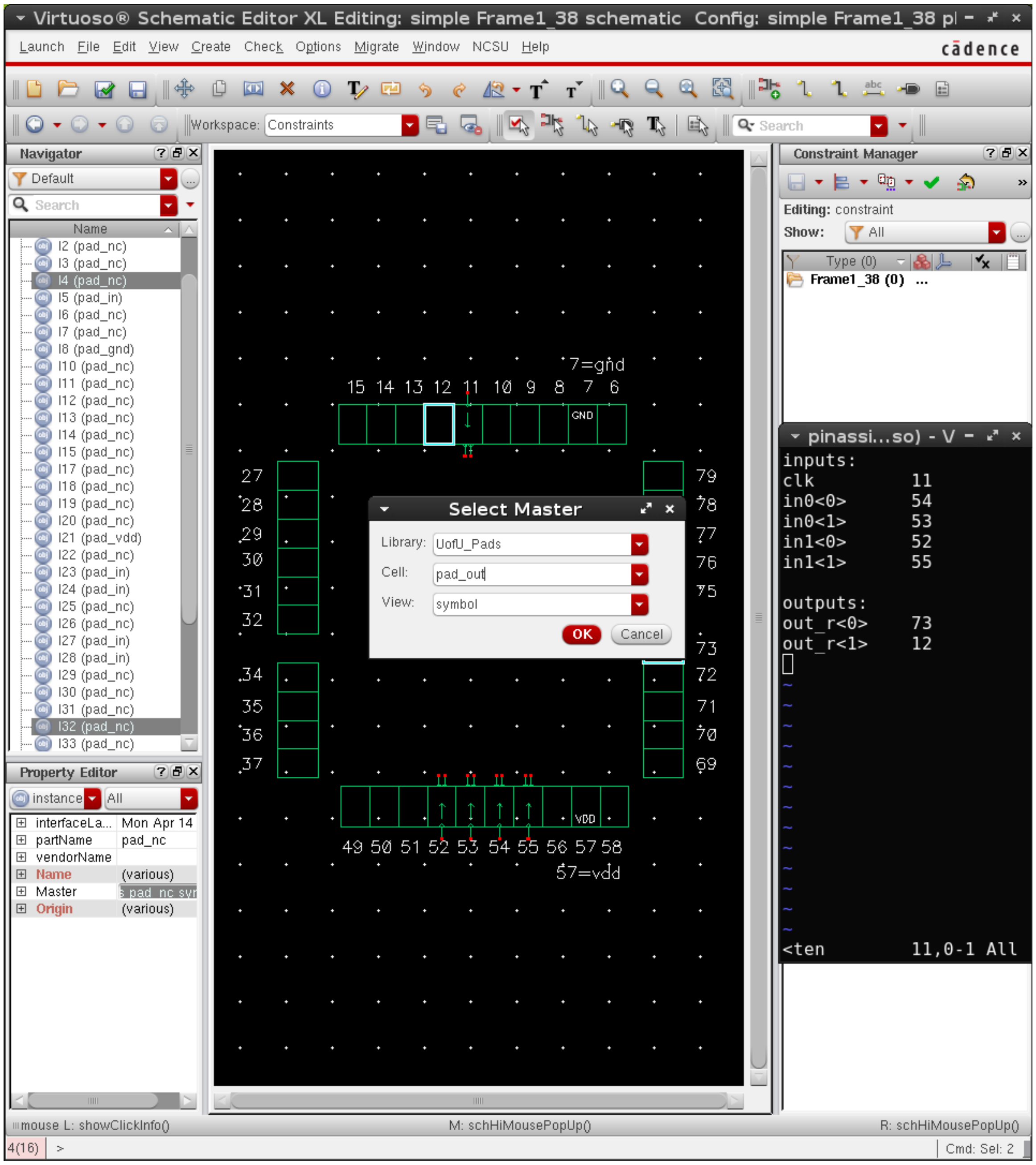
interfaceLa...	Mon Apr 14
partName	pad_nc
vendorName	
Name	(various)
Master	s pad_nc syr
Origin	(various)

Select Master
Library: UofU_Pads
Cell: pad_in
View: symbol

Editing: constraint
Show: All
Type (0)
Frame1_38 (0) ...

mouse L: showClickInfo() M: schHiMousePopUp() R: schHiMousePopUp()
4(16) > Cmd: Sel: 5

Figure 115: Add input pads in schematic.



interfaceLa...	Mon Apr 14
partName	pad_nc
vendorName	
Name	(various)
Master	s pad nc sy
Origin	(various)

Select Master

Library: UofU_Pads

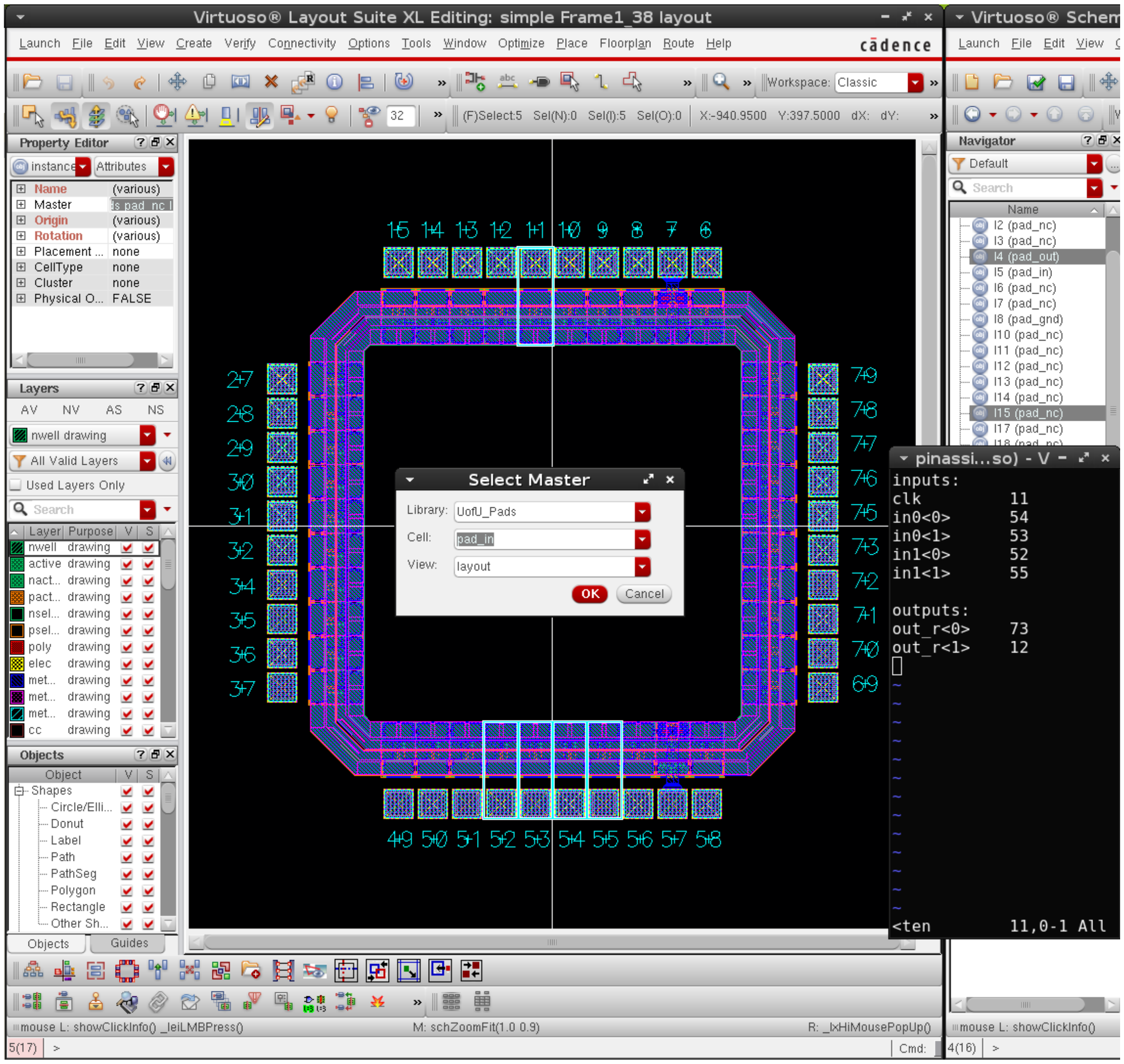
Cell: pad_out

View: symbol

OK Cancel

```
inputs:
clk         11
in0<0>      54
in0<1>      53
in1<0>      52
in1<1>      55

outputs:
out_r<0>    73
out_r<1>    12

<ten        11,0-1 All
```

Figure 116: Add output pads in schematic.

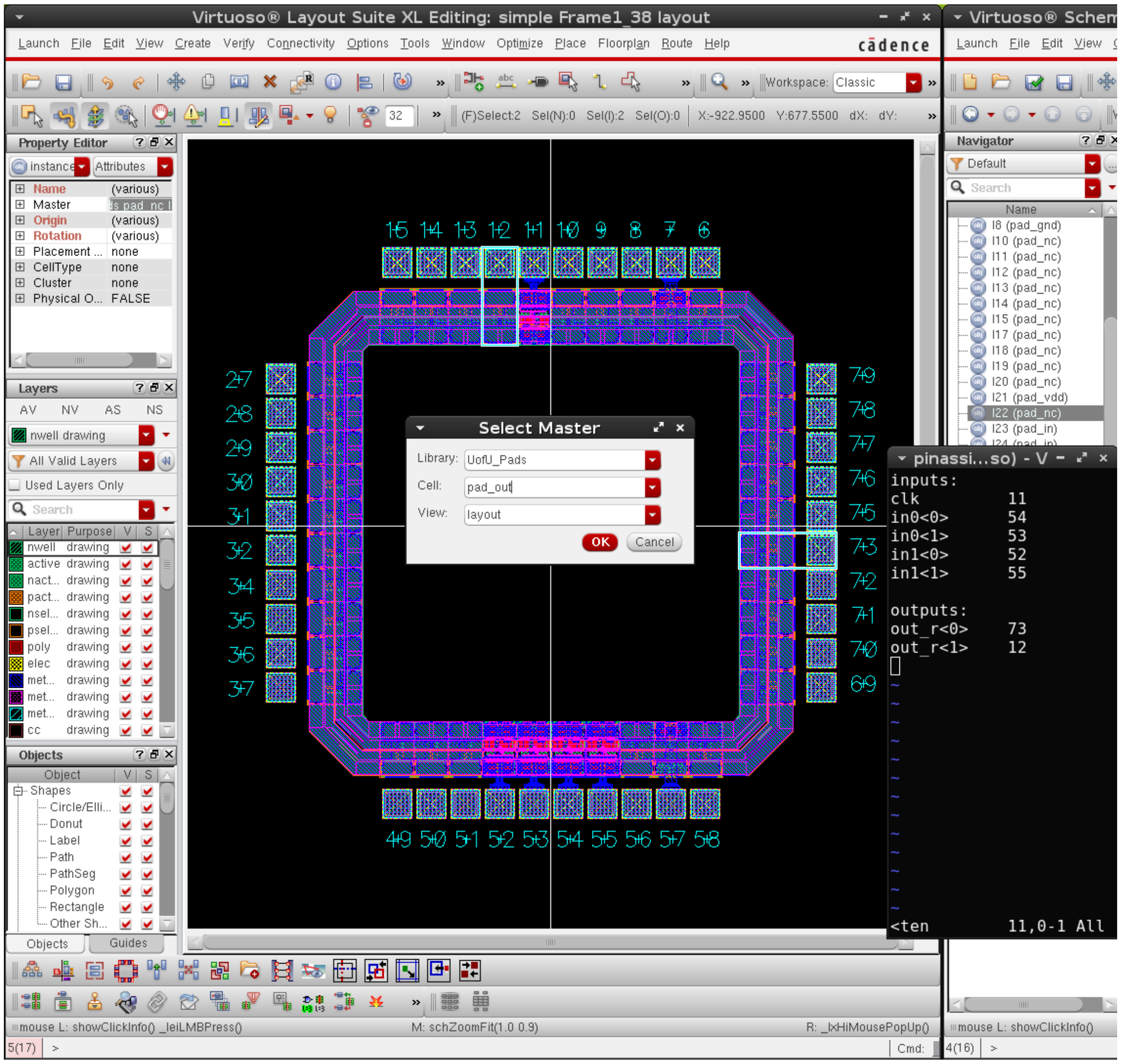


Library: UofU_Pads
Cell: pad_in
View: layout

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12
```

mouse L: showClickInfo() _leiLMBPress()   M: schZoomFit(1.0 0.9)   R: _lxHiMousePopUp()

Figure 117: Add input pads in layout.


Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
(F)Select:2 Sel(N):0 Sel(I):2 Sel(O):0 X:-922.9500 Y:677.5500 dX: dY:
Property Editor
Layers
Objects
Navigator
15 14 13 12 11 10 9 8 7 6
27 28 29 30 31 32 34 35 36 37
79 78 77 76 75 73 72 71 70 69
49 50 51 52 53 54 55 56 57 58
mouse L: showClickInfo() _leiLMBPress()
M: schZoomFit(1.0 0.9)
R: _lxHiMousePopUp()

Property Editor

Name	(various)
Master	s pad_nc l
Origin	(various)
Rotation	(various)
Placement ...	none
CellType	none
Cluster	none
Physical O...	FALSE

Layers

Layer	Purpose	V	S
nwell	drawing	✓	✓
active	drawing	✓	✓
nact...	drawing	✓	✓
pact...	drawing	✓	✓
nsel...	drawing	✓	✓
psel...	drawing	✓	✓
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓

Select Master

Library: UofU_Pads
Cell: pad_out
View: layout

OK Cancel

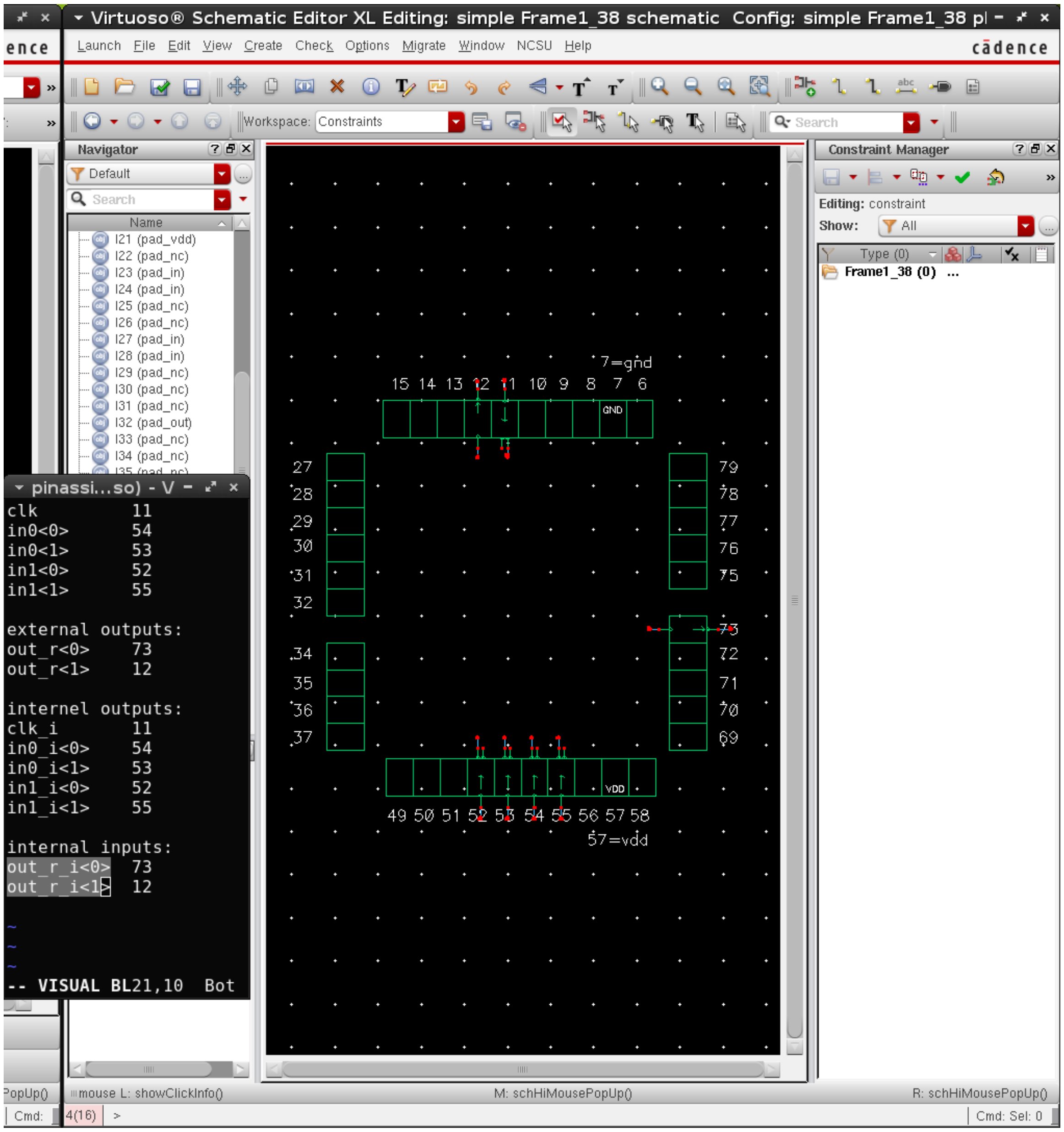
Navigator
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)
I18 (pad_nc)
I19 (pad_nc)
I20 (pad_nc)
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_in)

pinassi...so) - V

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12

<ten                11,0-1 All
```

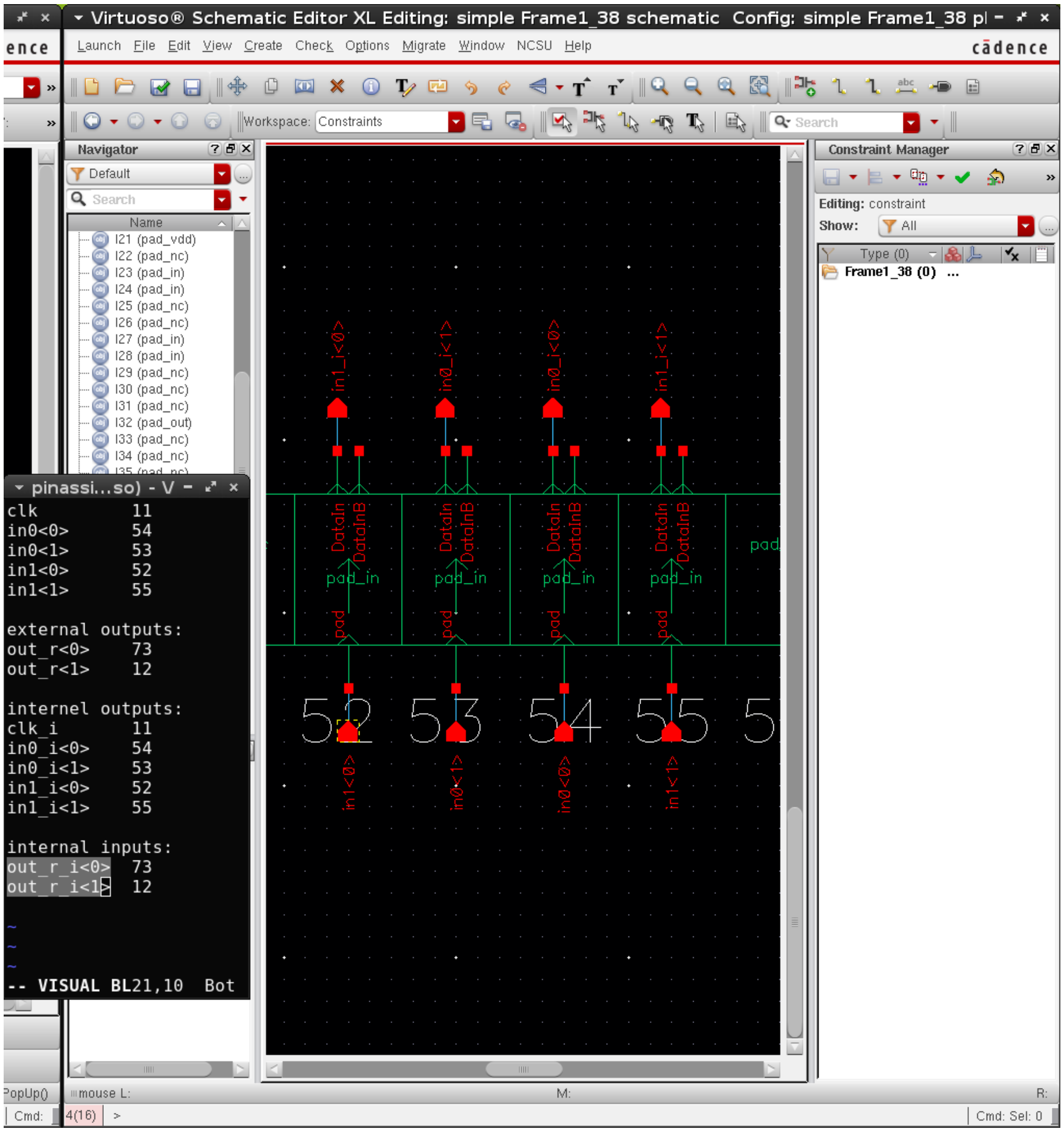
Figure 118: Add output pads in layout.

Add pins to pad frame (both schematic and layout). The pad frame works like a wraparound. The external pins of the pad frame (e.g. "Frame1_38") have the same name as the chip layout (e.g. "gpca40p"), and the internal pins of the pad frame connecting the chip layout use original names affixed by "_i" (Hence e.g. "clk" becomes "clk_i.) This procedure is shown in Figure 119 to Figure 132.



```
clk         11
in0<0>      54
in0<1>      53
in1<0>      52
in1<1>      55

external outputs:
out_r<0>    73
out_r<1>    12

internel outputs:
clk_i       11
in0_i<0>    54
in0_i<1>    53
in1_i<0>    52
in1_i<1>    55

internal inputs:
out_r_i<0>  73
out_r_i<1>  12

~
~
~
-- VISUAL BL21,10   Bot
```

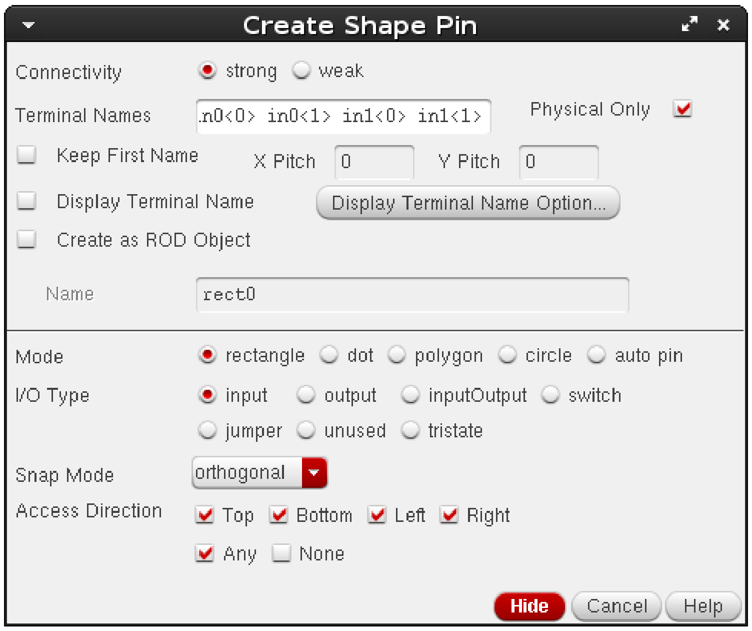
Figure 119: Add pins to the pad frame in schematic.



```
pinassi...so) - V
clk           11
in0<0>        54
in0<1>        53
in1<0>        52
in1<1>        55

external outputs:
out_r<0>      73
out_r<1>      12

internel outputs:
clk_i         11
in0_i<0>      54
in0_i<1>      53
in1_i<0>      52
in1_i<1>      55

internal inputs:
out_r_i<0>    73
out_r_i<1>    12

~
~
~
-- VISUAL BL21,10   Bot
```

Figure 120: Add pins to the pad frame schematic (detailed view).

Same for the layout by tapping the ports with pins


Create Shape Pin

Connectivity [✓ strong] weak

Terminal Names .n0<0> in0<1> in1<0> in1<1>

Physical Only [✓]

Keep First Name [ ] X Pitch 0 Y Pitch 0

Display Terminal Name [ ]

Create as ROD Object [ ]

Name rect0

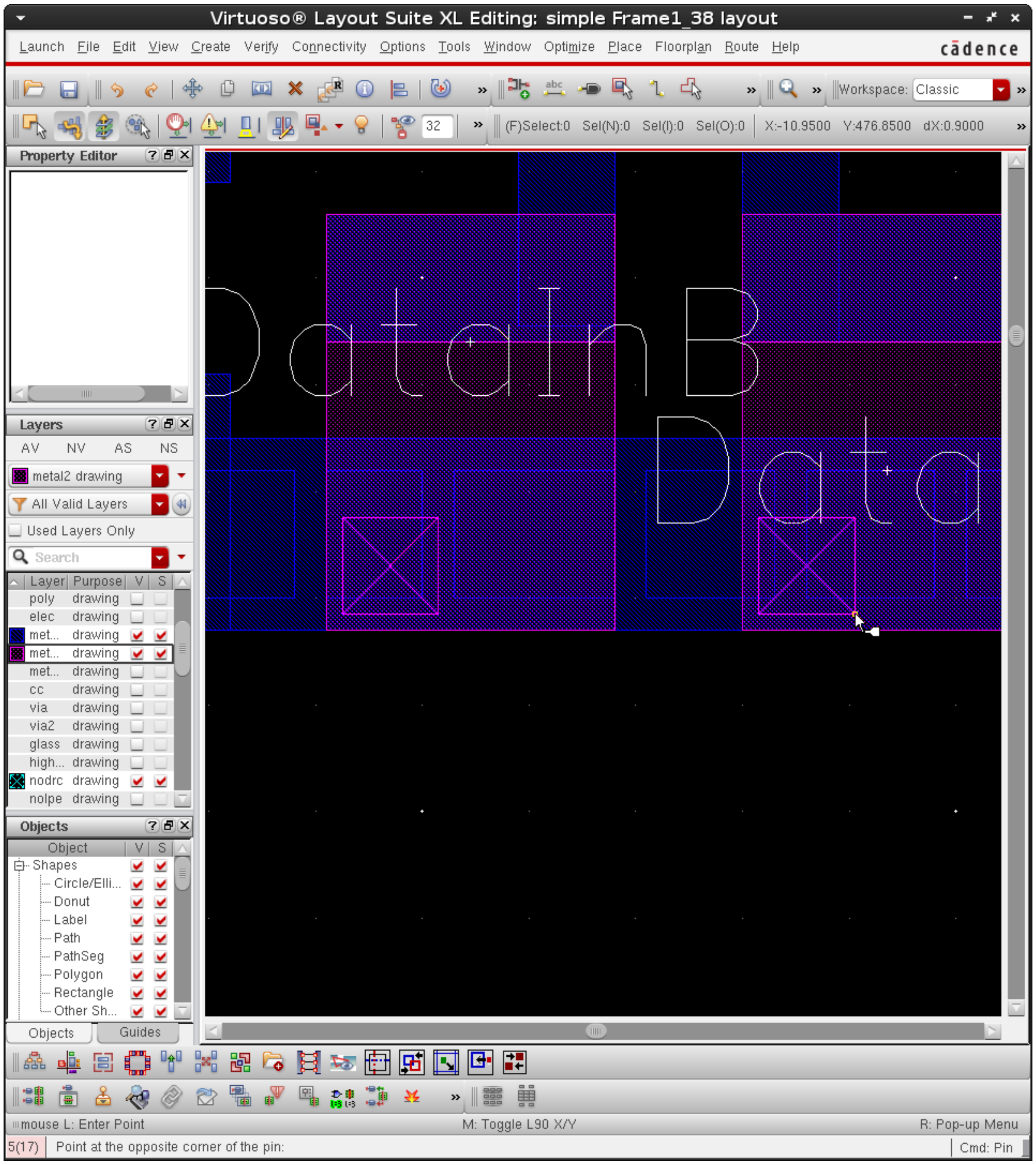
Mode [✓ rectangle] dot polygon circle auto pin

I/O Type [✓ input] output inputOutput switch jumper unused tristate

Snap Mode orthogonal

Access Direction [✓ Top] [✓ Bottom] [✓ Left] [✓ Right] [✓ Any] None

Figure 121: "Create Shape Pin" Window of Virtuoso.


Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:-10.9500 Y:476.8500 dX:0.9000
Property Editor
Layers
AV NV AS NS
metal2 drawing
All Valid Layers
Used Layers Only
Search
Objects
Shapes
Circle/Elli...
Donut
Label
Path
PathSeg
Polygon
Rectangle
Other Sh...
Objects Guides
DataInB
Data
mouse L: Enter Point
M: Toggle L90 X/Y
R: Pop-up Menu
5(17)
Point at the opposite corner of the pin:
Cmd: Pin

Layer	Purpose	V	S
poly	drawing		
elec	drawing		
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing		
cc	drawing		
via	drawing		
via2	drawing		
glass	drawing		
high...	drawing		
nodrc	drawing	✓	✓
nolpe	drawing		

Figure 122: Add pins to the pad frame layout (detailed view).

DRC, Extract, LVS to make sure no rule is violated.

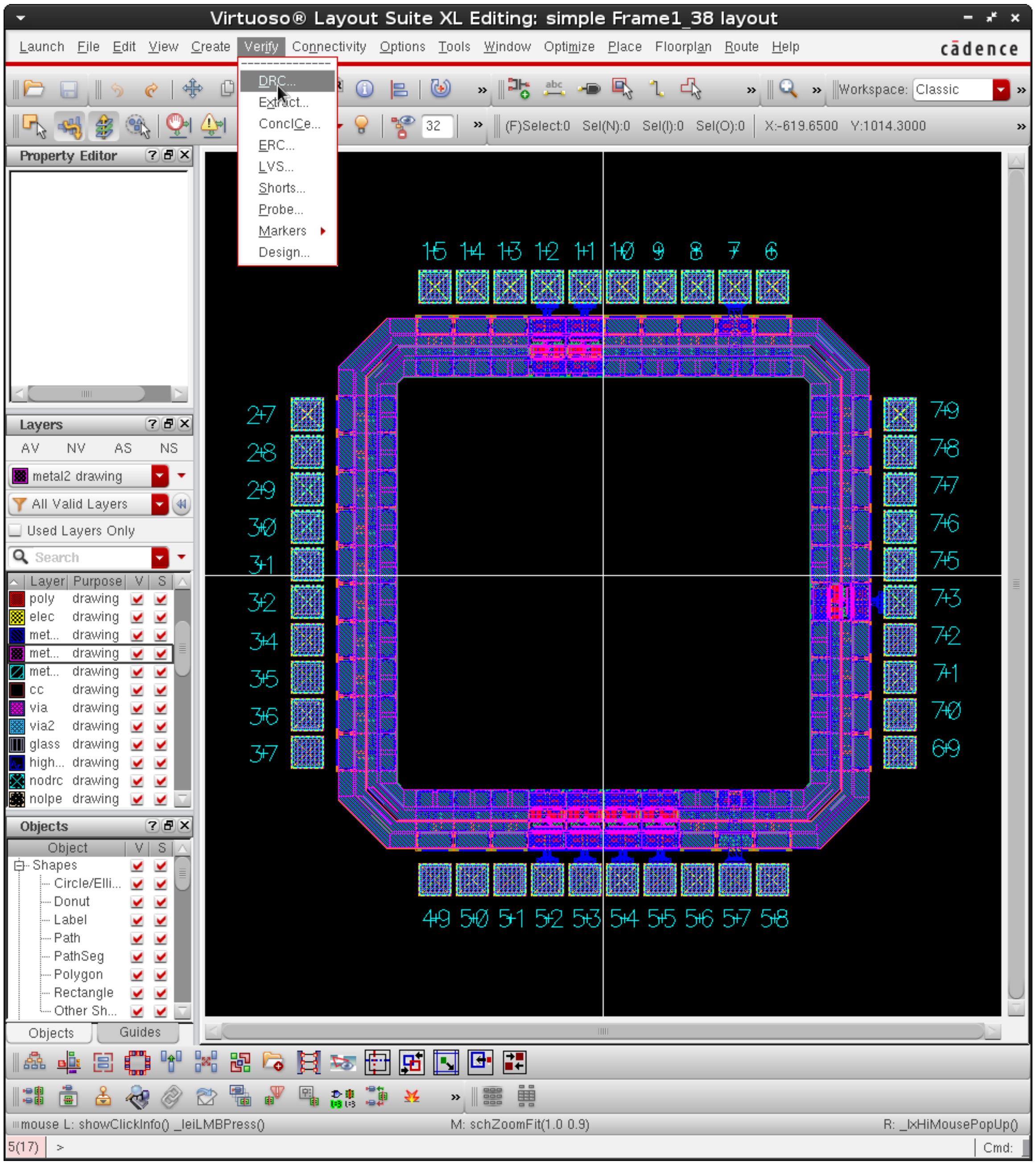
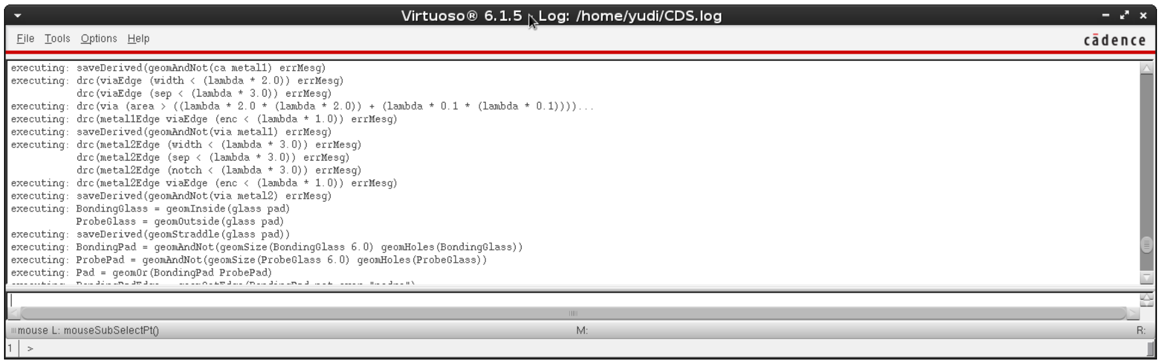


Figure 123: Start DRC from Virtuoso Layout Suite.



```
executing: saveDerived(geomAndNot(ca metal1) errMesg)
executing: drc(viaEdge (width < (lambda * 2.0)) errMesg)
           drc(viaEdge (sep < (lambda * 3.0)) errMesg)
executing: drc(via (area > ((lambda * 2.0 * (lambda * 2.0)) + (lambda * 0.1 * (lambda * 0.1))))...
executing: drc(metal1Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal1) errMesg)
executing: drc(metal2Edge (width < (lambda * 3.0)) errMesg)
           drc(metal2Edge (sep < (lambda * 3.0)) errMesg)
           drc(metal2Edge (notch < (lambda * 3.0)) errMesg)
executing: drc(metal2Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal2) errMesg)
executing: BondingGlass = geomInside(glass pad)
           ProbeGlass = geomOutside(glass pad)
executing: saveDerived(geomStraddle(glass pad))
executing: BondingPad = geomAndNot(geomSize(BondingGlass 6.0) geomHoles(BondingGlass))
executing: ProbePad = geomAndNot(geomSize(ProbeGlass 6.0) geomHoles(ProbeGlass))
executing: Pad = geomOr(BondingPad ProbePad)
```

Virtuoso® 6.1.5 Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: mouseSubSelectPt()
M:
R:
1 >

Figure 124: Report of successful DRC.

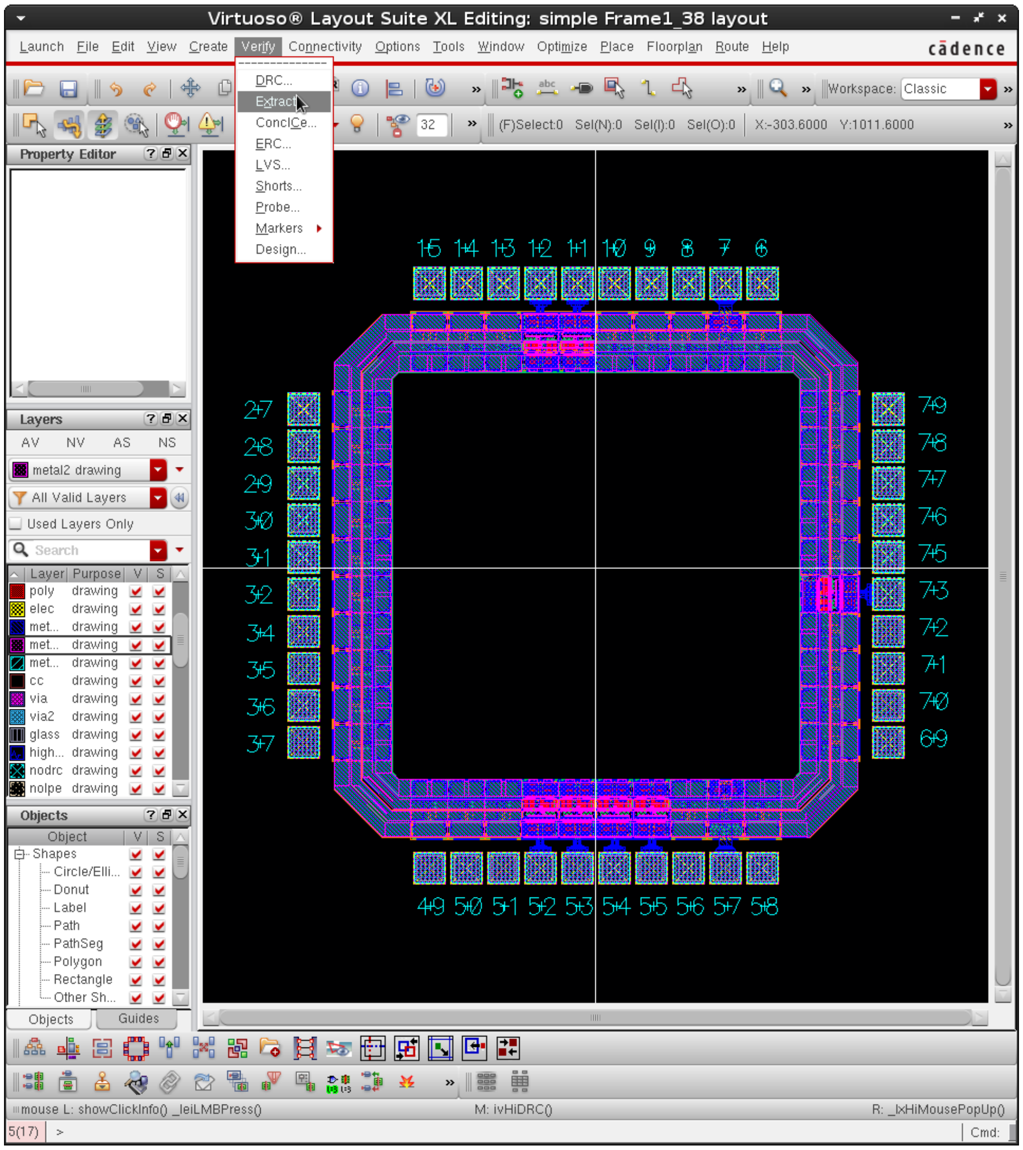
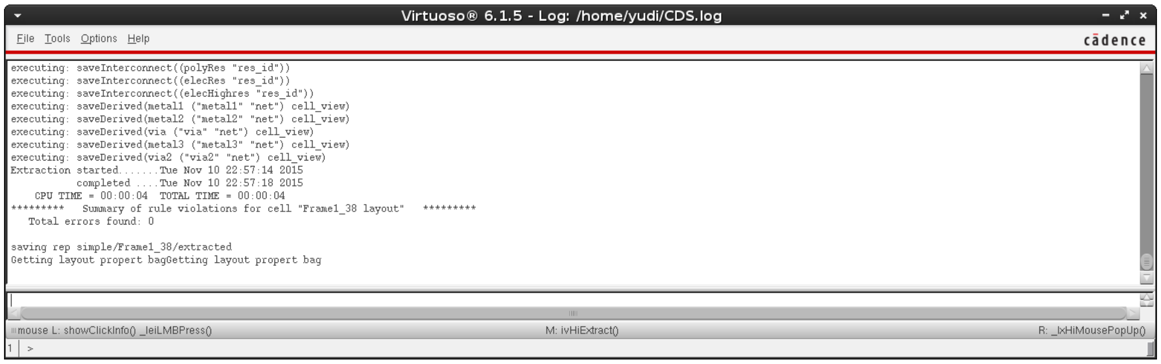


Figure 125: Start Extract from Virtuoso Layout Suite.



```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 22:57:14 2015
           completed ....Tue Nov 10 22:57:18 2015
    CPU TIME = 00:00:04  TOTAL TIME = 00:00:04
*********   Summary of rule violations for cell "Frame1_38 layout"   *********
   Total errors found: 0

saving rep simple/Frame1_38/extracted
Getting layout propert bagGetting layout propert bag
```

```
mouse L: showClickInfo() _leiLMBPress()          M: ivHiExtract()          R: _lxHiMousePopUp()
1 >
```

Figure 126: Report of successful Extract.

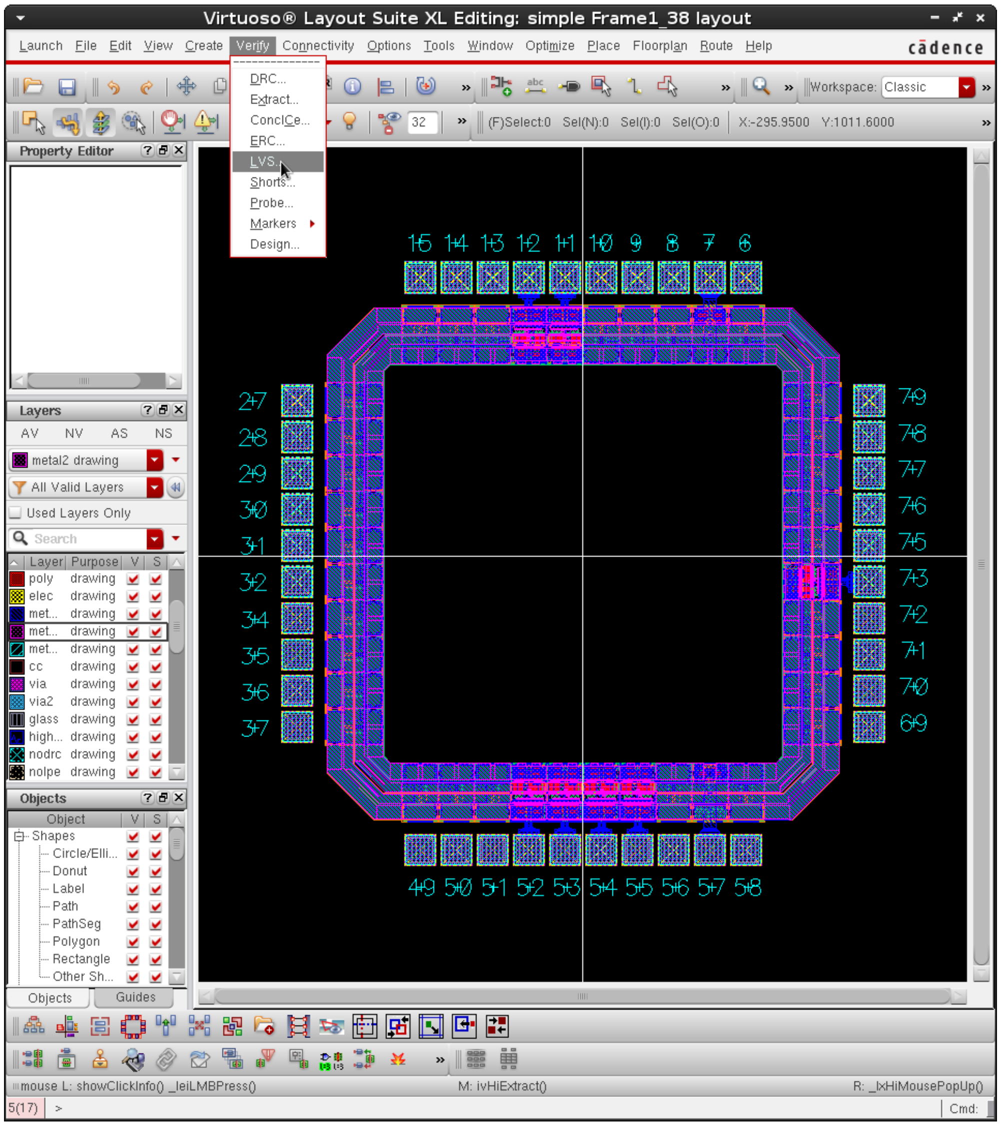
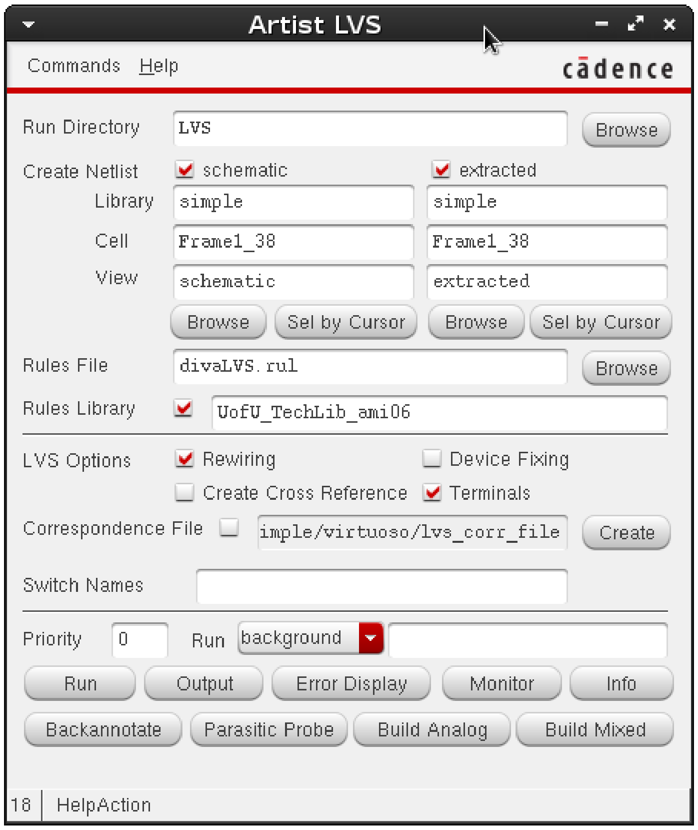


Figure 127: Start LVS from Virtuoso Layout Suite.


Artist LVS

Commands Help cādence

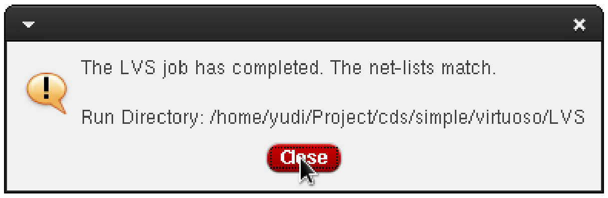
Run Directory	LVS		Browse
Create Netlist	[✓] schematic	[✓] extracted	
Library	simple	simple	
Cell	Frame1_38	Frame1_38	
View	schematic	extracted	
	Browse / Sel by Cursor	Browse / Sel by Cursor	
Rules File	divaLVS.rul		Browse
Rules Library	[✓] UofU_TechLib_ami06		
LVS Options	[✓] Rewiring	[ ] Device Fixing	
	[ ] Create Cross Reference	[✓] Terminals	
Correspondence File	[ ] imple/virtuoso/lvs_corr_file		Create
Switch Names			
Priority	0	Run background	

Run | Output | Error Display | Monitor | Info

Backannotate | Parasitic Probe | Build Analog | Build Mixed

18 HelpAction

Figure 128: "Artist LVS" window of Virtuoso.


The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

Close

Figure 129: "Artist LVS" successful LVS prompt.

Generate symbol for the pad frame.

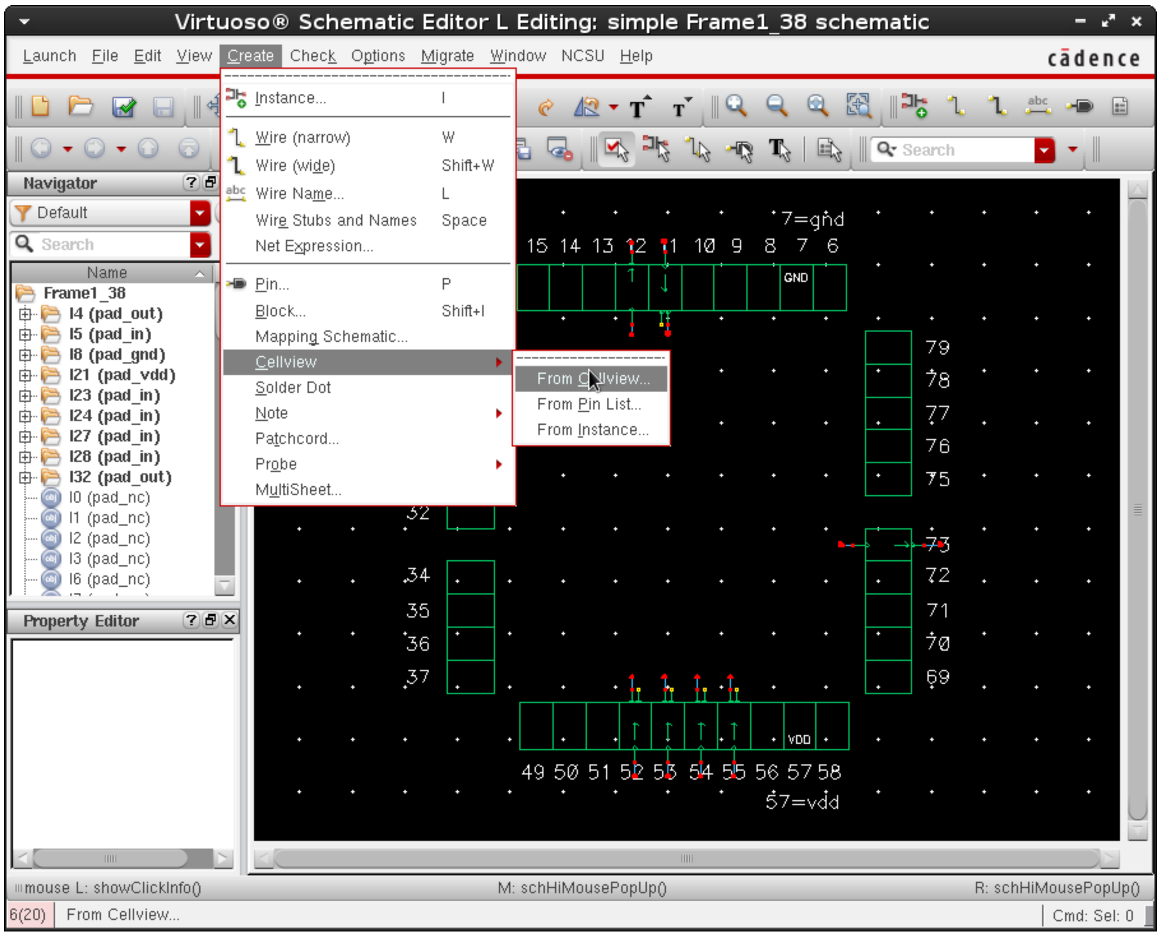
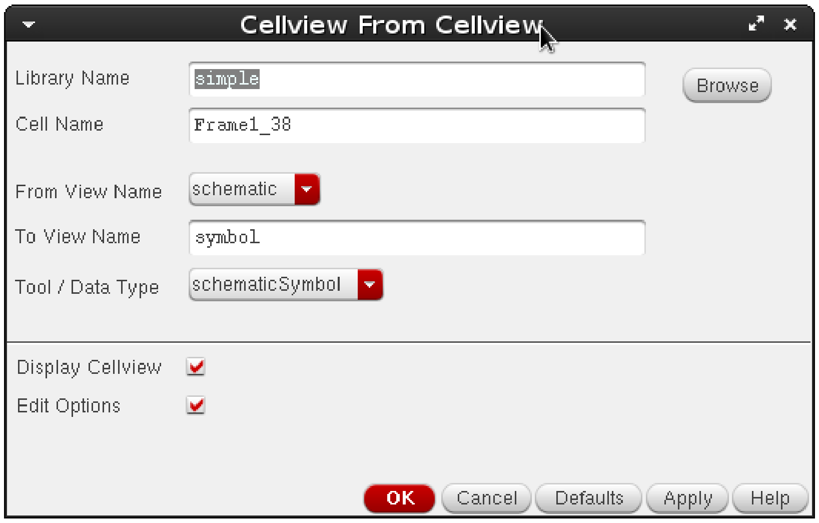
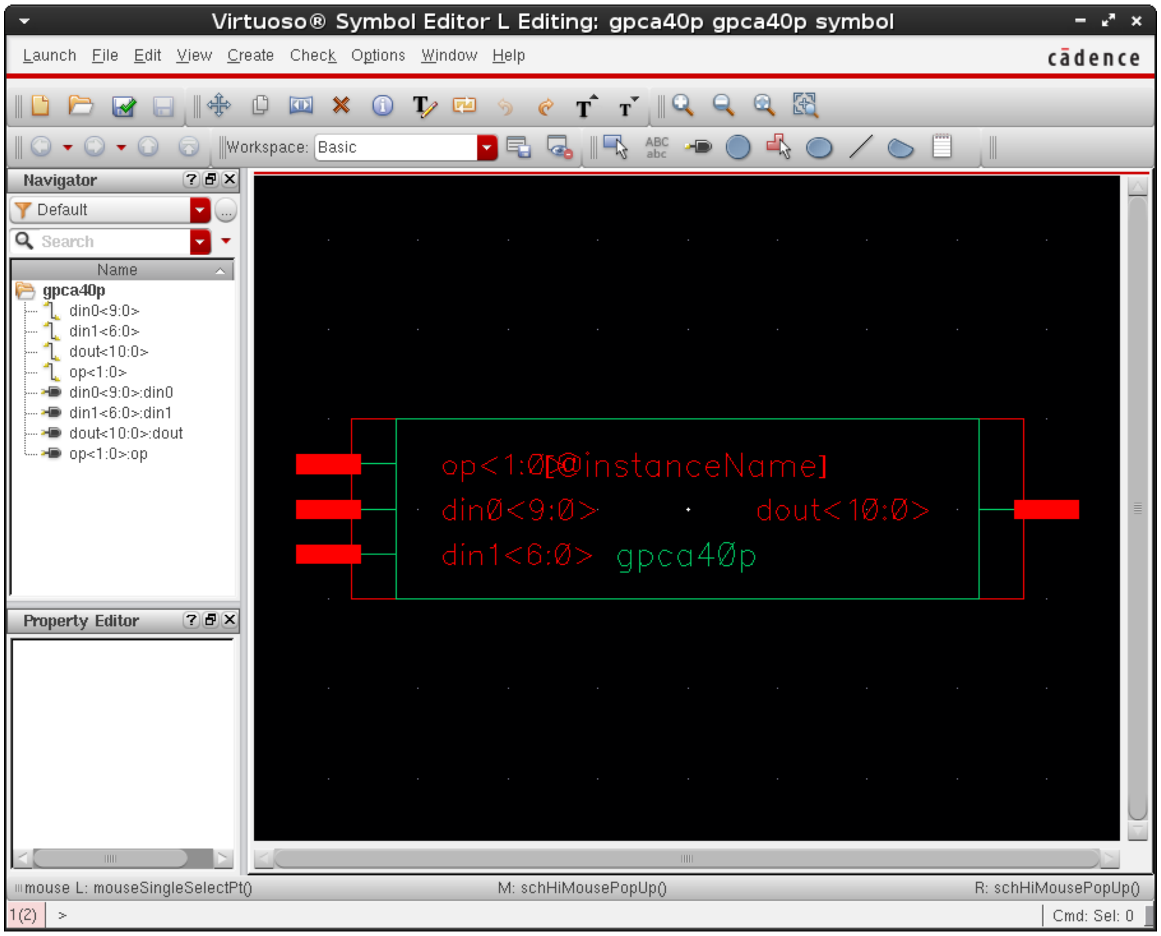


Figure 130: Create symbol view from schematic view.


Cellview From Cellview

Library Name: simple
Cell Name: Frame1_38
From View Name: schematic
To View Name: symbol
Tool / Data Type: schematicSymbol
Display Cellview: ✓
Edit Options: ✓

Figure 131: "CellView from CellView" window of Virtuoso.


Virtuoso® Symbol Editor L Editing: gpca40p gpca40p symbol

gpca40p
din0<9:0>
din1<6:0>
dout<10:0>
op<1:0>
din0<9:0>:din0
din1<6:0>:din1
dout<10:0>:dout
op<1:0>:op

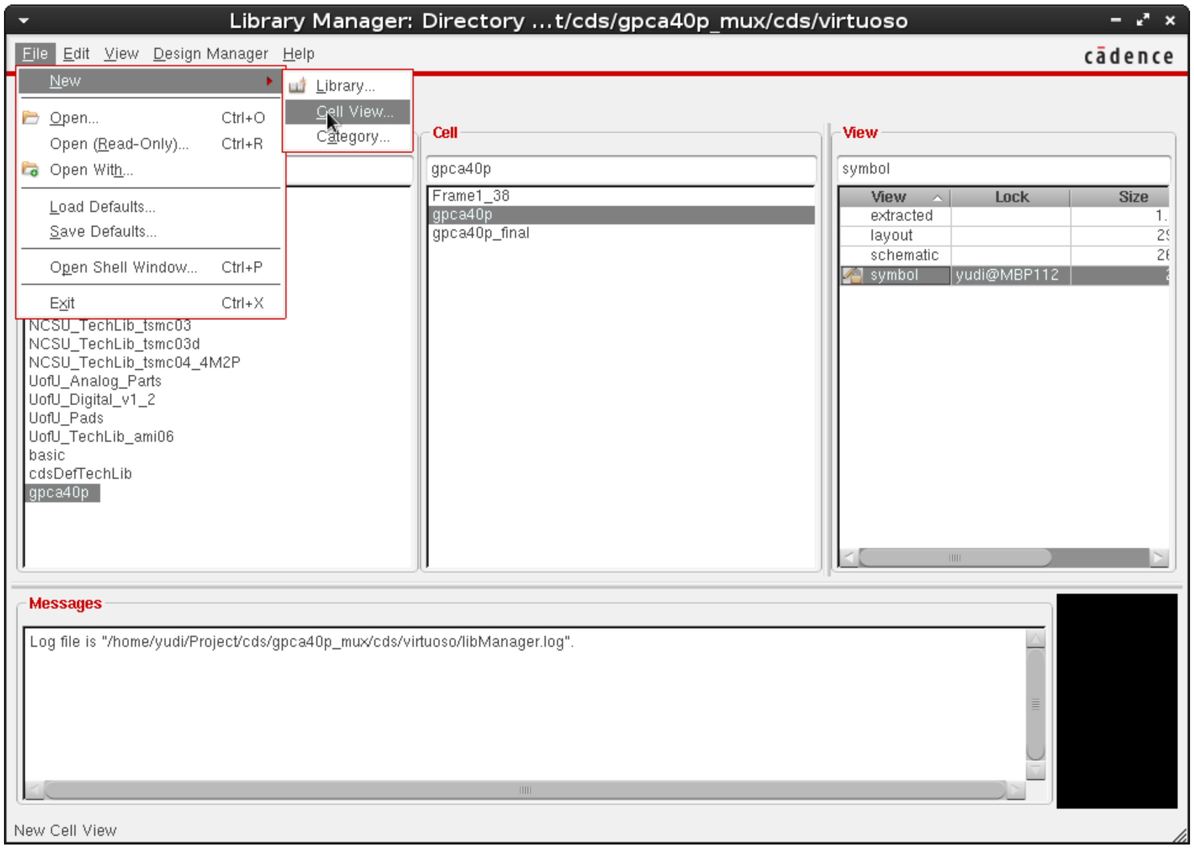
mouse L: mouseSingleSelectPt() M: schHiMousePopUp() R: schHiMousePopUp()
1(2) > Cmd: Sel: 0

Figure 132: Final symbol view of the pad frame.

e) Final Chip Assembly (Add Pad Frame)

With both the core of the chip ("simple") and the pad frame (modified "Frame1_38") are ready, creating a new cell to put them together. This procedure is shown in Figure 133 to 149.

Create a new Cell View (I call it "gpca40p_final").



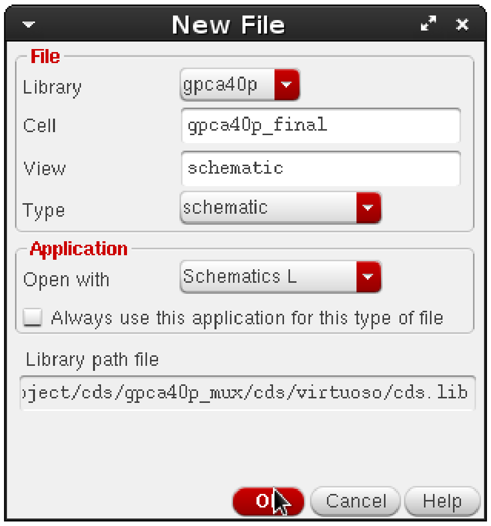
NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts
UofU_Digital_v1_2
UofU_Pads
UofU_TechLib_ami06
basic
cdsDefTechLib
gpca40p

Frame1_38
gpca40p
gpca40p_final

View	Lock	Size
extracted		1.
layout		2
schematic		2
symbol	yudi@MBP112	

```
Log file is "/home/yudi/Project/cds/gpca40p_mux/cds/virtuoso/libManager.log".
```

Figure 133: Create new "Cell View" from Virtuoso Layout Suite



Library gpca40p

Cell gpca40p_final

View schematic

Type schematic

Open with Schematics L

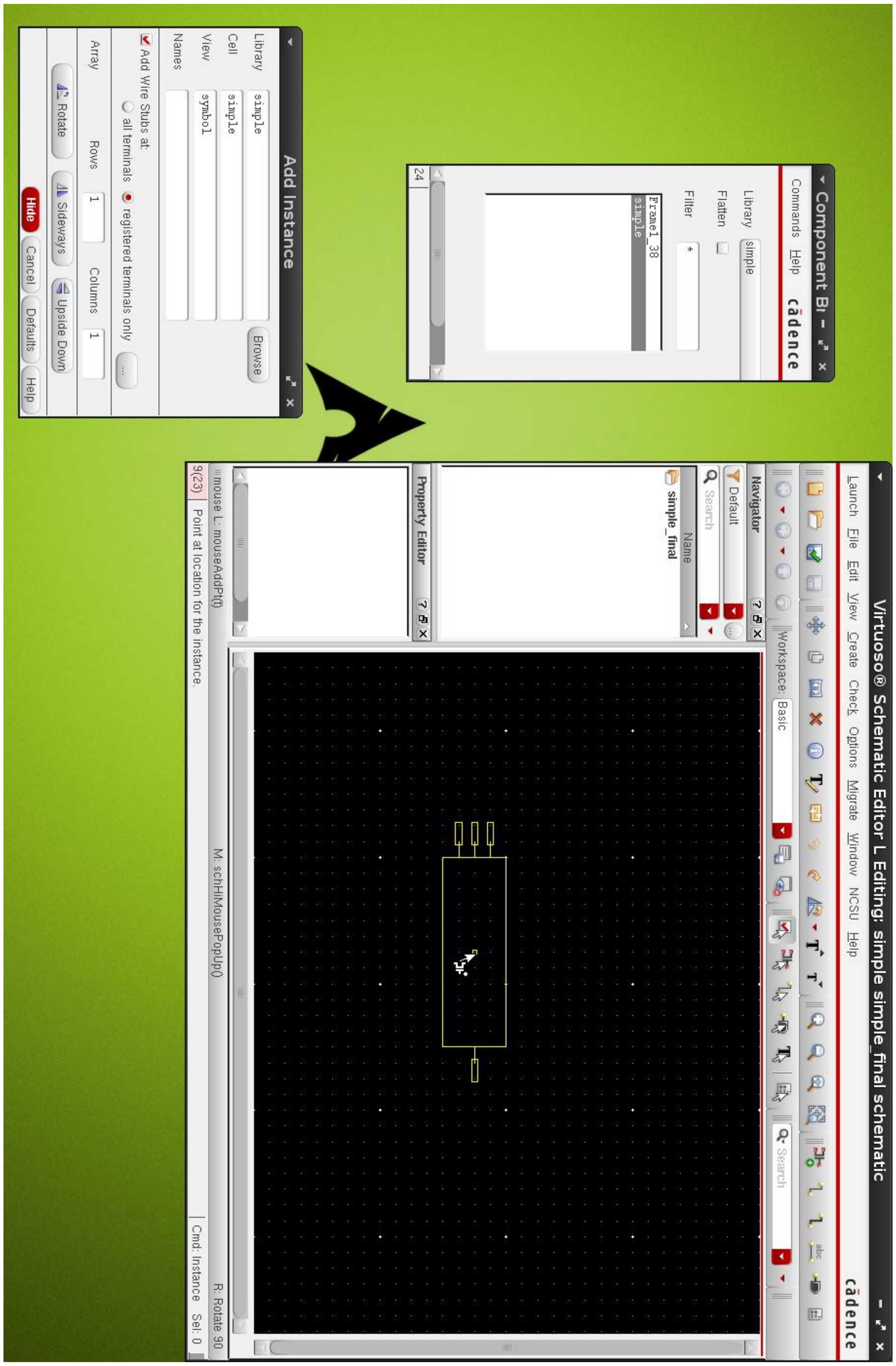
☐ Always use this application for this type of file

Library path file

```
›ject/cds/gpca40p_mux/cds/virtuoso/cds.lib
```

Figure 134: "New File" window of Virtuoso.

Instantiate both the core of the chip ("gpca40p") and the pad frame (modified "Frame1_38") (by pressing "I" to instantiate instances).



Library simple
Cell simple
View symbol
Names
Add Wire Stubs at: all terminals [✓ registered terminals only]
Array Rows 1 Columns 1

Library simple
Flatten
Filter *
Frame1_38
simple

mouse L: mouseAddPt() M: schHiMousePopUp() R: Rotate 90
9(23) Point at location for the instance. Cmd: Instance Sel: 0

Figure 135: Create the final schematic view.

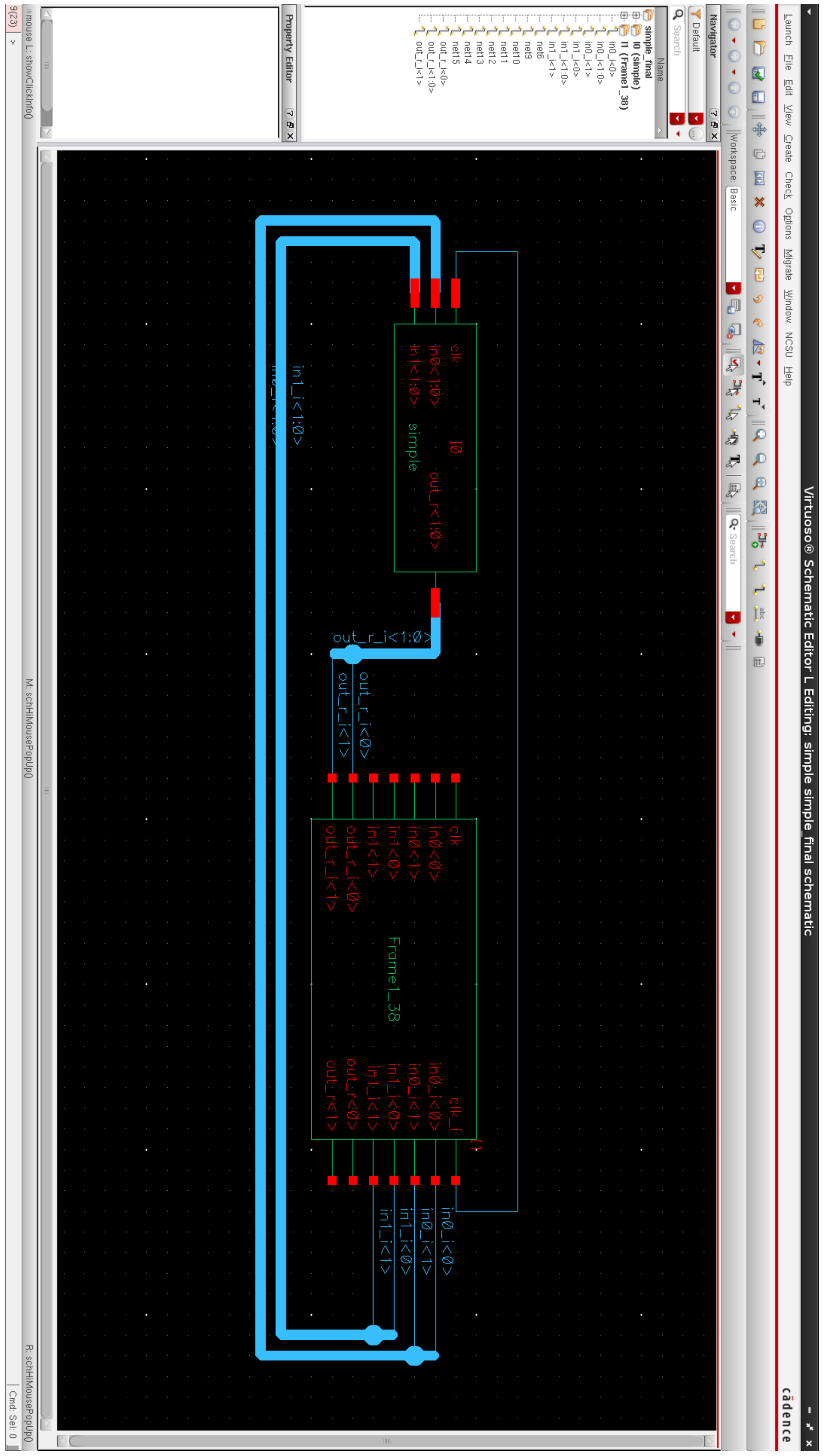
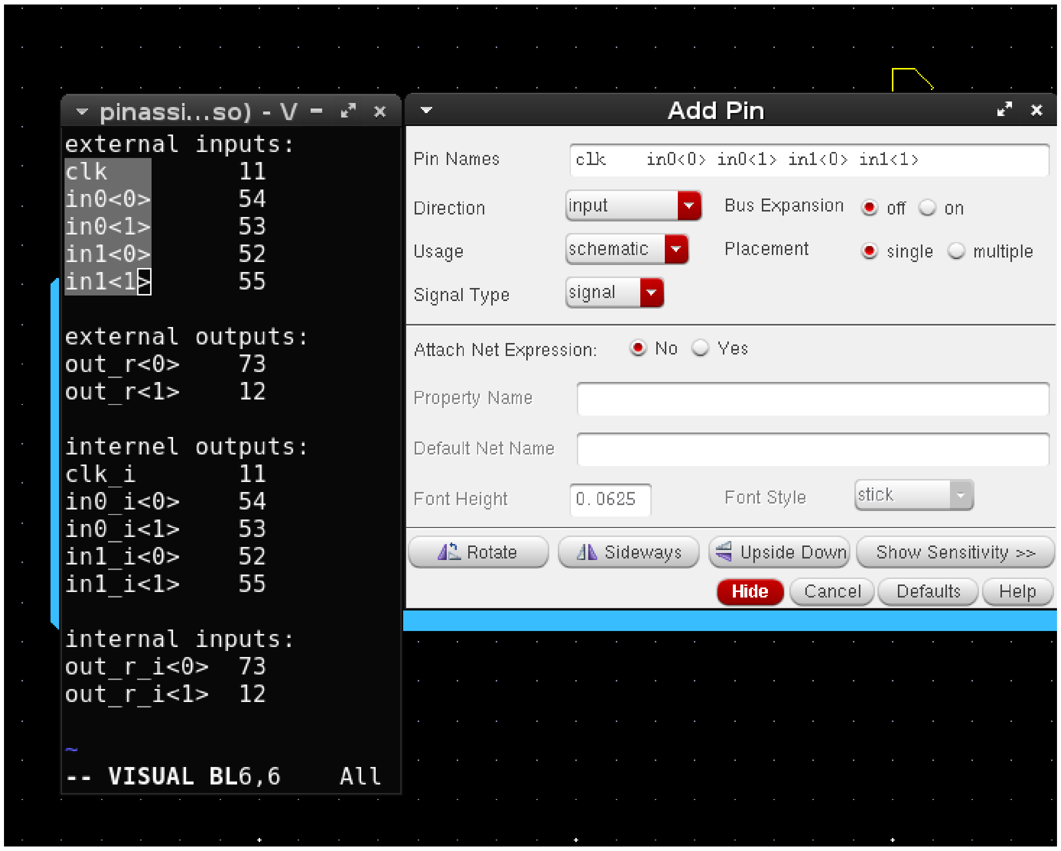


Figure 136: Routed schematic view.

Then add the pins.



```
external inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

external outputs:
out_r<0>     73
out_r<1>     12

internel outputs:
clk_i        11
in0_i<0>     54
in0_i<1>     53
in1_i<0>     52
in1_i<1>     55

internal inputs:
out_r_i<0>   73
out_r_i<1>   12

~
-- VISUAL BL6,6      All
```

Add Pin

Pin Names: clk in0<0> in0<1> in1<0> in1<1>

Direction: input

Usage: schematic

Signal Type: signal

Bus Expansion: [✓ off] on

Placement: [✓ single] multiple

Attach Net Expression: [✓ No] Yes

Property Name:

Default Net Name:

Font Height: 0.0625

Font Style: stick

Figure 137: "Add Pins" window of Virtuoso.

Final schematic view is shown below.

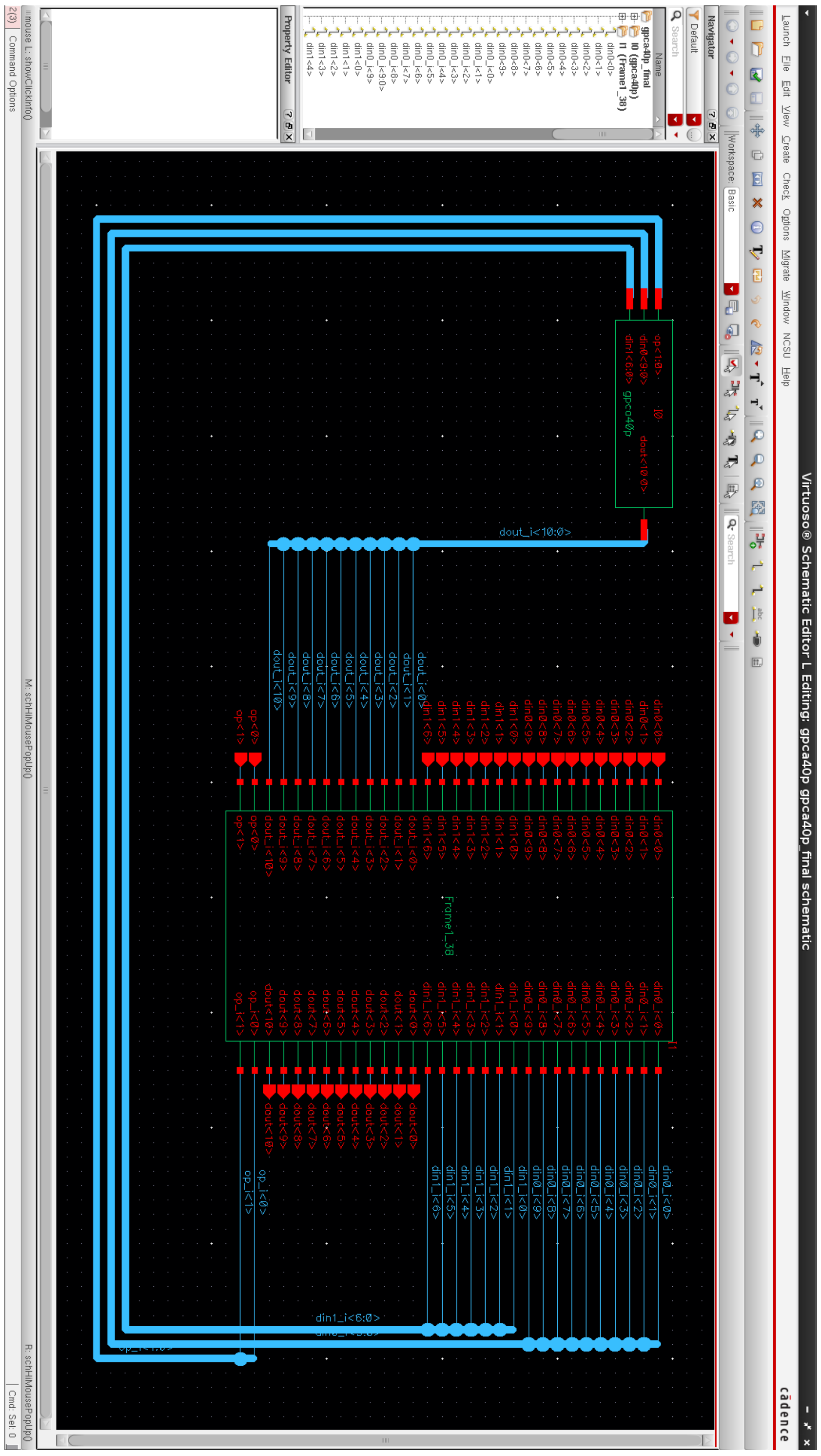
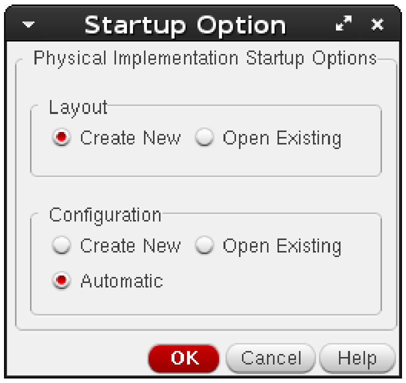


Figure 138: Final schematic view.

Then create the final layout view from the schematic.



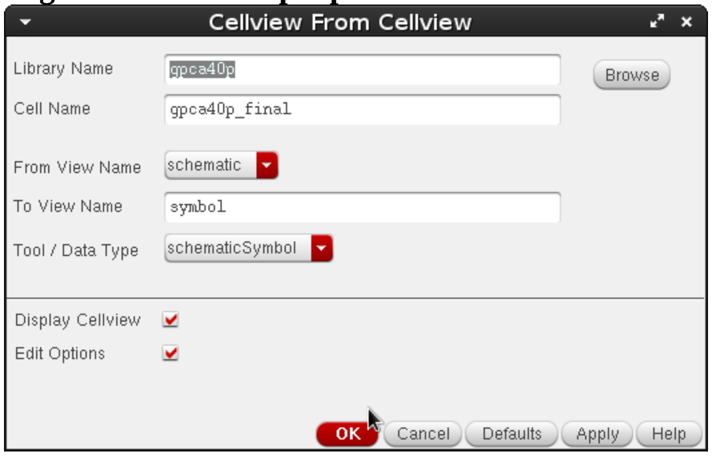
Startup Option

Physical Implementation Startup Options

Layout: [● Create New] ○ Open Existing

Configuration: ○ Create New ○ Open Existing [● Automatic]

Figure 139: "Startup Option" window of Virtuoso.



Cellview From Cellview

Library Name	gpca40p
Cell Name	gpca40p_final
From View Name	schematic
To View Name	symbol
Tool / Data Type	schematicSymbol
Display Cellview	✓
Edit Options	✓

Figure 140: "Startup Option" window of Virtuoso.

In the new empty layout view

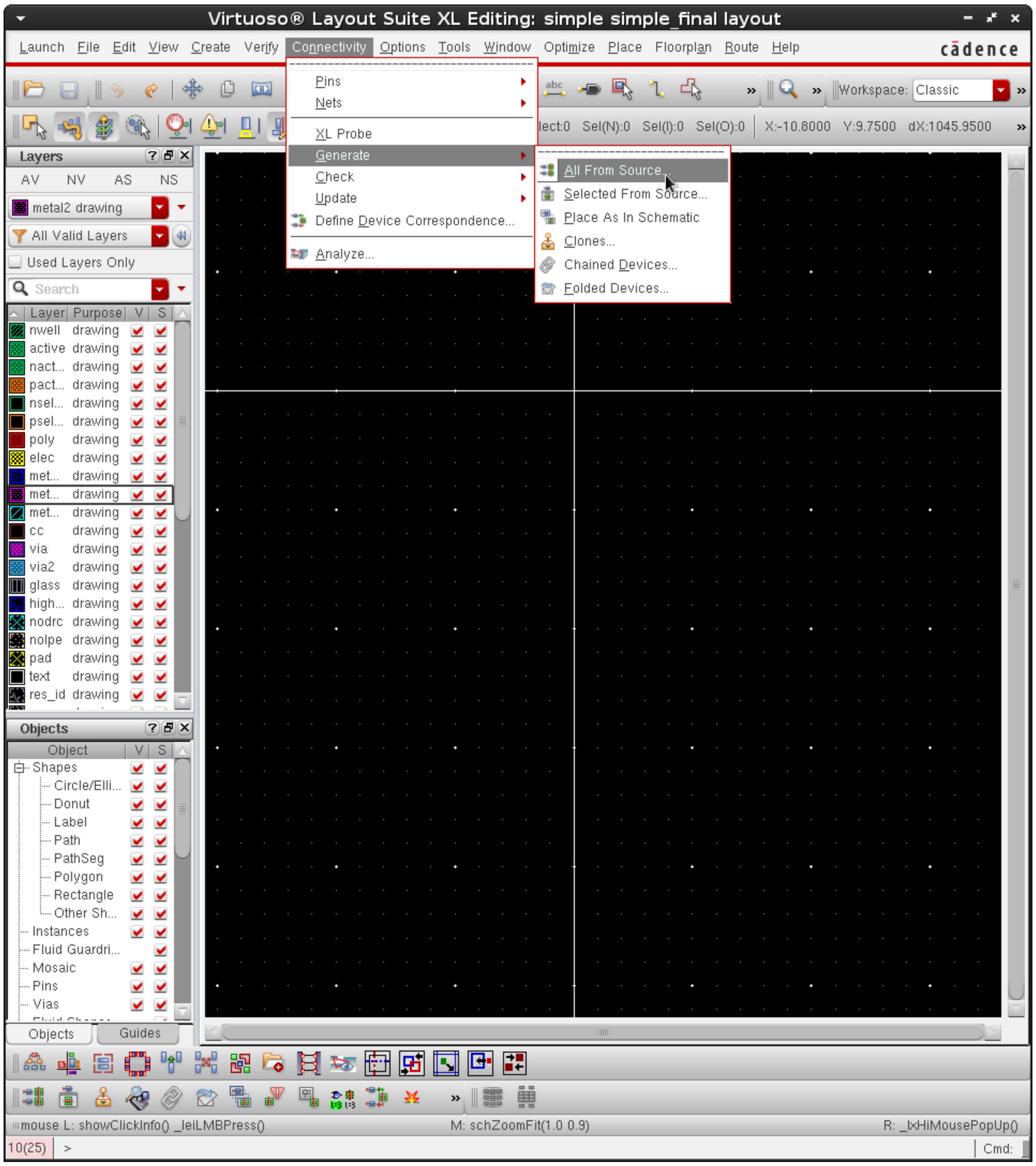
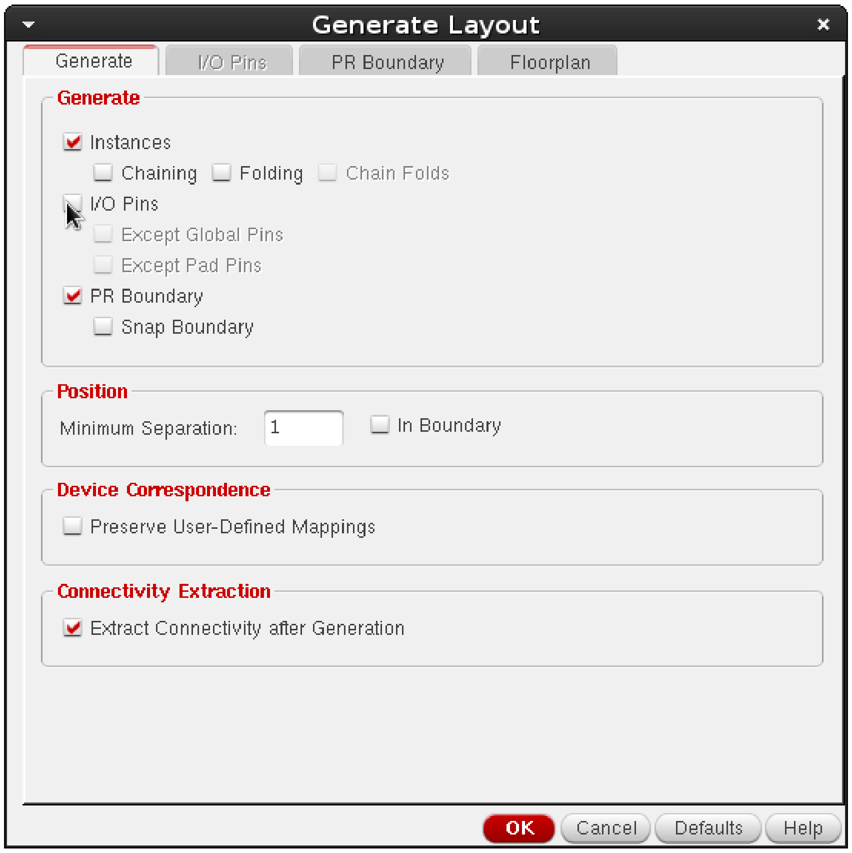

Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Pins
Nets
XL Probe
Generate
Check
Update
Define Device Correspondence...
Analyze...
All From Source...
Selected From Source...
Place As In Schematic
Clones...
Chained Devices...
Folded Devices...
Workspace: Classic
Layers
Objects
Guides

Figure 141: Start "Generate Layout" of Virtuoso.


Generate Layout

Generate | I/O Pins | PR Boundary | Floorplan

Generate

[✓] Instances
[ ] Chaining [ ] Folding [ ] Chain Folds
[ ] I/O Pins
[ ] Except Global Pins
[ ] Except Pad Pins
[✓] PR Boundary
[ ] Snap Boundary

Position

Minimum Separation: 1 [ ] In Boundary

Device Correspondence

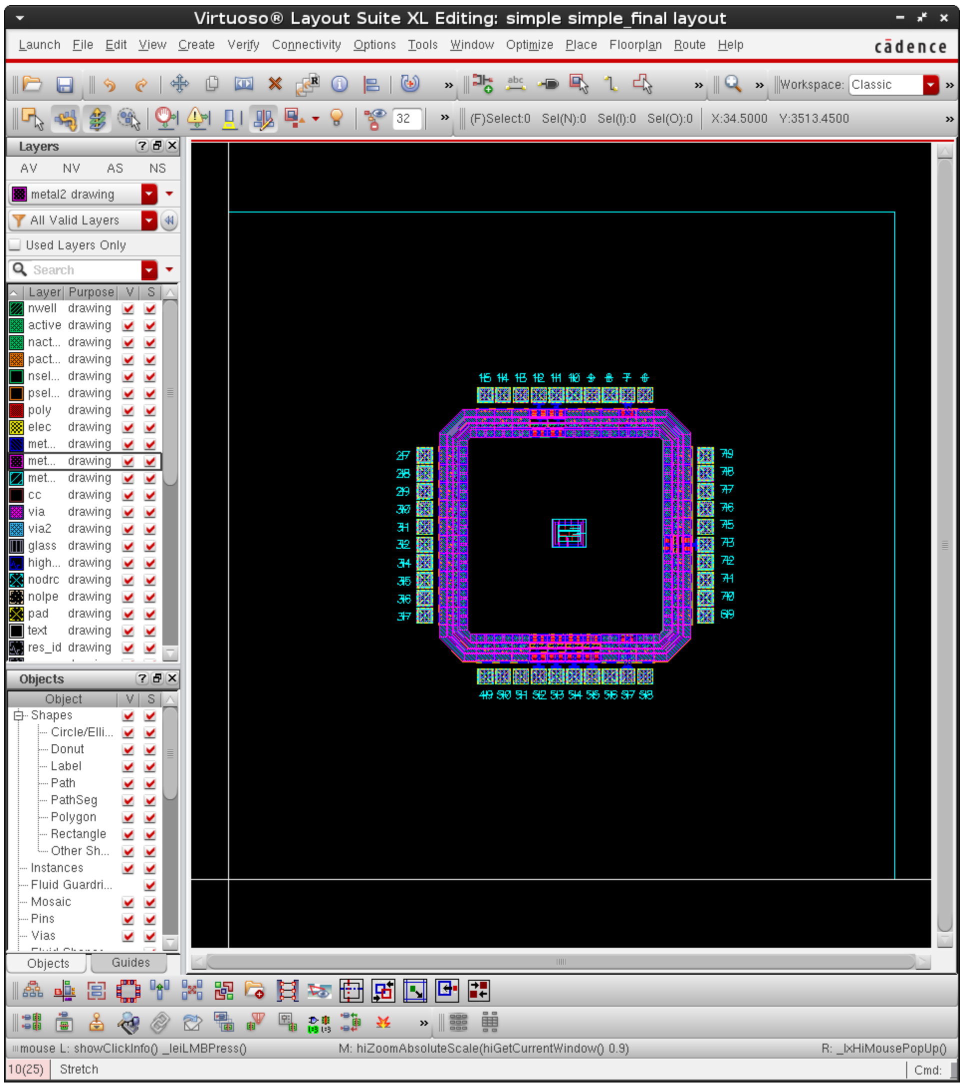
[ ] Preserve User-Defined Mappings

Connectivity Extraction

[✓] Extract Connectivity after Generation

Figure 142: "Generate Layout" window of Virtuoso.

Then place the pad frame ("Frame1_38") inside the "PR Boundary", and the core of the chip ("gpca40p") inside the pad frame.


Virtuoso® Layout Suite XL Editing: simple simple_final layout

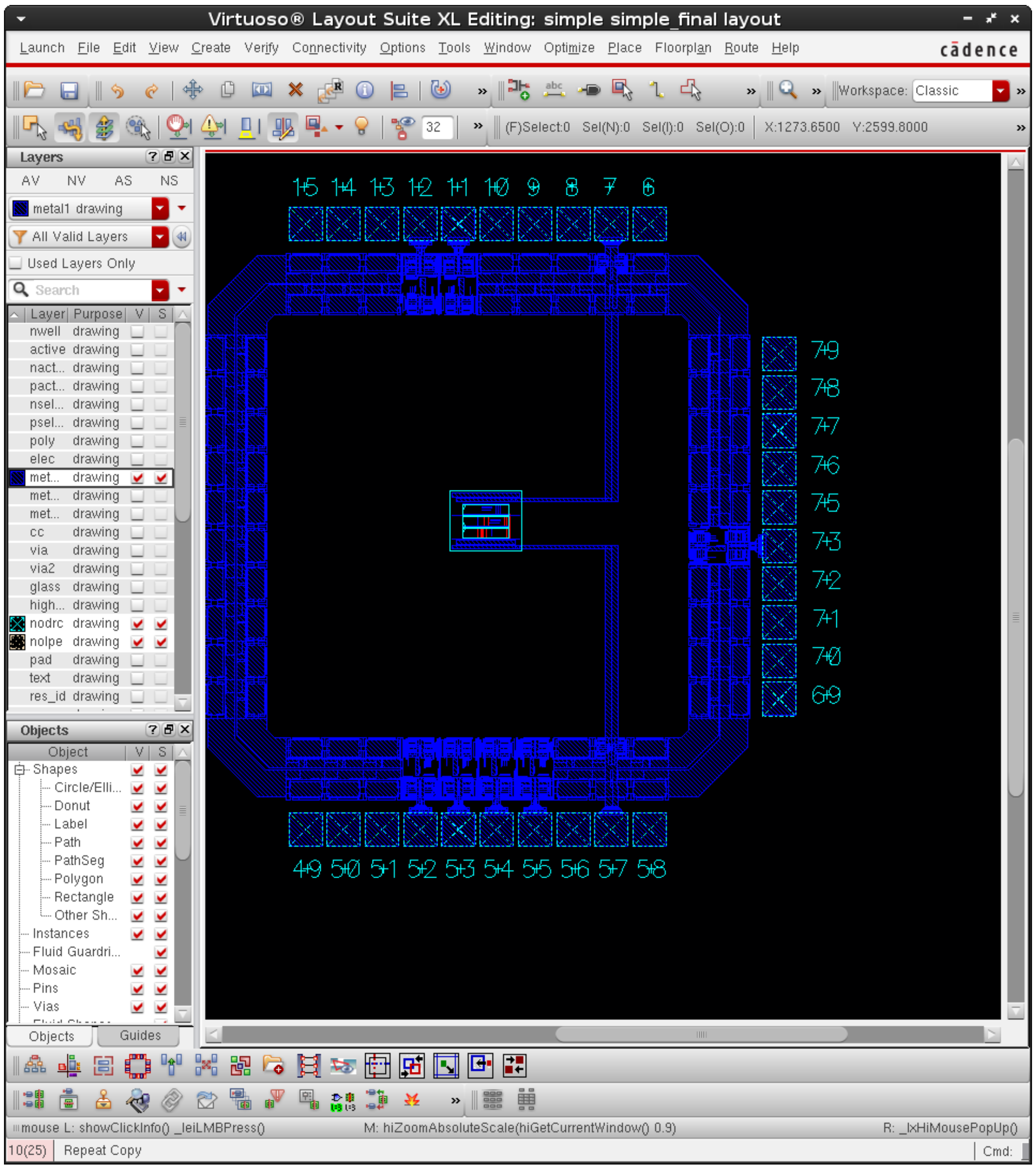
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:34.5000 Y:3513.4500

mouse L: showClickInfo() _leiLMBPress() M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9) R: _lxHiMousePopUp()

10(25) Stretch Cmd:

Figure 143: Component placement in Virtuoso.

Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.

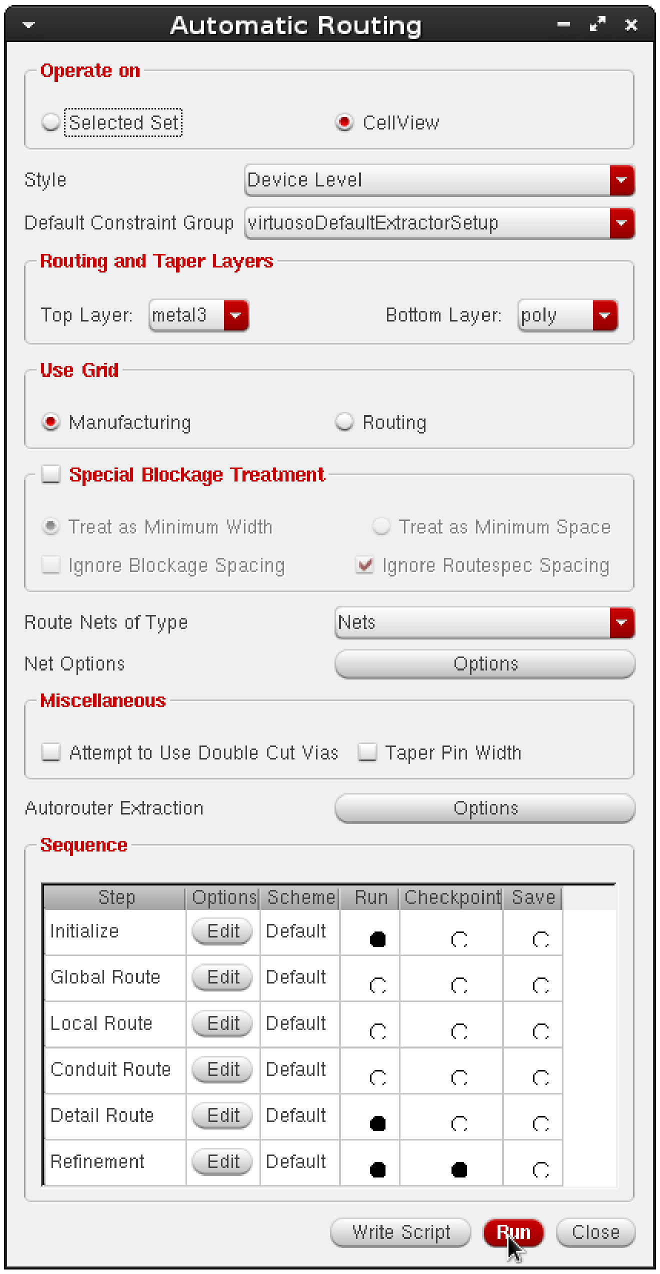


mouse L: showClickInfo() _leiLMBPress() M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9) R: _lxHiMousePopUp()

10(25) Repeat Copy Cmd:

Figure 144: Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.

Then invoke Automatic Routing.



Automatic Routing

Operate on

Selected Set [✓ CellView]

Style: Device Level

Default Constraint Group: virtuosoDefaultExtractorSetup

Routing and Taper Layers

Top Layer: metal3   Bottom Layer: poly

Use Grid

[✓ Manufacturing] Routing

☐ **Special Blockage Treatment**

[✓ Treat as Minimum Width] Treat as Minimum Space

☐ Ignore Blockage Spacing   ✓ Ignore Routespec Spacing

Route Nets of Type: Nets

Net Options: Options

Miscellaneous

☐ Attempt to Use Double Cut Vias   ☐ Taper Pin Width

Autorouter Extraction: Options

Sequence

Step	Options	Scheme	Run	Checkpoint	Save
Initialize	Edit	Default	●		
Global Route	Edit	Default			
Local Route	Edit	Default			
Conduit Route	Edit	Default			
Detail Route	Edit	Default	●		
Refinement	Edit	Default	●	●	

Figure 145: "Automatic Routing" window of Virtuoso.

The following is the final layout. Fill the empty space with poly fills to meet minimum polysilicon density required by AMI05 [35].

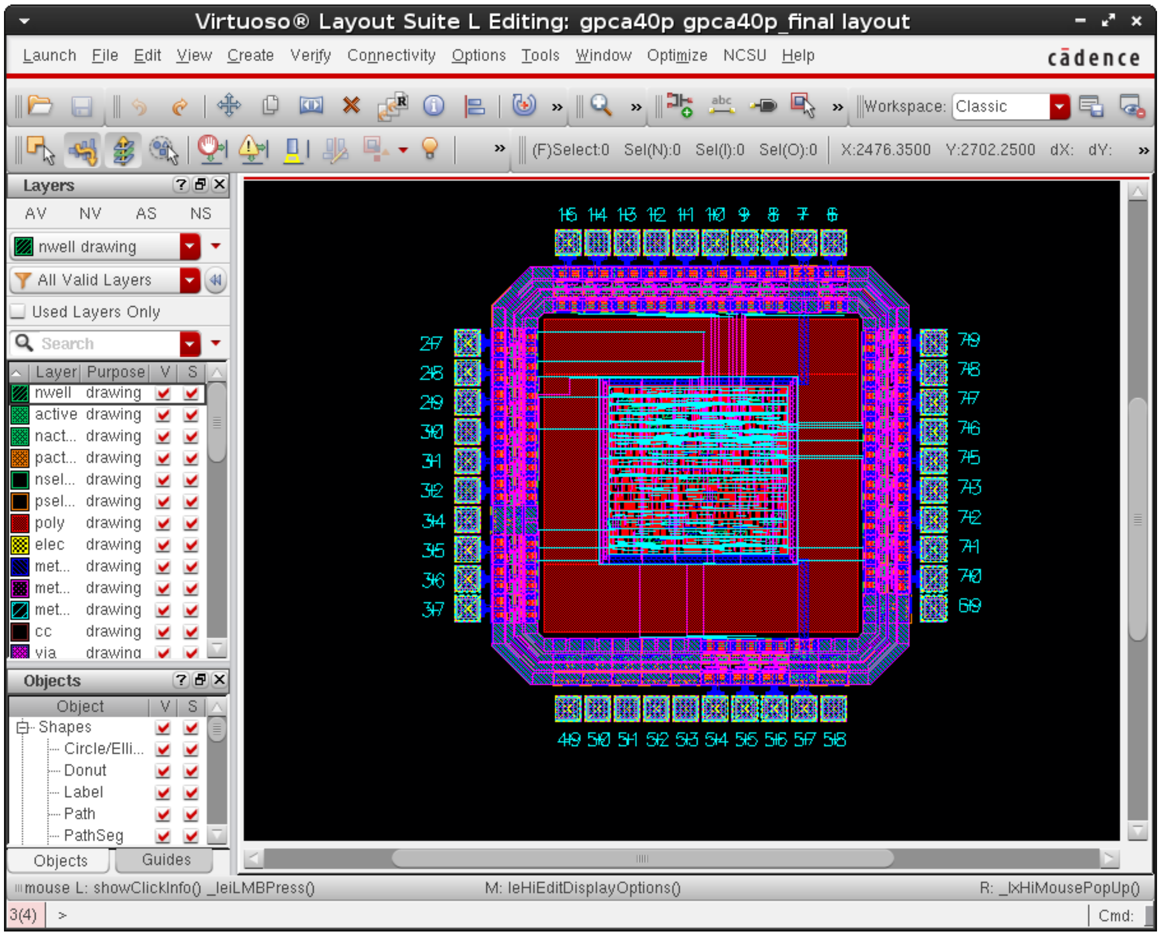
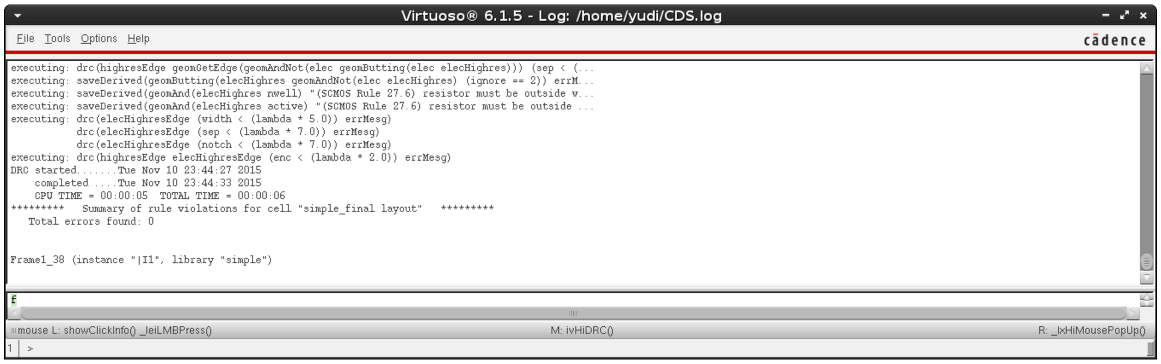


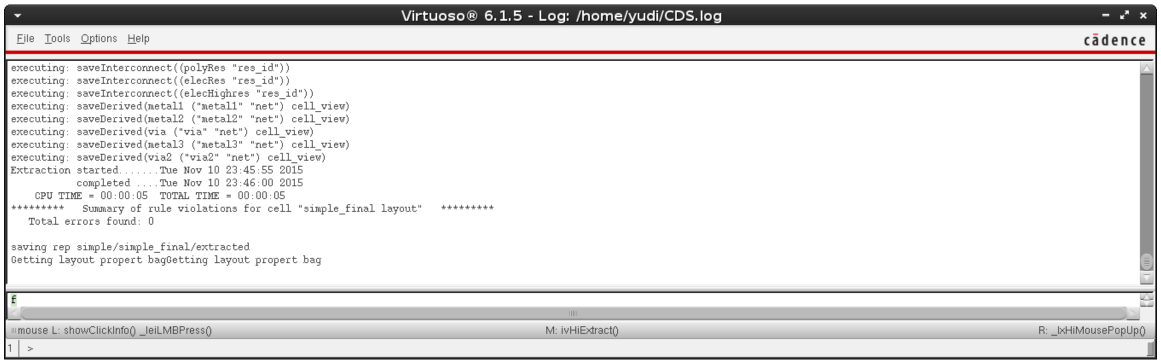
Figure 146: Final layout in Virtuoso.

DRC, Extract, LVS. They should give no error or warning.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiDRC()
R: _lxHiMousePopUp()

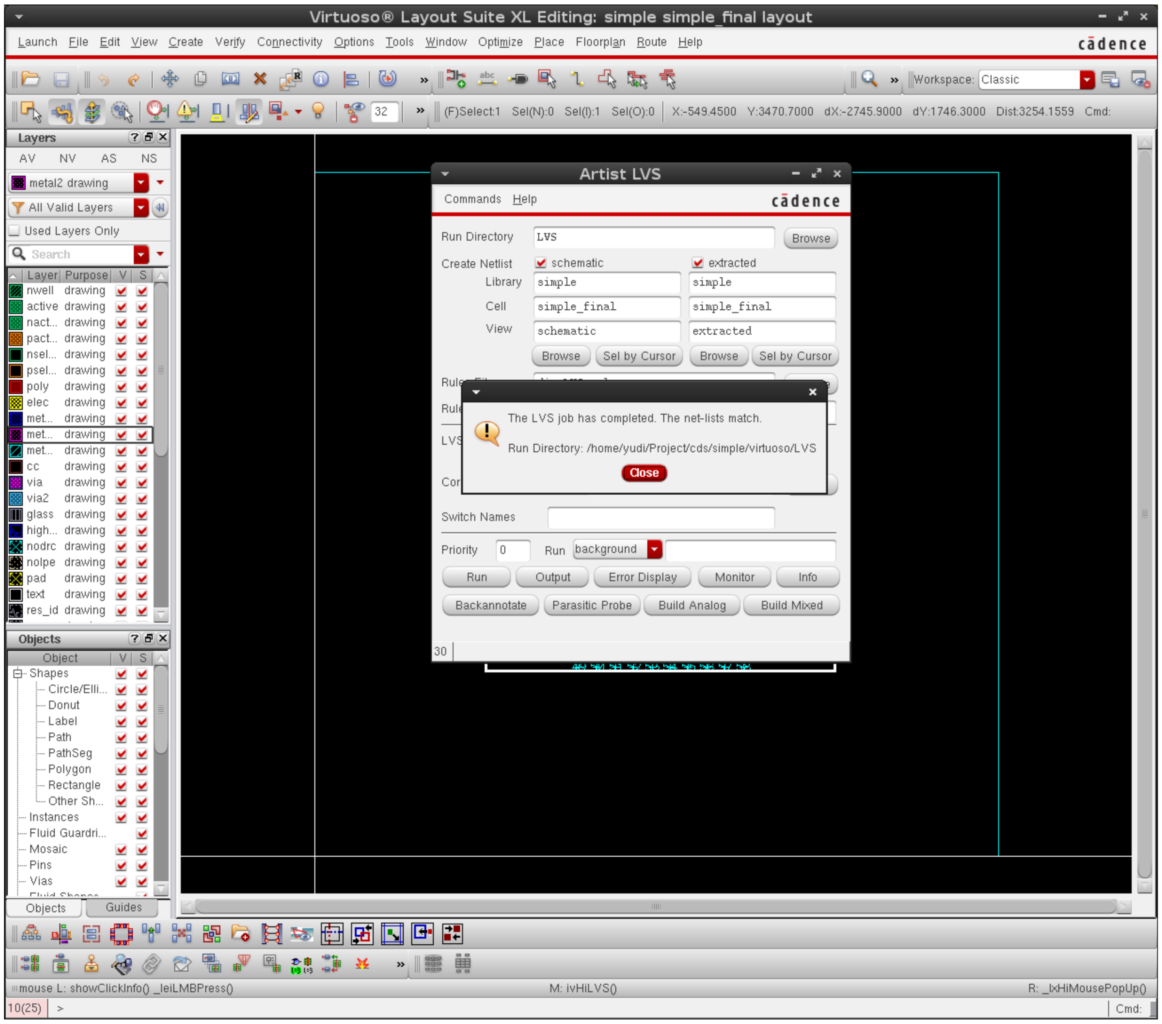
```
executing: drc(highresEdge geomGetEdge(geomAndNot(elec geomButting(elec elecHighres))) (sep < (...
executing: saveDerived(geomButting(elecHighres geomAndNot(elec elecHighres) (ignore == 2)) errM...
executing: saveDerived(geomAnd(elecHighres nwell) "(SCMOS Rule 27.6) resistor must be outside w...
executing: saveDerived(geomAnd(elecHighres active) "(SCMOS Rule 27.6) resistor must be outside ...
executing: drc(elecHighresEdge (width < (lambda * 5.0)) errMesg)
           drc(elecHighresEdge (sep < (lambda * 7.0)) errMesg)
           drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.......Tue Nov 10 23:44:27 2015
    completed ....Tue Nov 10 23:44:33 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:06
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0


Frame1_38 (instance "|I1", library "simple")
```

Figure 147: Report of successful DRC.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiExtract()
R: _lxHiMousePopUp()

```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 23:45:55 2015
           completed ....Tue Nov 10 23:46:00 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:05
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0

saving rep simple/simple_final/extracted
Getting layout propert bagGetting layout propert bag
```

Figure 148: Report of successful Extract.


Virtuoso® Layout Suite XL Editing: simple simple_final layout

(F)Select:1 Sel(N):0 Sel(I):1 Sel(O):0 X:-549.4500 Y:3470.7000 dX:-2745.9000 dY:1746.3000 Dist:3254.1559 Cmd:

Artist LVS

Run Directory: LVS

Create Netlist: [✓ schematic] [✓ extracted]

	schematic	extracted
Library	simple	simple
Cell	simple_final	simple_final
View	schematic	extracted

Priority: 0 Run: background

The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

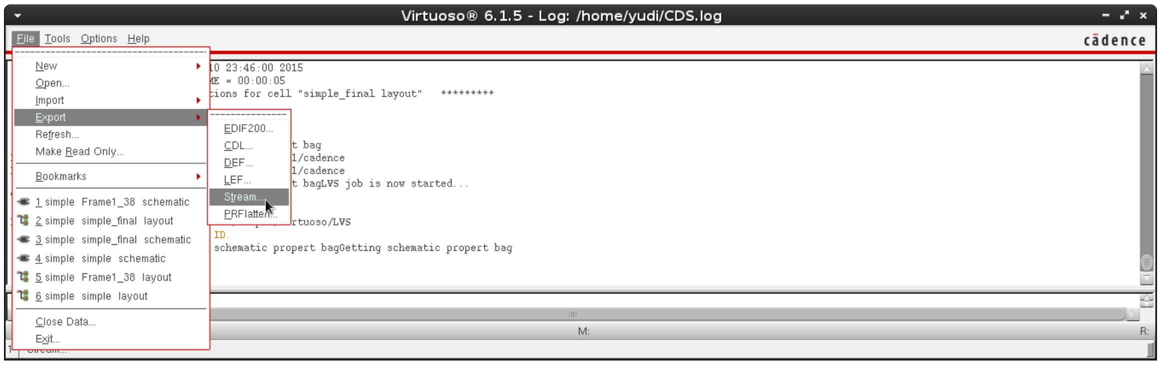
mouse L: showClickInfo() _leiLMBPress() M: ivHiLVS() R: _lxHiMousePopUp()

Figure 149: "Artist LVS" successful LVS prompt.

f) Export GDSII Stream File ("simple_final.gds")

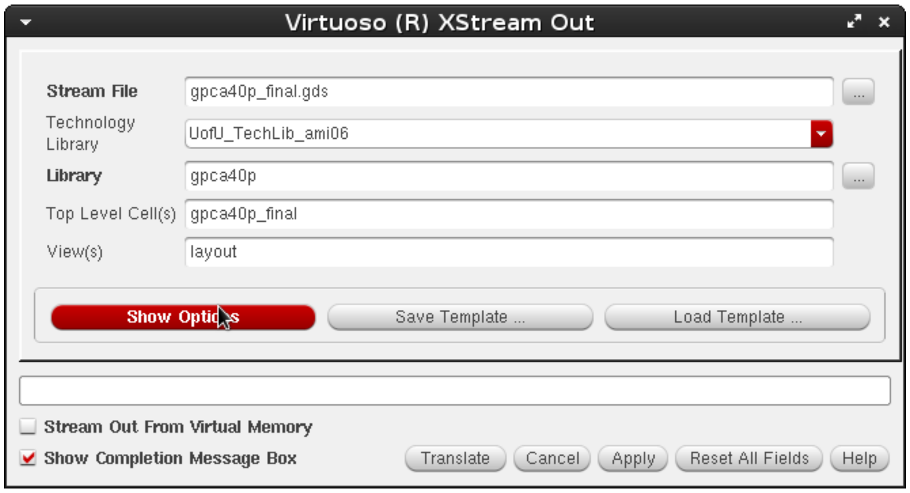
This procedure is shown in Figure 150 to Figure 155. This is the file to be sent for fabrication.

This is the file to be sent for fabrication.



```
10 23:46:00 2015
ME = 00:00:05
tions for cell "simple_final layout"   *********
t bag
1/cadence
1/cadence
t bagLVS job is now started...
rtuoso/LVS
ID.
schematic propert bagGetting schematic propert bag
```

Figure 150: Start "Stream Out" from Virtuoso Layout Suite.

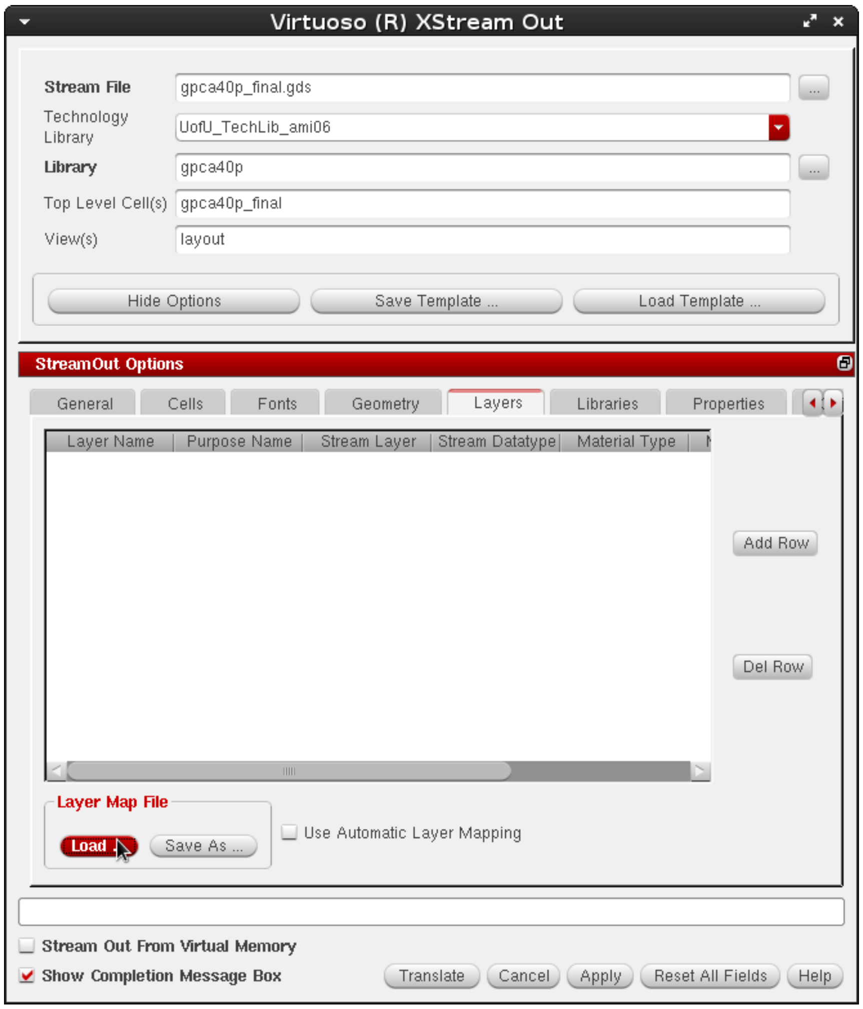


Stream File	gpca40p_final.gds
Technology Library	UofU_TechLib_ami06
Library	gpca40p
Top Level Cell(s)	gpca40p_final
View(s)	layout

[ ] Stream Out From Virtual Memory
[✓] Show Completion Message Box

Figure 151: "Show Options" in "XStream Out" window of Virtuoso.

"Show Options", then tab "Layers", "Load ..."

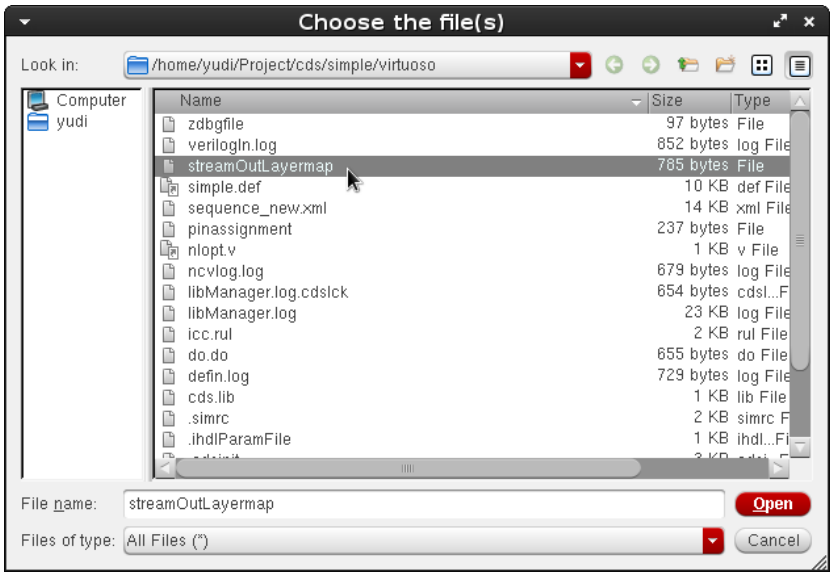

Virtuoso (R) XStream Out

Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

☐ Use Automatic Layer Mapping
☐ Stream Out From Virtual Memory
☑ Show Completion Message Box

Figure 152: "Load..." in "XStream Out" window of Virtuoso.

Select "streamOutLayermap" then hit "Open"



/home/yudi/Project/cds/simple/virtuoso

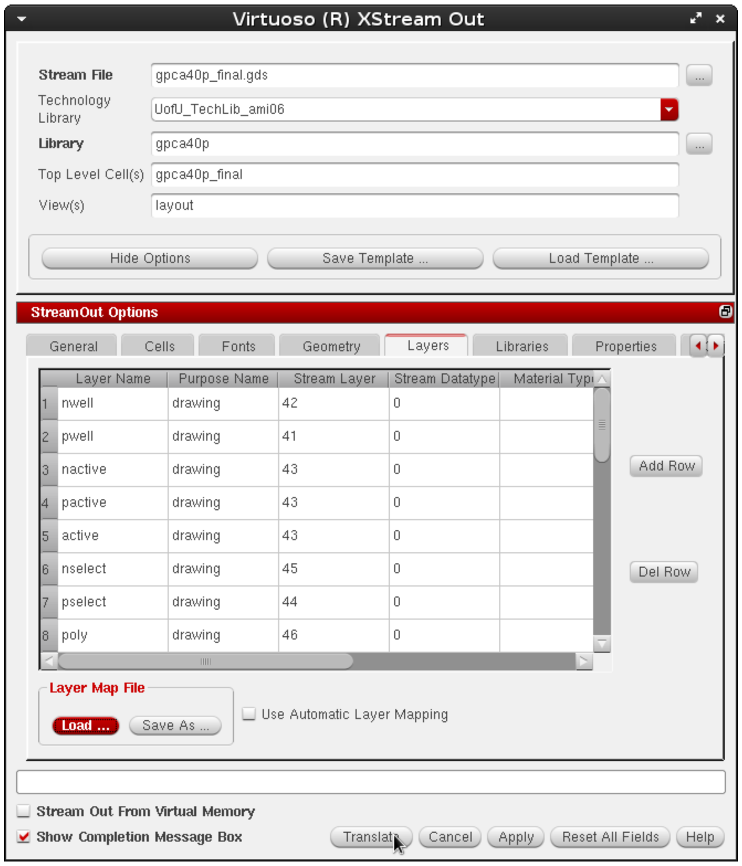
Name	Size	Type
zdbgfile	97 bytes	File
verilogIn.log	852 bytes	log File
streamOutLayermap	785 bytes	File
simple.def	10 KB	def File
sequence_new.xml	14 KB	xml File
pinassignment	237 bytes	File
nlopt.v	1 KB	v File
ncvlog.log	679 bytes	log File
libManager.log.cdslck	654 bytes	cdsl...F
libManager.log	23 KB	log File
icc.rul	2 KB	rul File
do.do	655 bytes	do File
defin.log	729 bytes	log File
cds.lib	1 KB	lib File
.simrc	2 KB	simrc F
.ihdlParamFile	1 KB	ihdl...Fi

File name: streamOutLayermap

Files of type: All Files (*)

Figure 153: Choose file for the "XStream Out".

Click "Translate".



Virtuoso (R) XStream Out

Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

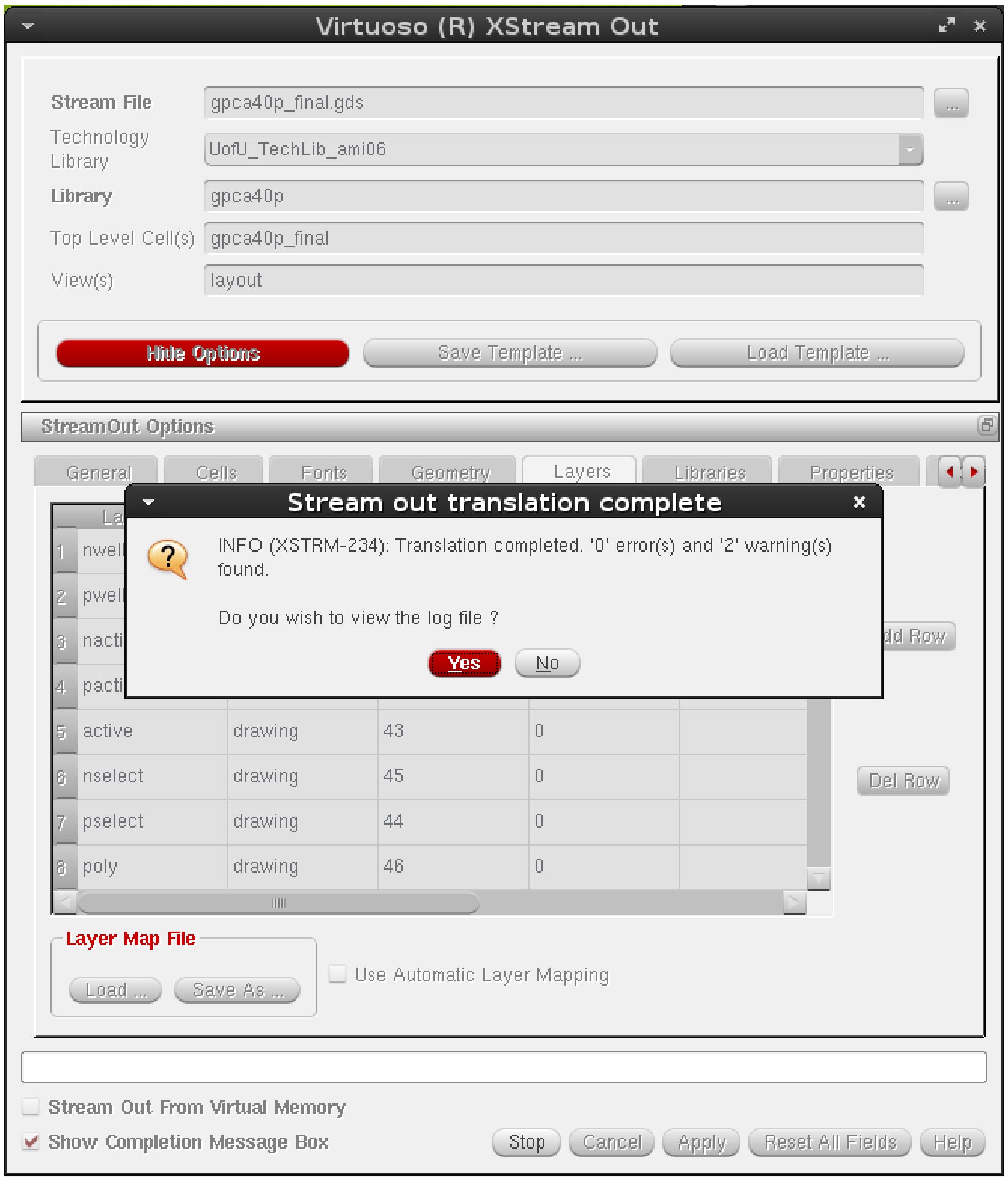
StreamOut Options — Layers

	Layer Name	Purpose Name	Stream Layer	Stream Datatype	Material Type
1	nwell	drawing	42	0	
2	pwell	drawing	41	0	
3	nactive	drawing	43	0	
4	pactive	drawing	43	0	
5	active	drawing	43	0	
6	nselect	drawing	45	0	
7	pselect	drawing	44	0	
8	poly	drawing	46	0	

Use Automatic Layer Mapping [ ]
Stream Out From Virtual Memory [ ]
Show Completion Message Box [✓]

Figure 154: "Translate" in "XStream Out" window of Virtuoso.

It should produce no error and the only warning is about "nodrc:drawing" (and a possible overwriting existing file warning).


Virtuoso (R) XStream Out

Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

1	nwell			
2	pwell			
3	nacti			
4	pacti			
5	active	drawing	43	0
6	nselect	drawing	45	0
7	pselect	drawing	44	0
8	poly	drawing	46	0

Stream out translation complete

INFO (XSTRM-234): Translation completed. '0' error(s) and '2' warning(s) found.

Do you wish to view the log file ?

Use Automatic Layer Mapping

Stream Out From Virtual Memory

✓ Show Completion Message Box

Figure 155: "Stream out translation complete" successful prompt.

Design Submission

In this section, the procedure for MOSIS submission with the MOSIS Educational Program (MEP) is given as a reference. More information is available in documents from MOSIS [37], and adapted from [38].

1) Fill in "MOSIS New Project Request Form":

Run Type: Shared IC Fabrication Run

Design Rules: Scalable CMOS

Technology: SCN3M_SUBM (if the second layer of poly is not used); SCN3ME_SUBM (if the second layer of poly is used).

Design Name and Password

Export Control: Standard

Substrate: none

Needs Library Installation: No

IP Included: none

Fill Authorized: Yes

Foundry: On Semi

Intended Disposition: Research

Design Size X and Y: Size including pads

Pad Count: How many pads used in the design (including signals and power)

Quantity Packaged: 5

Package Name: Depends on the design

Rotation in Package: None

Bonding Diagram Supplier: MOSIS

Downbond Locations: None

Quantity Unpackaged: 0

2) Fill in "Fabricate Form":

Go to Project Request -> Fabricate

Layout Transfer Method: I will upload layout via secure web form (HTTPS)

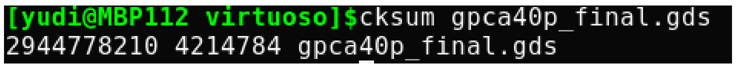
Compression/Encryption: Uncompressed

Generate the checksum and Count for the GDS file (Figure 156)

Layout Status: Final

Layout Format: GDS

Top Structure: the name of the top-level (e.g. "simple")



```
[yudi@MBP112 virtuoso]$cksum gpca40p_final.gds
2944778210 4214784 gpca40p_final.gds
```

Figure 156: Generating checksum with GNU cksum

Results

Using the estimation algorithm devised in this chapter, the design fully meets the design requirements. The result is summarized in Table 6 and Table 7 respectively. The design has been sent to MOSIS for fabrication.

Table 6. Summary of operations of the extension within constrains

Operation	**Original**	**Extended**
Multiply	7 by 5	12 by 10
Divide	7 by 4	12 by 9
Square	5	10
Square root	10	20

Table 7. Summary of pin count of the extension within constrains

	Original	**Extended**
Pin Number	86	30

Conclusion

In this chapter, an algorithm is devised, such that the pin count and the chip area of the pipeline can be estimated based on the number of rows. The expressions for this estimation algorithm based on least square are also given. The graphs explaining the algorithm are included. These graphs help in finding the relationship between the number of rows and the chip area. Such algorithms will help in the development of advanced arithmetic processors. In addition, the VLSI implementation of such a design has been discussed and detailed procedure for the implementation is included. The parameters have been met in the design. The design has been sent to MOSIS for fabrication. The chip will be tested once we get the fabricated chip back. The behavior Verilog code for this design is listed in the Appendix.

CHAPTER 4 EXTENSION FOR PIPELINING

Introduction

The original implementation proposed by [9] does not include the implementation of intermediate stage registers. The signals traverse through the array. The array is purely a combinational logic. To increase the throughput of the pipeline array to perform arithmetic computation, a technique called pipelining is used. The design and implementation of digital systems is taken up in this thesis. As a case study, the extension for pipelining upon the original generalized pipeline array is discussed. The generalized pipeline cellular array is introduced and extended such that it provides an alternative way in arithmetic processor design, in an environment of limited resources while achieving high performance and flexibility. These characteristics make it also suitable for being used as a case study in digital VLSI design education.

In computing, a pipeline is a series of data processing stages, in which the output of one stage is the input of the next one. The basic idea of pipelining is to split a major task into several balanced stages, in which the operations within stages of a pipeline are often carried out in parallel. In this fashion although the latency of data processing is slightly increased because the use of buffer, ideally the throughput of data processing is increased multiple times by the number of pipeline stages. This idea is illustrated in Figure 157 [43].

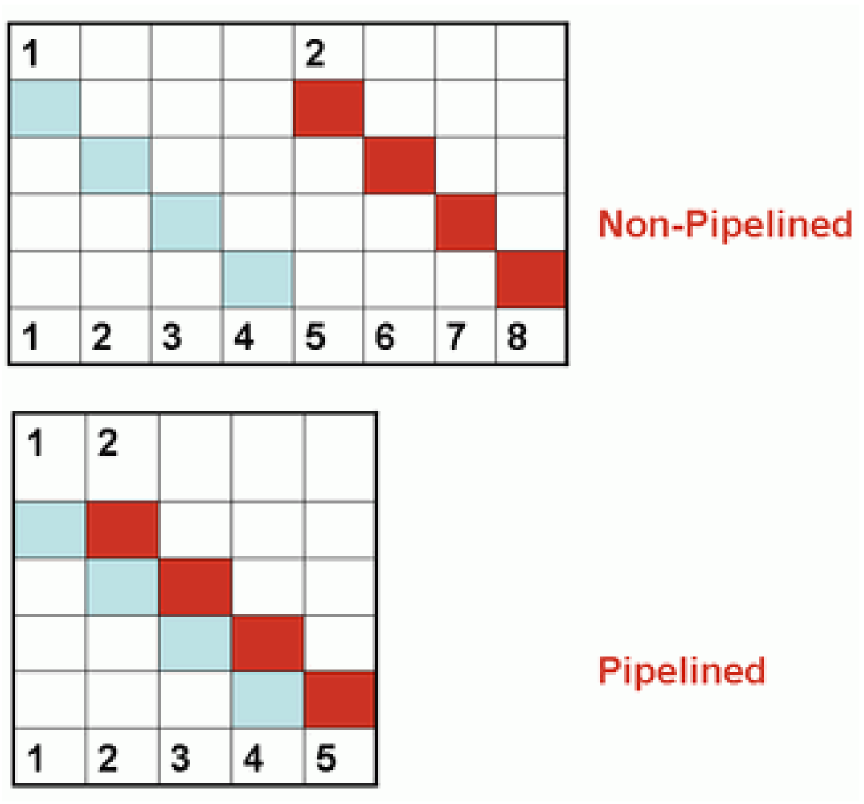


Figure 157: Pipelining technique.

Design

The design of this extension for pipelining based on the original array is illustrated in Figure 158.

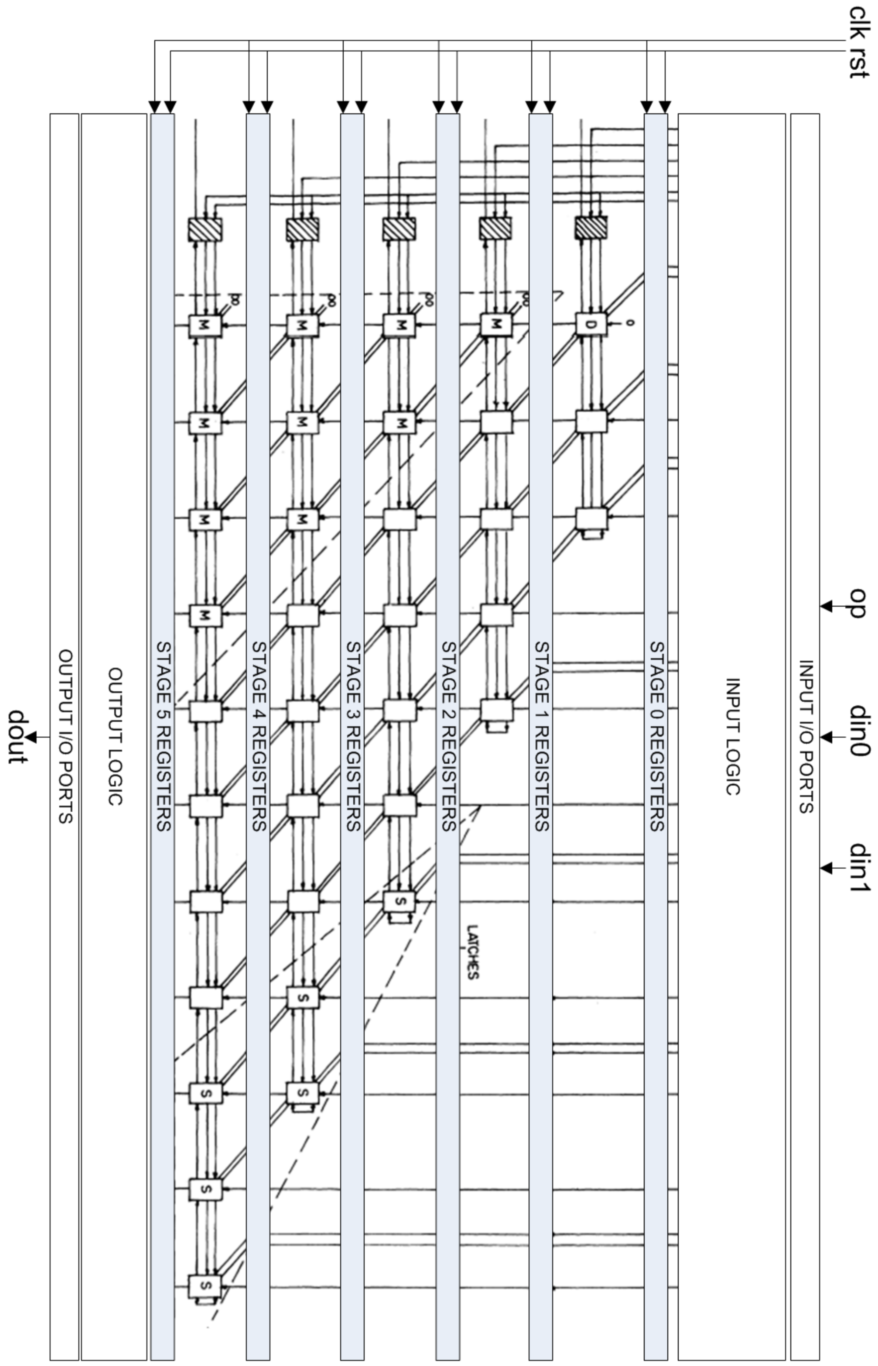


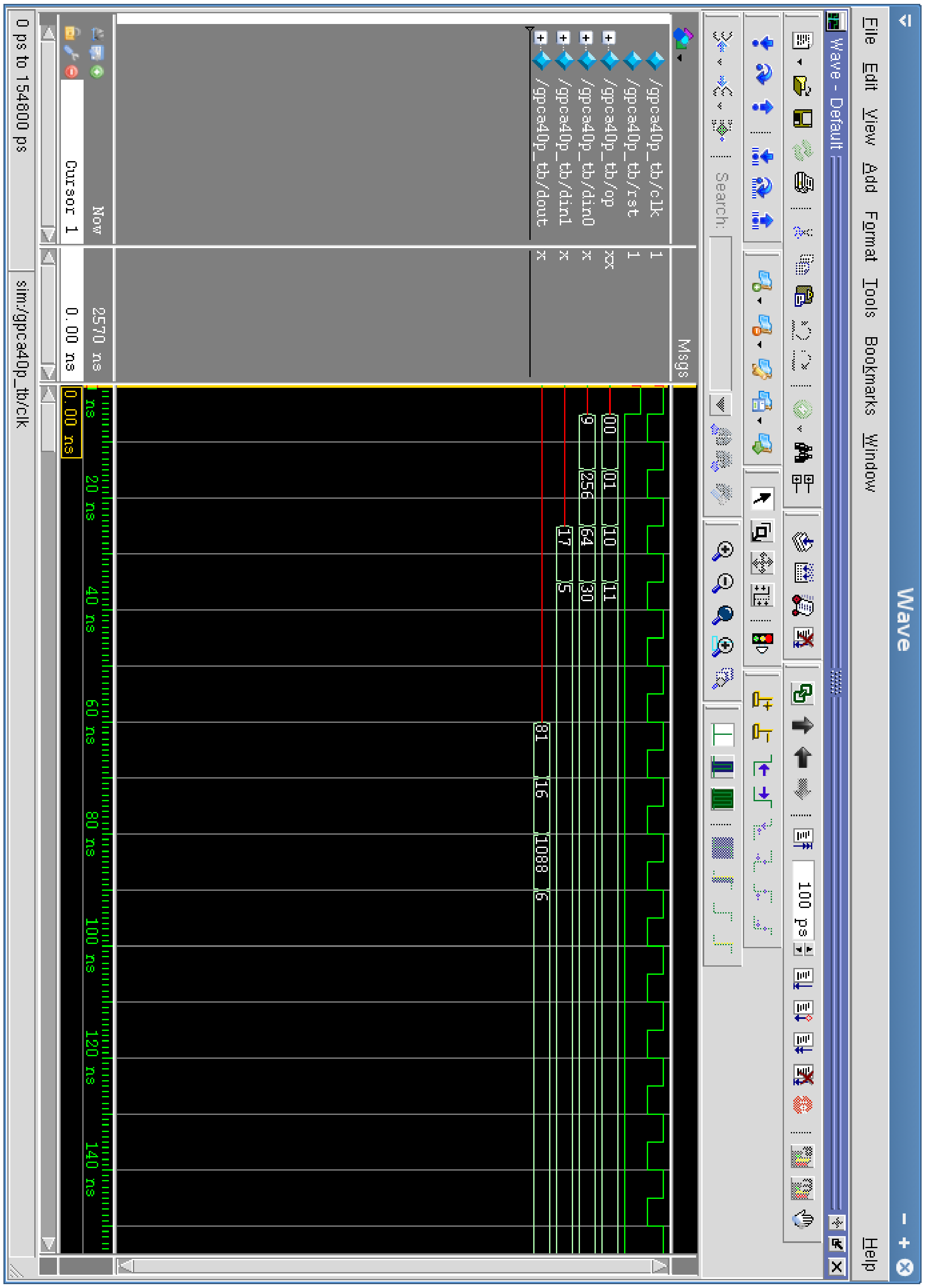
Figure 158: Design extension for pipelining.

The Verilog code in behavior model for this design is listed in the Appendix.

Before actual implementation, the correctness of this design should be verified. To achieve this, the design is simulated and then implemented on FPGA, in the same form as the designs in previous chapters.

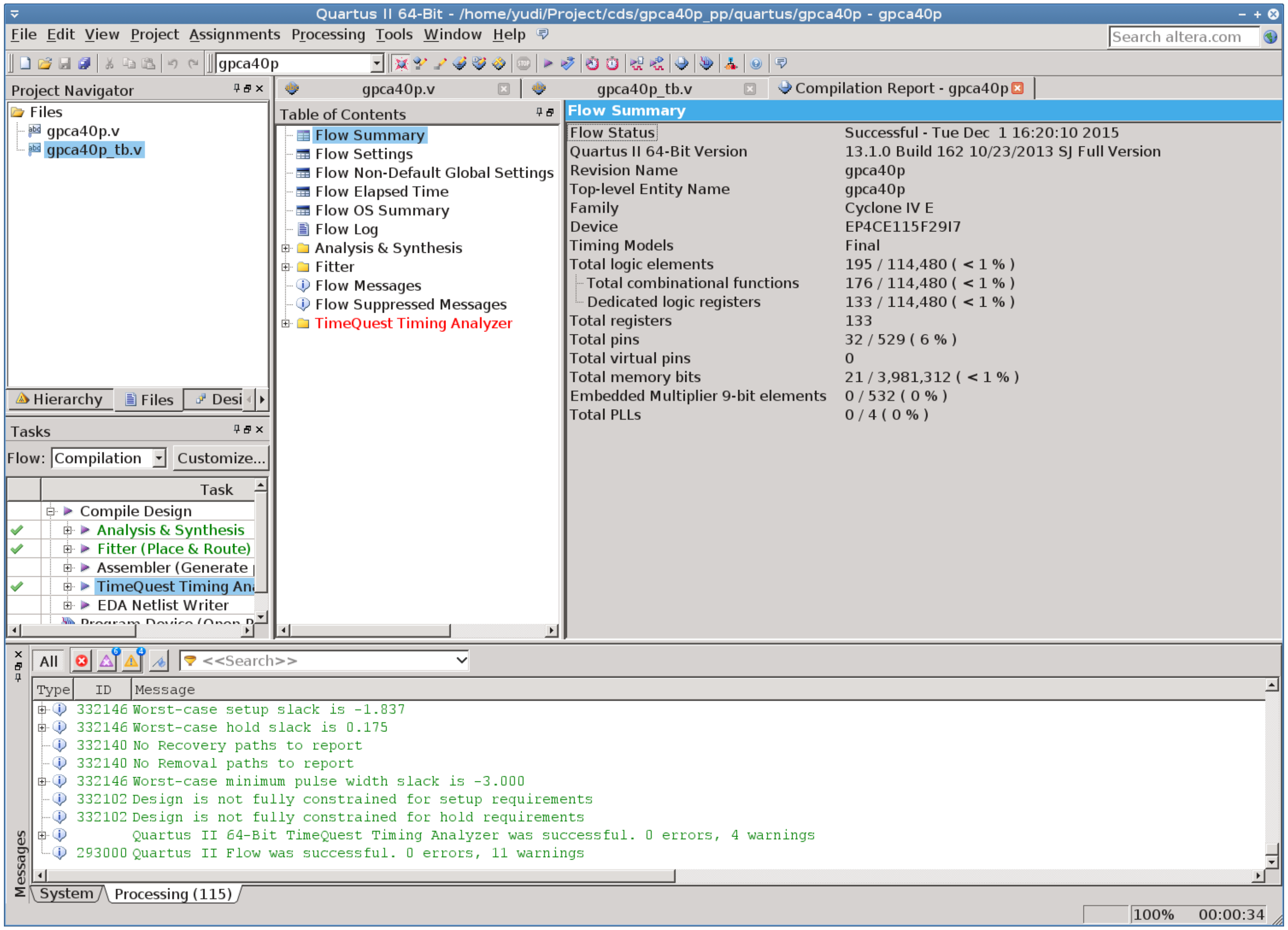
The functional simulation of this circuit is shown in Figure 159. In this simulation, all 4 operations (namely square, square root, multiplication and division) are tested with random data sets. According to the simulation result shown in Figure 159, all the results of calculations done by this circuit are mathematically correct, hence the circuit functions exactly the same as required. Compared to the previous designs, the output is delayed 5 cycles, as there are 5 stages in the pipelined design. In the meantime, since a new operation can be issued into the pipeline on every cycle, with simpler stages producing less delay, the throughput of data processing is achieved.

Result of FPGA synthesis and static timing analysis of maximum frequency on FPGA is shown in Figure 160 and Figure 161 respectively.



```
/gpca40p_tb/clk    1
/gpca40p_tb/rst    1
/gpca40p_tb/op     xx   00  01   10  11
/gpca40p_tb/din0   x    9   256  64  30
/gpca40p_tb/din1   x             17  5
/gpca40p_tb/dout   x    81  16  1088  6

Now       2570 ns
Cursor 1  0.00 ns   0.00 ns

ns   20 ns   40 ns   60 ns   80 ns   100 ns   120 ns   140 ns

0 ps to 154800 ps        sim:/gpca40p_tb/clk
```

Figure 159: Simulation result of the pipeline-extended design.

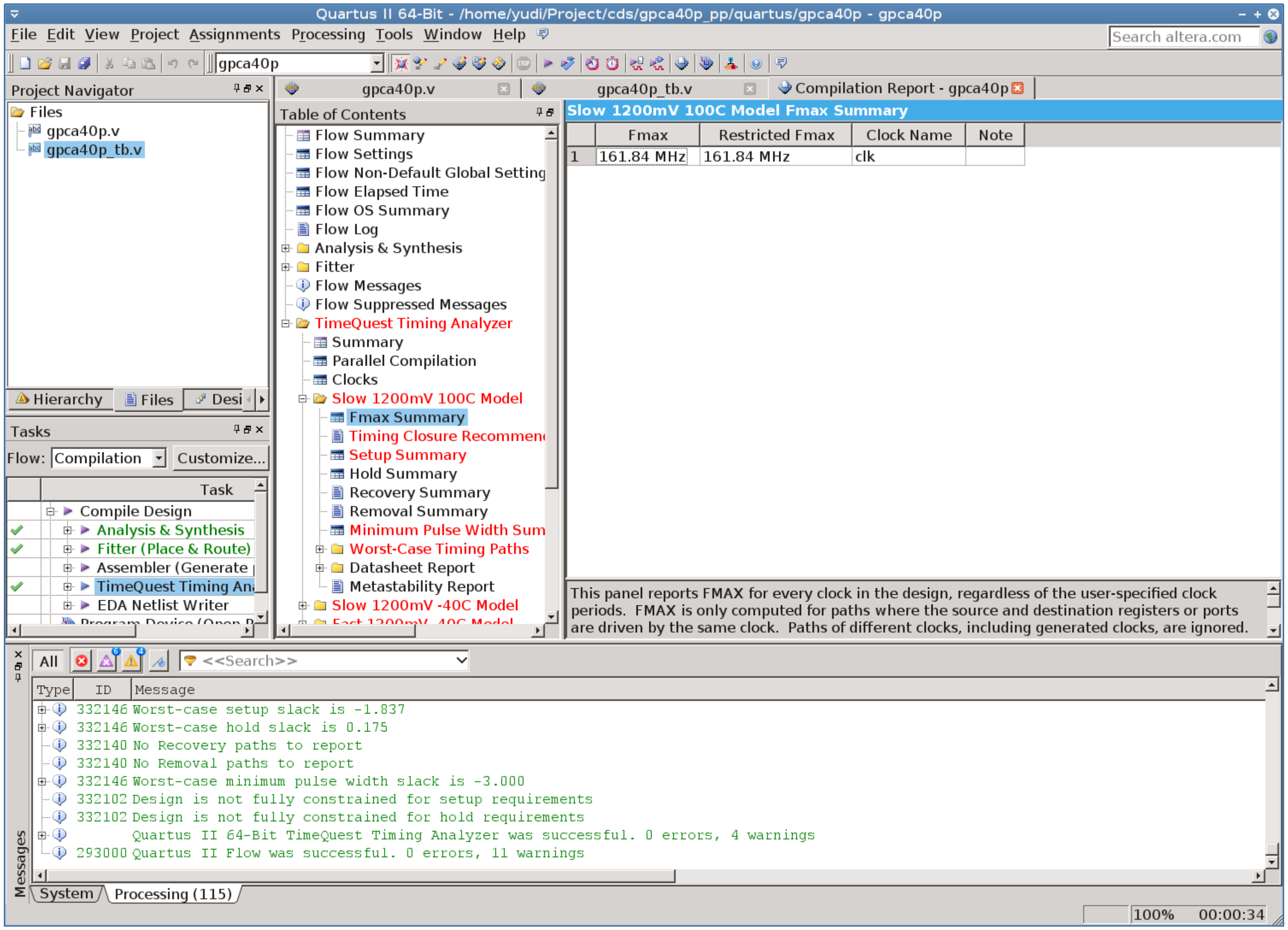


Flow Summary

Flow Status	Successful - Tue Dec 1 16:20:10 2015
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Full Version
Revision Name	gpca40p
Top-level Entity Name	gpca40p
Family	Cyclone IV E
Device	EP4CE115F29I7
Timing Models	Final
Total logic elements	195 / 114,480 (< 1 %)
Total combinational functions	176 / 114,480 (< 1 %)
Dedicated logic registers	133 / 114,480 (< 1 %)
Total registers	133
Total pins	32 / 529 (6 %)
Total virtual pins	0
Total memory bits	21 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

```
Type  ID     Message
      332146 Worst-case setup slack is -1.837
      332146 Worst-case hold slack is 0.175
      332140 No Recovery paths to report
      332140 No Removal paths to report
      332146 Worst-case minimum pulse width slack is -3.000
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
                    Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
      293000 Quartus II Flow was successful. 0 errors, 11 warnings
```

Figure 160: Result of FPGA synthesis.



Slow 1200mV 100C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	161.84 MHz	161.84 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored.

```
Type  ID     Message
      332146 Worst-case setup slack is -1.837
      332146 Worst-case hold slack is 0.175
      332140 No Recovery paths to report
      332140 No Removal paths to report
      332146 Worst-case minimum pulse width slack is -3.000
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
             Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
      293000 Quartus II Flow was successful. 0 errors, 11 warnings
```

Figure 161: Result of FPGA static timing analysis of maximum frequency.

Implementation

In this section, a step-by-step procedure for VLSI digital design is given, for the pipelined implementation of the pipeline array.

There are many fabrication technologies available nowadays, from various fabrication facilities such as GlobalFoundries and TSMC, in technologies such as 14 nm, 28 nm, 40 nm, 65 nm, 0.13 µm and 0.18 µm and so on [30]. Since we use the MOSIS Educational Program (MEP) for free fabrication service, which limits us to use the ON Semi 0.50 µm CMOS (C5N) technology [30], C5N is used in this procedure. However, the general procedure is the same for other technologies. There are many computer aided design (CAD) software available for VLSI design, such as Cadence and Synopsys. This procedure uses Cadence Encounter Digital Implementation Systems 14.00, Cadence Virtuoso Design Environment 6.15 along with NCSU CDK 1.6.0 [30], UofU Technology Library and UofU Standard Cell Library [24].

This procedure consists of 3 parts, each of which represents one major step for the VLSI design. They are logic synthesis, place and route and chip assembly respectively, which are introduced for the rest of this section. In the end, the procedure for MOSIS submission is given.

Logic Synthesis

In digital logic design, logic synthesis is a procedure by which a behavior-level HDL code describing the function of a circuit, is turned into a gate-level netlist which describes the implementation of a design in terms of logic gates, typically using a computer program called a synthesis tool.

In this subsection, a procedure for synthesis is given. For the concision of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [31]. There are other alternative RTL synthesizers available as well, such as Design Compiler by Synopsys. If tools other than what's described here are used, it's advised to refer to their respective manuals. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL Compiler

2) Prerequisites before This Step:

Behavior Verilog Code (e.g. "simple.v")

Tcl Script for RC Compiler ("rc.cmd", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Netlist Verilog Code ("nl.v")

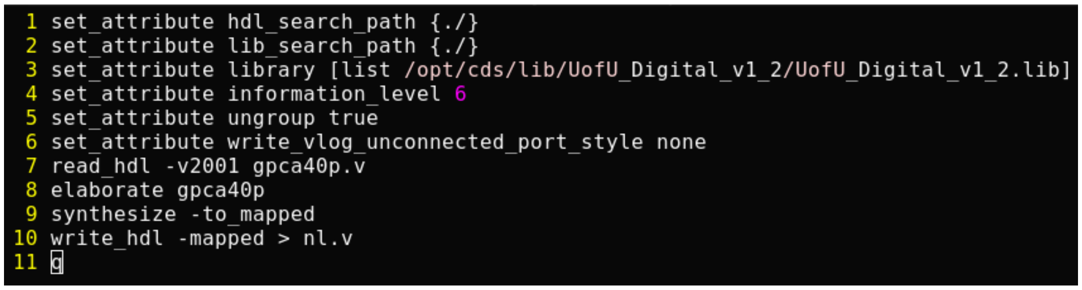
4) Steps:

a) Modify rc.cmd based on the requirement (as shown in Figure 162).

Line 3: Change UofU standard cell library path to where it's installed.

Line 7: Change "gpca40p.v" to the file name of Verilog code (e.g. "simple.v").

Line 8: Change "gpca40p" to the top-level entity name (e.g. "simple").



```
 1 set_attribute hdl_search_path {./}
 2 set_attribute lib_search_path {./}
 3 set_attribute library [list /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib]
 4 set_attribute information_level 6
 5 set_attribute ungroup true
 6 set_attribute write_vlog_unconnected_port_style none
 7 read_hdl -v2001 gpca40p.v
 8 elaborate gpca40p
 9 synthesize -to_mapped
10 write_hdl -mapped > nl.v
11 q
```

Figure 162: rc.cmd.

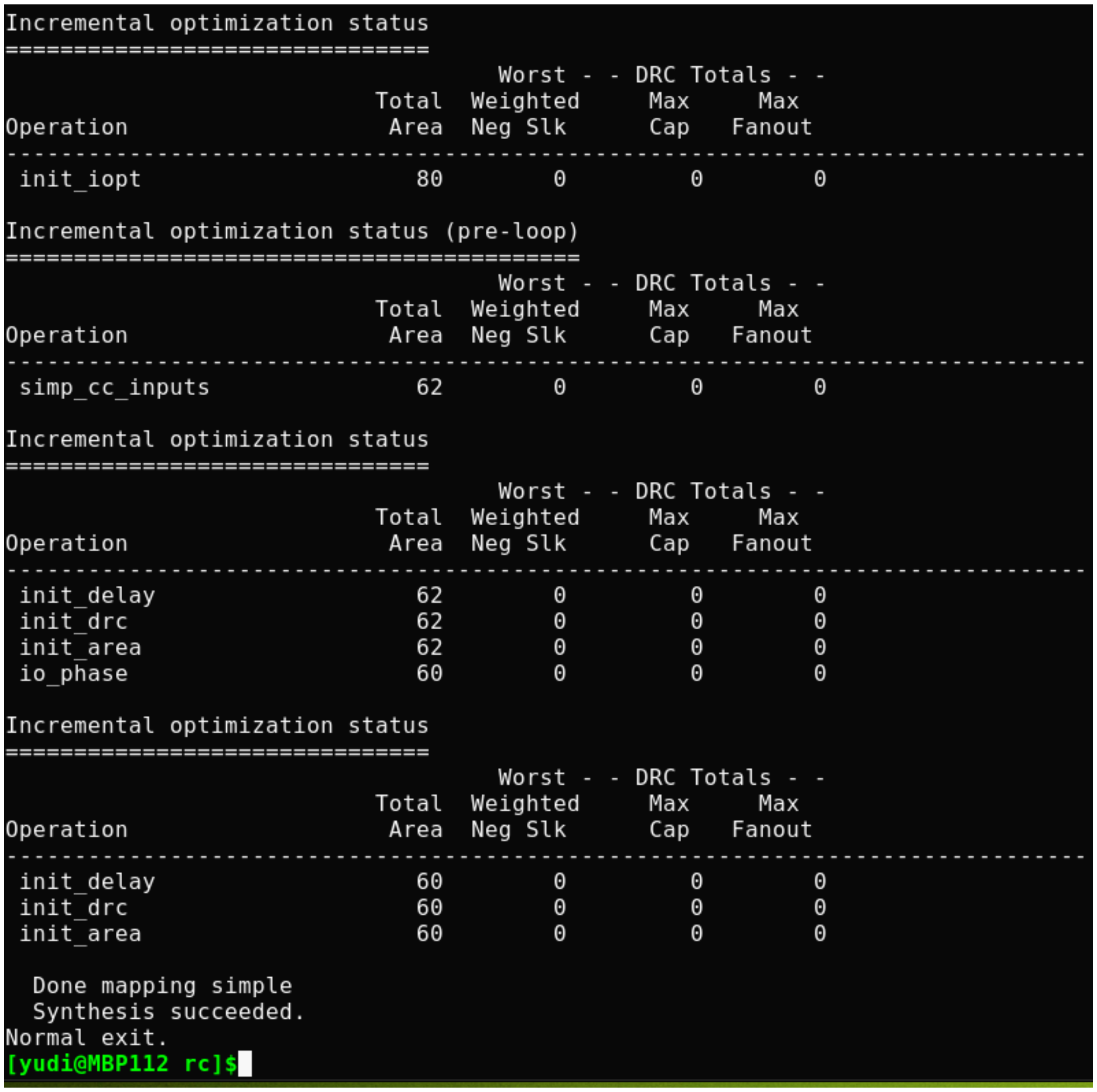
b) Run

$ rc -files rc.cmd

c) Check the result (Figure 163)

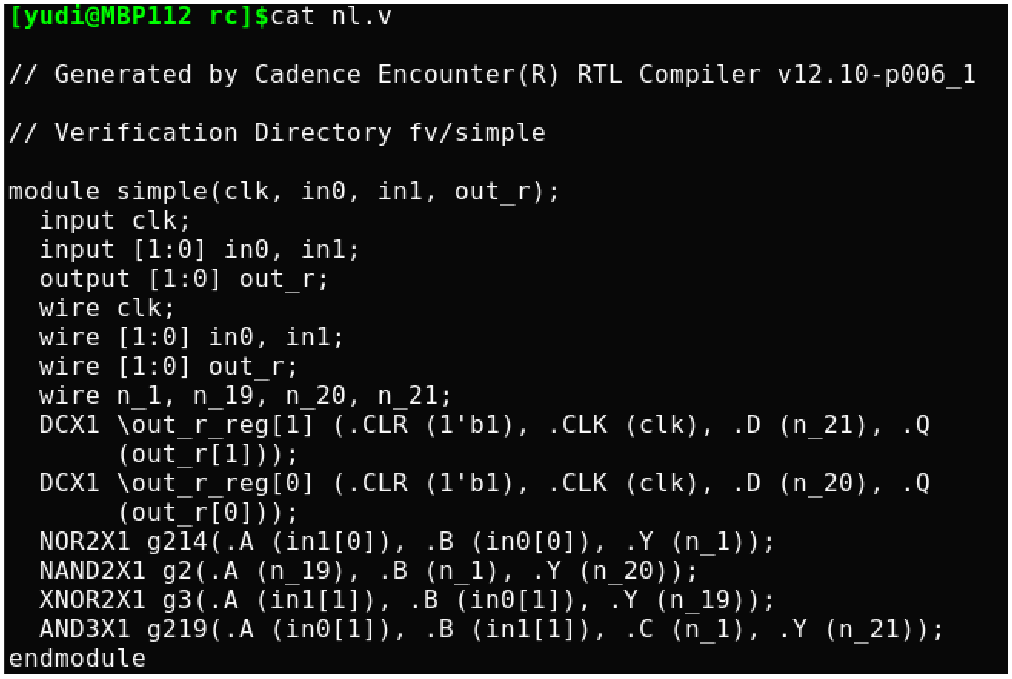
In the end, a netlist file called "nl.v" (Figure 164) is generated containing information which will be used later for place and route.

If everything goes smooth as above, continue to the next step. If anything goes wrong, fix it first before continuing further.



```
Incremental optimization status
===============================
                                    Worst  - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_iopt                    80        0        0        0

Incremental optimization status (pre-loop)
==========================================
                                    Worst  - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 simp_cc_inputs               62        0        0        0

Incremental optimization status
===============================
                                    Worst  - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   62        0        0        0
 init_drc                     62        0        0        0
 init_area                    62        0        0        0
 io_phase                     60        0        0        0

Incremental optimization status
===============================
                                    Worst  - - DRC Totals - -
                           Total  Weighted     Max     Max
Operation                   Area  Neg Slk      Cap   Fanout
-------------------------------------------------------------------------------
 init_delay                   60        0        0        0
 init_drc                     60        0        0        0
 init_area                    60        0        0        0

  Done mapping simple
  Synthesis succeeded.
Normal exit.
[yudi@MBP112 rc]$
```

Figure 163: Synthesis summary generated by RC Compiler.



```
[yudi@MBP112 rc]$cat nl.v

// Generated by Cadence Encounter(R) RTL Compiler v12.10-p006_1

// Verification Directory fv/simple

module simple(clk, in0, in1, out_r);
  input clk;
  input [1:0] in0, in1;
  output [1:0] out_r;
  wire clk;
  wire [1:0] in0, in1;
  wire [1:0] out_r;
  wire n_1, n_19, n_20, n_21;
  DCX1 \out_r_reg[1] (.CLR (1'b1), .CLK (clk), .D (n_21), .Q
       (out_r[1]));
  DCX1 \out_r_reg[0] (.CLR (1'b1), .CLK (clk), .D (n_20), .Q
       (out_r[0]));
  NOR2X1 g214(.A (in1[0]), .B (in0[0]), .Y (n_1));
  NAND2X1 g2(.A (n_19), .B (n_1), .Y (n_20));
  XNOR2X1 g3(.A (in1[1]), .B (in0[1]), .Y (n_19));
  AND3X1 g219(.A (in0[1]), .B (in1[1]), .C (n_1), .Y (n_21));
endmodule
```

Figure 164: Synthesis summary generated by RC Compiler.

Place and Route

"Place and route" is a stage in the process of VLSI design, in which the location to place all the logic elements within a generally limited amount of space and the way of all the wires needed to connect the logic elements are decided.

In this section, a procedure for "place and route" is given. For the brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [33]. If the reader is interested in using GUI commands instead of TCL scripts, please refer to EDI System Menu Reference [34] and textbooks [24] for more information. The detailed procedure and codes used are included in [32].

1) Tools: Cadence Encounter RTL-to-GDSII System

2) Prerequisites before This Step:

Netlist Verilog Code ("nl.v" from the last step)

Tcl Script for RC Compiler ("encounter.cmd", given in Appendix)

Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", given in Appendix)

Synopsys Design Constraints ("typical.sdc", given in Appendix)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

Optimized Netlist Verilog Code ("nlopt.v")

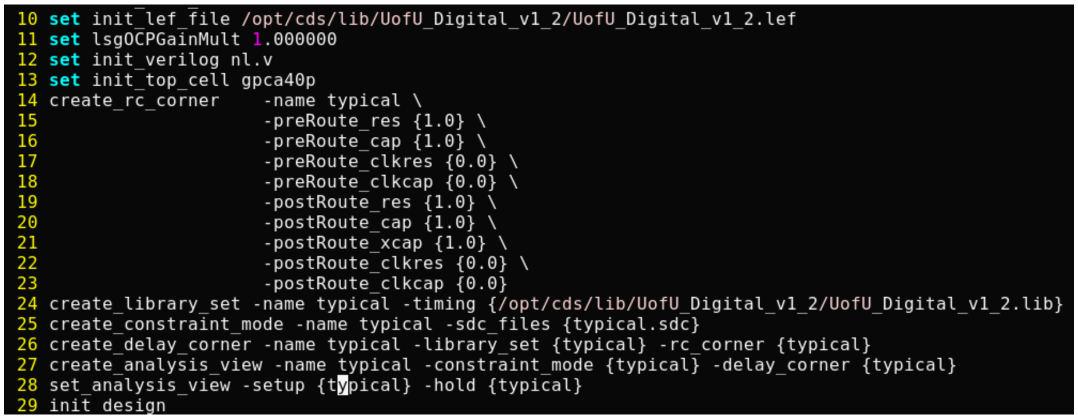
Design Exchange Format (DEF) File (e.g. "simple.def")

4) Steps:

a) Modify Tcl Script for RC Compiler ("encounter.cmd", Figure 165)

Line 10, 24: May change UofU standard cell library path to where it's installed

Line 13, 99: Change "gpca40p" to the top-level entity name (e.g. "simple")

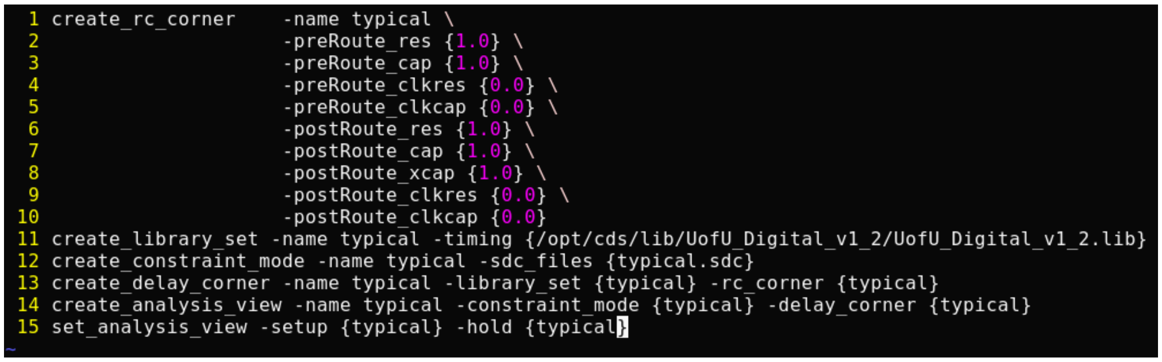


```
10 set init_lef_file /opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lef
11 set lsgOCPGainMult 1.000000
12 set init_verilog nl.v
13 set init_top_cell gpca40p
14 create_rc_corner    -name typical \
15                     -preRoute_res {1.0} \
16                     -preRoute_cap {1.0} \
17                     -preRoute_clkres {0.0} \
18                     -preRoute_clkcap {0.0} \
19                     -postRoute_res {1.0} \
20                     -postRoute_cap {1.0} \
21                     -postRoute_xcap {1.0} \
22                     -postRoute_clkres {0.0} \
23                     -postRoute_clkcap {0.0}
24 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
25 create_constraint_mode -name typical -sdc_files {typical.sdc}
26 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
27 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
28 set_analysis_view -setup {typical} -hold {typical}
29 init_design
```

Figure 165: encounter.cmd.

b) Modify Tcl Script for Multi-Mode Multi-Corner ("mmmc.tcl", Figure 166)

Line 11: May change UofU standard cell library path to where it's installed



```
 1 create_rc_corner    -name typical \
 2                     -preRoute_res {1.0} \
 3                     -preRoute_cap {1.0} \
 4                     -preRoute_clkres {0.0} \
 5                     -preRoute_clkcap {0.0} \
 6                     -postRoute_res {1.0} \
 7                     -postRoute_cap {1.0} \
 8                     -postRoute_xcap {1.0} \
 9                     -postRoute_clkres {0.0} \
10                     -postRoute_clkcap {0.0}
11 create_library_set -name typical -timing {/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_2.lib}
12 create_constraint_mode -name typical -sdc_files {typical.sdc}
13 create_delay_corner -name typical -library_set {typical} -rc_corner {typical}
14 create_analysis_view -name typical -constraint_mode {typical} -delay_corner {typical}
15 set_analysis_view -setup {typical} -hold {typical}
~
```

Figure 166: mmmc.tcl.

c) Run

$ encounter -init encounter.cmd

d) Check the result

It should run all the way to the final step without any errors if the above steps are

followed correctly, as shown in Figure 167 and Figure 168. In the end, a DEF file (e.g. "simple.def") for the chip layout (without pad frame) as well as a netlist file called "nlopt.v" is generated.

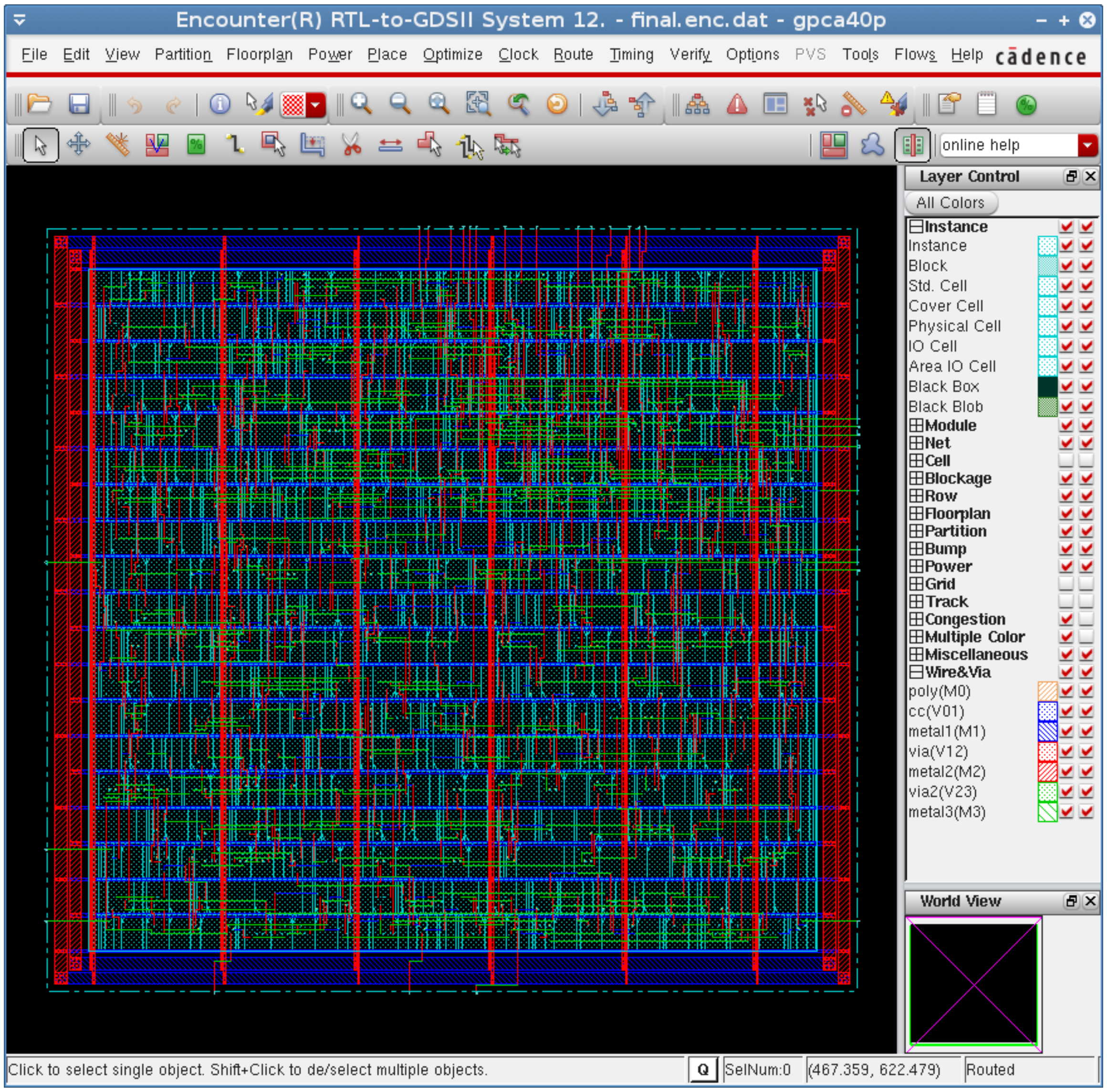
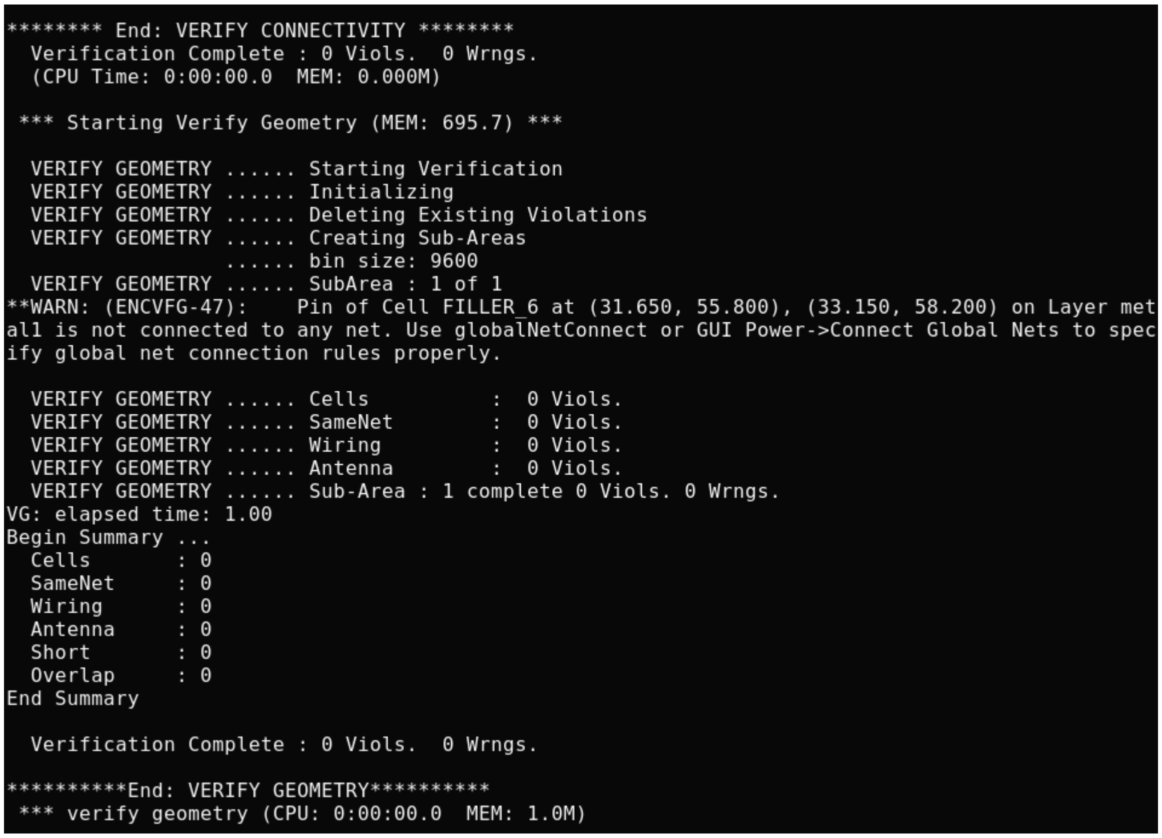


Figure 167: Final view in Encounter.



```
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)

 *** Starting Verify Geometry (MEM: 695.7) ***

  VERIFY GEOMETRY ...... Starting Verification
  VERIFY GEOMETRY ...... Initializing
  VERIFY GEOMETRY ...... Deleting Existing Violations
  VERIFY GEOMETRY ...... Creating Sub-Areas
                  ...... bin size: 9600
  VERIFY GEOMETRY ...... SubArea : 1 of 1
**WARN: (ENCVFG-47):    Pin of Cell FILLER_6 at (31.650, 55.800), (33.150, 58.200) on Layer met
al1 is not connected to any net. Use globalNetConnect or GUI Power->Connect Global Nets to spec
ify global net connection rules properly.

  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: 1.00
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: 0:00:00.0  MEM: 1.0M)
```

Figure 168: DRC report generated by Encounter.

Chip Assembly

As a requirement by AMI05 technology [35], design should be submitted along with pad frame. This section gives the procedure to assemble the pad frame with the chip. For the brevity of this procedure, the exact meanings of commands are not further explained. These commands are covered by the official manuals [36]. Materials such as textbooks [24] also have many useful information for reference. The detailed procedure and codes used are included in [32].

1) Tools:

Cadence Virtuoso Design Environment

2) Prerequisites before this step:

Optimized Netlist Verilog Code ("nlopt.v" from last step)

DEF File (e.g. "simple.def" from last step)

NCSU CDK Library ("ncsu-cdk-1.7.0.beta/")

UofU Technology Library ("UofU_TechLib_ami06/")

UofU Standard Cell Library ("UofU_Digital_v1_2/")

3) Destination Files Generated After This Step:

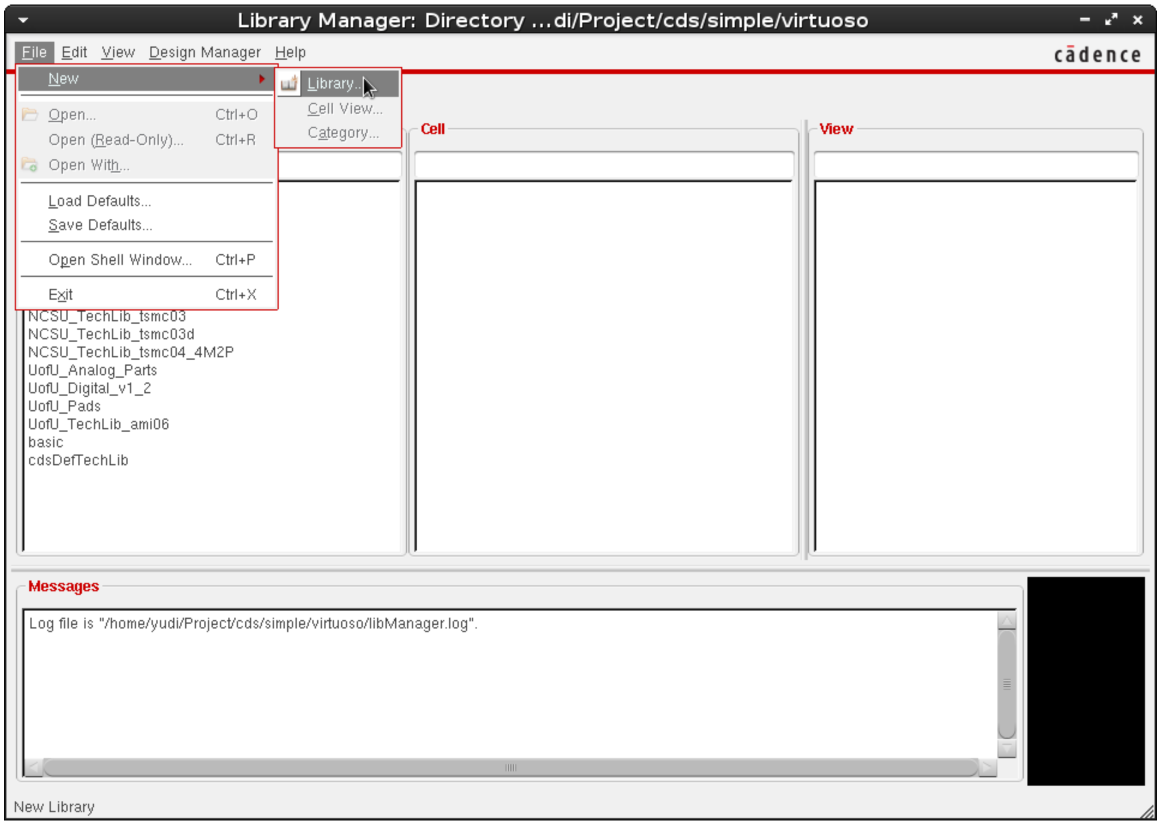
GDSII Stream File (e.g. "simple_final.gds")

4) Steps:

a) Launch Cadence Virtuoso Design Environment

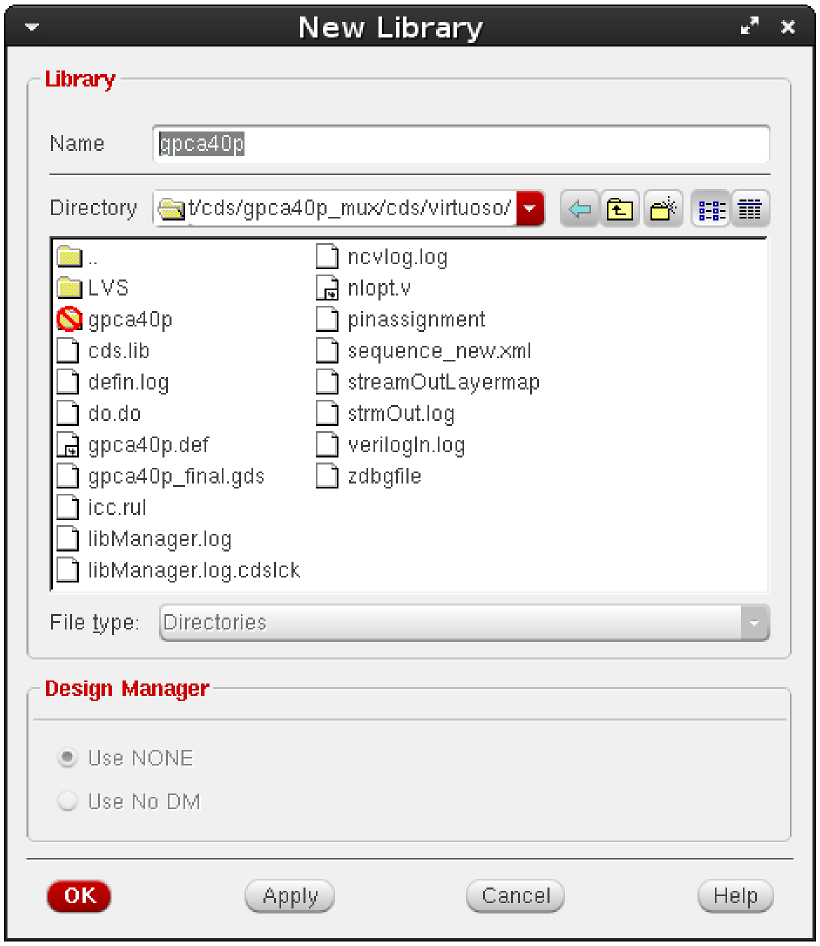
```
$ virtuoso
```

b) Create New Library (Figure 169 to Figure 172)



Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".

Figure 169: Create New Library with Virtuoso.



New Library

Library

Name: gpca40p

Directory: t/cds/gpca40p_mux/cds/virtuoso/

```
..                      ncvlog.log
LVS                     nlopt.v
gpca40p                 pinassignment
cds.lib                 sequence_new.xml
defin.log               streamOutLayermap
do.do                   strmOut.log
gpca40p.def             verilogIn.log
gpca40p_final.gds       zdbgfile
icc.rul
libManager.log
libManager.log.cdslck
```

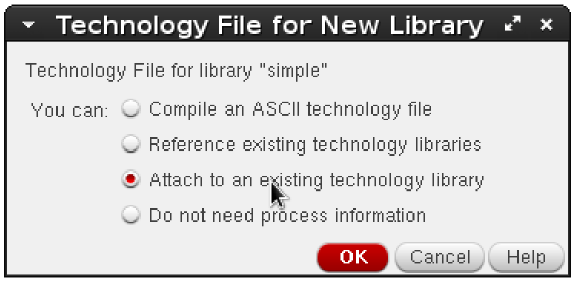
File type: Directories

Design Manager

[✓ Use NONE]

Use No DM

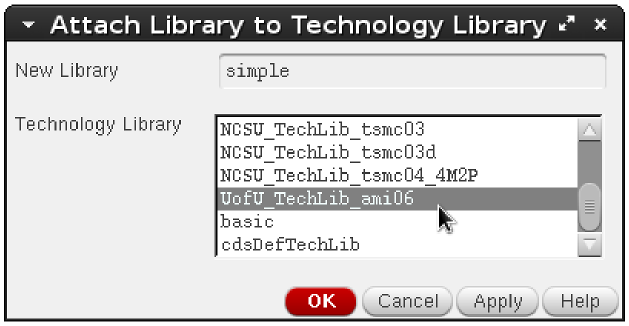
Figure 170: "New Library" window of Virtuoso.


Technology File for New Library

Technology File for library "simple"

You can: Compile an ASCII technology file [ ] / Reference existing technology libraries [ ] / [✓ Attach to an existing technology library] / Do not need process information [ ]

Figure 171: "Technology File for New Library" window of Virtuoso.


Attach Library to Technology Library

New Library: simple

Technology Library:

```
NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_TechLib_ami06
basic
cdsDefTechLib
```

Figure 172: "Technology File for New Library" window of Virtuoso.

c) Import Optimized Netlist Verilog Code ("nlopt.v" from last step) and DEF File (e.g. "simple.def" from last step). As shown in Figure 173 to Figure 177.

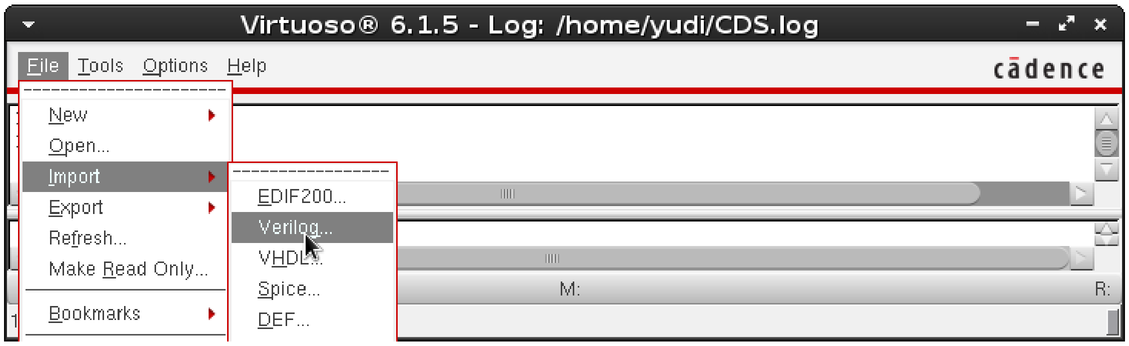
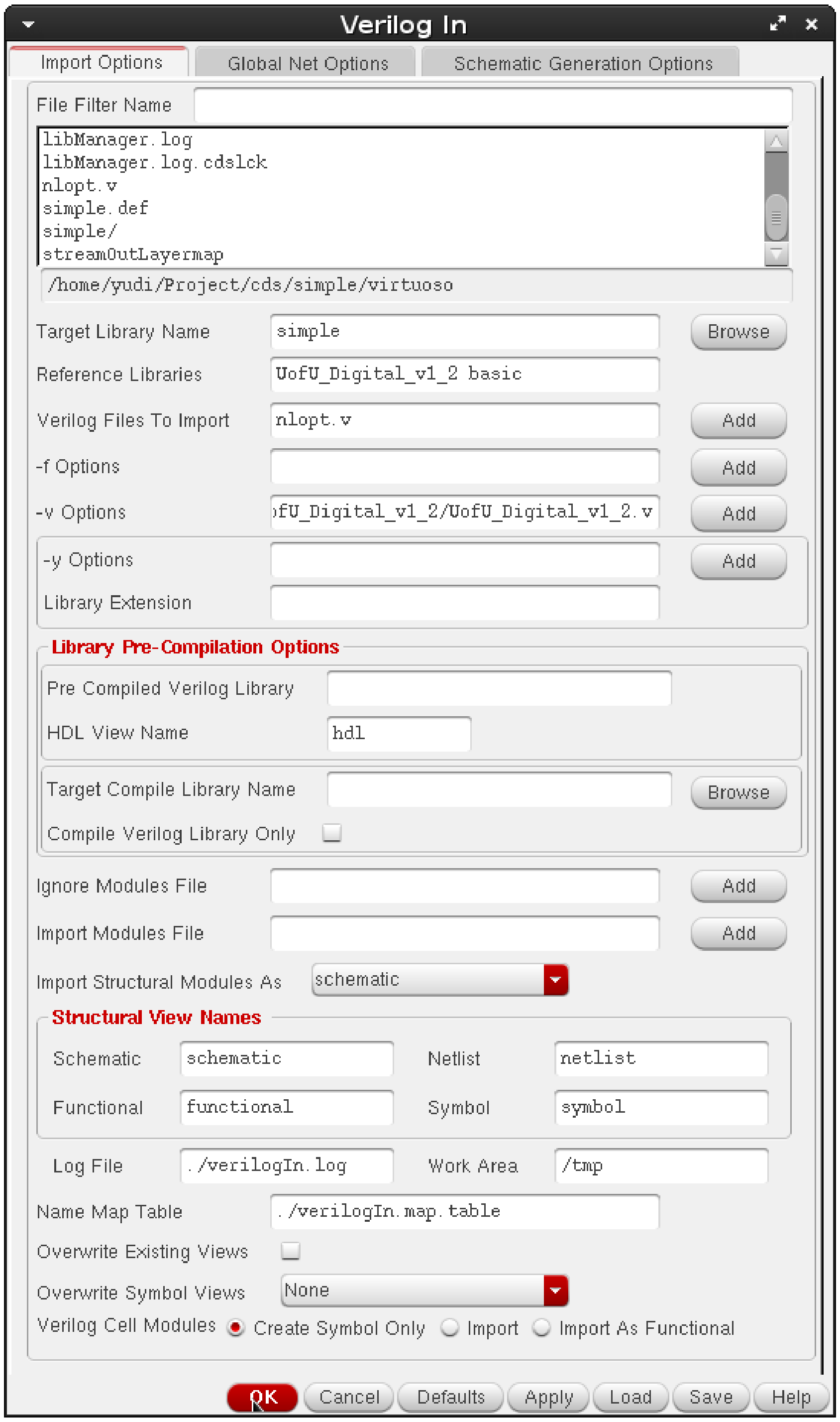


Figure 173: Import netlist with Virtuoso.



File Filter Name

```
libManager.log
libManager.log.cdslck
nlopt.v
simple.def
simple/
streamOutLayermap
```

/home/yudi/Project/cds/simple/virtuoso

Target Library Name	simple
Reference Libraries	UofU_Digital_v1_2 basic
Verilog Files To Import	nlopt.v
-f Options	
-v Options	›fU_Digital_v1_2/UofU_Digital_v1_2.v
-y Options	
Library Extension	

Library Pre-Compilation Options

Pre Compiled Verilog Library	
HDL View Name	hdl
Target Compile Library Name	
Compile Verilog Library Only	☐

Ignore Modules File	
Import Modules File	
Import Structural Modules As	schematic

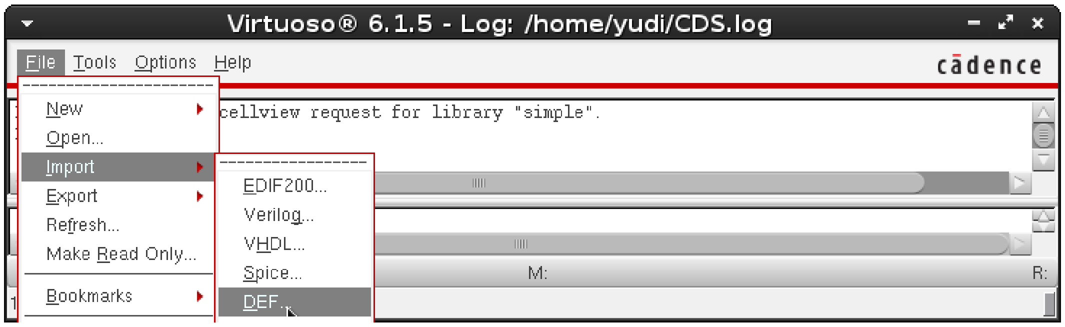
Structural View Names

Schematic	schematic	Netlist	netlist
Functional	functional	Symbol	symbol
Log File	./verilogIn.log	Work Area	/tmp

Name Map Table	./verilogIn.map.table
Overwrite Existing Views	☐
Overwrite Symbol Views	None

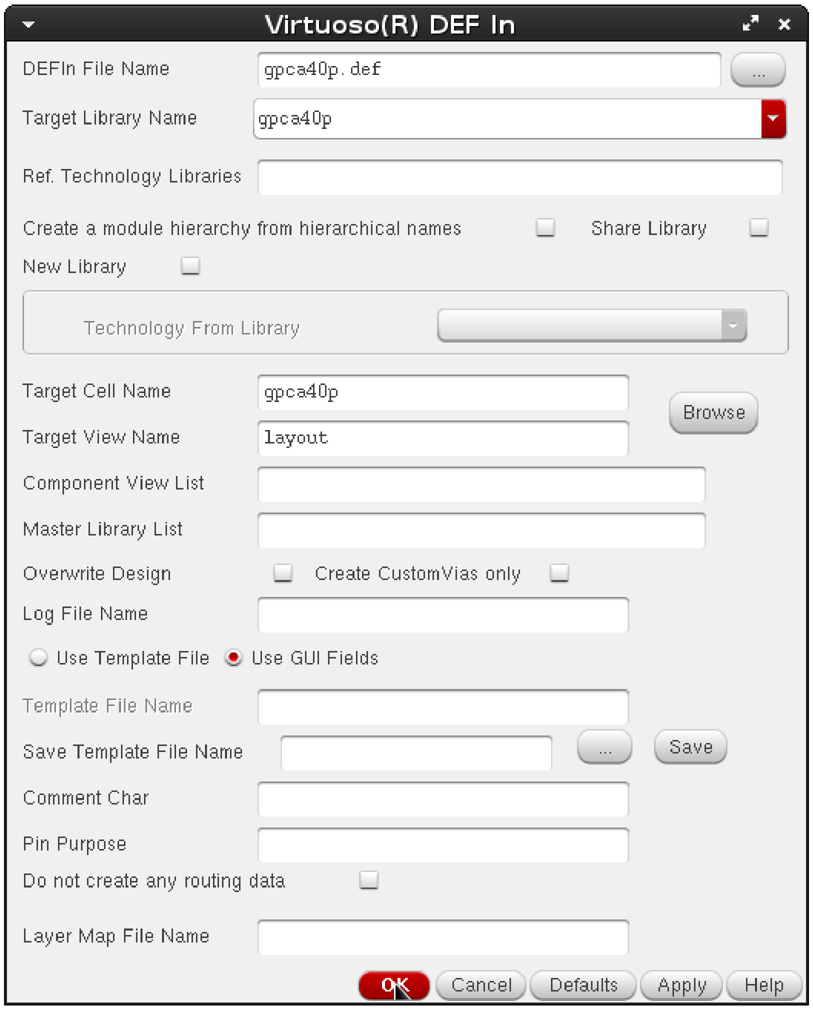
Verilog Cell Modules [✓ Create Symbol Only] Import Import As Functional

Figure 174: "Verilog In" window of Virtuoso.



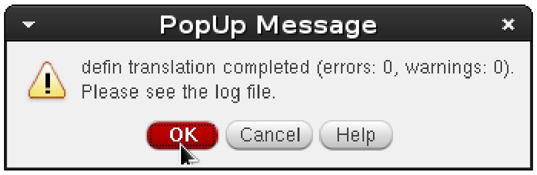
```
cellview request for library "simple".
```

Figure 175: Import DEF with Virtuoso.



DEFIn File Name: gpca40p.def
Target Library Name: gpca40p
Ref. Technology Libraries:
Create a module hierarchy from hierarchical names ☐ Share Library ☐
New Library ☐
Technology From Library
Target Cell Name: gpca40p
Target View Name: layout
Component View List:
Master Library List:
Overwrite Design ☐ Create CustomVias only ☐
Log File Name:
Use Template File [✓ Use GUI Fields]
Template File Name:
Save Template File Name:
Comment Char:
Pin Purpose:
Do not create any routing data ☐
Layer Map File Name:

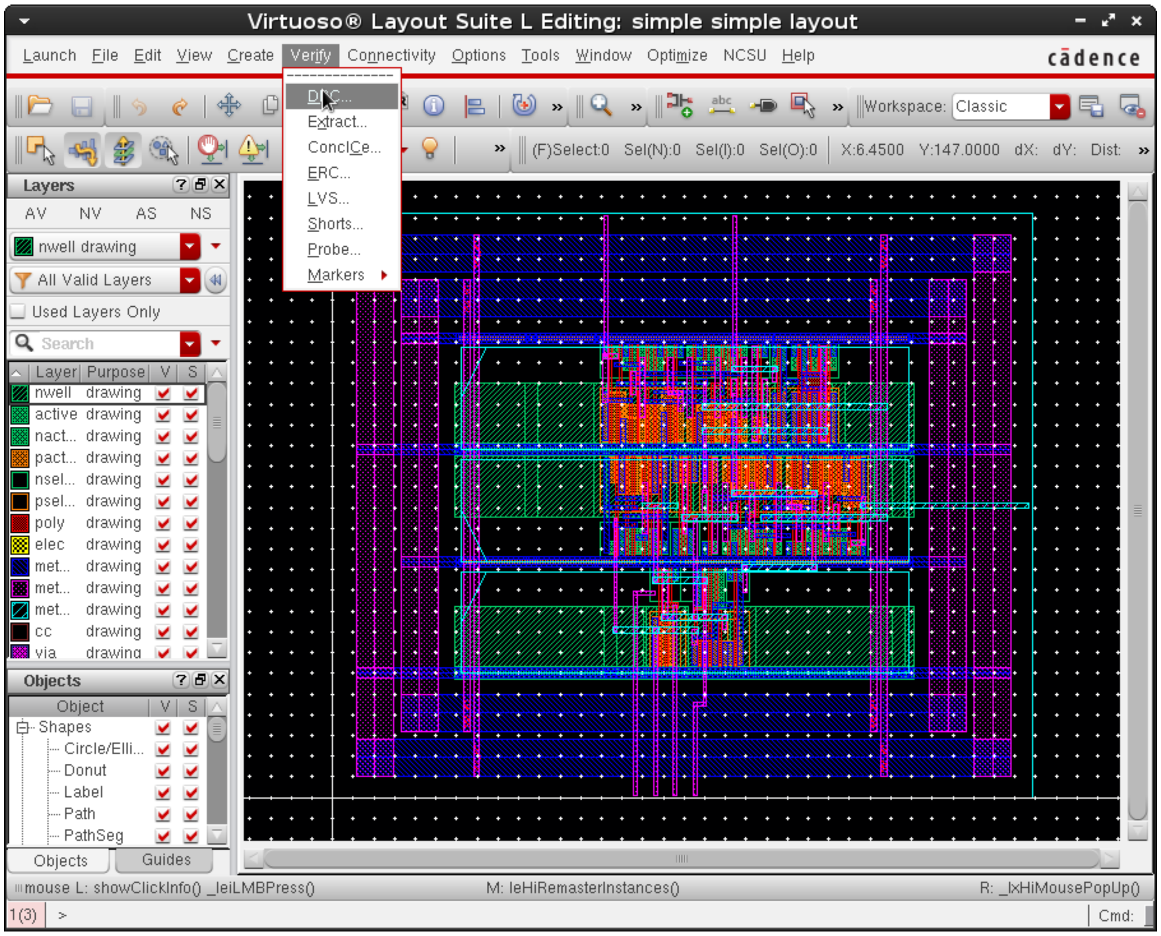
Figure 176: "DEF In" window of Virtuoso.



defin translation completed (errors: 0, warnings: 0).
Please see the log file.

Figure 177: "DEF In" successful translation prompt.

Once both layouts (from DEF) and schematic (from netlist in this section) are generated, DRC, Extract and LVS should be performed, as shown in Figure 178 to 193.


Virtuoso® Layout Suite L Editing: simple simple layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize NCSU Help
cādence
DRC...
Extract...
ConclCe...
ERC...
LVS...
Shorts...
Probe...
Markers
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:6.4500 Y:147.0000 dX: dY: Dist:
Layers
AV NV AS NS
nwell drawing
All Valid Layers
Used Layers Only
Search
Objects
Shapes
Circle/Elli...
Donut
Label
Path
PathSeg
Objects Guides
mouse L: showClickInfo() _leiLMBPress()
M: leHiRemasterInstances()
R: _lxHiMousePopUp()
1(3) >
Cmd:

Layer	Purpose	V	S
nwell	drawing	✓	✓
active	drawing	✓	✓
nact...	drawing	✓	✓
pact...	drawing	✓	✓
nsel...	drawing	✓	✓
psel...	drawing	✓	✓
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓
via	drawing	✓	✓

Figure 178: Start DRC from Virtuoso Layout Suite.

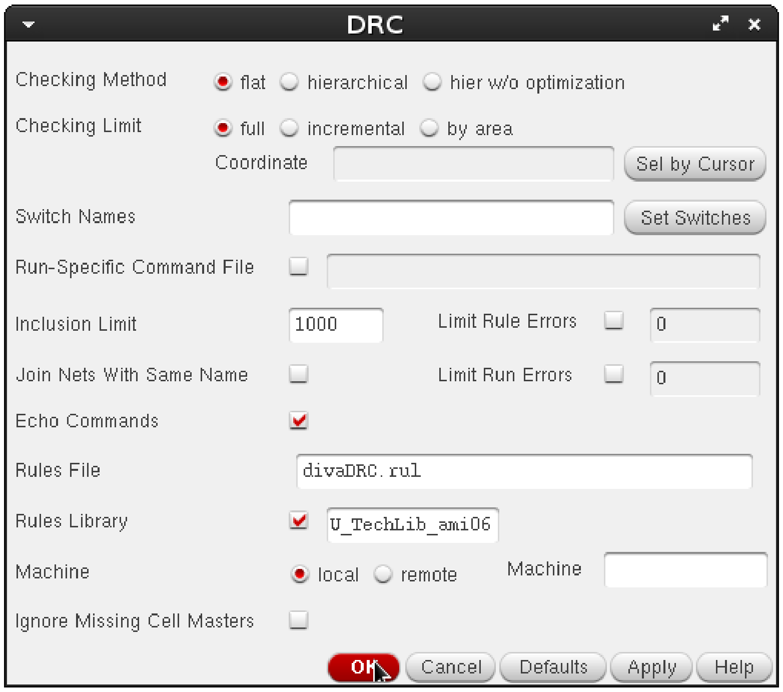
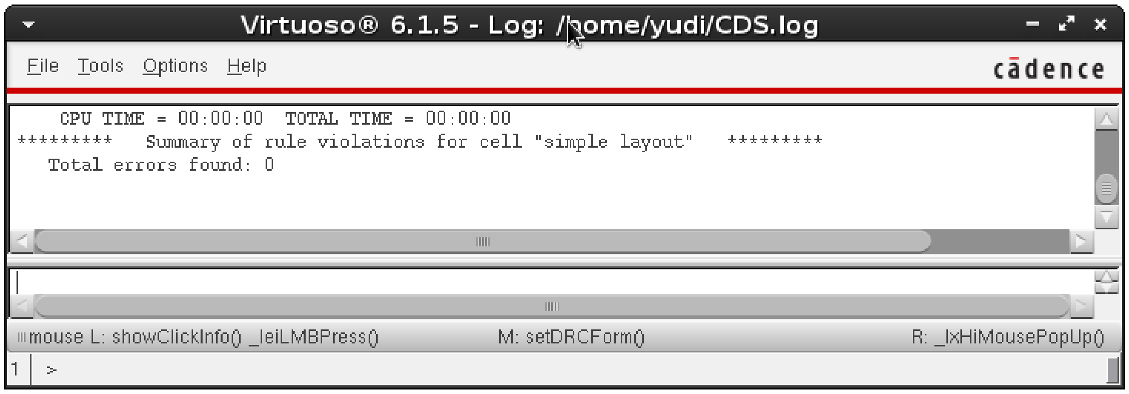


Figure 179: "DRC" window of Virtuoso.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File Tools Options Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: setDRCForm()
R: _lxHiMousePopUp()
1 >

```
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violations for cell "simple layout"   *********
   Total errors found: 0
```

Figure 180: Report of successful DRC.

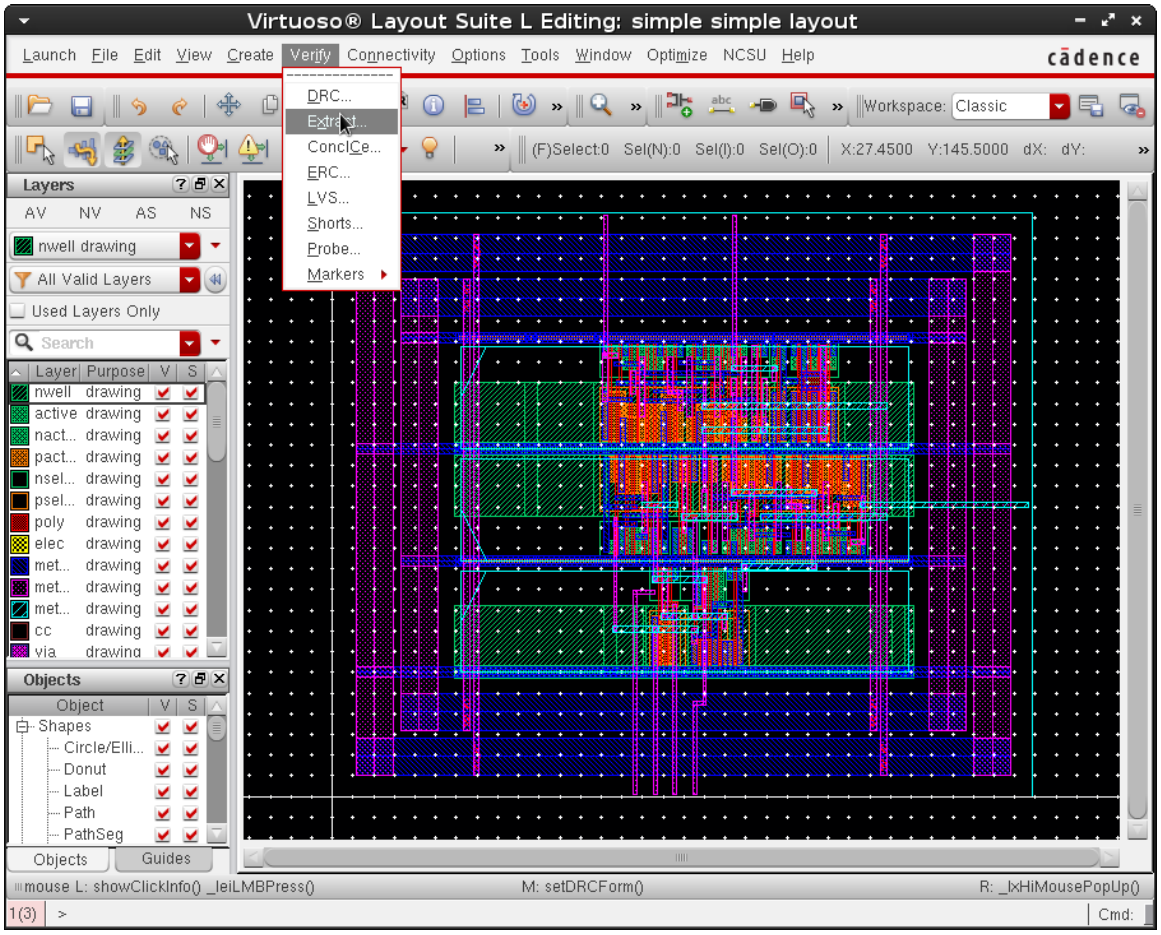
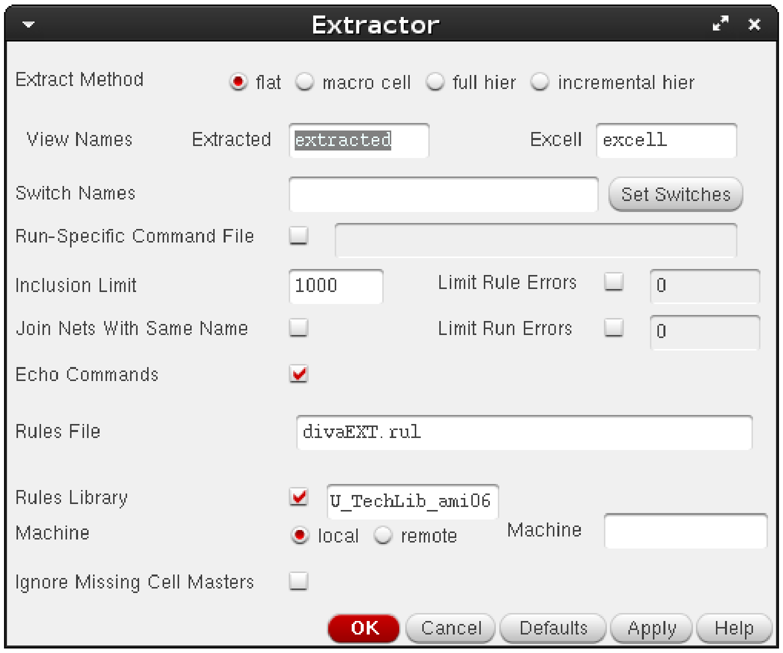


Figure 181: Start Extract from Virtuoso Layout Suite.



Extract Method [✓ flat] macro cell full hier incremental hier

View Names Extracted extracted Excell excell

Switch Names

Run-Specific Command File ☐

Inclusion Limit 1000 Limit Rule Errors ☐ 0

Join Nets With Same Name ☐ Limit Run Errors ☐ 0

Echo Commands ✓

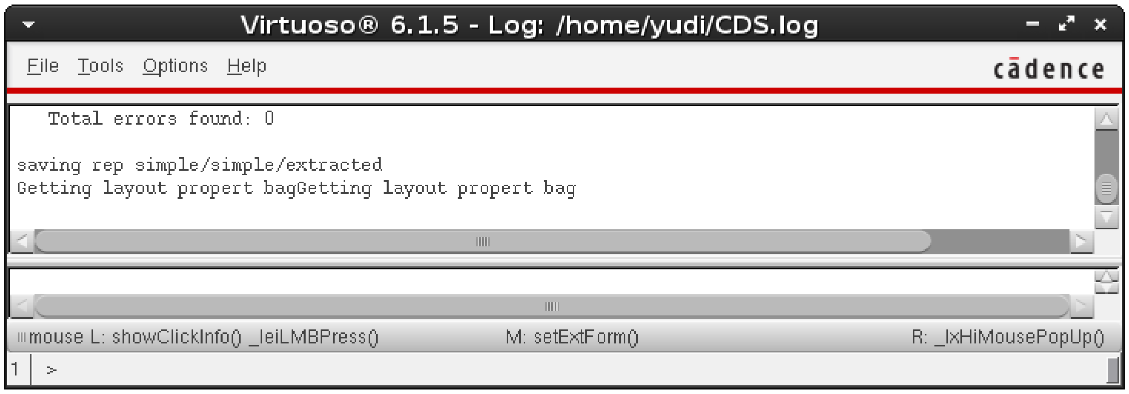
Rules File divaEXT.rul

Rules Library ✓ U_TechLib_ami06

Machine [✓ local] remote Machine

Ignore Missing Cell Masters ☐

Figure 182: "Extractor" window of Virtuoso.



```
   Total errors found: 0

saving rep simple/simple/extracted
Getting layout propert bagGetting layout propert bag
```

mouse L: showClickInfo() _leiLMBPress() M: setExtForm() R: _lxHiMousePopUp()

1 >

Figure 183: Report of successful Extract.

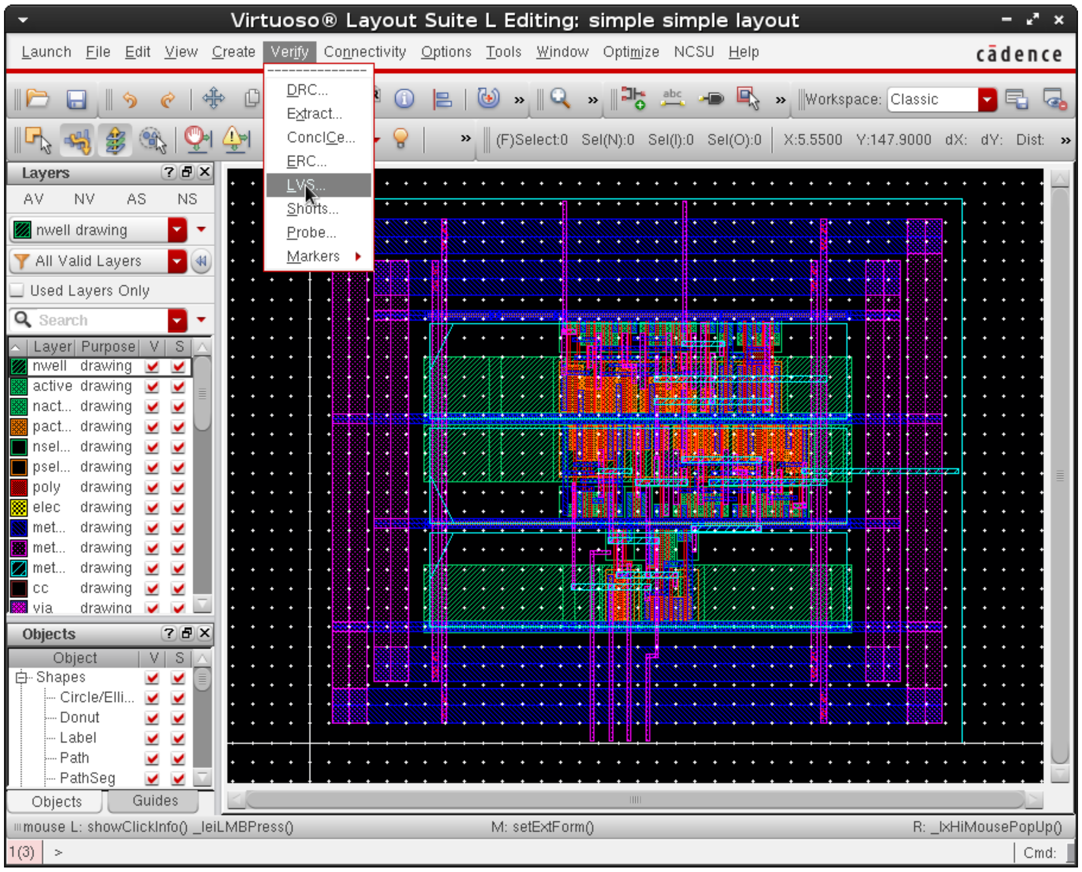
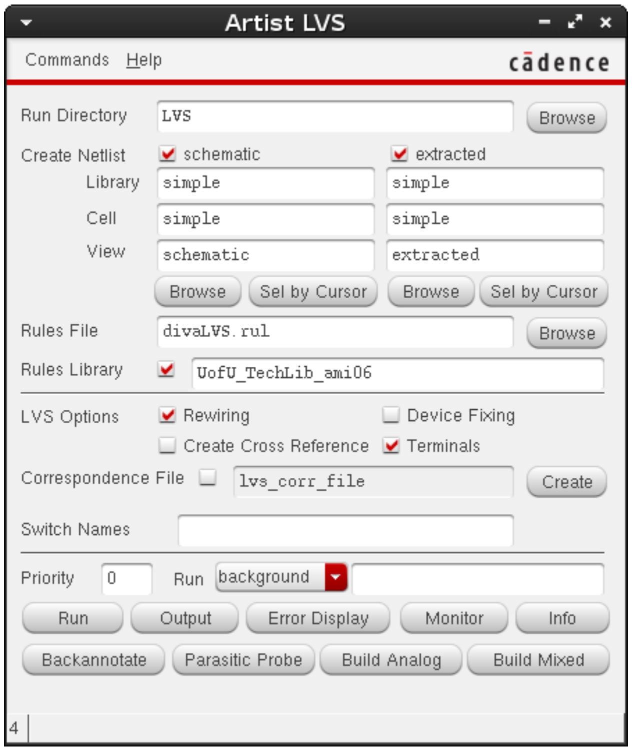


Figure 184: Start LVS from Virtuoso Layout Suite.



Run Directory: LVS

Create Netlist: [✓ schematic] [✓ extracted]

	schematic	extracted
Library	simple	simple
Cell	simple	simple
View	schematic	extracted

Rules File: divaLVS.rul

Rules Library: [✓] UofU_TechLib_ami06

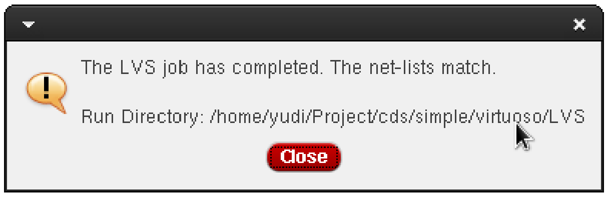
LVS Options: [✓ Rewiring] Device Fixing Create Cross Reference [✓ Terminals]

Correspondence File: lvs_corr_file

Switch Names:

Priority: 0 Run: background

Figure 185: "Artist LVS" window of Virtuoso.

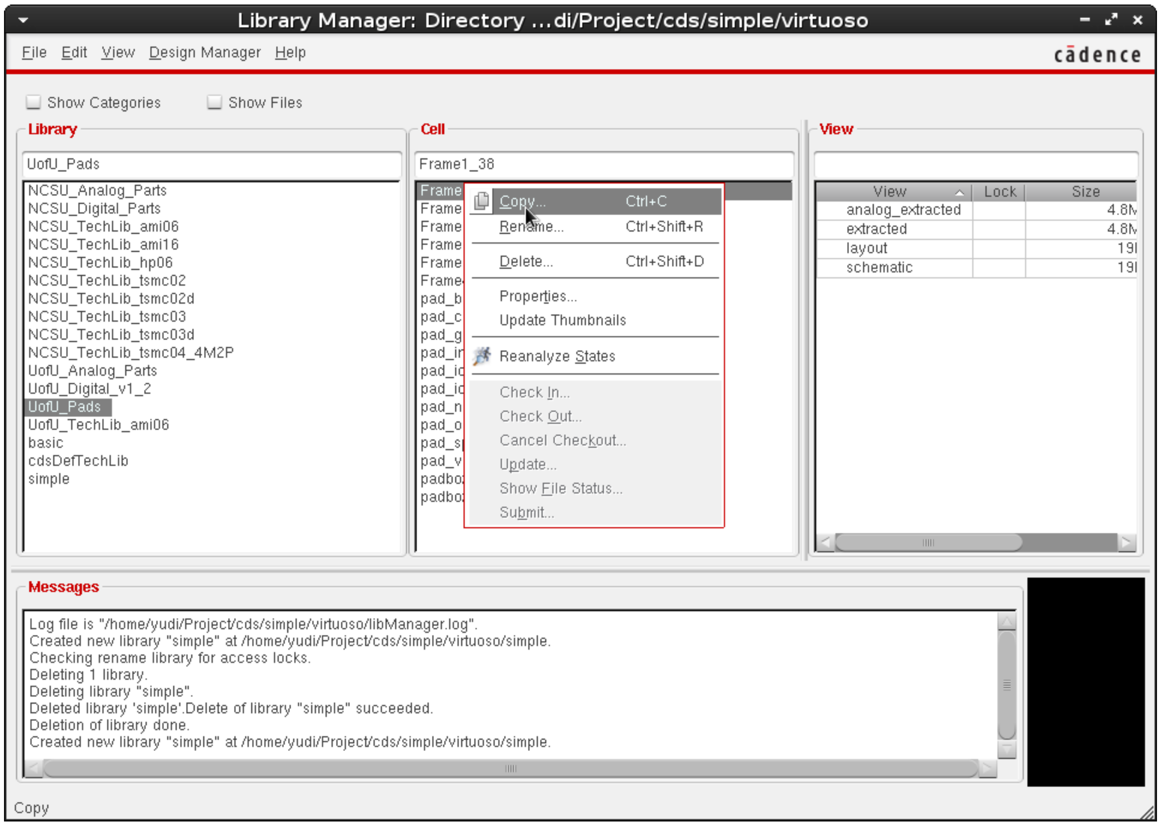


The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

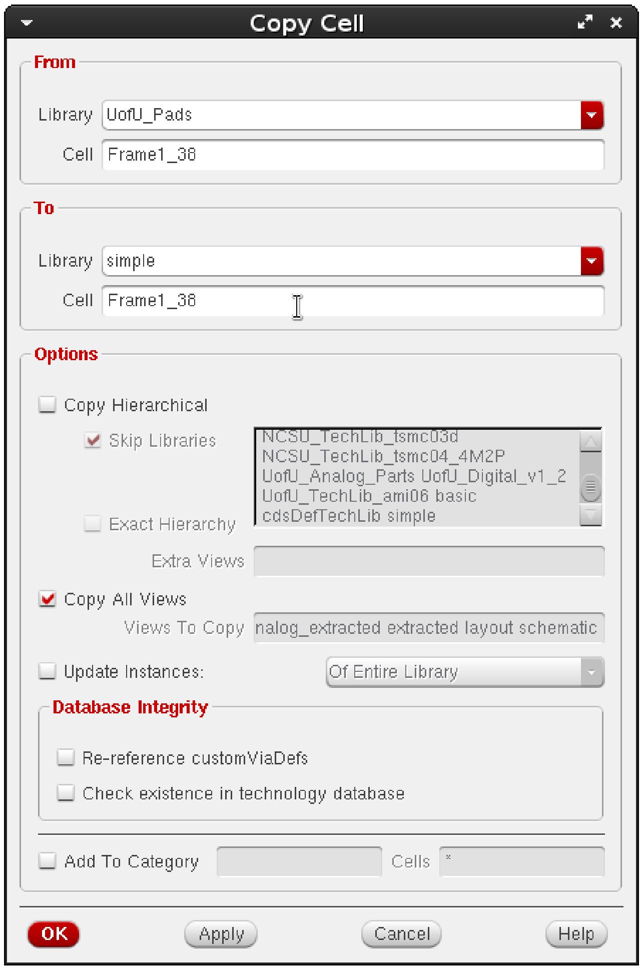
Figure 186: "Artist LVS" successful LVS prompt.

d) Customize Pad Frame (Figure 182 to 202)



```
Log file is "/home/yudi/Project/cds/simple/virtuoso/libManager.log".
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
Checking rename library for access locks.
Deleting 1 library.
Deleting library "simple".
Deleted library 'simple'.Delete of library "simple" succeeded.
Deletion of library done.
Created new library "simple" at /home/yudi/Project/cds/simple/virtuoso/simple.
```

Figure 187: Copy pad frame in Virtuoso.



Library UofU_Pads
Cell Frame1_38

Library simple
Cell Frame1_38

Copy Hierarchical
Skip Libraries [✓]
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts UofU_Digital_v1_2
UofU_TechLib_ami06 basic
cdsDefTechLib simple
Exact Hierarchy
Extra Views
Copy All Views [✓]
Views To Copy nalog_extracted extracted layout schematic
Update Instances: Of Entire Library

Re-reference customViaDefs
Check existence in technology database

Add To Category Cells *

Figure 188: "Copy Cell" window of Virtuoso.

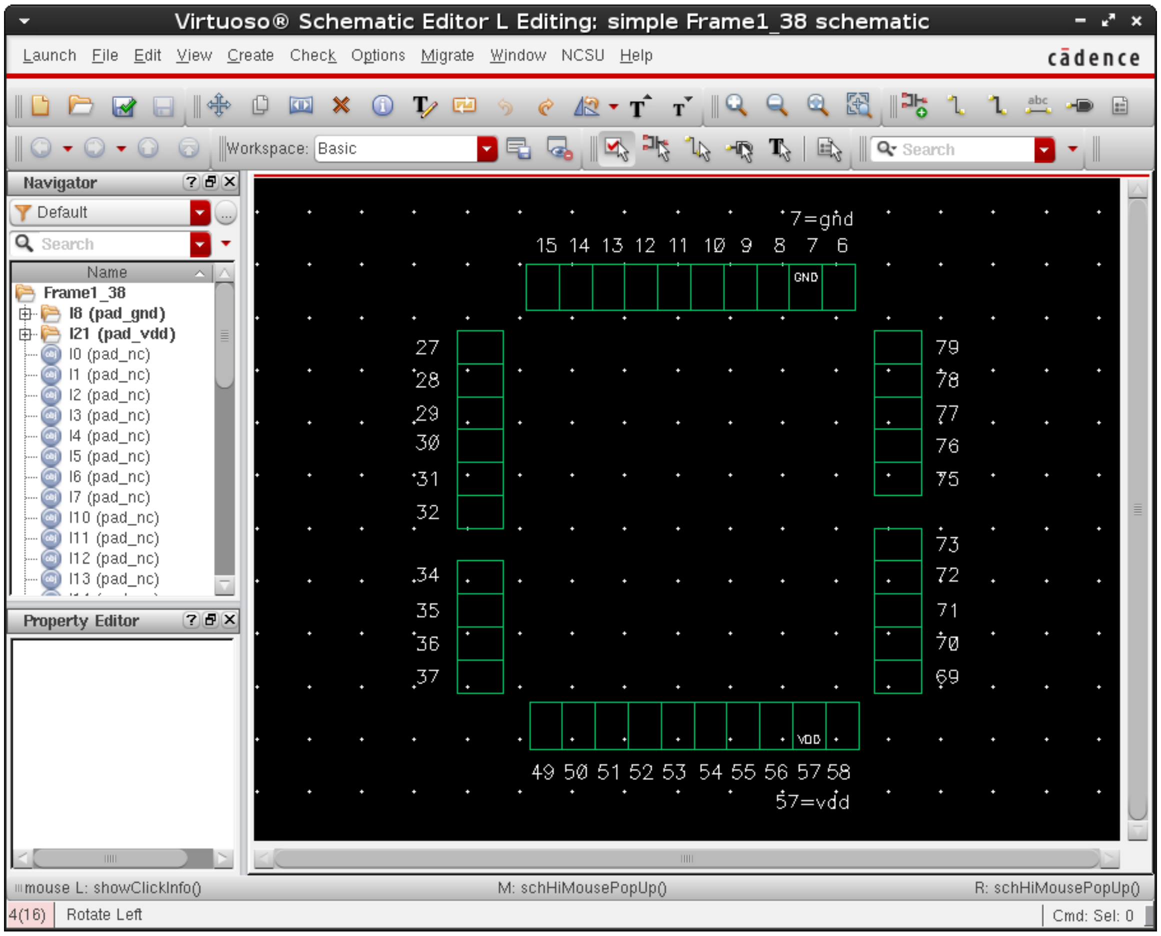
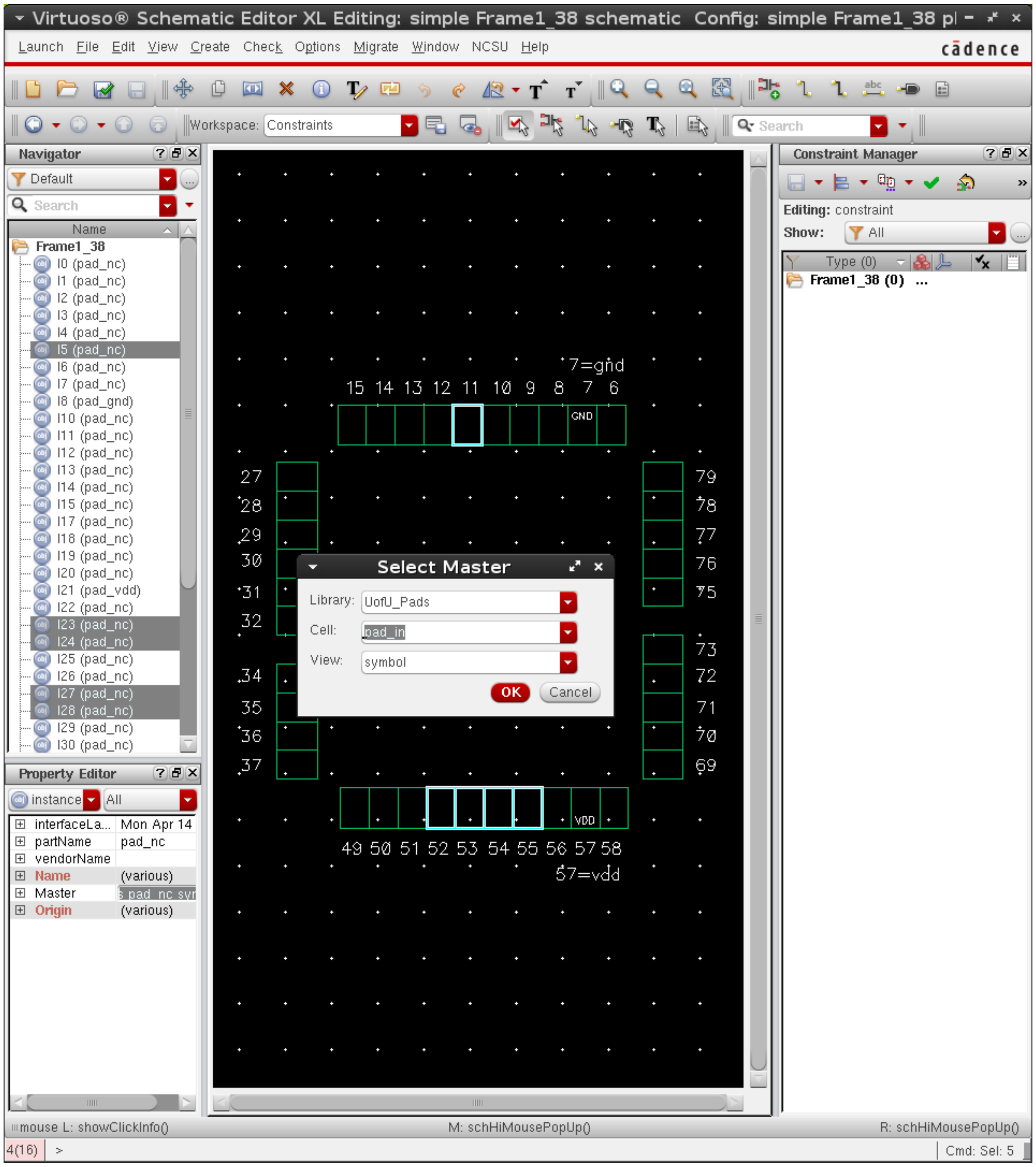


Figure 189: Initial schematic view of the pad frame.

Modify the used pads in the pad frame (both schematic and layout), from "pad_nc" to "pad_in" or "pad_out".



Workspace: Constraints

Navigator: Default

Name
Frame1_38
I0 (pad_nc)
I1 (pad_nc)
I2 (pad_nc)
I3 (pad_nc)
I4 (pad_nc)
I5 (pad_nc)
I6 (pad_nc)
I7 (pad_nc)
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)
I18 (pad_nc)
I19 (pad_nc)
I20 (pad_nc)
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_nc)
I24 (pad_nc)
I25 (pad_nc)
I26 (pad_nc)
I27 (pad_nc)
I28 (pad_nc)
I29 (pad_nc)
I30 (pad_nc)

Property Editor: instance, All

interfaceLa...	Mon Apr 14
partName	pad_nc
vendorName	
Name	(various)
Master	s pad_nc syr
Origin	(various)

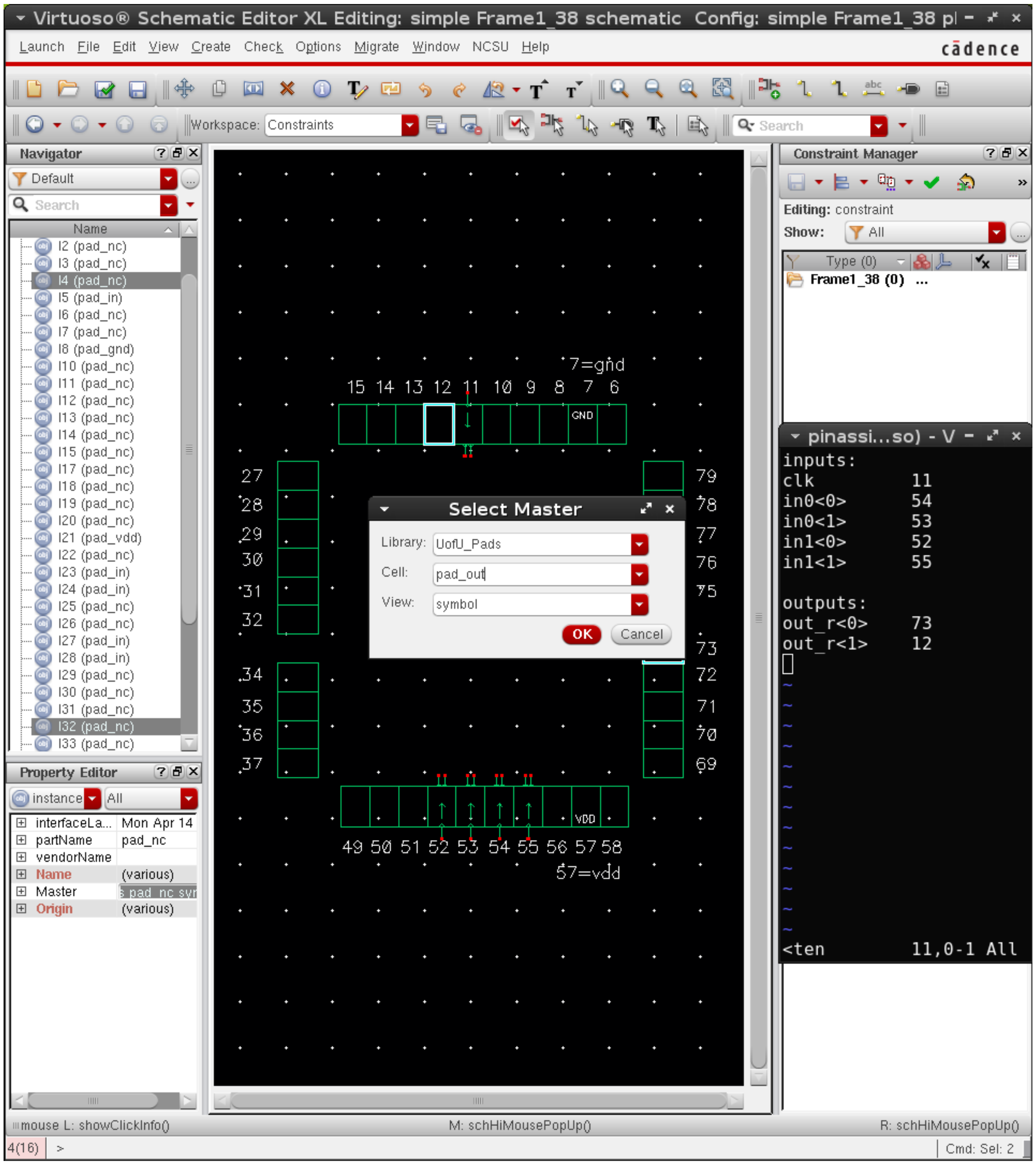
Select Master

Library: UofU_Pads
Cell: pad_in
View: symbol
OK Cancel

Constraint Manager — Editing: constraint; Show: All; Type (0); Frame1_38 (0) ...

mouse L: showClickInfo() M: schHiMousePopUp() R: schHiMousePopUp()
4(16) > Cmd: Sel: 5

Figure 190: Add input pads in schematic.



Navigator:

- I2 (pad_nc)
- I3 (pad_nc)
- I4 (pad_nc)
- I5 (pad_in)
- I6 (pad_nc)
- I7 (pad_nc)
- I8 (pad_gnd)
- I10 (pad_nc)
- I11 (pad_nc)
- I12 (pad_nc)
- I13 (pad_nc)
- I14 (pad_nc)
- I15 (pad_nc)
- I17 (pad_nc)
- I18 (pad_nc)
- I19 (pad_nc)
- I20 (pad_nc)
- I21 (pad_vdd)
- I22 (pad_nc)
- I23 (pad_in)
- I24 (pad_in)
- I25 (pad_nc)
- I26 (pad_nc)
- I27 (pad_in)
- I28 (pad_in)
- I29 (pad_nc)
- I30 (pad_nc)
- I31 (pad_nc)
- I32 (pad_nc)
- I33 (pad_nc)

Property Editor: instance / All

interfaceLa...	Mon Apr 14
partName	pad_nc
vendorName	
Name	(various)
Master	s pad_nc syr
Origin	(various)

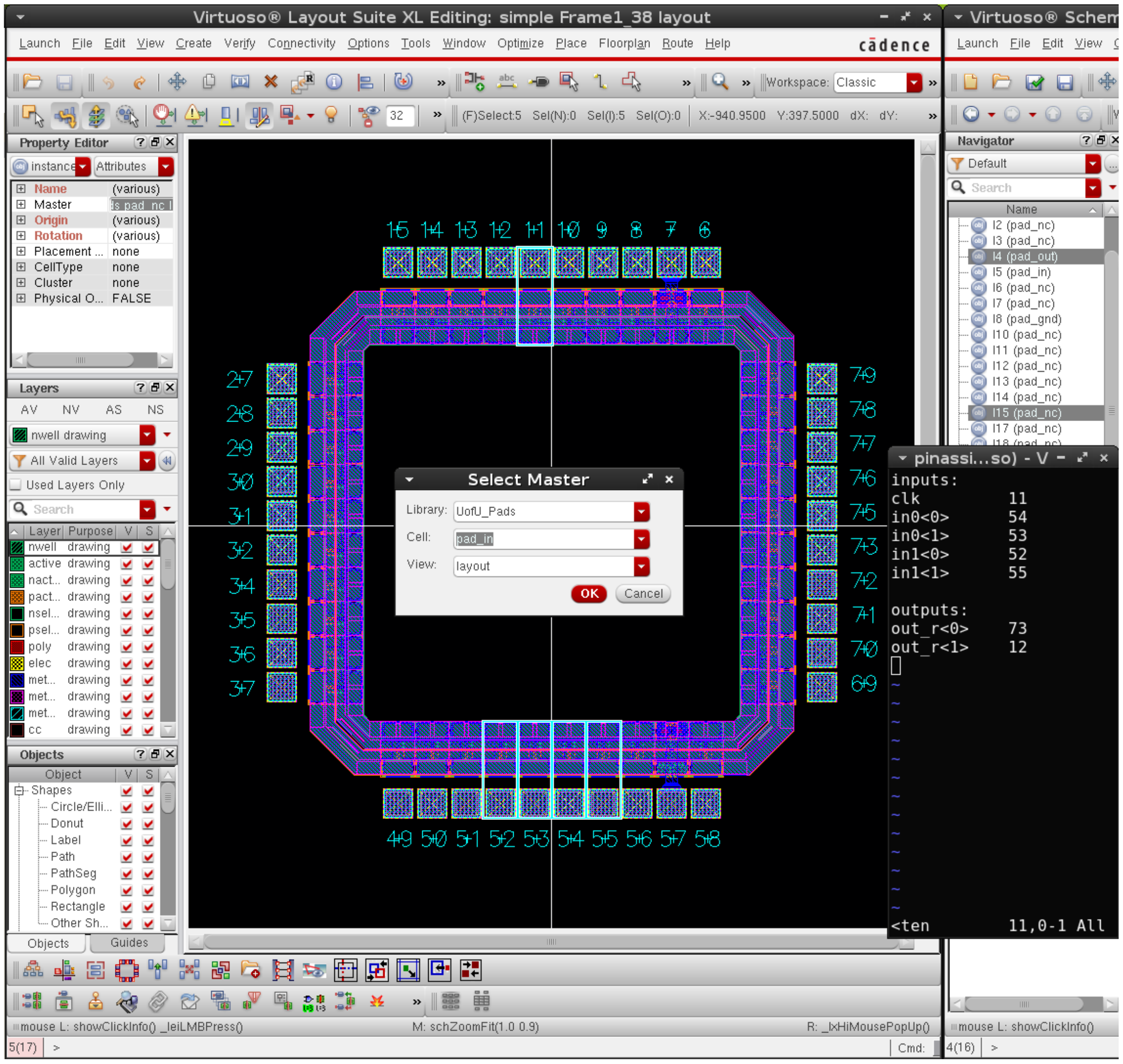
Select Master:

Library: UofU_Pads
Cell: pad_out
View: symbol

```
inputs:
clk        11
in0<0>     54
in0<1>     53
in1<0>     52
in1<1>     55

outputs:
out_r<0>   73
out_r<1>   12

<ten       11,0-1 All
```

mouse L: showClickInfo()   M: schHiMousePopUp()   R: schHiMousePopUp()
4(16) >   Cmd: Sel: 2

Figure 191: Add output pads in schematic.



Property Editor: instance, Attributes

Name	(various)
Master	ls pad_nc l
Origin	(various)
Rotation	(various)
Placement ...	none
CellType	none
Cluster	none
Physical O...	FALSE

Layers: nwell drawing; All Valid Layers; Used Layers Only

Layer	Purpose
nwell	drawing
active	drawing
nact...	drawing
pact...	drawing
nsel...	drawing
psel...	drawing
poly	drawing
elec	drawing
met...	drawing
met...	drawing
met...	drawing
cc	drawing

Objects: Shapes — Circle/Elli..., Donut, Label, Path, PathSeg, Polygon, Rectangle, Other Sh...

Select Master

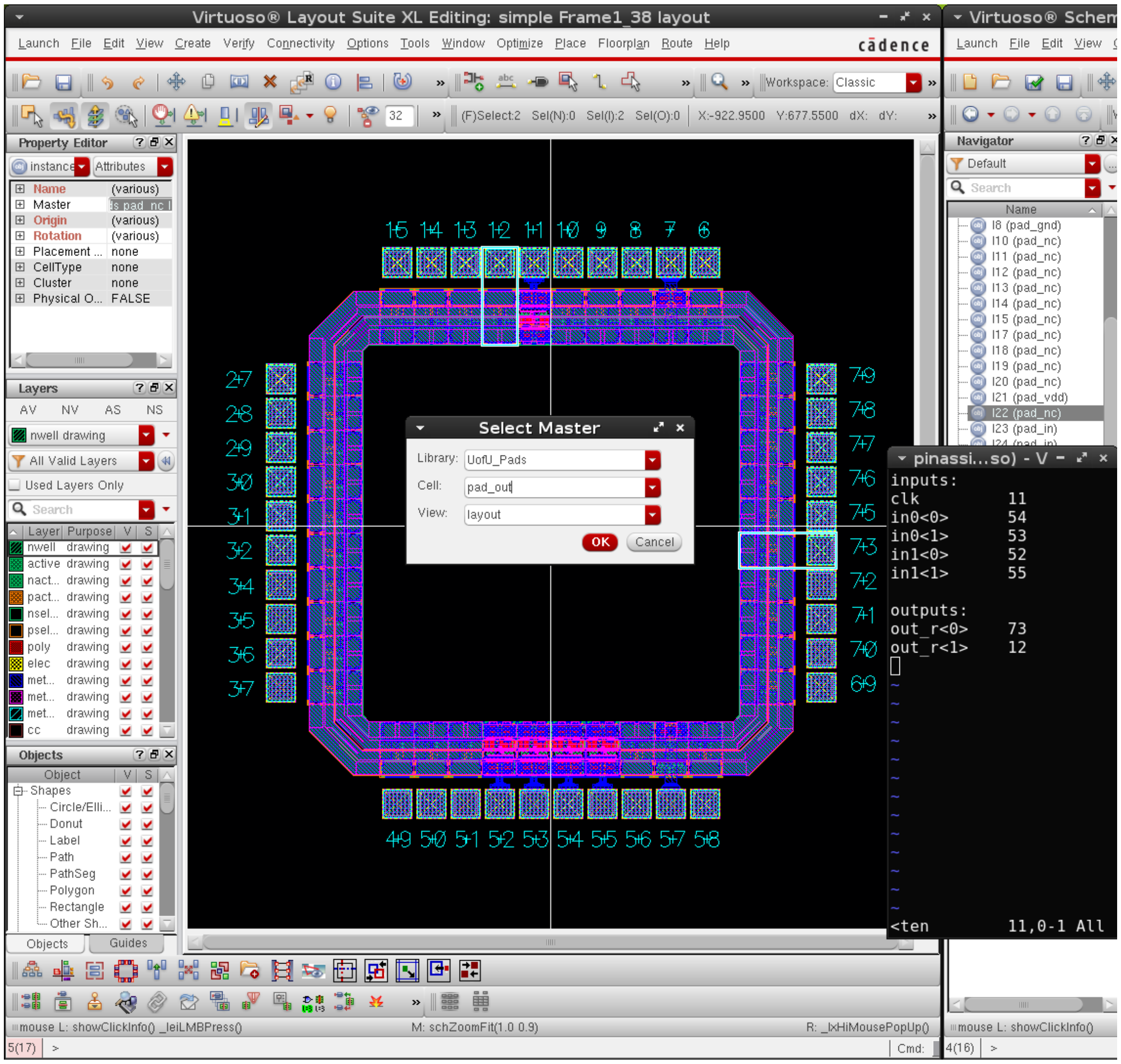
Library: UofU_Pads
Cell: pad_in
View: layout

Navigator: Default

I2 (pad_nc)
I3 (pad_nc)
I4 (pad_out)
I5 (pad_in)
I6 (pad_nc)
I7 (pad_nc)
I8 (pad_gnd)
I10 (pad_nc)
I11 (pad_nc)
I12 (pad_nc)
I13 (pad_nc)
I14 (pad_nc)
I15 (pad_nc)
I17 (pad_nc)

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12

<ten              11,0-1 All
```

Figure 192: Add input pads in layout.


Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
(F)Select:2 Sel(N):0 Sel(I):2 Sel(O):0 X:-922.9500 Y:677.5500 dX: dY:
Property Editor
Layers
Objects
Navigator
Virtuoso® Schem
mouse L: showClickInfo() _leiLMBPress()
M: schZoomFit(1.0 0.9)
R: _lxHiMousePopUp()
15 14 13 12 11 10 9 8 7 6
27 28 29 30 31 32 34 35 36 37
79 78 77 76 75 73 72 71 70 69
49 50 51 52 53 54 55 56 57 58

Select Master

Library: UofU_Pads
Cell: pad_out
View: layout

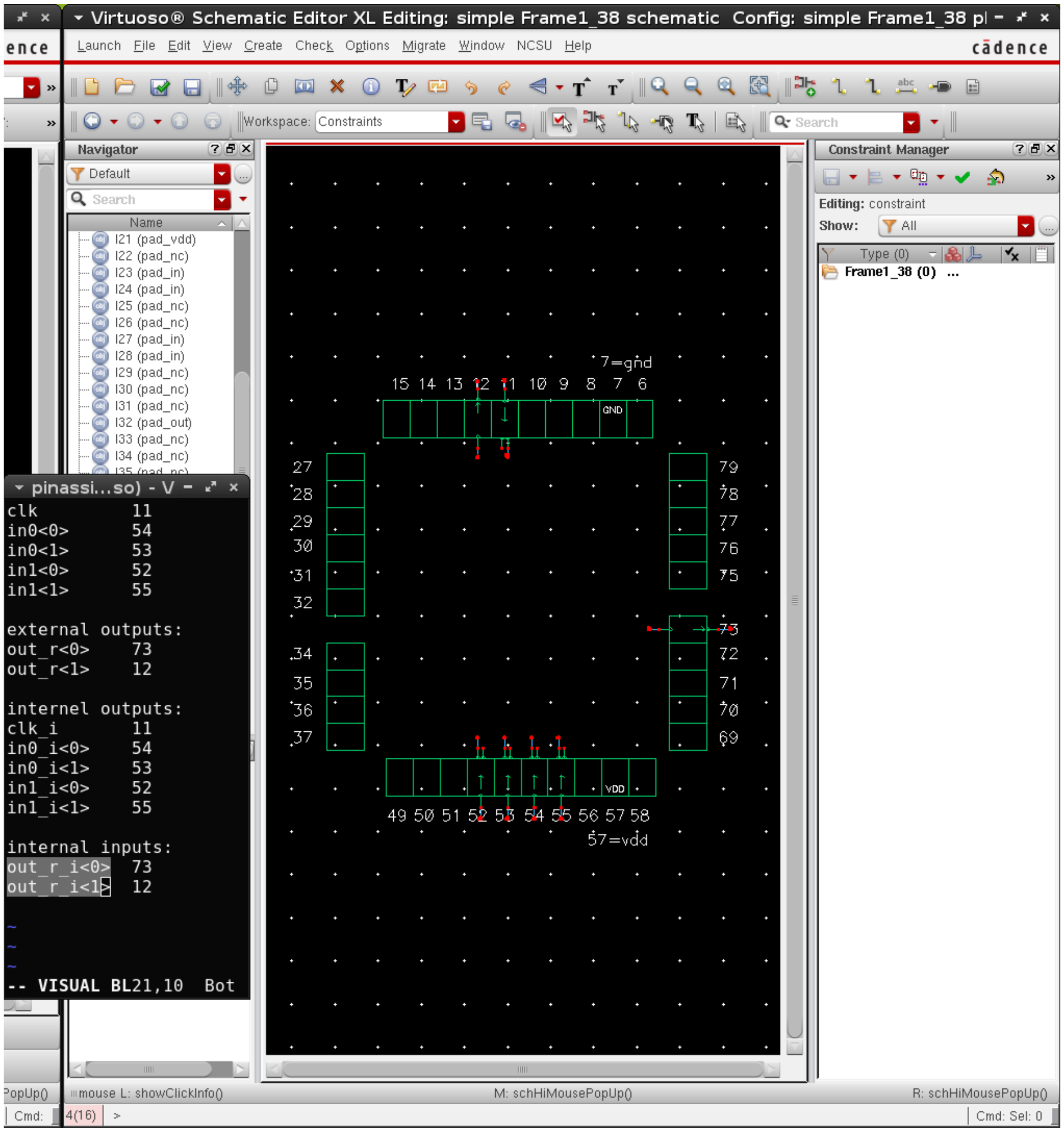
OK Cancel

pinassi...so) - V

```
inputs:
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

outputs:
out_r<0>     73
out_r<1>     12
```

Figure 193: Add output pads in layout.

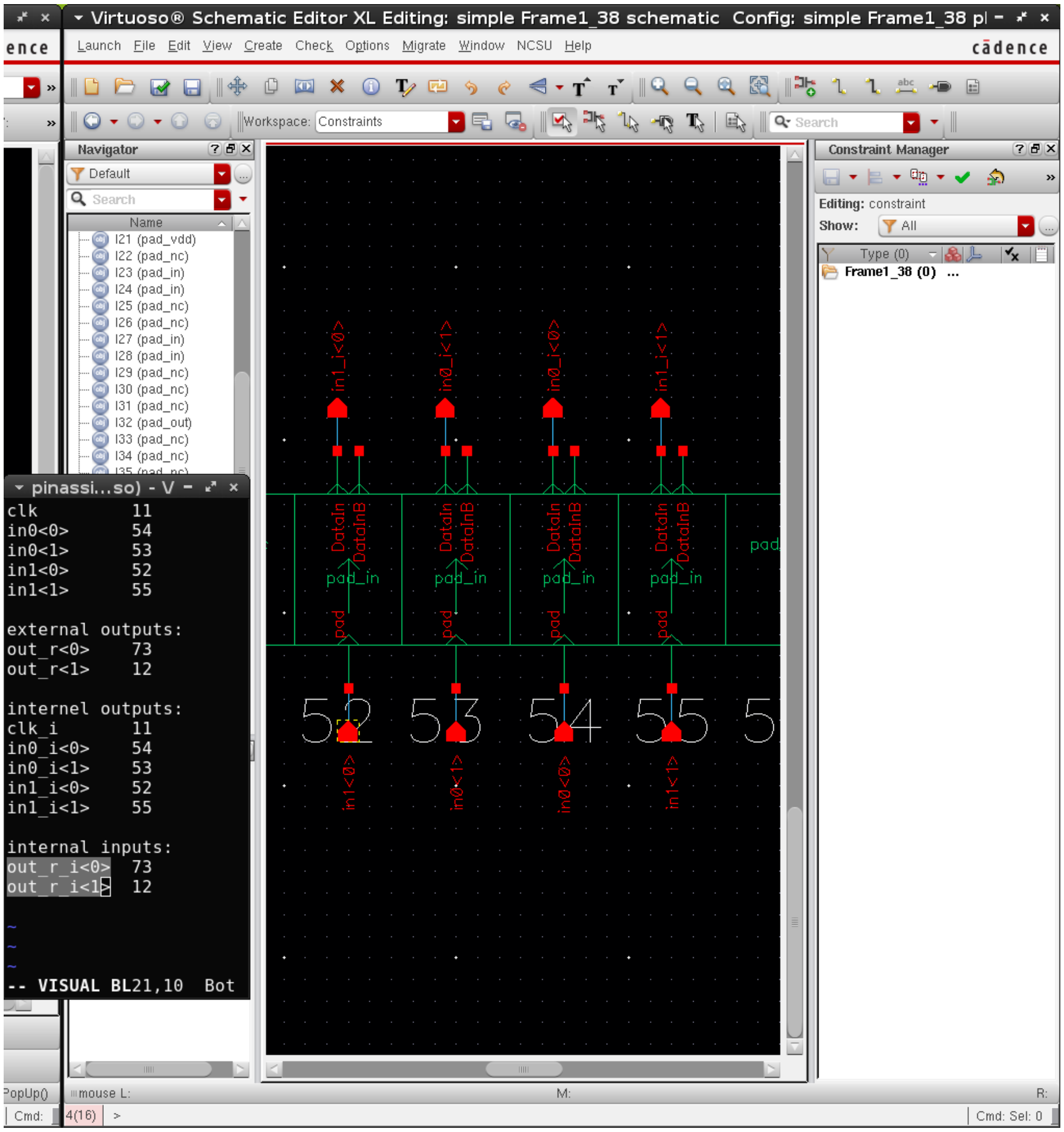
Add pins to pad frame (both schematic and layout). The pad frame works like a wraparound. The external pins of the pad frame (e.g. "Frame1_38") have the same name as the chip layout (e.g. "gpca40p"), and the internal pins of the pad frame connecting the chip layout use original names affixed by "_i" (Hence e.g. "clk" becomes "clk_i.) This procedure is shown in Figure 194 to Figure 207.



```
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_in)
I24 (pad_in)
I25 (pad_nc)
I26 (pad_nc)
I27 (pad_in)
I28 (pad_in)
I29 (pad_nc)
I30 (pad_nc)
I31 (pad_nc)
I32 (pad_out)
I33 (pad_nc)
I34 (pad_nc)
I35 (pad_nc)
```

```
clk          11
in0<0>       54
in0<1>       53
in1<0>       52
in1<1>       55

external outputs:
out_r<0>     73
out_r<1>     12

internel outputs:
clk_i        11
in0_i<0>     54
in0_i<1>     53
in1_i<0>     52
in1_i<1>     55

internal inputs:
out_r_i<0>   73
out_r_i<1>   12

~
~
~
-- VISUAL BL21,10   Bot
```

Figure 194: Add pins to the pad frame in schematic.

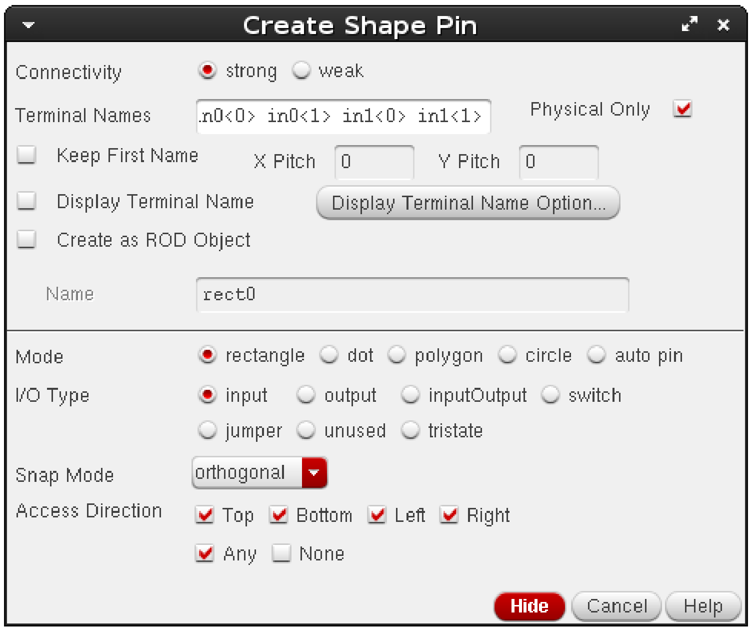


```
I21 (pad_vdd)
I22 (pad_nc)
I23 (pad_in)
I24 (pad_in)
I25 (pad_nc)
I26 (pad_nc)
I27 (pad_in)
I28 (pad_in)
I29 (pad_nc)
I30 (pad_nc)
I31 (pad_nc)
I32 (pad_out)
I33 (pad_nc)
I34 (pad_nc)
I35 (pad_nc)
```

```
clk           11
in0<0>        54
in0<1>        53
in1<0>        52
in1<1>        55

external outputs:
out_r<0>      73
out_r<1>      12

internel outputs:
clk_i         11
in0_i<0>      54
in0_i<1>      53
in1_i<0>      52
in1_i<1>      55

internal inputs:
out_r_i<0>    73
out_r_i<1>    12

~
~
~
-- VISUAL BL21,10   Bot
```

Figure 195: Add pins to the pad frame schematic (detailed view).

Same for the layout by tapping the ports with pins



Connectivity [✓ strong] weak

Terminal Names .n0<0> in0<1> in1<0> in1<1>

Physical Only [✓]

Keep First Name [ ] X Pitch 0 Y Pitch 0

Display Terminal Name [ ]

Create as ROD Object [ ]

Name rect0

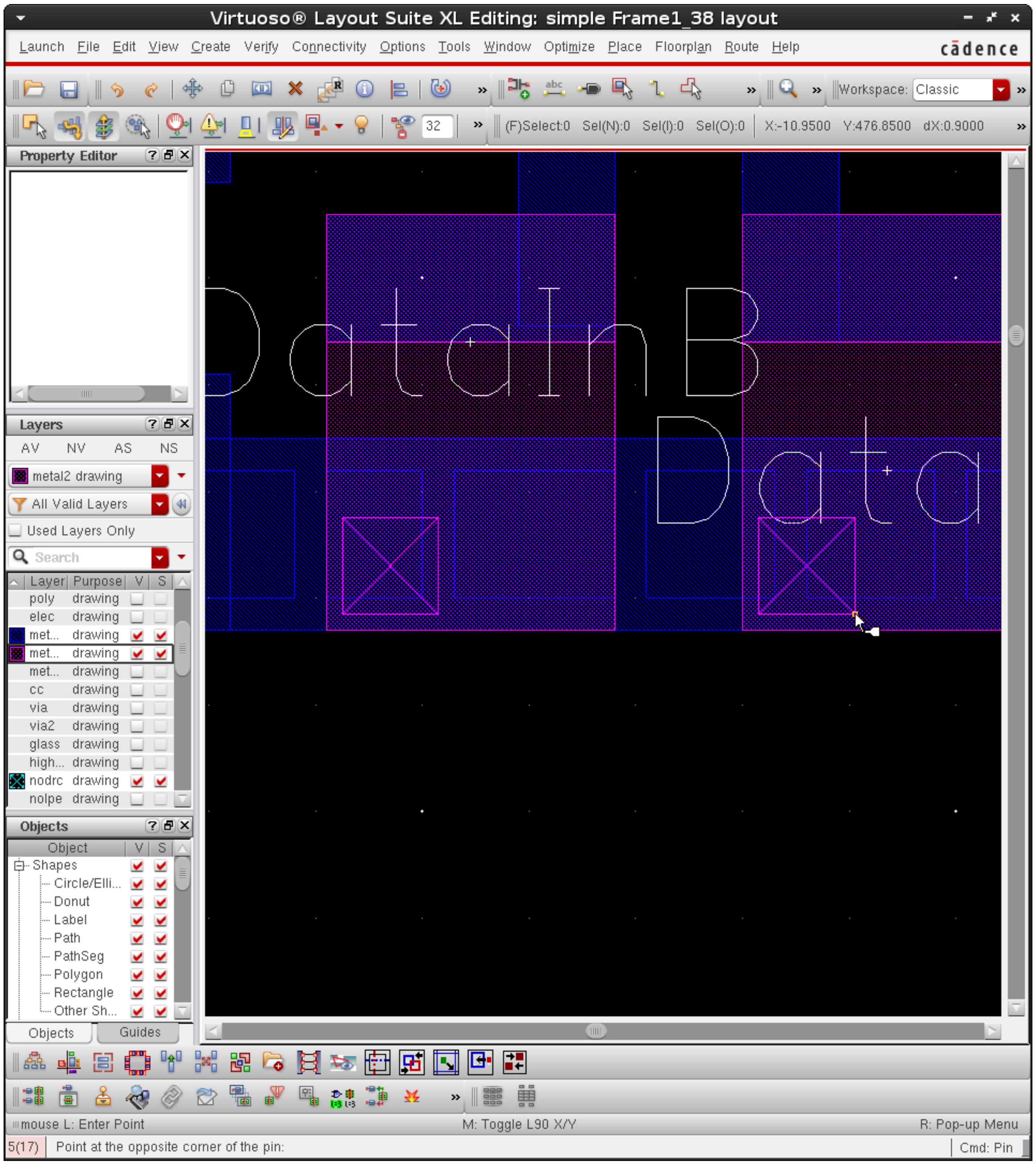
Mode [✓ rectangle] dot polygon circle auto pin

I/O Type [✓ input] output inputOutput switch jumper unused tristate

Snap Mode orthogonal

Access Direction [✓ Top] [✓ Bottom] [✓ Left] [✓ Right] [✓ Any] None

Figure 196: "Create Shape Pin" Window of Virtuoso.


Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cadence
Workspace: Classic
Property Editor
Layers
AV NV AS NS
metal2 drawing
All Valid Layers
Used Layers Only
Search
Objects
Objects Guides
DataInB
Data

(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:-10.9500 Y:476.8500 dX:0.9000

Layer	Purpose	V	S
poly	drawing	☐	☐
elec	drawing	☐	☐
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	☐	☐
cc	drawing	☐	☐
via	drawing	☐	☐
via2	drawing	☐	☐
glass	drawing	☐	☐
high...	drawing	☐	☐
nodrc	drawing	✓	✓
nolpe	drawing	☐	☐

Object	V	S
Shapes	✓	✓
Circle/Elli...	✓	✓
Donut	✓	✓
Label	✓	✓
Path	✓	✓
PathSeg	✓	✓
Polygon	✓	✓
Rectangle	✓	✓
Other Sh...	✓	✓

mouse L: Enter Point M: Toggle L90 X/Y R: Pop-up Menu

5(17) Point at the opposite corner of the pin: Cmd: Pin

Figure 197: Add pins to the pad frame layout (detailed view).

DRC, Extract, LVS to make sure no rule is violated.

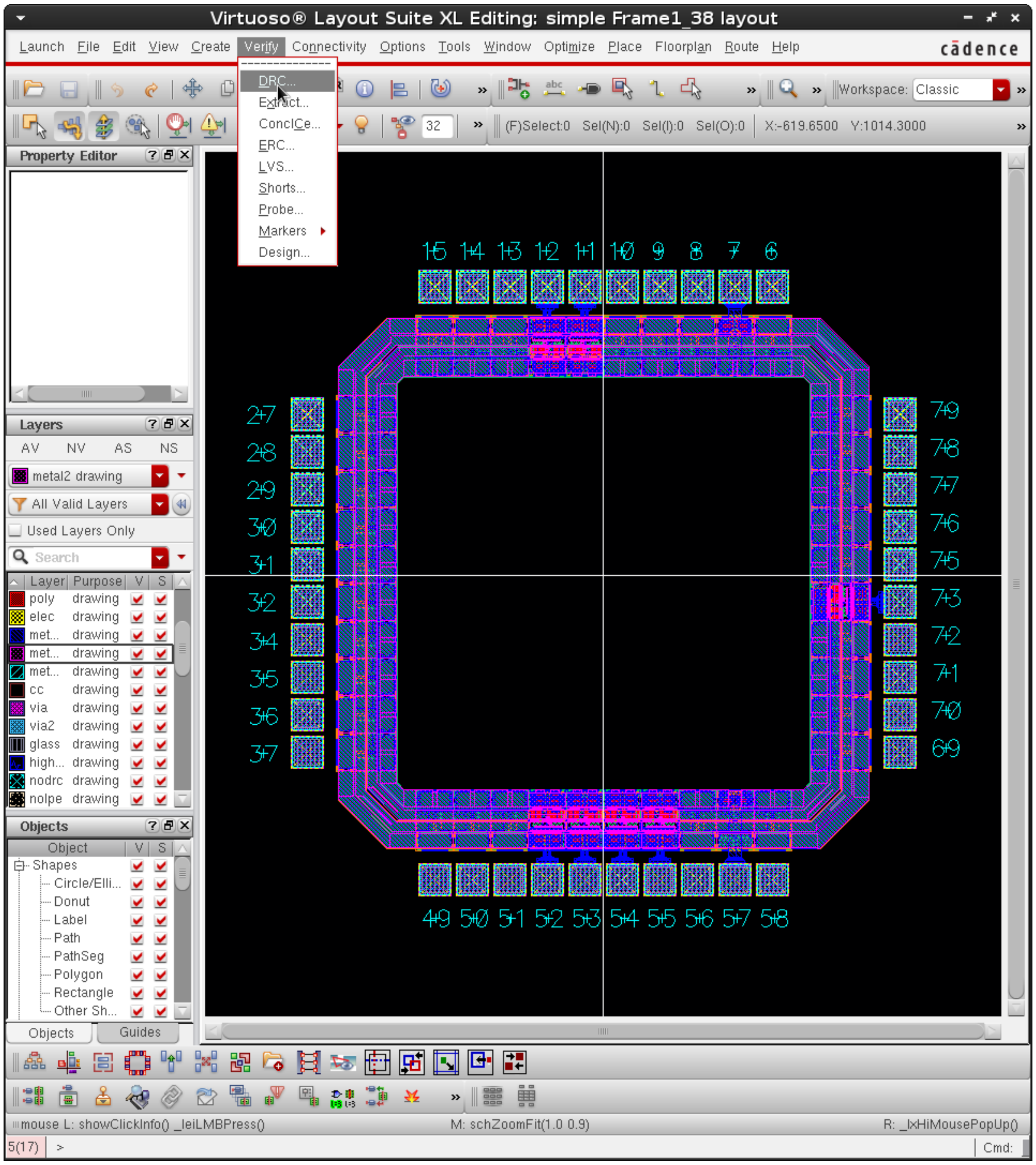
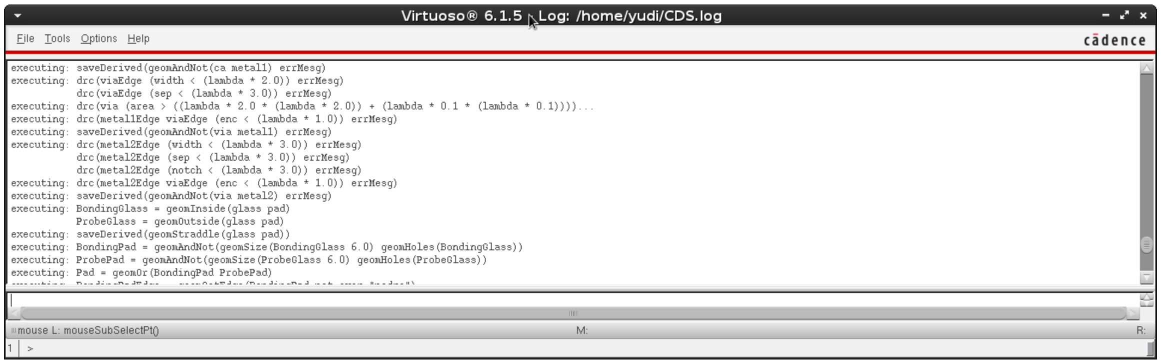

Virtuoso® Layout Suite XL Editing: simple Frame1_38 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
DRC...
Extract...
ConclCe...
ERC...
LVS...
Shorts...
Probe...
Markers
Design...
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:-619.6500 Y:1014.3000
Property Editor
Layers
metal2 drawing
All Valid Layers
Used Layers Only
Objects
Guides
mouse L: showClickInfo() _leiLMBPress()
M: schZoomFit(1.0 0.9)
R: _lxHiMousePopUp()
Cmd:

Figure 198: Start DRC from Virtuoso Layout Suite.



```
executing: saveDerived(geomAndNot(ca metal1) errMesg)
executing: drc(viaEdge (width < (lambda * 2.0)) errMesg)
           drc(viaEdge (sep < (lambda * 3.0)) errMesg)
executing: drc(via (area > ((lambda * 2.0 * (lambda * 2.0)) + (lambda * 0.1 * (lambda * 0.1)))...
executing: drc(metal1Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal1) errMesg)
executing: drc(metal2Edge (width < (lambda * 3.0)) errMesg)
           drc(metal2Edge (sep < (lambda * 3.0)) errMesg)
           drc(metal2Edge (notch < (lambda * 3.0)) errMesg)
executing: drc(metal2Edge viaEdge (enc < (lambda * 1.0)) errMesg)
executing: saveDerived(geomAndNot(via metal2) errMesg)
executing: BondingGlass = geomInside(glass pad)
           ProbeGlass = geomOutside(glass pad)
executing: saveDerived(geomStraddle(glass pad))
executing: BondingPad = geomAndNot(geomSize(BondingGlass 6.0) geomHoles(BondingGlass))
executing: ProbePad = geomAndNot(geomSize(ProbeGlass 6.0) geomHoles(ProbeGlass))
executing: Pad = geomOr(BondingPad ProbePad)
```

```
mouse L: mouseSubSelectPt()          M:          R:
1 >
```

Figure 199: Report of successful DRC.

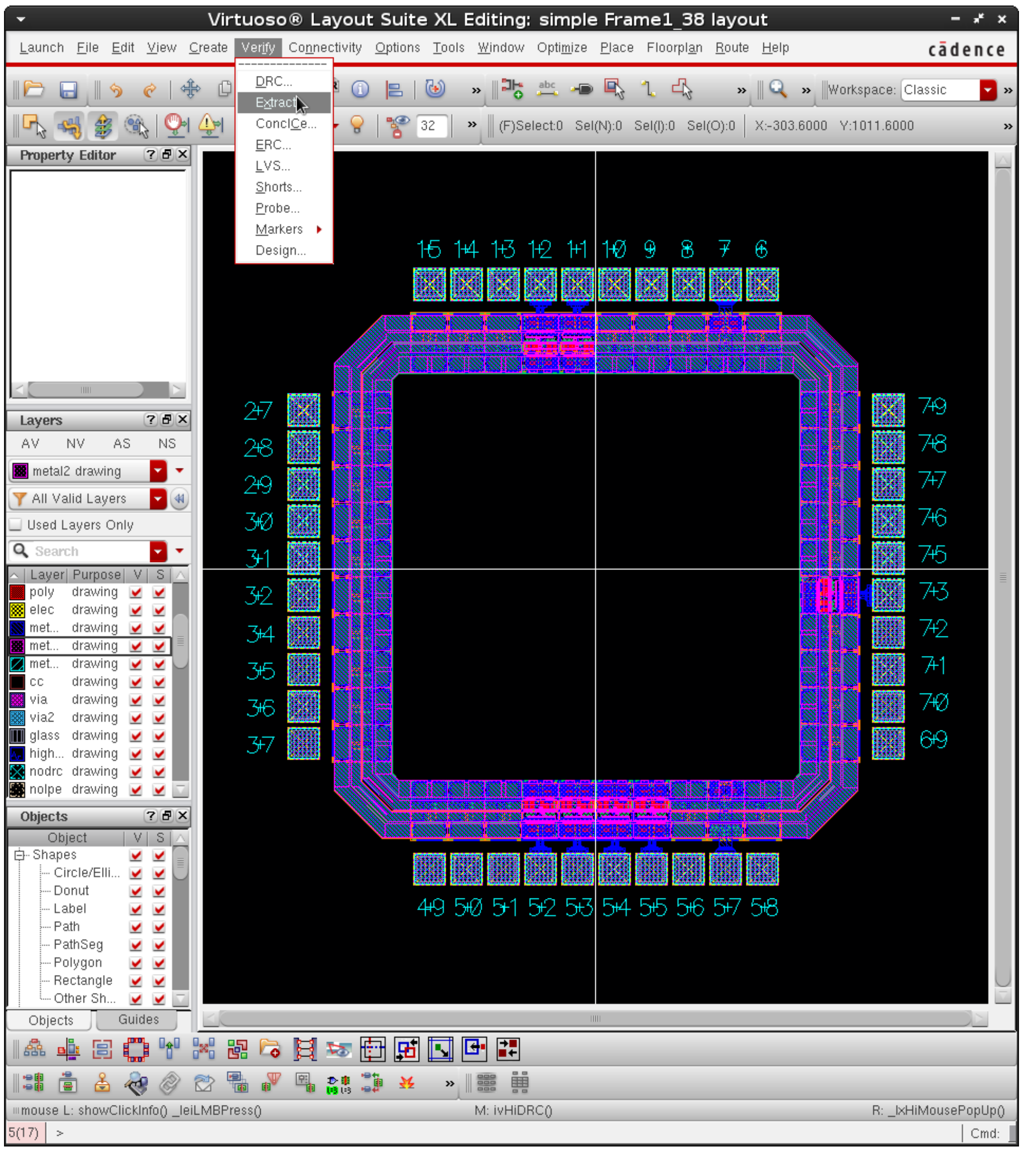
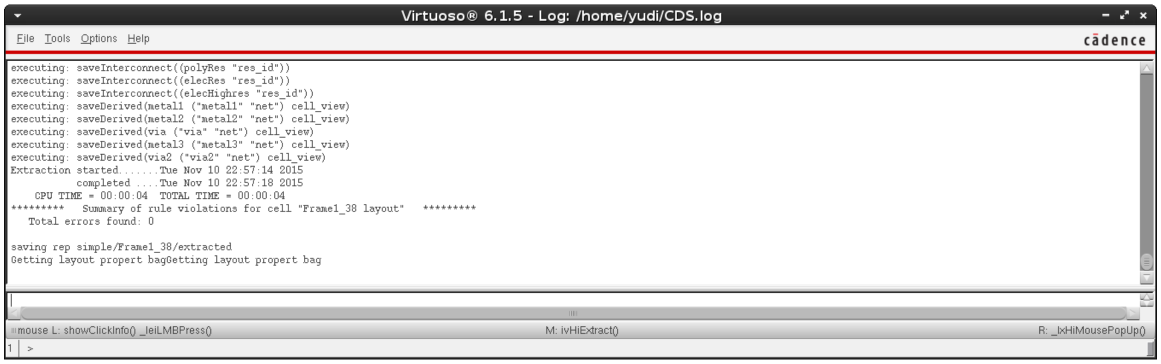


Figure 200: Start Extract from Virtuoso Layout Suite.



```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 22:57:14 2015
           completed ....Tue Nov 10 22:57:18 2015
    CPU TIME = 00:00:04  TOTAL TIME = 00:00:04
*********   Summary of rule violations for cell "Frame1_38 layout"   *********
   Total errors found: 0

saving rep simple/Frame1_38/extracted
Getting layout propert bagGetting layout propert bag
```

```
mouse L: showClickInfo() _leiLMBPress()          M: ivHiExtract()          R: _lxHiMousePopUp()
1  >
```

Figure 201: Report of successful Extract.

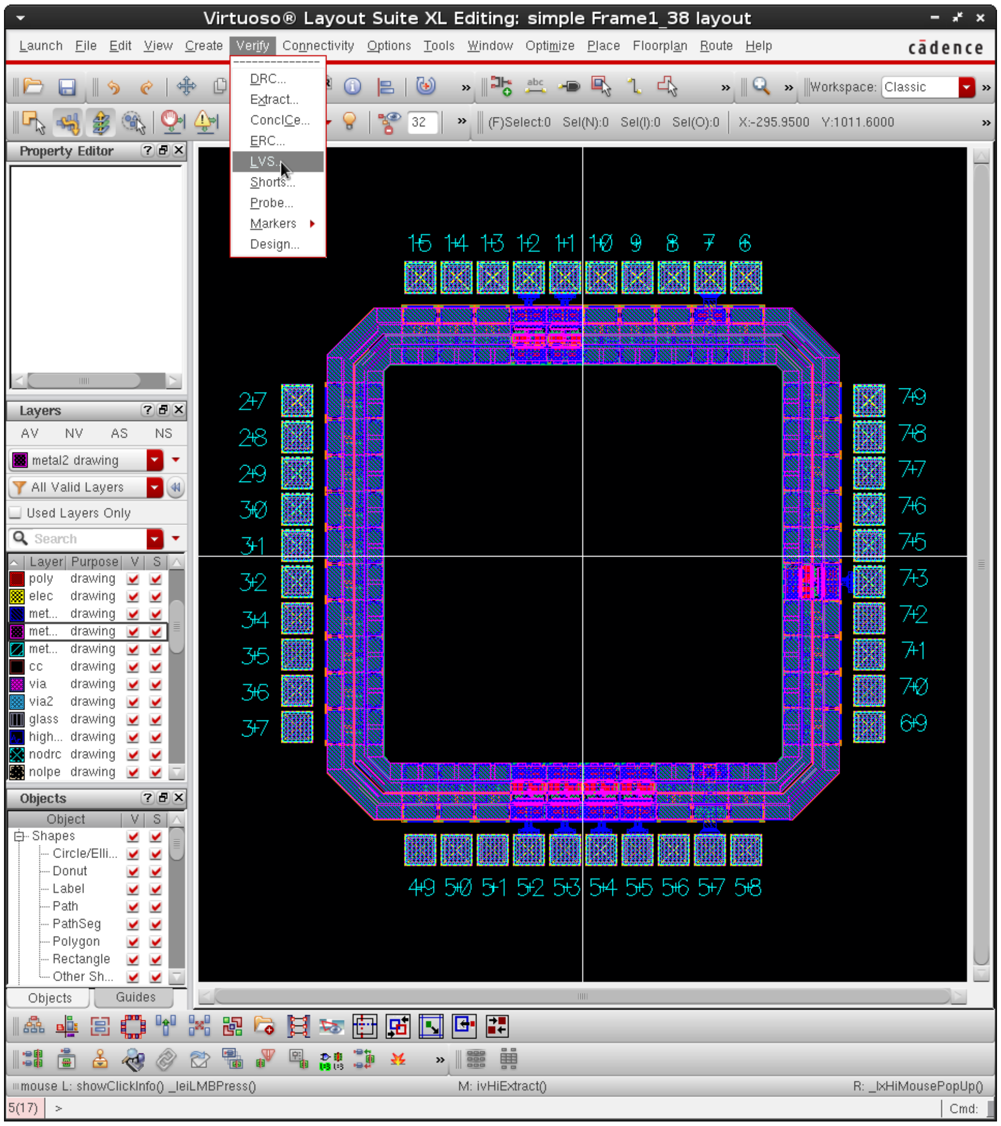
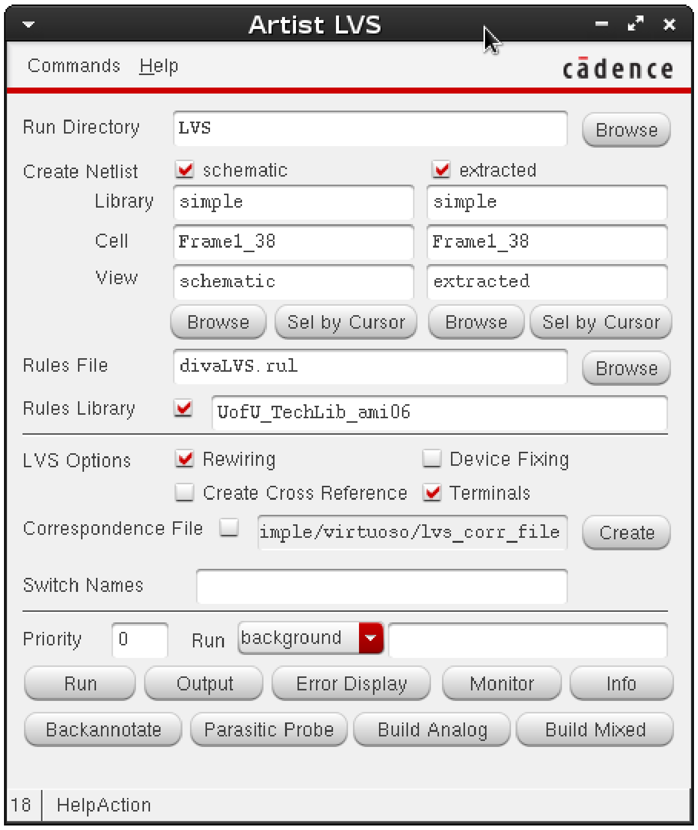


Figure 202: Start LVS from Virtuoso Layout Suite.


Artist LVS

Commands Help

Run Directory: LVS

Create Netlist: ✓ schematic ✓ extracted

	schematic	extracted
Library	simple	simple
Cell	Frame1_38	Frame1_38
View	schematic	extracted

Rules File: divaLVS.rul

Rules Library: ✓ UofU_TechLib_ami06

LVS Options: ✓ Rewiring, Device Fixing, Create Cross Reference, ✓ Terminals

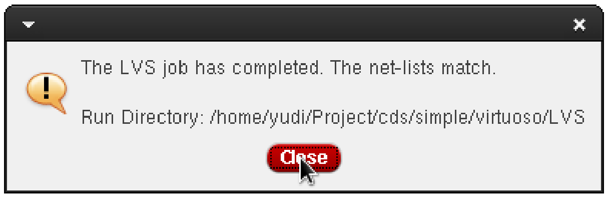
Correspondence File: imple/virtuoso/lvs_corr_file

Switch Names:

Priority: 0 Run: background

18 HelpAction

Figure 203: "Artist LVS" window of Virtuoso.


The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

Figure 204: "Artist LVS" successful LVS prompt.

Generate symbol for the pad frame.

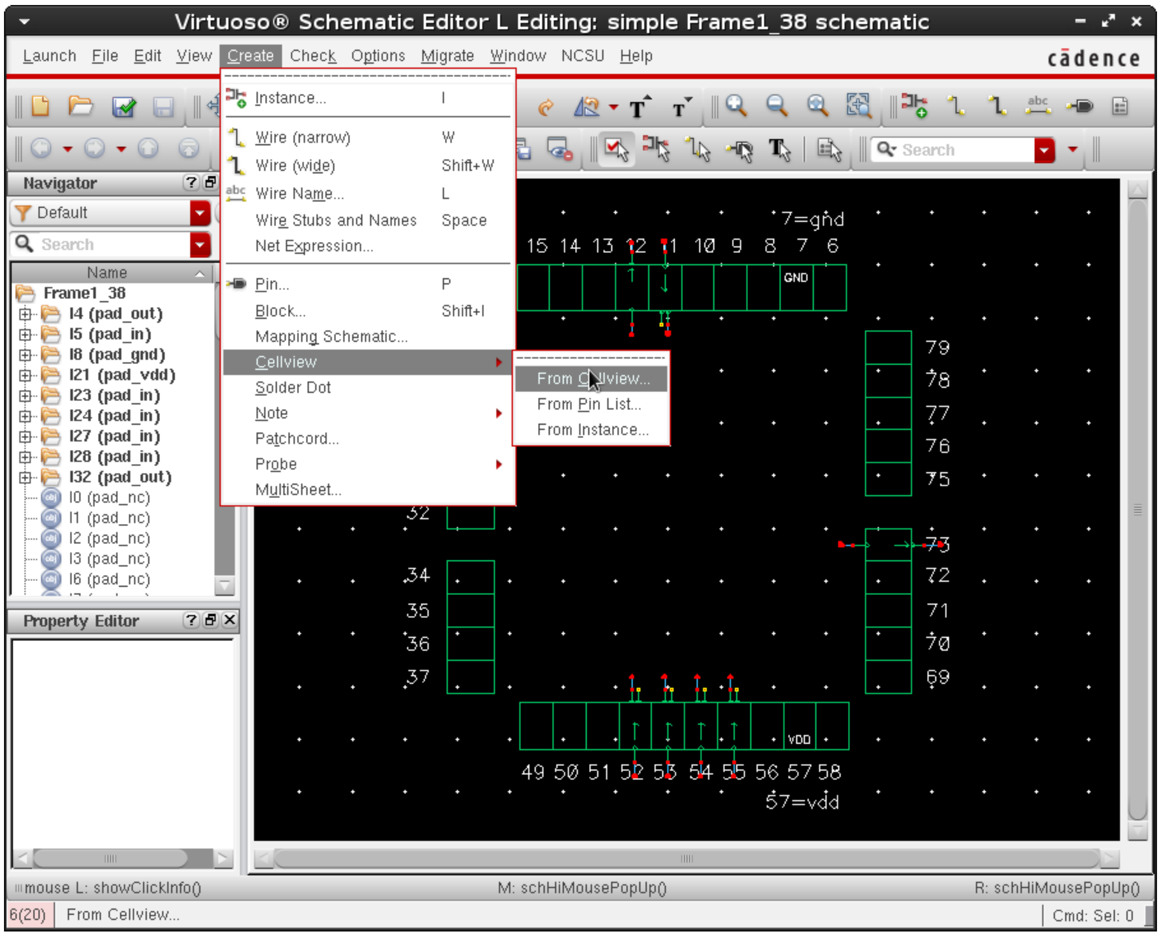
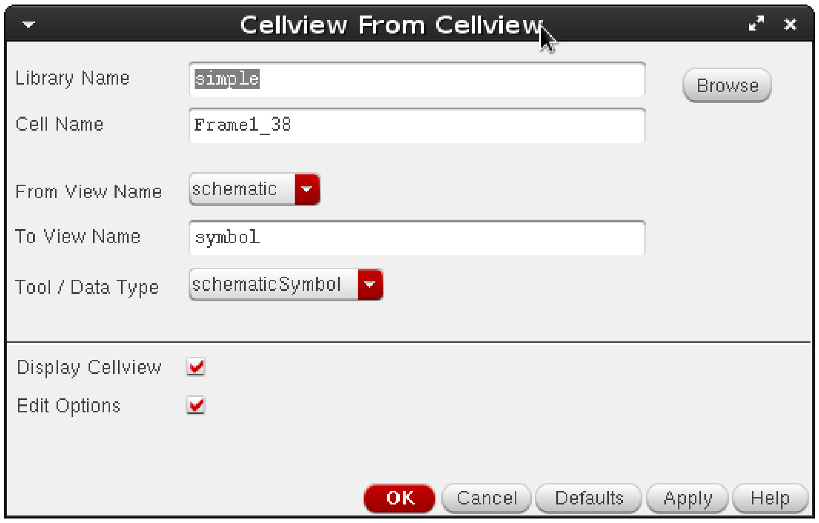


Figure 205: Create symbol view from schematic view.


Cellview From Cellview

Library Name: simple

Cell Name: Frame1_38

From View Name: schematic

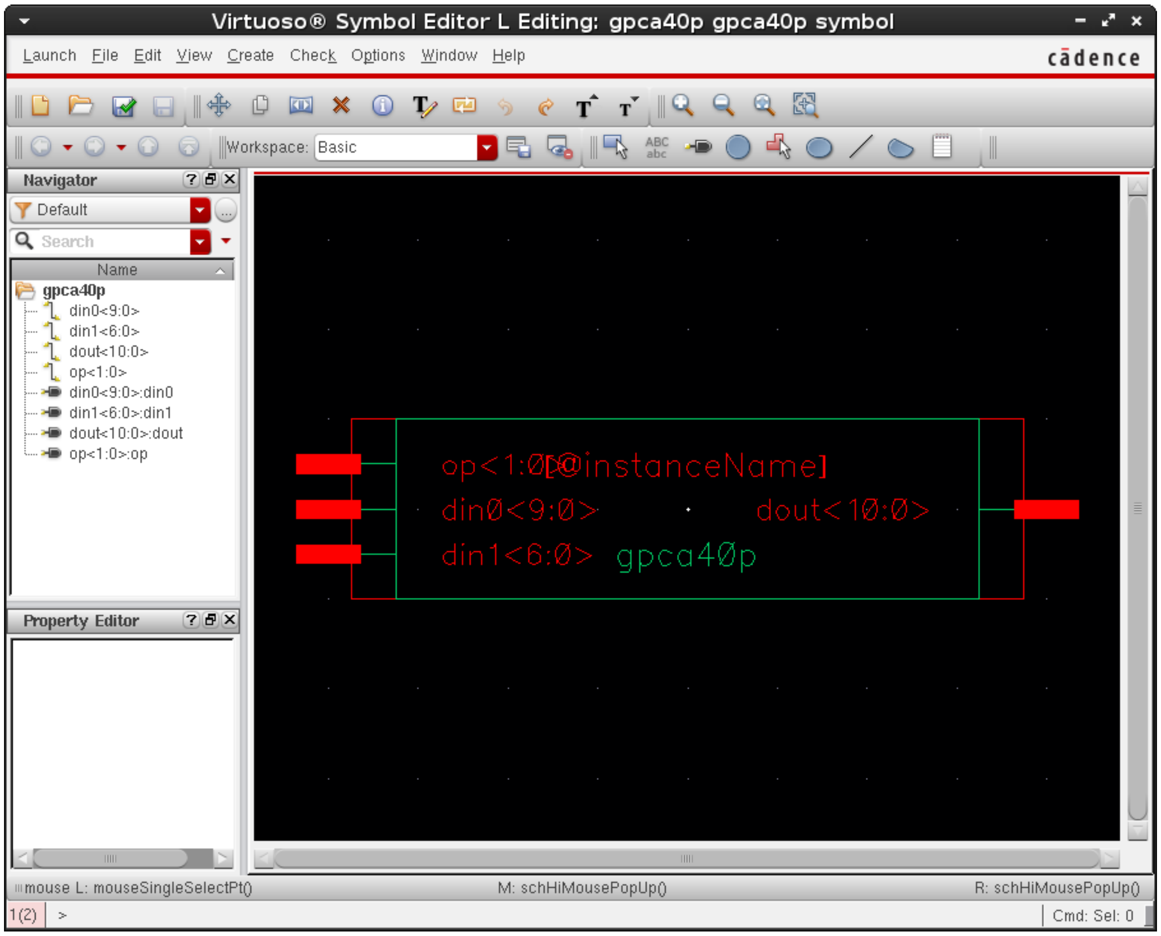
To View Name: symbol

Tool / Data Type: schematicSymbol

Display Cellview: ✓

Edit Options: ✓

Figure 206: "CellView from CellView" window of Virtuoso.


Virtuoso® Symbol Editor L Editing: gpca40p gpca40p symbol

mouse L: mouseSingleSelectPt() M: schHiMousePopUp() R: schHiMousePopUp()

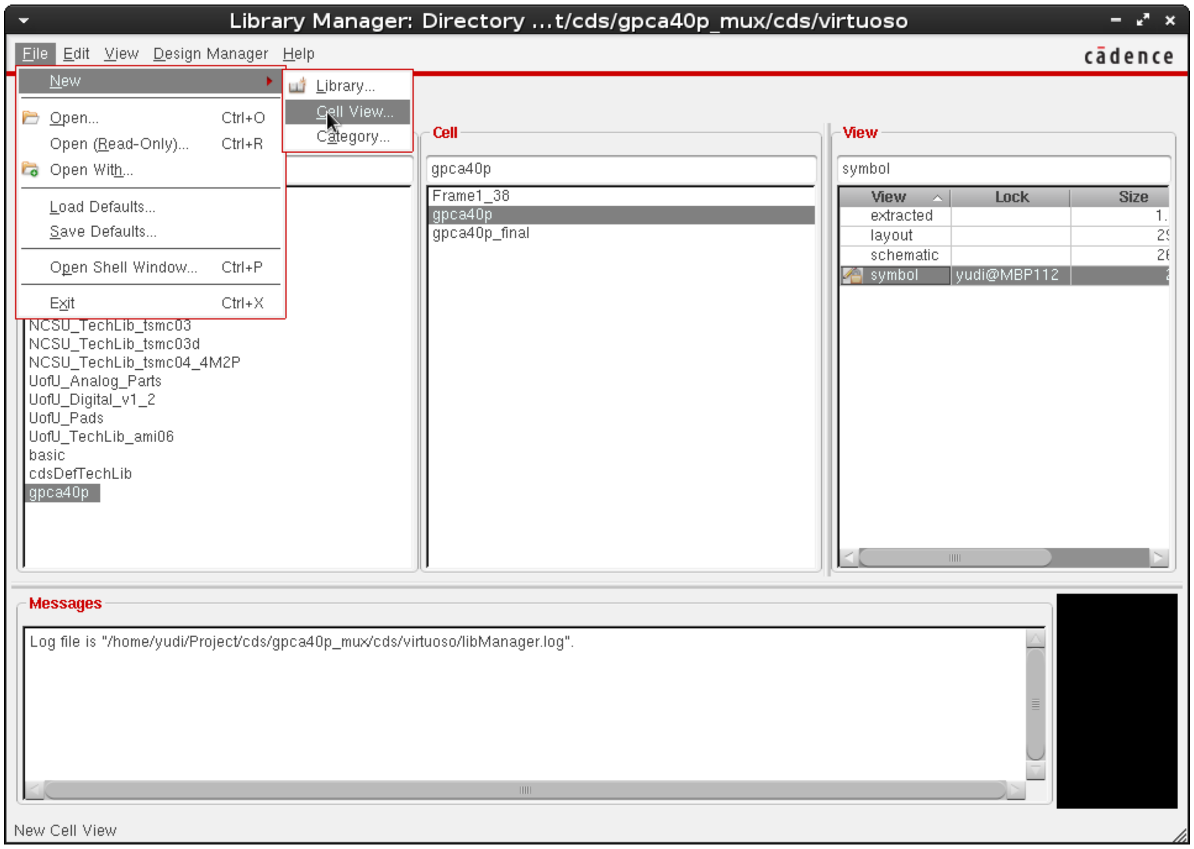
1(2) > Cmd: Sel: 0

Figure 207: Final symbol view of the pad frame.

e) Final Chip Assembly (Add Pad Frame)

With both the core of the chip ("simple") and the pad frame (modified "Frame1_38") are ready, creating a new cell to put them together. This procedure is shown in Figure 208 to 224.

Create a new Cell View (I call it "gpca40p_final").



NCSU_TechLib_tsmc03
NCSU_TechLib_tsmc03d
NCSU_TechLib_tsmc04_4M2P
UofU_Analog_Parts
UofU_Digital_v1_2
UofU_Pads
UofU_TechLib_ami06
basic
cdsDefTechLib
gpca40p

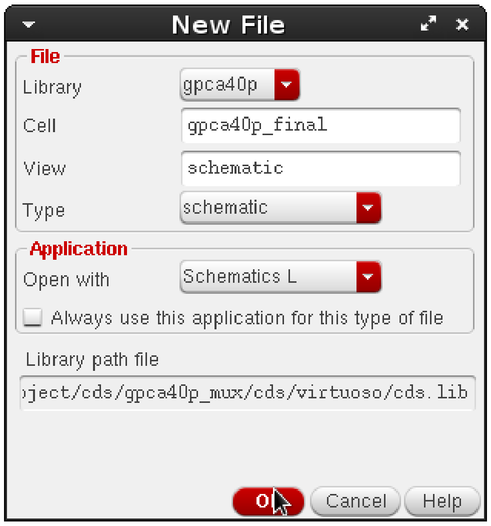
Cell: gpca40p
Frame1_38
gpca40p
gpca40p_final

View: symbol

View	Lock	Size
extracted		1.
layout		2
schematic		2
symbol	yudi@MBP112	

```
Log file is "/home/yudi/Project/cds/gpca40p_mux/cds/virtuoso/libManager.log".
```

Figure 208: Create new "Cell View" from Virtuoso Layout Suite



File

Library: gpca40p

Cell: gpca40p_final

View: schematic

Type: schematic

Application

Open with: Schematics L

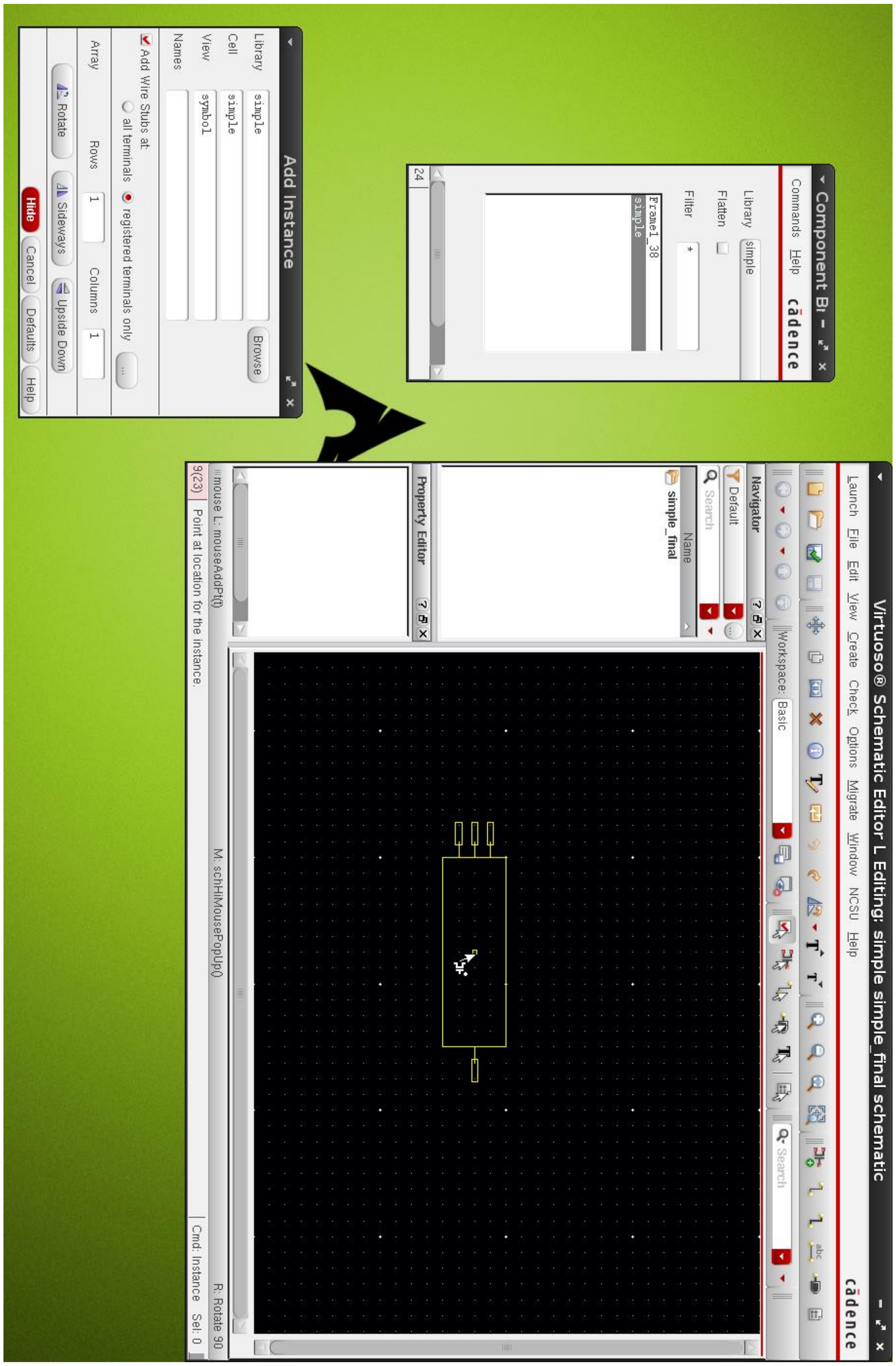
☐ Always use this application for this type of file

Library path file

ject/cds/gpca40p_mux/cds/virtuoso/cds.lib

Figure 209: "New File" window of Virtuoso.

Instantiate both the core of the chip ("gpca40p") and the pad frame (modified "Frame1_38") (by pressing "I" to instantiate instances).



Library simple
Flatten
Filter *
Frame1_38
simple
24

Library simple
Cell simple
View symbol
Names
[✓] Add Wire Stubs at: all terminals [● registered terminals only]
Array Rows 1 Columns 1

mouse L: mouseAddPt() M: schHiMousePopUp() R: Rotate 90
9(23) Point at location for the instance. Cmd: Instance Sel: 0

Figure 210: Create the final schematic view.

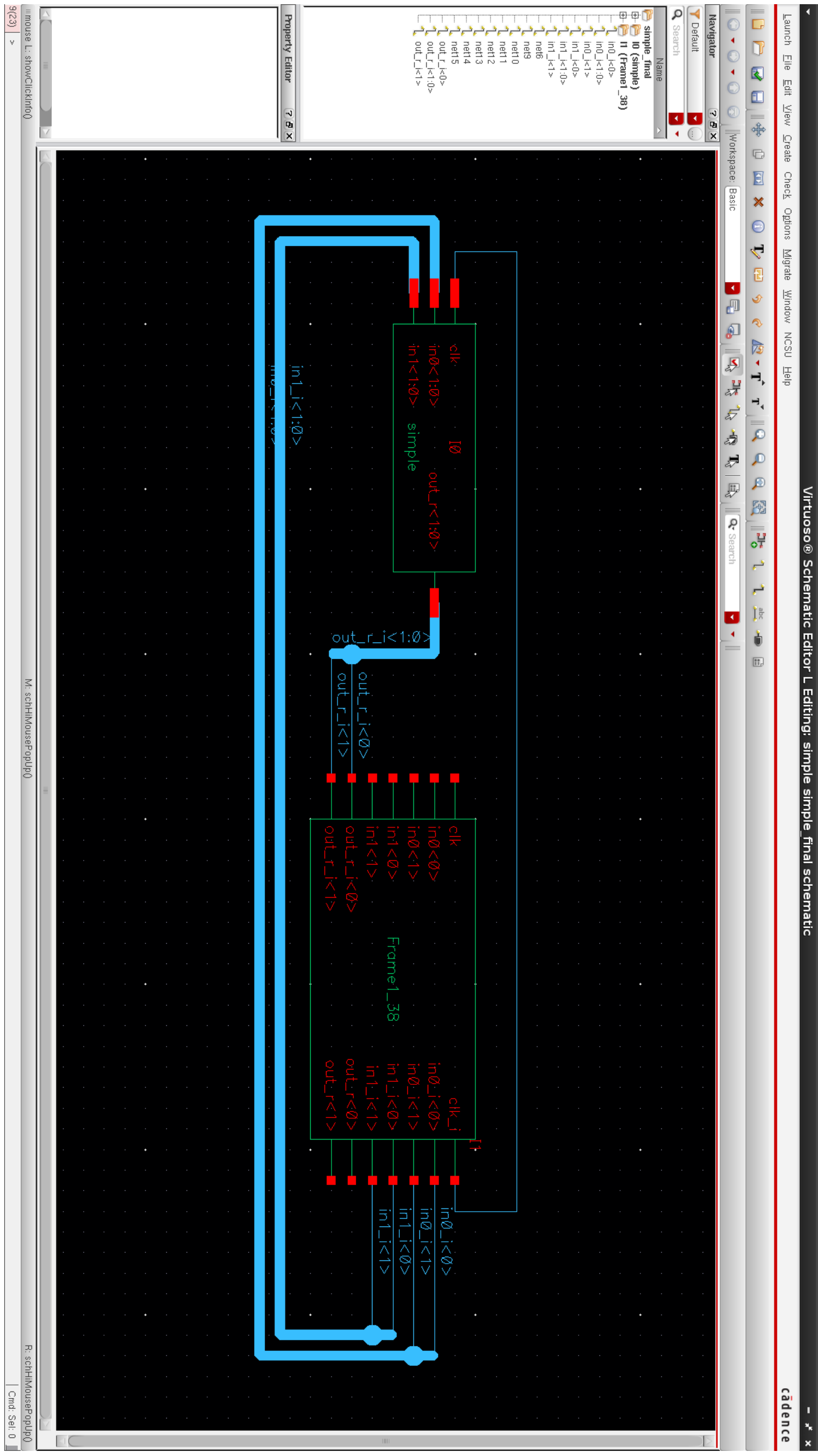
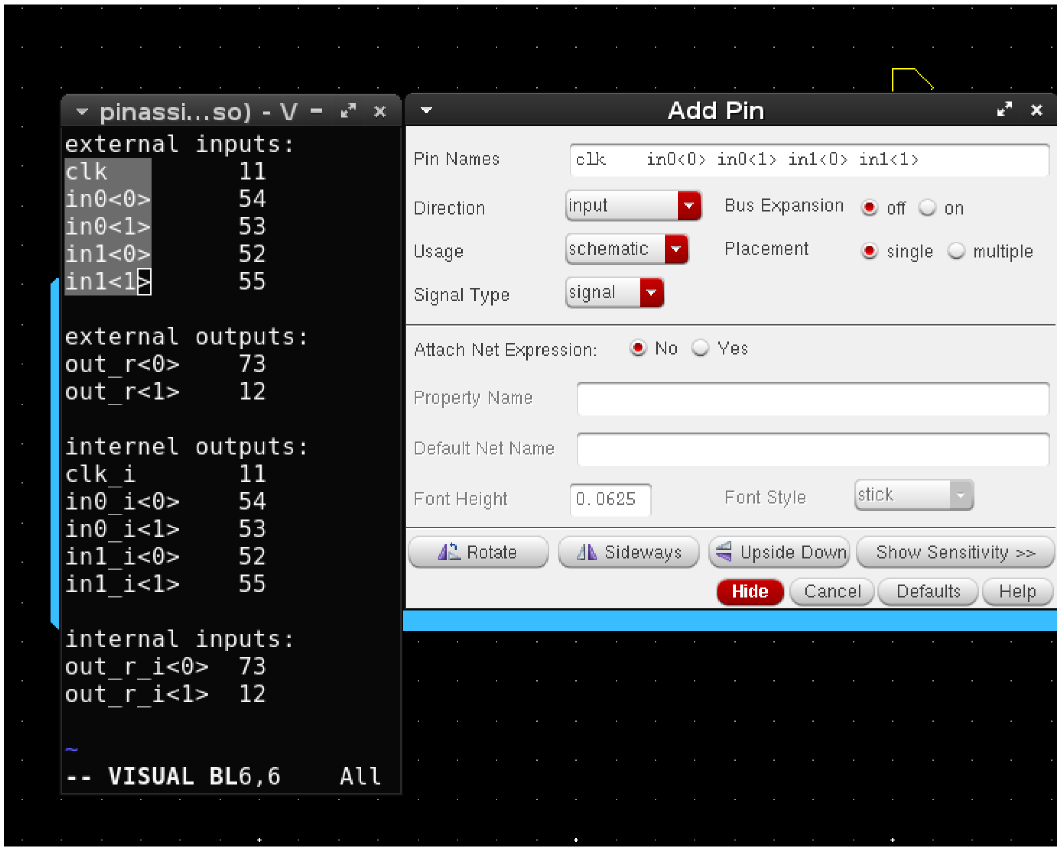


Figure 211: Routed schematic view.

Then add the pins.



```
external inputs:
clk        11
in0<0>     54
in0<1>     53
in1<0>     52
in1<1>     55

external outputs:
out_r<0>   73
out_r<1>   12

internel outputs:
clk_i      11
in0_i<0>   54
in0_i<1>   53
in1_i<0>   52
in1_i<1>   55

internal inputs:
out_r_i<0>  73
out_r_i<1>  12

-- VISUAL BL6,6      All
```

Add Pin

Pin Names: clk in0<0> in0<1> in1<0> in1<1>

Direction: input

Bus Expansion: [✓ off] on

Usage: schematic

Placement: [✓ single] multiple

Signal Type: signal

Attach Net Expression: [✓ No] Yes

Property Name:

Default Net Name:

Font Height: 0.0625

Font Style: stick

Figure 212: "Add Pins" window of Virtuoso.

Final schematic view is shown below.

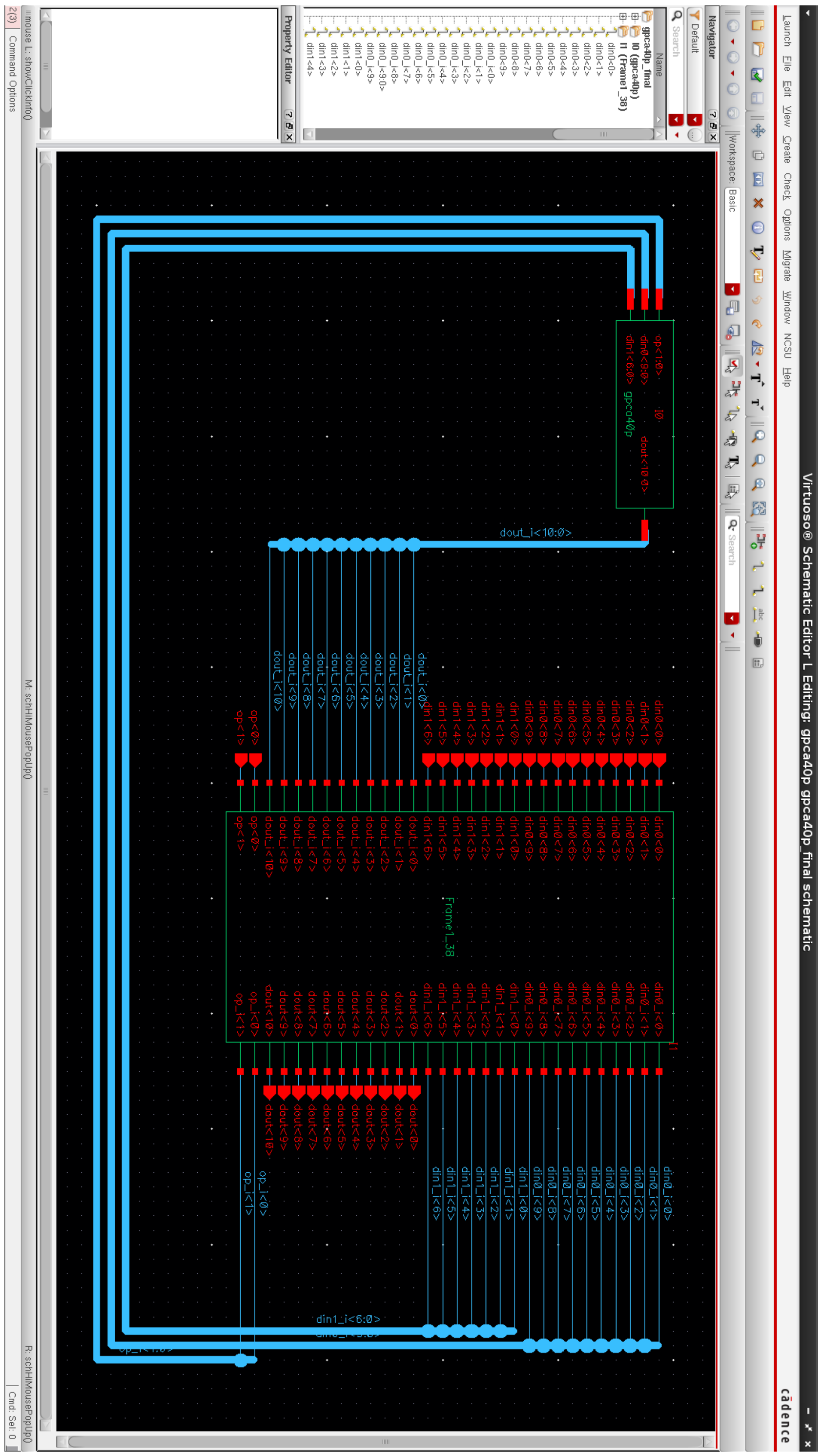
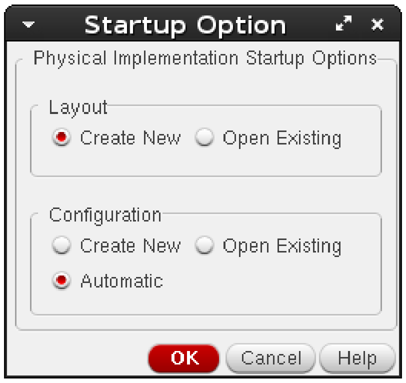

Virtuoso® Schematic Editor L Editing: gpca40p gpca40p_final schematic
Launch File Edit View Create Check Options Migrate Window NCSU Help
Navigator
Property Editor
gpca40p
I0
Frame1_38
I1
dout_i<10:0>
din1_i<6:0>
din0_i<9:0>
op_i<1:0>
cadence

Figure 213: Final schematic view.

Then create the final layout view from the schematic.



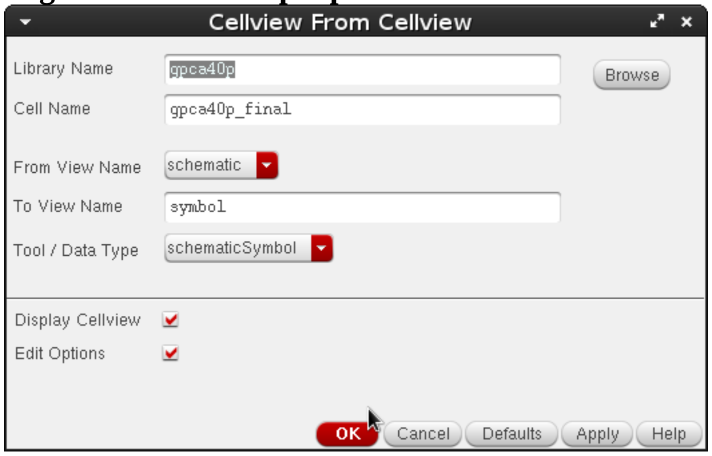
Startup Option

Physical Implementation Startup Options

Layout: [✓ Create New] Open Existing

Configuration: Create New Open Existing [✓ Automatic]

Figure 214: "Startup Option" window of Virtuoso.



Cellview From Cellview

Library Name: gpca40p

Cell Name: gpca40p_final

From View Name: schematic

To View Name: symbol

Tool / Data Type: schematicSymbol

Display Cellview: ✓

Edit Options: ✓

Figure 215: "Startup Option" window of Virtuoso.

In the new empty layout view

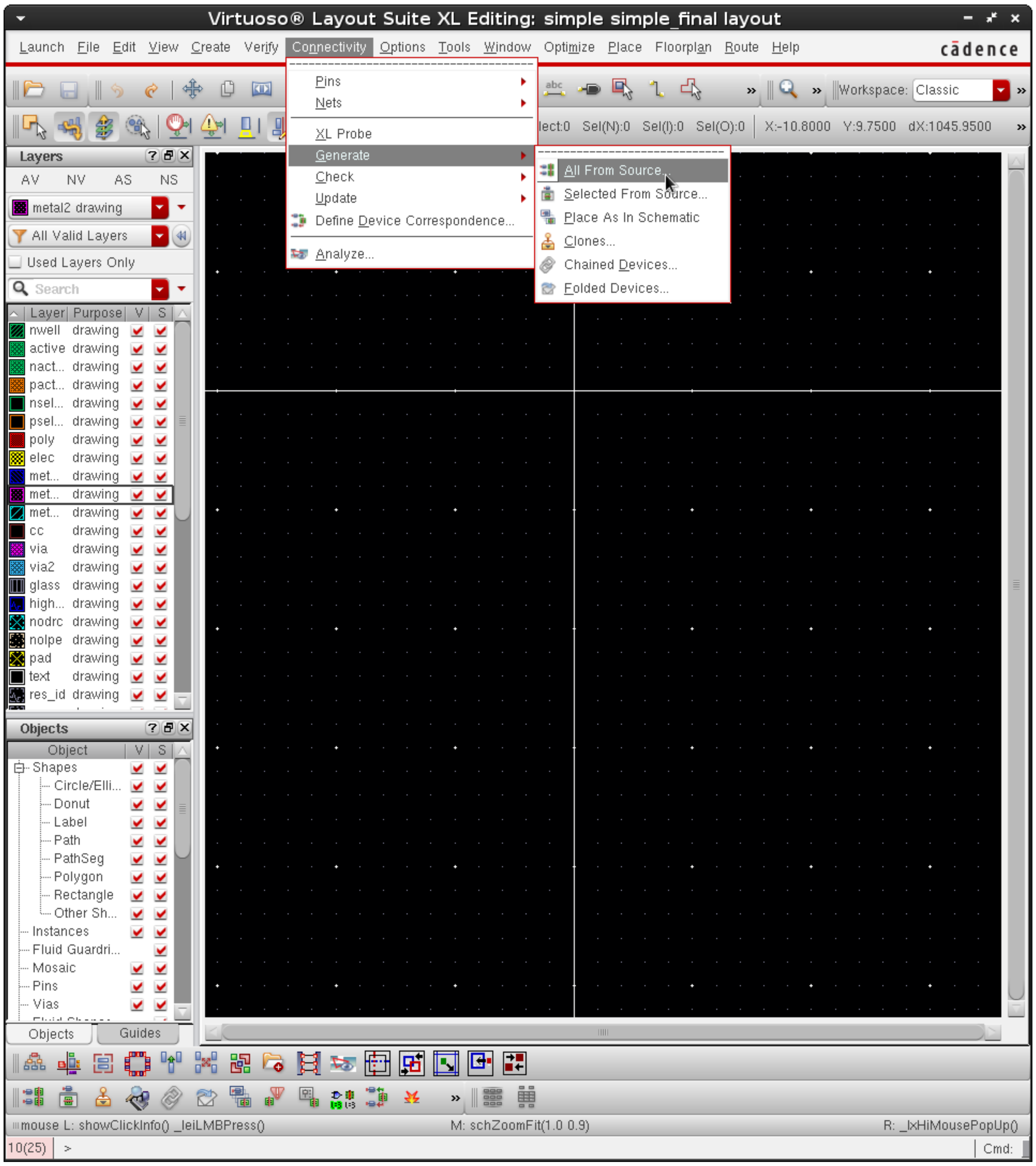
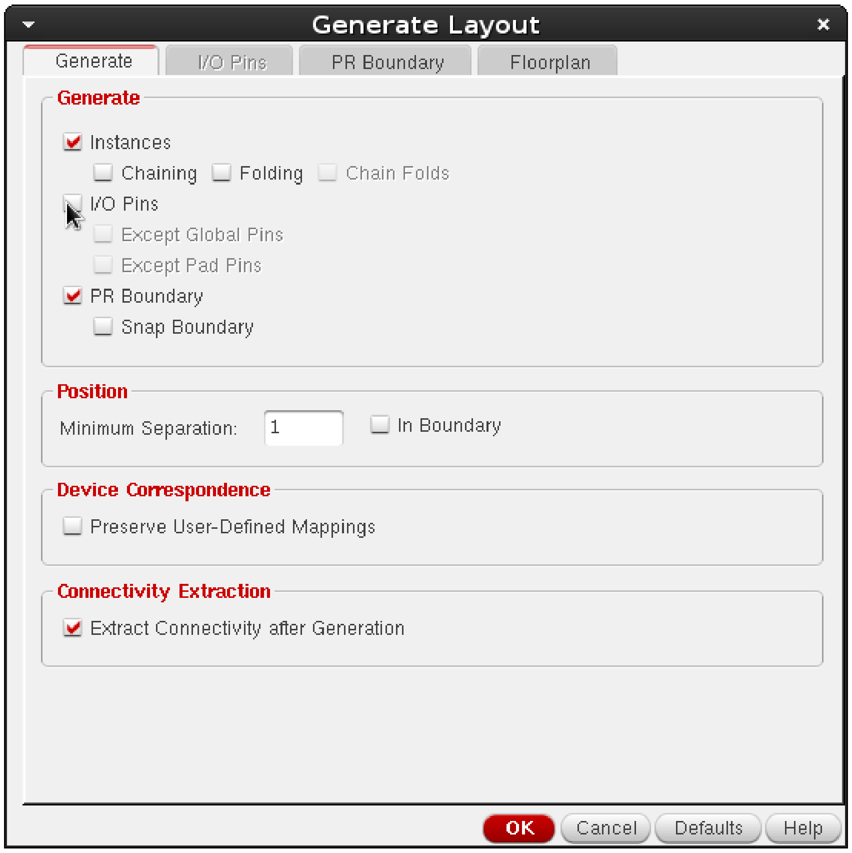

Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
Pins
Nets
XL Probe
Generate
Check
Update
Define Device Correspondence...
Analyze...
All From Source...
Selected From Source...
Place As In Schematic
Clones...
Chained Devices...
Folded Devices...
Workspace: Classic
Layers
Objects
Guides

Figure 216: Start "Generate Layout" of Virtuoso.


Generate Layout

Generate | I/O Pins | PR Boundary | Floorplan

Generate

- [✓] Instances
  - [ ] Chaining
  - [ ] Folding
  - [ ] Chain Folds
- [ ] I/O Pins
  - [ ] Except Global Pins
  - [ ] Except Pad Pins
- [✓] PR Boundary
  - [ ] Snap Boundary

Position

Minimum Separation: 1 [ ] In Boundary

Device Correspondence

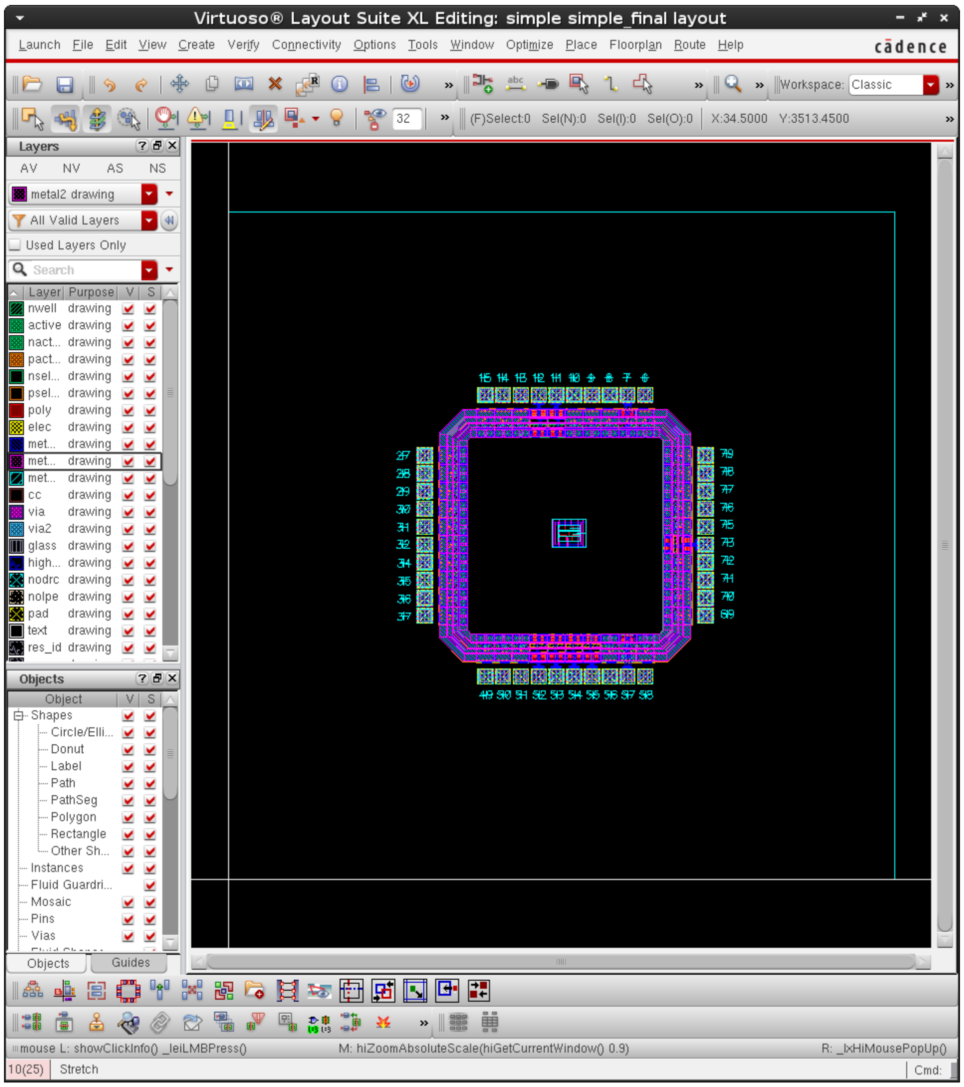
- [ ] Preserve User-Defined Mappings

Connectivity Extraction

- [✓] Extract Connectivity after Generation

Figure 217: "Generate Layout" window of Virtuoso.

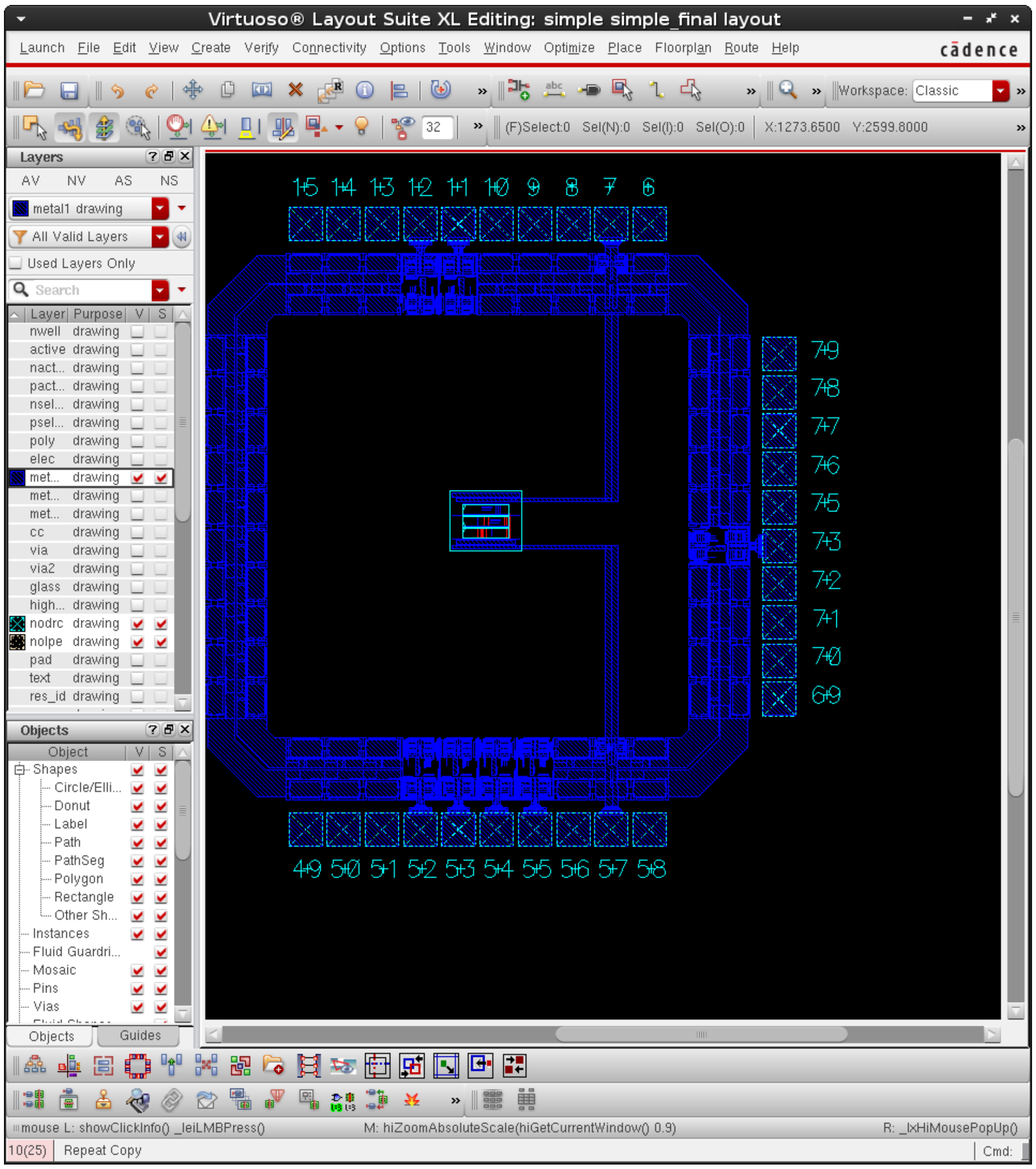
Then place the pad frame ("Frame1_38") inside the "PR Boundary", and the core of the chip ("gpca40p") inside the pad frame.


Virtuoso® Layout Suite XL Editing: simple simple_final layout
Launch File Edit View Create Verify Connectivity Options Tools Window Optimize Place Floorplan Route Help
cādence
Workspace: Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X:34.5000 Y:3513.4500
Layers
metal2 drawing
All Valid Layers
Used Layers Only
Search
Objects
Shapes
Instances
Pins
Vias
Guides

mouse L: showClickInfo() _leiLMBPress()
M: hiZoomAbsoluteScale(hiGetCurrentWindow() 0.9)
R: _lxHiMousePopUp()
10(25) Stretch
Cmd:

Figure 218: Component placement in Virtuoso.

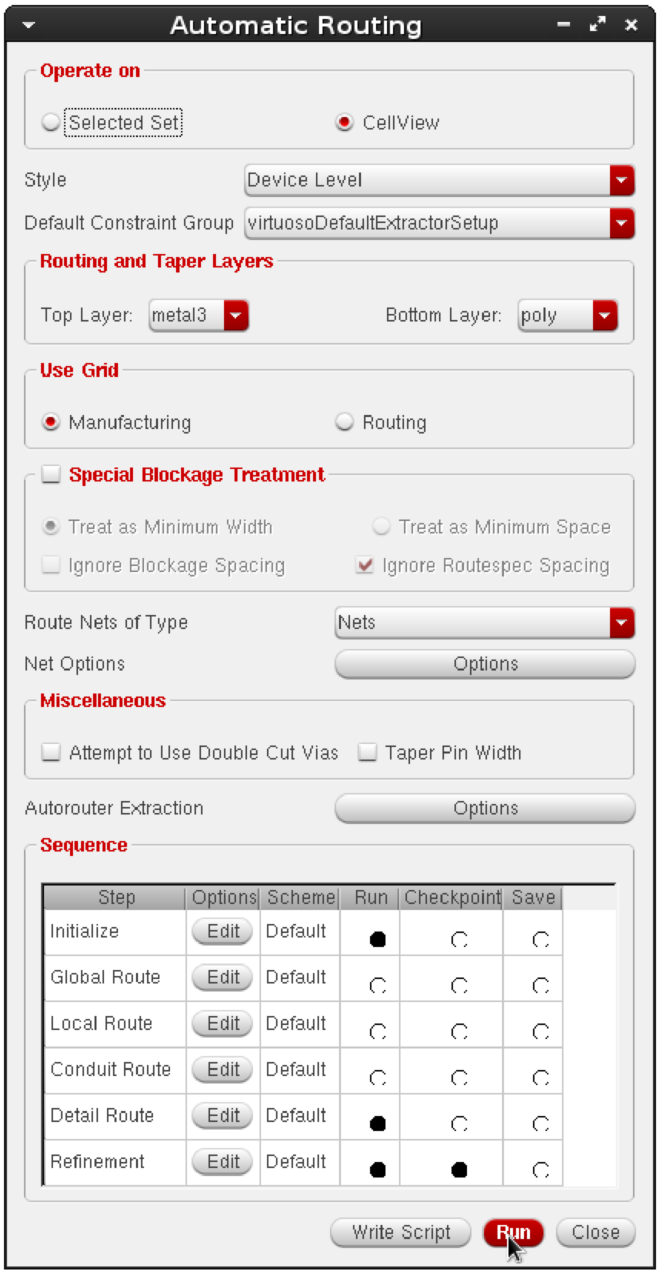
Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.



Layer	Purpose
nwell	drawing
active	drawing
nact...	drawing
pact...	drawing
nsel...	drawing
psel...	drawing
poly	drawing
elec	drawing
met...	drawing
met...	drawing
met...	drawing
cc	drawing
via	drawing
via2	drawing
glass	drawing
high...	drawing
nodrc	drawing
nolpe	drawing
pad	drawing
text	drawing
res_id	drawing

Figure 219: Connect the "add!" (Pad 57) and "god!" (Pad 7) to the power ring.

Then invoke Automatic Routing.



Automatic Routing

Operate on

Selected Set [✓ CellView]

Style: Device Level

Default Constraint Group: virtuosoDefaultExtractorSetup

Routing and Taper Layers

Top Layer: metal3 Bottom Layer: poly

Use Grid

[✓ Manufacturing] Routing

☐ Special Blockage Treatment

[✓ Treat as Minimum Width] Treat as Minimum Space

☐ Ignore Blockage Spacing ✓ Ignore Routespec Spacing

Route Nets of Type: Nets

Net Options: Options

Miscellaneous

☐ Attempt to Use Double Cut Vias ☐ Taper Pin Width

Autorouter Extraction: Options

Sequence

Step	Options	Scheme	Run	Checkpoint	Save
Initialize	Edit	Default	●	◌	◌
Global Route	Edit	Default	◌	◌	◌
Local Route	Edit	Default	◌	◌	◌
Conduit Route	Edit	Default	◌	◌	◌
Detail Route	Edit	Default	●	◌	◌
Refinement	Edit	Default	●	●	◌

Write Script Run Close

Figure 220: "Automatic Routing" window of Virtuoso.

The following is the final layout. Fill the empty space with poly fills to meet minimum polysilicon density required by AMI05 [35].

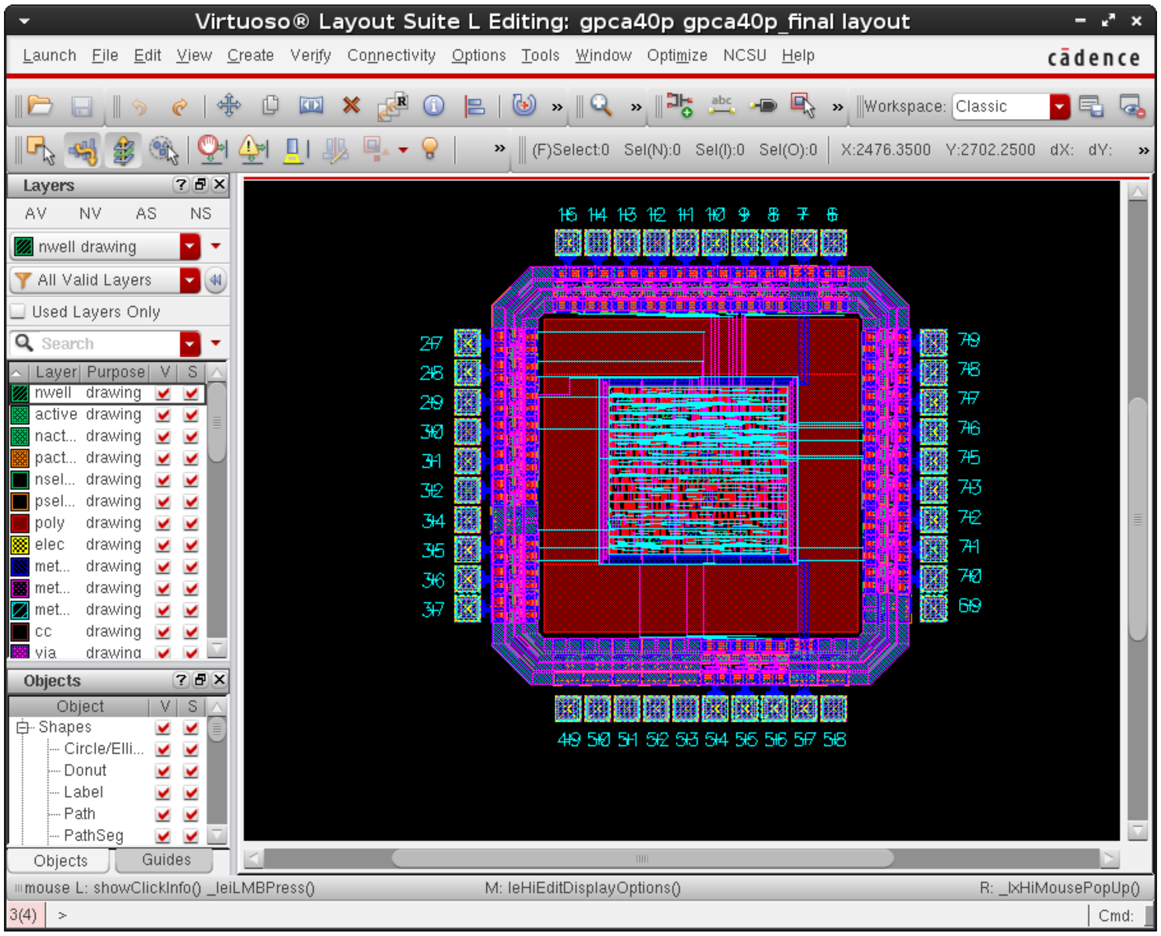
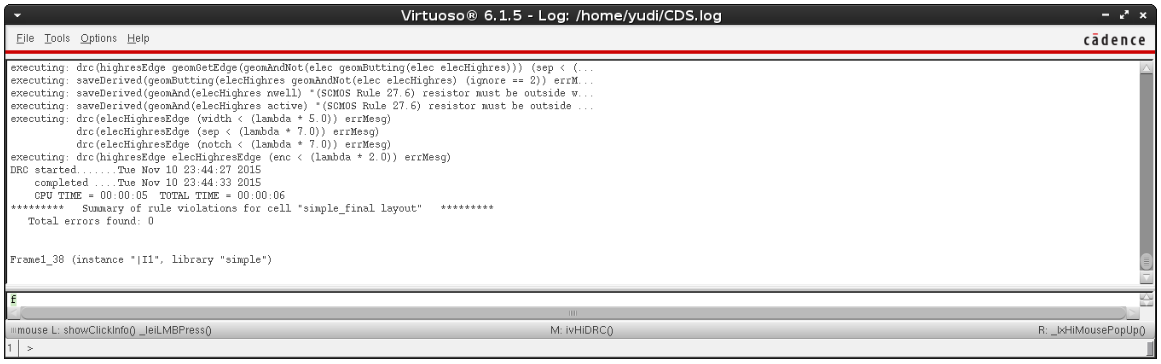


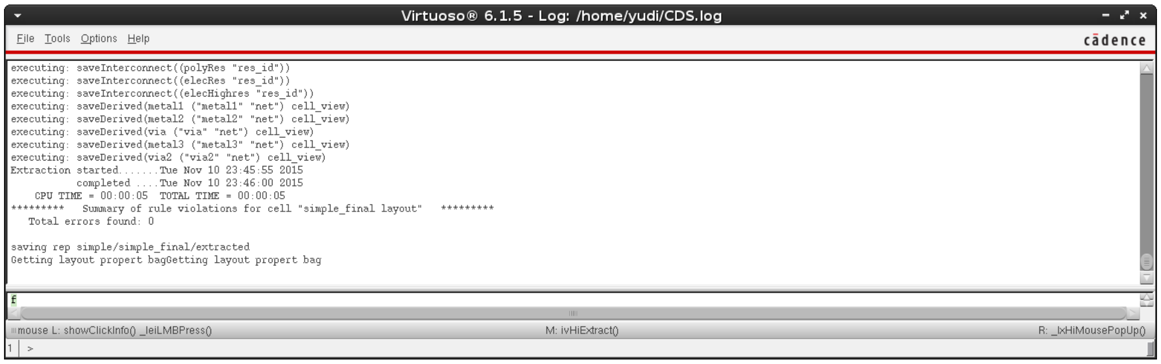
Figure 221: Final layout in Virtuoso.

DRC, Extract, LVS. They should give no error or warning.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File
Tools
Options
Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiDRC()
R: _lxHiMousePopUp()

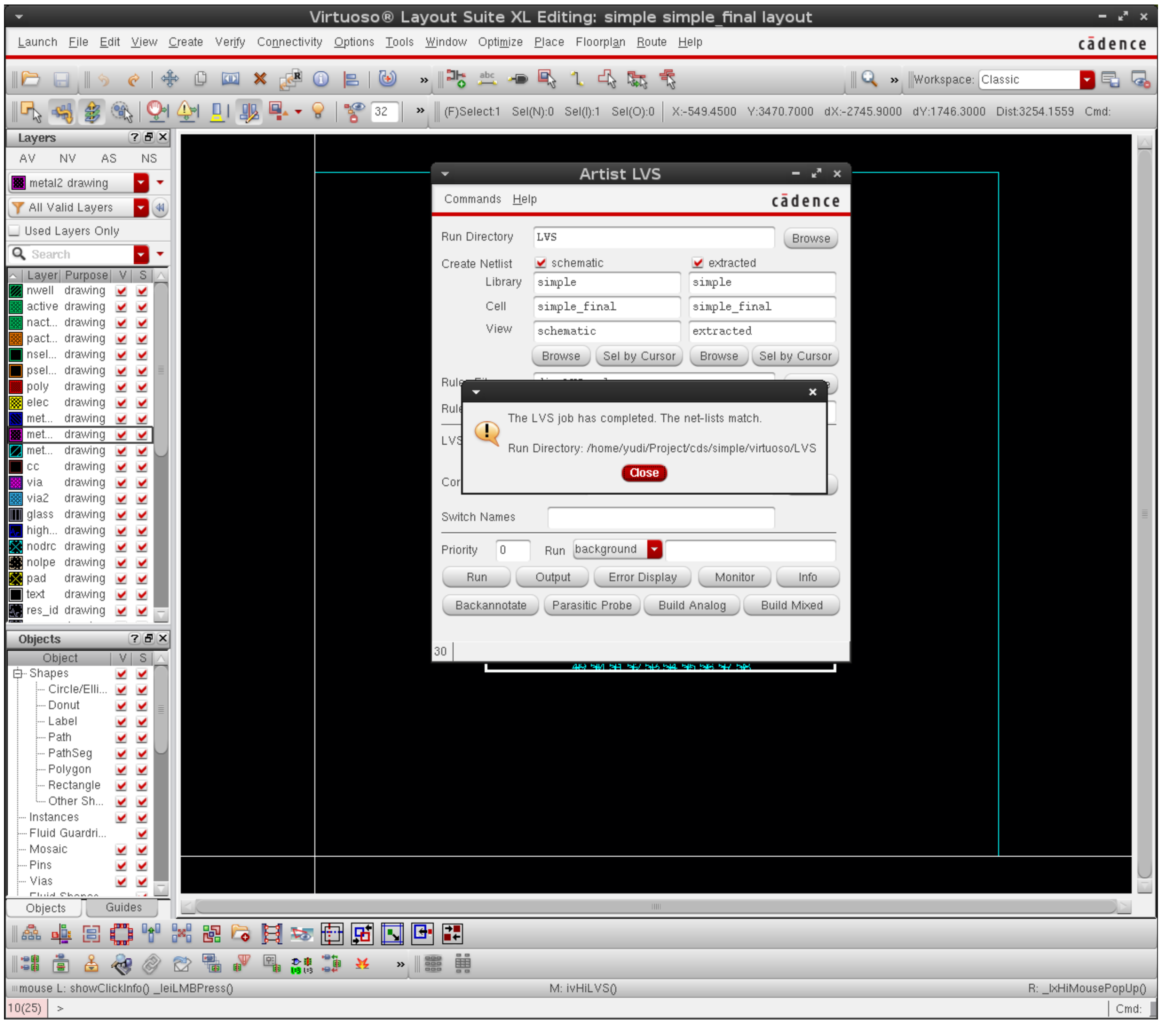
```
executing: drc(highresEdge geomGetEdge(geomAndNot(elec geomButting(elec elecHighres))) (sep < (...
executing: saveDerived(geomButting(elecHighres geomAndNot(elec elecHighres) (ignore == 2)) errM...
executing: saveDerived(geomAnd(elecHighres nwell) "(SCMOS Rule 27.6) resistor must be outside w...
executing: saveDerived(geomAnd(elecHighres active) "(SCMOS Rule 27.6) resistor must be outside ...
executing: drc(elecHighresEdge (width < (lambda * 5.0)) errMesg)
           drc(elecHighresEdge (sep < (lambda * 7.0)) errMesg)
           drc(elecHighresEdge (notch < (lambda * 7.0)) errMesg)
executing: drc(highresEdge elecHighresEdge (enc < (lambda * 2.0)) errMesg)
DRC started.......Tue Nov 10 23:44:27 2015
    completed ....Tue Nov 10 23:44:33 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:06
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0


Frame1_38 (instance "|I1", library "simple")
```

Figure 222: Report of successful DRC.


Virtuoso® 6.1.5 - Log: /home/yudi/CDS.log
File
Tools
Options
Help
cādence
mouse L: showClickInfo() _leiLMBPress()
M: ivHiExtract()
R: _lxHiMousePopUp()

```
executing: saveInterconnect((polyRes "res_id"))
executing: saveInterconnect((elecRes "res_id"))
executing: saveInterconnect((elecHighres "res_id"))
executing: saveDerived(metal1 ("metal1" "net") cell_view)
executing: saveDerived(metal2 ("metal2" "net") cell_view)
executing: saveDerived(via ("via" "net") cell_view)
executing: saveDerived(metal3 ("metal3" "net") cell_view)
executing: saveDerived(via2 ("via2" "net") cell_view)
Extraction started.......Tue Nov 10 23:45:55 2015
           completed ....Tue Nov 10 23:46:00 2015
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:05
*********   Summary of rule violations for cell "simple_final layout"   *********
   Total errors found: 0

saving rep simple/simple_final/extracted
Getting layout propert bagGetting layout propert bag
```

Figure 223: Report of successful Extract.



Virtuoso® Layout Suite XL Editing: simple simple_final layout

Workspace: Classic

(F)Select:1 Sel(N):0 Sel(I):1 Sel(O):0 X:-549.4500 Y:3470.7000 dX:-2745.9000 dY:1746.3000 Dist:3254.1559 Cmd:

metal2 drawing
All Valid Layers
Used Layers Only

Layer	Purpose	V	S
nwell	drawing	✓	✓
active	drawing	✓	✓
nact...	drawing	✓	✓
pact...	drawing	✓	✓
nsel...	drawing	✓	✓
psel...	drawing	✓	✓
poly	drawing	✓	✓
elec	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
met...	drawing	✓	✓
cc	drawing	✓	✓
via	drawing	✓	✓
via2	drawing	✓	✓
glass	drawing	✓	✓
high...	drawing	✓	✓
nodrc	drawing	✓	✓
nolpe	drawing	✓	✓
pad	drawing	✓	✓
text	drawing	✓	✓
res_id	drawing	✓	✓

Object	V	S
Shapes	✓	✓
Circle/Elli...	✓	✓
Donut	✓	✓
Label	✓	✓
Path	✓	✓
PathSeg	✓	✓
Polygon	✓	✓
Rectangle	✓	✓
Other Sh...	✓	✓
Instances	✓	✓
Fluid Guardri...		✓
Mosaic	✓	✓
Pins	✓	✓
Vias	✓	✓

Artist LVS

Run Directory: LVS

Create Netlist	✓ schematic	✓ extracted
Library	simple	simple
Cell	simple_final	simple_final
View	schematic	extracted

Switch Names:

Priority: 0 Run: background

The LVS job has completed. The net-lists match.

Run Directory: /home/yudi/Project/cds/simple/virtuoso/LVS

mouse L: showClickInfo() _leiLMBPress() M: ivHiLVS() R: _lxHiMousePopUp()

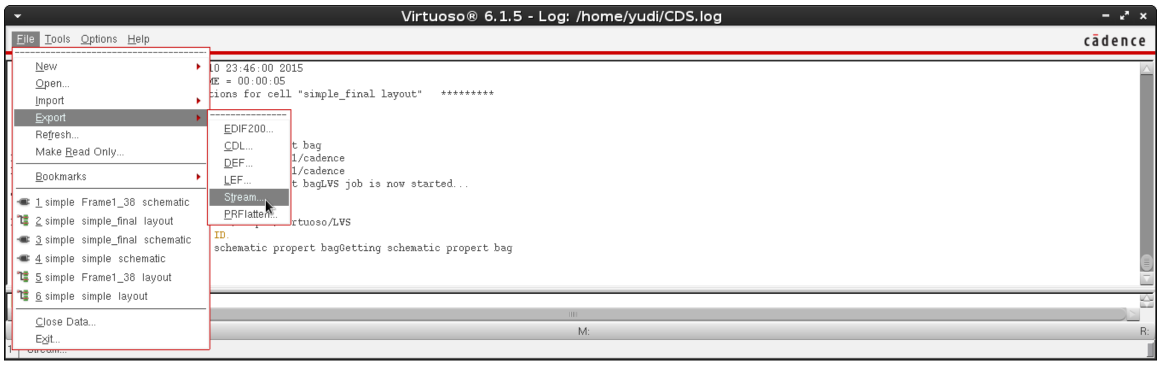
10(25) > Cmd:

Figure 224: "Artist LVS" successful LVS prompt.

f) Export GDSII Stream File ("simple_final.gds")

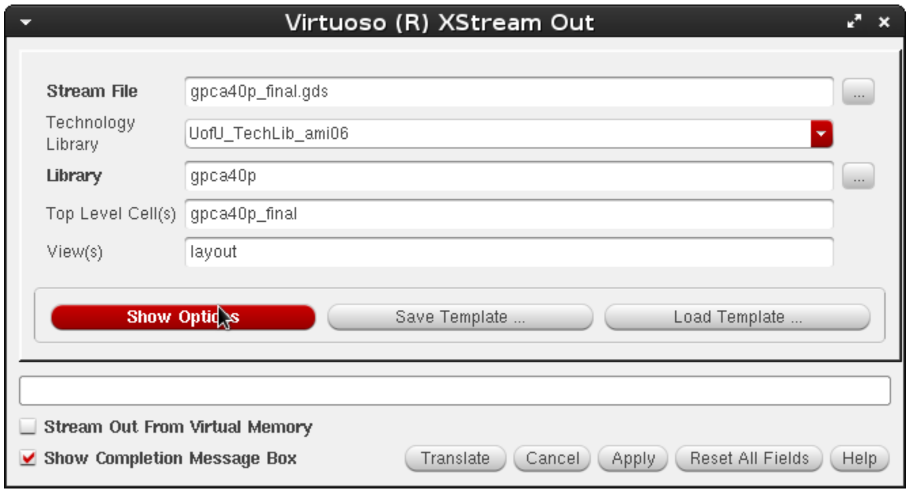
This procedure is shown in Figure 225 to Figure 230. This is the file to be sent for fabrication.

This is the file to be sent for fabrication.



```
10 23:46:00 2015
ME = 00:00:05
tions for cell "simple_final layout"   *********

t bag
1/cadence
1/cadence
t bagLVS job is now started...

rtuoso/LVS
ID.
schematic propert bagGetting schematic propert bag
```

Figure 225: Start "Stream Out" from Virtuoso Layout Suite.

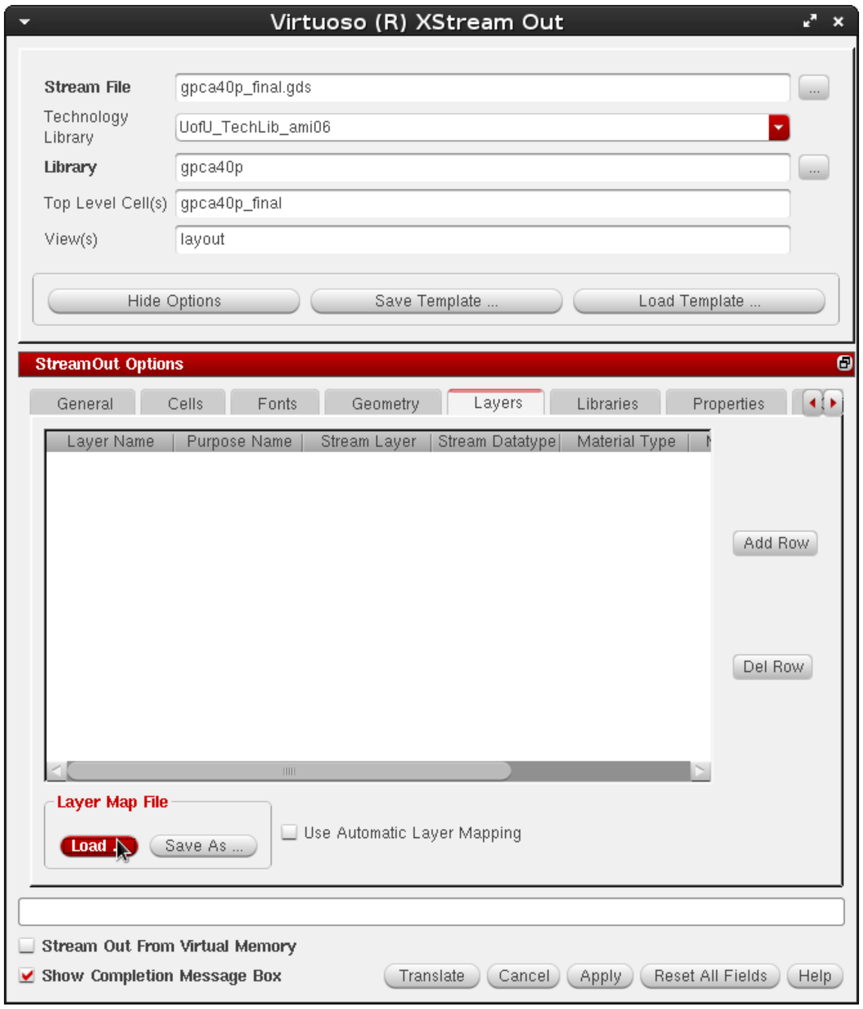


Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

☐ Stream Out From Virtual Memory
✓ Show Completion Message Box

Figure 226: "Show Options" in "XStream Out" window of Virtuoso.

"Show Options", then tab "Layers", "Load ..."

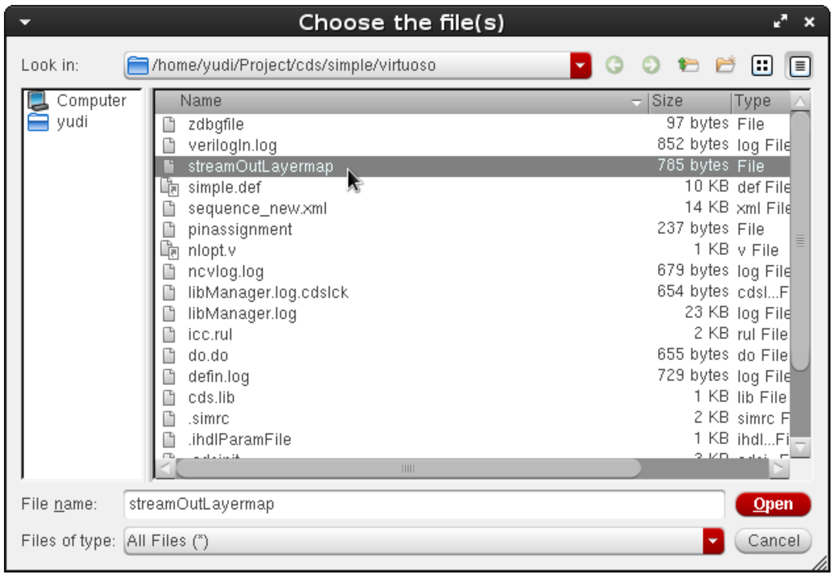


Stream File: gpca40p_final.gds
Technology Library: UofU_TechLib_ami06
Library: gpca40p
Top Level Cell(s): gpca40p_final
View(s): layout

☐ Use Automatic Layer Mapping
☐ Stream Out From Virtual Memory
☑ Show Completion Message Box

Figure 227: "Load..." in "XStream Out" window of Virtuoso.

Select "streamOutLayermap" then hit "Open"



/home/yudi/Project/cds/simple/virtuoso

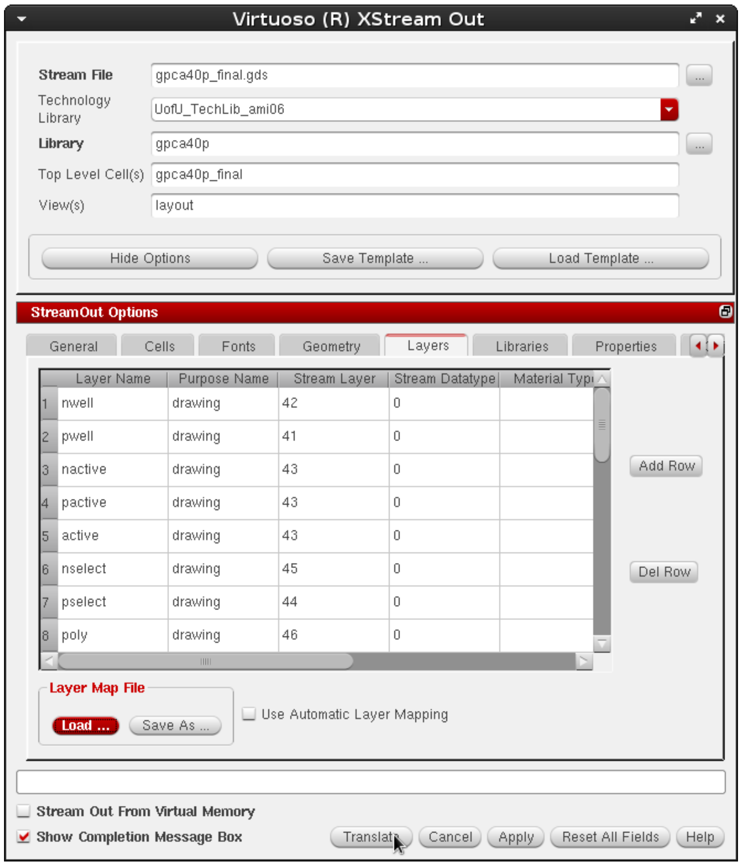
Name	Size	Type
zdbgfile	97 bytes	File
verilogIn.log	852 bytes	log File
streamOutLayermap	785 bytes	File
simple.def	10 KB	def File
sequence_new.xml	14 KB	xml File
pinassignment	237 bytes	File
nlopt.v	1 KB	v File
ncvlog.log	679 bytes	log File
libManager.log.cdslck	654 bytes	cdsl...F
libManager.log	23 KB	log File
icc.rul	2 KB	rul File
do.do	655 bytes	do File
defin.log	729 bytes	log File
cds.lib	1 KB	lib File
.simrc	2 KB	simrc F
.ihdlParamFile	1 KB	ihdl...Fi

File name: streamOutLayermap

Files of type: All Files (*)

Figure 228: Choose file for the "XStream Out".

Click "Translate".


Virtuoso (R) XStream Out

Stream File	gpca40p_final.gds
Technology Library	UofU_TechLib_ami06
Library	gpca40p
Top Level Cell(s)	gpca40p_final
View(s)	layout

	Layer Name	Purpose Name	Stream Layer	Stream Datatype	Material Type
1	nwell	drawing	42	0	
2	pwell	drawing	41	0	
3	nactive	drawing	43	0	
4	pactive	drawing	43	0	
5	active	drawing	43	0	
6	nselect	drawing	45	0	
7	pselect	drawing	44	0	
8	poly	drawing	46	0	

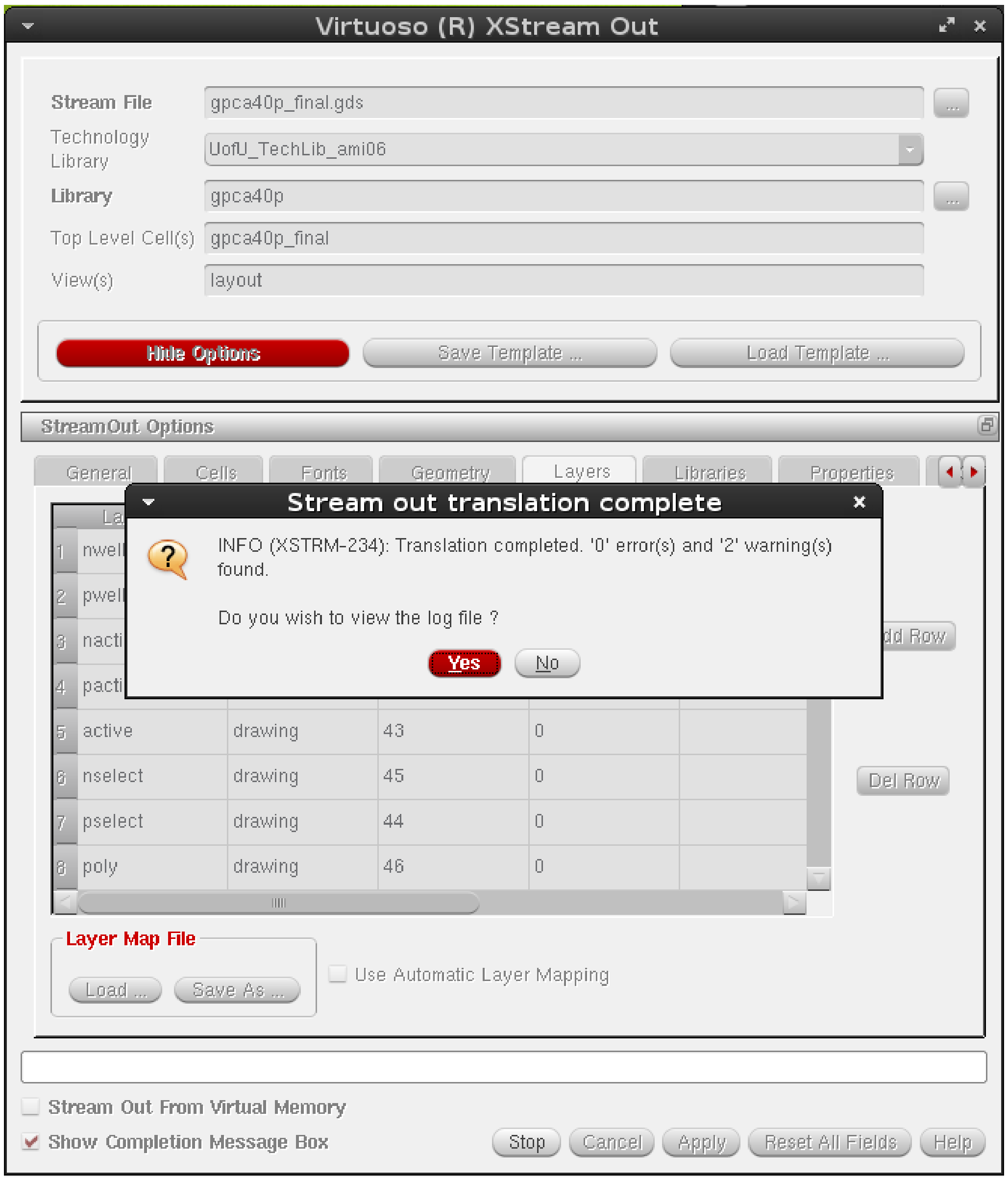
[ ] Use Automatic Layer Mapping

[ ] Stream Out From Virtual Memory

[✓] Show Completion Message Box

Figure 229: "Translate" in "XStream Out" window of Virtuoso.

It should produce no error and the only warning is about "nodrc:drawing" (and a possible overwriting existing file warning).


Virtuoso (R) XStream Out

Stream File	gpca40p_final.gds
Technology Library	UofU_TechLib_ami06
Library	gpca40p
Top Level Cell(s)	gpca40p_final
View(s)	layout

Stream out translation complete

INFO (XSTRM-234): Translation completed. '0' error(s) and '2' warning(s) found.

Do you wish to view the log file ?

1	nwell			
2	pwell			
3	nacti			
4	pacti			
5	active	drawing	43	0
6	nselect	drawing	45	0
7	pselect	drawing	44	0
8	poly	drawing	46	0

☐ Use Automatic Layer Mapping

☐ Stream Out From Virtual Memory

✓ Show Completion Message Box

Figure 230: "Stream out translation complete" successful prompt.

Design Submission

In this section, the procedure for MOSIS submission with the MOSIS Educational Program (MEP) is given as a reference. More information is available in documents from MOSIS [37], and adapted from [38].

1) Fill in "MOSIS New Project Request Form":

Run Type: Shared IC Fabrication Run

Design Rules: Scalable CMOS

Technology: SCN3M_SUBM (if the second layer of poly is not used); SCN3ME_SUBM (if the second layer of poly is used).

Design Name and Password

Export Control: Standard

Substrate: none

Needs Library Installation: No

IP Included: none

Fill Authorized: Yes

Foundry: On Semi

Intended Disposition: Research

Design Size X and Y: Size including pads

Pad Count: How many pads used in the design (including signals and power)

Quantity Packaged: 5

Package Name: Depends on the design

Rotation in Package: None

Bonding Diagram Supplier: MOSIS

Downbond Locations: None

Quantity Unpackaged: 0

2) Fill in "Fabricate Form":

Go to Project Request -> Fabricate

Layout Transfer Method: I will upload layout via secure web form (HTTPS)

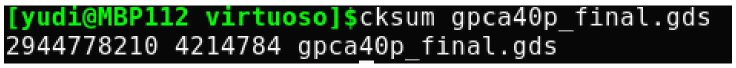
Compression/Encryption: Uncompressed

Generate the checksum and Count for the GDS file (Figure 231)

Layout Status: Final

Layout Format: GDS

Top Structure: the name of the top-level (e.g. "simple")



```
[yudi@MBP112 virtuoso]$cksum gpca40p_final.gds
2944778210 4214784 gpca40p_final.gds
```

Figure 231: Generating checksum with GNU cksum

Results

The extended new design achieves higher throughput then the original design while fully meets the design constrains. The result is summarized in Table 8. The design has been sent to MOSIS for fabrication.

Table 8: Implementation Summary of the Pipelined Design

	Original	**Extended**
Maximum delay (ns)	44.13	14.24
Maximum frequency (MHz)	22.7	70.2
Maximum throughput (million operations per second)	22.7	70.20

Conclusion

In this chapter, the concept of pipelining is briefly discussed. The original pipeline array does not discuss the implementation of the intermediate stage registers used in the pipeline array. This is a merely a combinational circuit. This array is improved to include intermediate stage registers. Instead of a pure combination circuit, it is now a sequential circuit and this circuit also has a clock. Such an improvement will help in the development of advanced pipelined arithmetic processors. The array is also implemented on FPGA and VLSI. The design has been sent to MOSIS for fabrication. The behavior Verilog code for this design is listed in the Appendix.

CHAPTER 5 SUMMARY AND CONCLUSION

Introduction

In this thesis, a VLSI implementation of a generalized pipeline array has been discussed and detailed procedure for the implementation is included. The procedure requires the simulation on FPGA. The approaches for extending the array so as to meet 40-pin requirements of the MOSIS design are discussed. The parameters such as expected delays, size, and memory have been met in the design. Then the original design is further extended for pipelining operation. The design has been sent to MOSIS for fabrication. The chip will be tested once we get the fabricated chip back.

Summary and Conclusion

A generalized pipeline array appeared in IEEE transaction in 1974. The array appeared in a few textbooks on computer arithmetic. From time to time, a number of papers appeared which reflected the modifications of this array. The objective of this thesis is to present the design and VLSI implementation of this array. The array can add, subtract, multiply, divide, square and square root of binary numbers. In this thesis, we suggest various extensions upon the original design, and step-by-step procedures by which the design can be sent to MOSIS and to get the fabricated chip back.

In Chapter 2, the array has been extended from 5 rows to 7 rows so that the extended operations can be performed. In particular, a procedure is developed by which the design and the implementation methodologies are suitable for 40 pin and 500 nm technologies.

In Chapter 3, an algorithm has been developed by which one can predict and advance the maximum size and performance of the array. A procedure for VLSI implementation of such a design using such an algorithm is also given.

In Chapter 4, the extension of pipelining is conducted based on the original design to increase data processing throughput. A procedure for VLSI implementation of the pipelined design is also given.

In particular, in order to achieve the following operations, the parameters of the design are listed in Table 9. The derivation process is given in the thesis.

It is hoped that the design and implementation done here will go a long way in the development of arithmetic units of advanced processors.

Table 9: Result of Extended generalized pipeline cellular array

Operations	# of bytes	# of rows	# of cols	Area (μm^2)
Multiply	1	8	17	523060
Divide	2	8	17	523060
Square	1 by 1	8	17	523060
Square root	2 by 1	9	19	647571

Contribution

The main contributions of this thesis are:

1. The review of existing literatures with the view to develop advanced arithmetic processors. Such processors can possibly be used in future computers.

2. Development of algorithms which can improve the design of arithmetic processors based on the number of rows or columns with a view to have a specified chip area and the number of input/output pins.

3. Extension of pipeline arrays so as to include pipelined part.

4. Development of unified procedure for VLSI implementation of chips which can be sent to MOSIS for fabrication and get the fabricated chip back and tested. Please note that the developed procedure herein works if the Verilog code is synthesizable. Future efforts are needed so as to decide in advance whether the code is synthesizable or not. If not, the code can be modified so as to be synthesizable.

This thesis first introduces the generalized cellular pipeline array, followed by analysis of various extensions of the original design, which makes the design more suitable for modern arithmetic processors, including extension from the original 5-row design to 7-row design (and any number of rows beyond), an algorithm developed to estimate the maximum number of rows in advance with given technology constrains, and an optimized pipeline implementation based on the original design. These designs can be used as the arithmetic unit within a project, and it also provides references for modern arithmetic processors designs.

The second part of this thesis introduces a step-by-step procedure by which the VLSI implementation is carried out, using the generalized cellular pipeline array as a case study in this process. This design and implementation procedure can serve as a reference material for VLSI designers and students in digital design courses in universities.

Problems for Future Work

Due to the scope of this thesis, since the requirement of timing is not demanding, no timing analysis and optimization during physical implementation is conducted. To further optimize the operation of the pipeline array, timing analysis and optimization is to be conducted in the future.

Due to the technical constrains when conducting this thesis, only 500 nm technology is available. In addition, the chip area is limited to 1500 µm by 1500 µm and the pin count is limited to 40 pins. When new technologies are available in the future, the estimation algorithm is to be extended to adapt the new technologies.

Due to the scope of this thesis, since the requirement of timing is not demanding, the pipeline is not fully balanced and optimized. To further optimize the operation of the pipeline array, techniques such as timing analysis are to be incorporated to further optimize the operation of the pipeline.

Problems such as “parallelism versus pipelining, help of FPGAs in parallelism, using 2 Spartan 3s in parallel” are also possible works of future study.

APPENDIX A Verilog Code for the 7 Row Extension

```
// Verilog implementation of "A GENERALIZED PIPELINE CELLULAR
ARRAY" by
// Harpreet Singh, Shashank Kamthan, Dharma Agarwal, Lubna
Alazzawi
// All codes here are imitating what's described in the papers
given by Dr. Singh, and it is claimed by Dr. Singh to be at least
useful at all
// top-level entity: "gpca"

// arithmetic cell
module ac
(
input wire  A, B, C, X, F, C1,
output wire S, D, E, C0
);

assign S = ((A ^ (B ^ X) ^ C1) & F) | (A & ~F);
assign C0 = ((B ^ X) & (A | C1)) | (A & C1);
assign D = C & (B | F);
assign E = B | (C & F);
endmodule

module cc
(
input wire  X, P, C0,
output wire F
);

assign F = (C0 & X) | (P & ~X);
endmodule

module gpca
(
input wire        X,
input wire [1:7]  P,
input wire [1:9]  B, C,
input wire [1:14] A,
output wire [1:7] F,
output wire [1:15]   S
);

wire [1:7]  FI;
wire [1:3]  C1;
wire [1:5]  C2;
wire [1:7]  C3;
wire [1:9]  C4;
wire [1:11] C5;
wire [1:13] C6;
wire [1:15] C7;
wire [1:3]  S1;
wire [1:5]  S2;
wire [1:7]  S3;
wire [1:9]  S4;
wire [1:11] S5;
wire [1:13] S6;
wire [1:15] S7;
wire [1:3]  D1;
wire [1:5]  D2;
wire [1:7]  D3;
wire [1:9]  D4;
wire [1:11] D5;
wire [1:13] D6;
wire [1:3]  E1;
wire [1:5]  E2;
wire [1:7]  E3;
wire [1:9]  E4;
wire [1:11] E5;
wire [1:13] E6;

assign F[1] = C1[1];
assign F[2] = C2[1];
assign F[3] = C3[1];
assign F[4] = C4[1];
assign F[5] = C5[1];
assign F[6] = C6[1];
assign F[7] = C7[1];

assign S = S7;

// control cells (X,P,C0 / F)
cc cc1(X, P[1], C1[1], FI[1]);
cc cc2(X, P[2], C2[1], FI[2]);
cc cc3(X, P[3], C3[1], FI[3]);
cc cc4(X, P[4], C4[1], FI[4]);
cc cc5(X, P[5], C5[1], FI[5]);
cc cc6(X, P[6], C6[1], FI[6]);
cc cc7(X, P[7], C6[1], FI[7]);

// arithmetic cells of row 1(A,B,C,X,F,C1/S,D,E,C0)
ac ac11(.A(1'b0), .B(B[1]),   .C(C[1]),   .X(X),   .F(FI[1]),
   .C1(C1[2]), .S(S1[1]),  .D(D1[1]),  .E(E1[1]),
   .C0(C1[1]));
ac ac12(.A(A[1]), .B(B[2]),   .C(C[2]),   .X(X),   .F(FI[1]),
   .C1(C1[3]), .S(S1[2]),  .D(D1[2]),  .E(E1[2]),
   .C0(C1[2]));
ac ac13(.A(A[2]), .B(B[3]),   .C(C[3]),   .X(X),   .F(FI[1]),
   .C1(X),     .S(S1[3]),  .D(D1[3]),  .E(E1[3]),
   .C0(C1[3]));

// arithmetic cells of row 2(A,B,C,X,F,C1/S,D,E,C0)
ac ac21(.A(S1[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[2]),  .C1(C2[2]), .S(S2[1]),  .D(D2[1]),  .E(E2[1]),
   .C0(C2[1]));
ac ac22(.A(S1[2]),  .B(D1[1]),  .C(E1[1]),  .X(X),
   .F(FI[2]),  .C1(C2[3]), .S(S2[2]),  .D(D2[2]),  .E(E2[2]),
   .C0(C2[2]));
ac ac23(.A(S1[3]),  .B(D1[2]),  .C(E1[2]),  .X(X),
   .F(FI[2]),  .C1(C2[4]), .S(S2[3]),  .D(D2[3]),  .E(E2[3]),
   .C0(C2[3]));
ac ac24(.A(A[3]), .B(D1[3]),  .C(E1[3]),  .X(X),   .F(FI[2]),
   .C1(C2[5]), .S(S2[4]),  .D(D2[4]),  .E(E2[4]),
   .C0(C2[4]));
ac ac25(.A(A[4]), .B(B[4]),   .C(C[4]),   .X(X),   .F(FI[2]),
   .C1(X),     .S(S2[5]),  .D(D2[5]),  .E(E2[5]),
   .C0(C2[5]));

// arithmetic cells of row 3(A,B,C,X,F,C1/S,D,E,C0)
ac ac31(.A(S2[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[3]),  .C1(C3[2]), .S(S3[1]),  .D(D3[1]),  .E(E3[1]),
   .C0(C3[1]));
ac ac32(.A(S2[2]),  .B(D2[1]),  .C(E2[1]),  .X(X),
   .F(FI[3]),  .C1(C3[3]), .S(S3[2]),  .D(D3[2]),  .E(E3[2]),
   .C0(C3[2]));
ac ac33(.A(S2[3]),  .B(D2[2]),  .C(E2[2]),  .X(X),
   .F(FI[3]),  .C1(C3[4]), .S(S3[3]),  .D(D3[3]),  .E(E3[3]),
   .C0(C3[3]));
ac ac34(.A(S2[4]),  .B(D2[3]),  .C(E2[3]),  .X(X),
   .F(FI[3]),  .C1(C3[5]), .S(S3[4]),  .D(D3[4]),  .E(E3[4]),
   .C0(C3[4]));
ac ac35(.A(S2[5]),  .B(D2[4]),  .C(E2[4]),  .X(X),
   .F(FI[3]),  .C1(C3[6]), .S(S3[5]),  .D(D3[5]),  .E(E3[5]),
   .C0(C3[5]));
ac ac36(.A(A[5]), .B(D2[5]),  .C(E2[5]),  .X(X),   .F(FI[3]),
   .C1(C3[7]), .S(S3[6]),  .D(D3[6]),  .E(E3[6]),
   .C0(C3[6]));
ac ac37(.A(A[6]), .B(B[5]),   .C(C[5]),   .X(X),   .F(FI[3]),
   .C1(X),     .S(S3[7]),  .D(D3[7]),  .E(E3[7]),
   .C0(C3[7]));

// arithmetic cells of row 4(A,B,C,X,F,C1/S,D,E,C0)
ac ac41(.A(S3[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[4]),  .C1(C4[2]), .S(S4[1]),  .D(D4[1]),  .E(E4[1]),
   .C0(C4[1]));
ac ac42(.A(S3[2]),  .B(D3[1]),  .C(E3[1]),  .X(X),
   .F(FI[4]),  .C1(C4[3]), .S(S4[2]),  .D(D4[2]),  .E(E4[2]),
   .C0(C4[2]));
ac ac43(.A(S3[3]),  .B(D3[2]),  .C(E3[2]),  .X(X),
   .F(FI[4]),  .C1(C4[4]), .S(S4[3]),  .D(D4[3]),  .E(E4[3]),
   .C0(C4[3]));
ac ac44(.A(S3[4]),  .B(D3[3]),  .C(E3[3]),  .X(X),
   .F(FI[4]),  .C1(C4[5]), .S(S4[4]),  .D(D4[4]),  .E(E4[4]),
   .C0(C4[4]));
ac ac45(.A(S3[5]),  .B(D3[4]),  .C(E3[4]),  .X(X),
   .F(FI[4]),  .C1(C4[6]), .S(S4[5]),  .D(D4[5]),  .E(E4[5]),
   .C0(C4[5]));
ac ac46(.A(S3[6]),  .B(D3[5]),  .C(E3[5]),  .X(X),
   .F(FI[4]),  .C1(C4[7]), .S(S4[6]),  .D(D4[6]),  .E(E4[6]),
   .C0(C4[6]));
ac ac47(.A(S3[7]),  .B(D3[6]),  .C(E3[6]),  .X(X),
   .F(FI[4]),  .C1(C4[8]), .S(S4[7]),  .D(D4[7]),  .E(E4[7]),
   .C0(C4[7]));
ac ac48(.A(A[7]), .B(D3[7]),  .C(E3[7]),  .X(X),   .F(FI[4]),
   .C1(C4[9]), .S(S4[8]),  .D(D4[8]),  .E(E4[8]),
   .C0(C4[8]));
ac ac49(.A(A[8]), .B(B[6]),   .C(C[6]),   .X(X),   .F(FI[4]),
   .C1(X),     .S(S4[9]),  .D(D4[9]),  .E(E4[9]),
   .C0(C4[9]));

// arithmetic cells of row 5(A,B,C,X,F,C1/S,D,E,C0)
ac ac51(.A(S4[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[5]),  .C1(C5[2]), .S(S5[1]),  .D(D5[1]),  .E(E5[1]),
   .C0(C5[1]));
ac ac52(.A(S4[2]),  .B(D4[1]),  .C(E4[1]),  .X(X),
   .F(FI[5]),  .C1(C5[3]), .S(S5[2]),  .D(D5[2]),  .E(E5[2]),
   .C0(C5[2]));
ac ac53(.A(S4[3]),  .B(D4[2]),  .C(E4[2]),  .X(X),
   .F(FI[5]),  .C1(C5[4]), .S(S5[3]),  .D(D5[3]),  .E(E5[3]),
   .C0(C5[3]));
ac ac54(.A(S4[4]),  .B(D4[3]),  .C(E4[3]),  .X(X),
   .F(FI[5]),  .C1(C5[5]), .S(S5[4]),  .D(D5[4]),  .E(E5[4]),
   .C0(C5[4]));
ac ac55(.A(S4[5]),  .B(D4[4]),  .C(E4[4]),  .X(X),
   .F(FI[5]),  .C1(C5[6]), .S(S5[5]),  .D(D5[5]),  .E(E5[5]),
   .C0(C5[5]));
ac ac56(.A(S4[6]),  .B(D4[5]),  .C(E4[5]),  .X(X),
   .F(FI[5]),  .C1(C5[7]), .S(S5[6]),  .D(D5[6]),  .E(E5[6]),
   .C0(C5[6]));
ac ac57(.A(S4[7]),  .B(D4[6]),  .C(E4[6]),  .X(X),
   .F(FI[5]),  .C1(C5[8]), .S(S5[7]),  .D(D5[7]),  .E(E5[7]),
   .C0(C5[7]));
ac ac58(.A(S4[8]),  .B(D4[7]),  .C(E4[7]),  .X(X),
   .F(FI[5]),  .C1(C5[9]), .S(S5[8]),  .D(D5[8]),  .E(E5[8]),
   .C0(C5[8]));
ac ac59(.A(S4[9]),  .B(D4[8]),  .C(E4[8]),  .X(X),
   .F(FI[5]),  .C1(C5[10]),.S(S5[9]),  .D(D5[9]),  .E(E5[9]),
   .C0(C5[9]));
ac ac5a(.A(A[9]),.B(D4[9]),  .C(E4[9]),  .X(X),   .F(FI[5]),
   .C1(C5[11]),.S(S5[10]),.D(D5[10]),.E(E5[10]),
   .C0(C5[10]));
ac ac5b(.A(A[10]),  .B(B[7]),   .C(C[7]),   .X(X),
   .F(FI[5]),  .C1(X),     .S(S5[11]),.D(D5[11]),.E(E5[11]),
   .C0(C5[11]));

// arithmetic cells of row 6(A,B,C,X,F,C1/S,D,E,C0)
ac ac61(.A(S5[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[6]),  .C1(C6[2]), .S(S6[1]),  .D(D6[1]),  .E(E6[1]),
   .C0(C6[1]));
ac ac62(.A(S5[2]),  .B(D5[1]),  .C(E5[1]),  .X(X),
   .F(FI[6]),  .C1(C6[3]), .S(S6[2]),  .D(D6[2]),  .E(E6[2]),
   .C0(C6[2]));
ac ac63(.A(S5[3]),  .B(D5[2]),  .C(E5[2]),  .X(X),
   .F(FI[6]),  .C1(C6[4]), .S(S6[3]),  .D(D6[3]),  .E(E6[3]),
   .C0(C6[3]));
ac ac64(.A(S5[4]),  .B(D5[3]),  .C(E5[3]),  .X(X),
   .F(FI[6]),  .C1(C6[5]), .S(S6[4]),  .D(D6[4]),  .E(E6[4]),
   .C0(C6[4]));
ac ac65(.A(S5[5]),  .B(D5[4]),  .C(E5[4]),  .X(X),
   .F(FI[6]),  .C1(C6[6]), .S(S6[5]),  .D(D6[5]),  .E(E6[5]),
   .C0(C6[5]));
ac ac66(.A(S5[6]),  .B(D5[5]),  .C(E5[5]),  .X(X),
   .F(FI[6]),  .C1(C6[7]), .S(S6[6]),  .D(D6[6]),  .E(E6[6]),
   .C0(C6[6]));
ac ac67(.A(S5[7]),  .B(D5[6]),  .C(E5[6]),  .X(X),
   .F(FI[6]),  .C1(C6[8]), .S(S6[7]),  .D(D6[7]),  .E(E6[7]),
   .C0(C6[7]));
ac ac68(.A(S5[8]),  .B(D5[7]),  .C(E5[7]),  .X(X),
   .F(FI[6]),  .C1(C6[9]), .S(S6[8]),  .D(D6[8]),  .E(E6[8]),
   .C0(C6[8]));
ac ac69(.A(S5[9]),  .B(D5[8]),  .C(E5[8]),  .X(X),
   .F(FI[6]),  .C1(C6[10]),.S(S6[9]),  .D(D6[9]),  .E(E6[9]),
   .C0(C6[9]));
ac ac6a(.A(S5[10]), .B(D5[9]),  .C(E5[9]),  .X(X),
   .F(FI[6]),  .C1(C6[11]),.S(S6[10]),.D(D6[10]),.E(E6[10]),
   .C0(C6[10]));
ac ac6b(.A(S5[11]), .B(D5[10]), .C(E5[10]), .X(X),
   .F(FI[6]),  .C1(C6[12]),.S(S6[11]),.D(D6[11]),.E(E6[11]),
   .C0(C6[11]));
ac ac6c(.A(A[11]),  .B(D5[11]), .C(E5[11]), .X(X),
   .F(FI[6]),  .C1(C6[13]),.S(S6[12]),.D(D6[12]),.E(E6[12]),
   .C0(C6[12]));
ac ac6d(.A(A[12]),  .B(B[8]),   .C(C[8]),   .X(X),
   .F(FI[6]),  .C1(X),     .S(S6[13]), .D(D6[13]),.E(E6[13]),
   .C0(C6[13]));

// arithmetic cells of row 7(A,B,C,X,F,C1/S,D,E,C0)
ac ac71(.A(S6[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[7]),  .C1(C7[2]), .S(S7[1]),  .D(), .E(),
   .C0(C7[1]));
ac ac72(.A(S6[2]),  .B(D6[1]),  .C(E6[1]),  .X(X),
   .F(FI[7]),  .C1(C7[3]), .S(S7[2]),  .D(), .E(),
   .C0(C7[2]));
ac ac73(.A(S6[3]),  .B(D6[2]),  .C(E6[2]),  .X(X),
   .F(FI[7]),  .C1(C7[4]), .S(S7[3]),  .D(), .E(),
   .C0(C7[3]));
ac ac74(.A(S6[4]),  .B(D6[3]),  .C(E6[3]),  .X(X),
   .F(FI[7]),  .C1(C7[5]), .S(S7[4]),  .D(), .E(),
   .C0(C7[4]));
ac ac75(.A(S6[5]),  .B(D6[4]),  .C(E6[4]),  .X(X),
   .F(FI[7]),  .C1(C7[6]), .S(S7[5]),  .D(), .E(),
   .C0(C7[5]));
ac ac76(.A(S6[6]),  .B(D6[5]),  .C(E6[5]),  .X(X),
   .F(FI[7]),  .C1(C7[7]), .S(S7[6]),  .D(), .E(),
   .C0(C7[6]));
ac ac77(.A(S6[7]),  .B(D6[6]),  .C(E6[6]),  .X(X),
   .F(FI[7]),  .C1(C7[8]), .S(S7[7]),  .D(), .E(),
   .C0(C7[7]));
ac ac78(.A(S6[8]),  .B(D6[7]),  .C(E6[7]),  .X(X),
   .F(FI[7]),  .C1(C7[9]), .S(S7[8]),  .D(), .E(),
   .C0(C7[8]));
ac ac79(.A(S6[9]),  .B(D6[8]),  .C(E6[8]),  .X(X),
   .F(FI[7]),  .C1(C7[10]),.S(S7[9]),  .D(), .E(),
   .C0(C7[9]));
ac ac7a(.A(S6[10]), .B(D6[9]),  .C(E6[9]),  .X(X),
   .F(FI[7]),  .C1(C7[11]),.S(S7[10]),.D(), .E(),
   .C0(C7[10]));
ac ac7b(.A(S6[11]), .B(D6[10]), .C(E6[10]), .X(X),
   .F(FI[7]),  .C1(C7[12]),.S(S7[11]),.D(), .E(),
   .C0(C7[11]));
ac ac7c(.A(S6[12]), .B(D6[11]), .C(E6[11]), .X(X),
   .F(FI[7]),  .C1(C7[13]),.S(S7[12]),.D(), .E(),
   .C0(C7[12]));
ac ac7d(.A(S6[13]), .B(D6[12]), .C(E6[12]), .X(X),
   .F(FI[7]),  .C1(C7[14]),.S(S7[13]),.D(), .E(),
   .C0(C7[13]));
ac ac7e(.A(A[13]),  .B(D6[13]), .C(E6[13]), .X(X),
   .F(FI[7]),  .C1(C7[15]),.S(S7[14]),.D(), .E(),
   .C0(C7[14]));
ac ac7f(.A(A[14]),  .B(B[9]),   .C(C[9]),   .X(X),
   .F(FI[7]),  .C1(X),     .S(S7[15]), .D(), .E(),
   .C0(C7[15]));
endmodule
```

APPENDIX B Verilog Code for the Extension to Meet Design Constrains

```
// Verilog implementation of "A GENERALIZED PIPELINE CELLULAR
ARRAY" by
// Harpreet Singh, Shashank Kamthan, Dharma Agarwal, Lubna
Alazzawi
// All codes here are imitating what's described in the papers
given by Dr. Singh, and it is claimed by Dr. Singh to be at least
useful at all
// top-level entity: "gpca40p"

// arithmetic cell
module ac
(
input wire  A, B, C, X, F, C1,
output wire S, D, E, C0
);

assign S = ((A ^ (B ^ X) ^ C1) & F) | (A & ~F);
assign C0 = ((B ^ X) & (A | C1)) | (A & C1);
assign D = C & (B | F);
assign E = B | (C & F);
endmodule

module cc
(
input wire  X, P, C0,
output wire F
);

assign F = (C0 & X) | (P & ~X);
endmodule

module gpca
(
input wire        X,
input wire [1:5]  P,
input wire [1:7]  B, C,
input wire [1:10] A,
output wire [1:5] F,
output wire [1:11]   S
);

wire [1:5]  FI;
wire [1:3]  C1;
wire [1:5]  C2;
wire [1:7]  C3;
wire [1:9]  C4;
wire [1:11] C5;
wire [1:3]  S1;
wire [1:5]  S2;
wire [1:7]  S3;
wire [1:9]  S4;
wire [1:3]  D1;
wire [1:5]  D2;
wire [1:7]  D3;
wire [1:9]  D4;
wire [1:3]  E1;
wire [1:5]  E2;
wire [1:7]  E3;
wire [1:9]  E4;

assign F[1] = C1[1];
assign F[2] = C2[1];
assign F[3] = C3[1];
assign F[4] = C4[1];
assign F[5] = C5[1];

// control cells (X,P,C0 / F)
cc cc1(X, P[1], C1[1], FI[1]);
cc cc2(X, P[2], C2[1], FI[2]);
cc cc3(X, P[3], C3[1], FI[3]);
cc cc4(X, P[4], C4[1], FI[4]);
cc cc5(X, P[5], C5[1], FI[5]);

// arithmetic cells of row 1(A,B,C,X,F,C1/S,D,E,C0)
ac ac11(.A(1'b0), .B(B[1]),   .C(C[1]),   .X(X),   .F(FI[1]),
   .C1(C1[2]), .S(S1[1]),  .D(D1[1]),  .E(E1[1]),
   .C0(C1[1]));
ac ac12(.A(A[1]), .B(B[2]),   .C(C[2]),   .X(X),   .F(FI[1]),
   .C1(C1[3]), .S(S1[2]),  .D(D1[2]),  .E(E1[2]),
   .C0(C1[2]));
ac ac13(.A(A[2]), .B(B[3]),   .C(C[3]),   .X(X),   .F(FI[1]),
   .C1(X),     .S(S1[3]),  .D(D1[3]),  .E(E1[3]),
   .C0(C1[3]));

// arithmetic cells of row 2(A,B,C,X,F,C1/S,D,E,C0)
ac ac21(.A(S1[1]),  .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[2]),  .C1(C2[2]), .S(S2[1]),  .D(D2[1]),  .E(E2[1]),
   .C0(C2[1]));
ac ac22(.A(S1[2]),   .B(D1[1]),  .C(E1[1]),  .X(X),
   .F(FI[2]),  .C1(C2[3]), .S(S2[2]),  .D(D2[2]),  .E(E2[2]),
   .C0(C2[2]));
ac ac23(.A(S1[3]),   .B(D1[2]),  .C(E1[2]),  .X(X),
   .F(FI[2]),  .C1(C2[4]), .S(S2[3]),  .D(D2[3]),  .E(E2[3]),
   .C0(C2[3]));
ac ac24(.A(A[3]), .B(D1[3]),  .C(E1[3]),  .X(X),   .F(FI[2]),
   .C1(C2[5]), .S(S2[4]),  .D(D2[4]),  .E(E2[4]),
   .C0(C2[4]));
ac ac25(.A(A[4]), .B(B[4]),   .C(C[4]),   .X(X),   .F(FI[2]),
   .C1(X),     .S(S2[5]),  .D(D2[5]),  .E(E2[5]),
   .C0(C2[5]));

// arithmetic cells of row 3(A,B,C,X,F,C1/S,D,E,C0)
ac ac31(.A(S2[1]),   .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[3]),  .C1(C3[2]), .S(S3[1]),  .D(D3[1]),  .E(E3[1]),
   .C0(C3[1]));
ac ac32(.A(S2[2]),   .B(D2[1]),  .C(E2[1]),  .X(X),
   .F(FI[3]),  .C1(C3[3]), .S(S3[2]),  .D(D3[2]),  .E(E3[2]),
   .C0(C3[2]));
ac ac33(.A(S2[3]),   .B(D2[2]),  .C(E2[2]),  .X(X),
   .F(FI[3]),  .C1(C3[4]), .S(S3[3]),  .D(D3[3]),  .E(E3[3]),
   .C0(C3[3]));
ac ac34(.A(S2[4]),   .B(D2[3]),  .C(E2[3]),  .X(X),
   .F(FI[3]),  .C1(C3[5]), .S(S3[4]),  .D(D3[4]),  .E(E3[4]),
   .C0(C3[4]));
ac ac35(.A(S2[5]),   .B(D2[4]),  .C(E2[4]),  .X(X),
   .F(FI[3]),  .C1(C3[6]), .S(S3[5]),  .D(D3[5]),  .E(E3[5]),
   .C0(C3[5]));
ac ac36(.A(A[5]), .B(D2[5]),  .C(E2[5]),  .X(X),   .F(FI[3]),
   .C1(C3[7]), .S(S3[6]),  .D(D3[6]),  .E(E3[6]),
   .C0(C3[6]));
ac ac37(.A(A[6]), .B(B[5]),   .C(C[5]),   .X(X),   .F(FI[3]),
   .C1(X),     .S(S3[7]),  .D(D3[7]),  .E(E3[7]),
   .C0(C3[7]));

// arithmetic cells of row 4(A,B,C,X,F,C1/S,D,E,C0)
ac ac41(.A(S3[1]),   .B(1'b0),   .C(1'b0),   .X(X),
   .F(FI[4]),  .C1(C4[2]), .S(S4[1]),  .D(D4[1]),  .E(E4[1]),
   .C0(C4[1]));
ac ac42(.A(S3[2]),   .B(D3[1]),  .C(E3[1]),  .X(X),
   .F(FI[4]),  .C1(C4[3]), .S(S4[2]),  .D(D4[2]),  .E(E4[2]),
   .C0(C4[2]));
ac ac43(.A(S3[3]),  .B(D3[2]), .C(E3[2]), .X(X),
   .F(FI[4]), .C1(C4[4]), .S(S4[3]), .D(D4[3]), .E(E4[3]),
   .C0(C4[3]));
ac ac44(.A(S3[4]),  .B(D3[3]), .C(E3[3]), .X(X),
   .F(FI[4]), .C1(C4[5]), .S(S4[4]), .D(D4[4]), .E(E4[4]),
   .C0(C4[4]));
ac ac45(.A(S3[5]),  .B(D3[4]), .C(E3[4]), .X(X),
   .F(FI[4]), .C1(C4[6]), .S(S4[5]), .D(D4[5]), .E(E4[5]),
   .C0(C4[5]));
ac ac46(.A(S3[6]),  .B(D3[5]), .C(E3[5]), .X(X),
   .F(FI[4]), .C1(C4[7]), .S(S4[6]), .D(D4[6]), .E(E4[6]),
   .C0(C4[6]));
ac ac47(.A(S3[7]),  .B(D3[6]), .C(E3[6]), .X(X),
   .F(FI[4]), .C1(C4[8]), .S(S4[7]), .D(D4[7]), .E(E4[7]),
   .C0(C4[7]));
ac ac48(.A(A[7]),.B(D3[7]), .C(E3[7]), .X(X),  .F(FI[4]),
   .C1(C4[9]),.S(S4[8]), .D(D4[8]), .E(E4[8]),
   .C0(C4[8]));
ac ac49(.A(A[8]),.B(B[6]),  .C(C[6]),  .X(X),  .F(FI[4]),
   .C1(X),    .S(S4[9]), .D(D4[9]), .E(E4[9]),
   .C0(C4[9]));

// arithmetic cells of row 5(A,B,C,X,F,C1/S,D,E,C0)
ac ac51(.A(S4[1]),  .B(1'b0),  .C(1'b0),  .X(X),
   .F(FI[5]), .C1(C5[2]), .S(S[1]),  .D(), .E(),
   .C0(C5[1]));
ac ac52(.A(S4[2]),  .B(D4[1]), .C(E4[1]), .X(X),
   .F(FI[5]), .C1(C5[3]), .S(S[2]),  .D(), .E(),
   .C0(C5[2]));
ac ac53(.A(S4[3]),  .B(D4[2]), .C(E4[2]), .X(X),
   .F(FI[5]), .C1(C5[4]), .S(S[3]),  .D(), .E(),
   .C0(C5[3]));
ac ac54(.A(S4[4]),  .B(D4[3]), .C(E4[3]), .X(X),
   .F(FI[5]), .C1(C5[5]), .S(S[4]),  .D(), .E(),
   .C0(C5[4]));
ac ac55(.A(S4[5]),  .B(D4[4]), .C(E4[4]), .X(X),
   .F(FI[5]), .C1(C5[6]), .S(S[5]),  .D(), .E(),
   .C0(C5[5]));
ac ac56(.A(S4[6]),  .B(D4[5]), .C(E4[5]), .X(X),
   .F(FI[5]), .C1(C5[7]), .S(S[6]),  .D(), .E(),
   .C0(C5[6]));
ac ac57(.A(S4[7]),  .B(D4[6]), .C(E4[6]), .X(X),
   .F(FI[5]), .C1(C5[8]), .S(S[7]),  .D(), .E(),
   .C0(C5[7]));
ac ac58(.A(S4[8]),   .B(D4[7]),  .C(E4[7]),  .X(X),
   .F(FI[5]),  .C1(C5[9]), .S(S[8]),   .D(), .E(),
   .C0(C5[8]));
ac ac59(.A(S4[9]),   .B(D4[8]),  .C(E4[8]),  .X(X),
   .F(FI[5]),  .C1(C5[10]),.S(S[9]),   .D(), .E(),
   .C0(C5[9]));
ac ac5a(.A(A[9]),.B(D4[9]),  .C(E4[9]),  .X(X),   .F(FI[5]),
   .C1(C5[11]),.S(S[10]),  .D(), .E(), .C0(C5[10]));
ac ac5b(.A(A[10]),   .B(B[7]),   .C(C[7]),   .X(X),
   .F(FI[5]),  .C1(X),     .S(S[11]),  .D(), .E(),
   .C0(C5[11]));

endmodule

module gpca40p
(
input wire [1:0] op,
input wire [9:0] din0,
input wire [6:0] din1,
output reg [10:0] dout
);
//       op[1:0]  din0[9:0]      din1[6:0]      dout[10:0]
(TBD)
//
// sq    00    P[1:5]         7'bx        S[1:11]
   (X=1'b0, B=7'b0011111, C=7'b0100000, P=din0[4:0], A=10'b0)
// sqr      01    A[1:10]        7'bx        {6'bx, F[1:5]}
   (X=1'b1, B=7'b0011111, C=7'b0100000, P=5'b0, A=din0[9:0])
// mult  10    B[1:7], C[1:7] {2'bx, P[1:5]} S[1:11]
   (X=1'b0, B=C=din0[6:0], P=din1[4:0], A=10'b0)
// div      11    A[10:1]        B[7:1], C[7:1] {7'bx, F[2:5]}
      (X=1'b1, B=C=din1[0:6], P=5'b0, A=din0[0:9])

/* input */
reg          X;
reg [1:5]    P;
reg [1:7]    B, C;
reg [1:10]   A;

/* output */
wire [1:5]   F;
wire [1:11] S;
```

```
/* original circuit */
gpca inst_gpca (X, P, B, C, A, F, S);

/* multiplexer */
always @* begin
/* default */
X     = 1'bx;
P[1:5]   = 5'bx;
B[1:7]   = 7'bx;
C[1:7]   = 7'bx;
A[1:10]  = 10'bx;

dout[10:0] = 11'bx;

case(op)
2'b00: begin
X     = 1'b0;
P[1:5]   = din0[4:0];
B[1:7]   = 7'b0011111;
C[1:7]   = 7'b0100000;
A[1:10]  = 10'b0;

dout[10:0] = S[1:11];
end
2'b01: begin
X     = 1'b1;
P[1:5]   = 5'b0;
B[1:7]   = 7'b0011111;
C[1:7]   = 7'b0100000;
A[1:10]  = din0[9:0];

dout[10:0] = {6'b0, F[1:5]};
end
2'b10: begin
X     = 1'b0;
P[1:5]   = din1[4:0];
B[1:7]   = din0[6:0];
C[1:7]   = din0[6:0];
A[1:10]  = 10'b0;

dout[10:0] = S[1:11];
end
2'b11: begin
X     = 1'b1;
```

```
P[1:5]   = 5'b0;
B[1:7]   =  {din1[0],  din1[1],  din1[2],  din1[3],  din1[4],
din1[5], din1[6]};
C[1:7]   =  {din1[0],  din1[1],  din1[2],  din1[3],  din1[4],
din1[5], din1[6]};
A[1:10]  =  {din0[0],  din0[1],  din0[2],  din0[3],  din0[4],
din0[5], din0[6], din0[7], din0[8], din0[9]};

dout[10:0] = {7'b0, F[5], F[4], F[3], F[2]};
end
default: begin
end
endcase
end
endmodule
```

APPENDIX C Verilog Code for the Extension of Pipelining

```
// Verilog implementation of "A GENERALIZED PIPELINE CELLULAR
ARRAY" by
// Harpreet Singh, Shashank Kamthan, Dharma Agarwal, Lubna
Alazzawi
// All codes here are imitating what's described in the papers
given by Dr. Singh, and it is claimed by Dr. Singh to be at least
useful at all
// top-level entity: "gpca40p"

// arithmetic cell
module ac
(
input wire  A, B, C, X, F, C1,
output wire S, D, E, C0
);

assign S = ((A ^ (B ^ X) ^ C1) & F) | (A & ~F);
assign C0 = ((B ^ X) & (A | C1)) | (A & C1);
assign D = C & (B | F);
assign E = B | (C & F);
endmodule

module cc
(
input wire  X, P, C0,
output wire F
);

assign F = (C0 & X) | (P & ~X);
endmodule

module gpca
(
input wire       clk, rst,
input wire       X,
input wire [1:5]  P,    // [1:nrow]
input wire [1:7]  C,    // [1:(nrow + 2)]
input wire [1:7]  B,    // [1:(nrow + 2)]
input wire [1:10] A,    // [1:(2*nrow)]
output reg [1:5]  F,    // [1:nrow]
output reg [1:11] S     // [1:(2*nrow + 1)]
);
```

```
/* stage 0 */
reg          s0_X_r;
reg [1:5]    s0_P_r;
reg [1:7]    s0_C_r;
reg [1:7]    s0_B_r;
reg [1:10]   s0_A_r;

wire      s0_F;
wire [1:3]   s0_C0;

/* stage 1 */
reg          s1_X_r, s1_X_n;
reg [2:5]    s1_P_r, s1_P_n;
reg [4:7]    s1_C_r, s1_C_n;
reg [4:7]    s1_B_r, s1_B_n;
reg [3:10]   s1_A_r, s1_A_n;

reg [1:3]    s1_S_r;
reg [1:3]    s1_D_r;
reg [1:3]    s1_E_r;
reg [1:1]    s1_F_r, s1_F_n;

wire [1:3]   s1_S_n;
wire [1:3]   s1_D_n;
wire [1:3]   s1_E_n;

wire      s1_F;
wire [1:5]   s1_C0;

/* stage 2 */
reg          s2_X_r, s2_X_n;
reg [3:5]    s2_P_r, s2_P_n;
reg [5:7]    s2_C_r, s2_C_n;
reg [5:7]    s2_B_r, s2_B_n;
reg [5:10]   s2_A_r, s2_A_n;

reg [1:5]    s2_S_r;
reg [1:5]    s2_D_r;
reg [1:5]    s2_E_r;
reg [1:2]    s2_F_r, s2_F_n;

wire [1:5]   s2_S_n;
wire [1:5]   s2_D_n;
wire [1:5]  s2_E_n;

wire     s2_F;
wire [1:7]  s2_C0;

/* stage 3 */
reg         s3_X_r, s3_X_n;
reg [4:5]   s3_P_r, s3_P_n;
reg [6:7]   s3_C_r, s3_C_n;
reg [6:7]   s3_B_r, s3_B_n;
reg [7:10]  s3_A_r, s3_A_n;

reg [1:7]   s3_S_r;
reg [1:7]   s3_D_r;
reg [1:7]   s3_E_r;
reg [1:3]   s3_F_r, s3_F_n;

wire [1:7]  s3_S_n;
wire [1:7]  s3_D_n;
wire [1:7]  s3_E_n;

wire     s3_F;
wire [1:9]  s3_C0;


/* stage 4 */
reg         s4_X_r, s4_X_n;
reg [5:5]   s4_P_r, s4_P_n;
reg [7:7]   s4_C_r, s4_C_n;
reg [7:7]   s4_B_r, s4_B_n;
reg [9:10]  s4_A_r, s4_A_n;

reg [1:9]   s4_S_r;
reg [1:9]   s4_D_r;
reg [1:9]   s4_E_r;
reg [1:4]   s4_F_r, s4_F_n;

wire [1:9]  s4_S_n;
wire [1:9]  s4_D_n;
wire [1:9]  s4_E_n;

wire     s4_F;
wire [1:11] s4_C0;
```

```
/* stage 5 */
reg [1:11]  s5_S_r;
reg [1:11]  s5_D_r;
reg [1:11]  s5_E_r;
reg [1:5]   s5_F_r, s5_F_n;

wire [1:11] s5_S_n;
wire [1:11] s5_D_n;
wire [1:11] s5_E_n;

/*** stage 0 ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s0_X_r   <= 1'bx;
s0_P_r   <= 5'bx;
s0_C_r   <= 7'bx;
s0_B_r   <= 7'bx;
s0_A_r   <= 10'bx;
end else begin
s0_X_r   <= X;
s0_P_r   <= P;
s0_C_r   <= C;
s0_B_r   <= B;
s0_A_r   <= A;
end
end

always @* begin
s1_X_n        = s0_X_r            ;
s1_P_n[2:5]   = s0_P_r[2:5]          ;
s1_C_n[4:7]   = s0_C_r[4:7]          ;
s1_B_n[4:7]   = s0_B_r[4:7]          ;
s1_A_n[3:10]  = s0_A_r[3:10]         ;
s1_F_n[1:1]   = {s0_C0[1]}           ;
end

/* control cell of row 1 (X,P,C0 / F) */
cc cc1(s0_X_r, s0_P_r[1],  s0_C0[1],   s0_F);

/* arithmetic cells of row 1 (A,B,C,X,F,C1/S,D,E,C0) */
ac ac11(.A(1'b0),    .B(s0_B_r[1]), .C(s0_C_r[1]), .X(s0_X_r),
   .F(s0_F),   .C1(s0_C0[2]), .S(s1_S_n[1]), .D(s1_D_n[1]),
   .E(s1_E_n[1]), .C0(s0_C0[1])  );
ac ac12(.A(s0_A_r[1]), .B(s0_B_r[2]), .C(s0_C_r[2]),
   .X(s0_X_r), .F(s0_F),   .C1(s0_C0[3]), .S(s1_S_n[2]),
   .D(s1_D_n[2]), .E(s1_E_n[2]), .C0(s0_C0[2]) );
ac ac13(.A(s0_A_r[2]), .B(s0_B_r[3]), .C(s0_C_r[3]),
   .X(s0_X_r), .F(s0_F),   .C1(s0_X_r),   .S(s1_S_n[3]),
   .D(s1_D_n[3]), .E(s1_E_n[3]), .C0(s0_C0[3]) );

/*** stage 1 ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s1_X_r   <= 1'bx;
s1_P_r   <= 4'bx;
s1_C_r   <= 4'bx;
s1_B_r   <= 4'bx;
s1_A_r   <= 8'bx;
s1_S_r   <= 3'bx;
s1_D_r   <= 3'bx;
s1_E_r   <= 3'bx;
s1_F_r   <= 1'bx;
end else begin
s1_X_r   <= s1_X_n;
s1_P_r   <= s1_P_n;
s1_C_r   <= s1_C_n;
s1_B_r   <= s1_B_n;
s1_A_r   <= s1_A_n;
s1_S_r   <= s1_S_n;
s1_D_r   <= s1_D_n;
s1_E_r   <= s1_E_n;
s1_F_r   <= s1_F_n;
end
end

always @* begin
s2_X_n         = s1_X_r            ;
s2_P_n[3:5]    = s1_P_r[3:5]          ;
s2_C_n[5:7]    = s1_C_r[5:7]          ;
s2_B_n[5:7]    = s1_B_r[5:7]          ;
s2_A_n[5:10]   = s1_A_r[5:10]         ;
s2_F_n[1:2]    = {s1_F_r[1:1], s1_C0[1]} ;
end

/* control cell of row 2 (X,P,C0 / F) */
cc cc2(s1_X_r, s1_P_r[2],  s1_C0[1],   s1_F);

/* arithmetic cells of row 2 (A,B,C,X,F,C1/S,D,E,C0) */
```

```
ac ac21(.A(s1_S_r[1]), .B(1'b0),      .C(1'b0),
   .X(s1_X_r), .F(s1_F),   .C1(s1_C0[2]), .S(s2_S_n[1]),
   .D(s2_D_n[1]), .E(s2_E_n[1]), .C0(s1_C0[1]) );
ac ac22(.A(s1_S_r[2]), .B(s1_D_r[1]), .C(s1_E_r[1]),
   .X(s1_X_r), .F(s1_F),   .C1(s1_C0[3]), .S(s2_S_n[2]),
   .D(s2_D_n[2]), .E(s2_E_n[2]), .C0(s1_C0[2]) );
ac ac23(.A(s1_S_r[3]), .B(s1_D_r[2]), .C(s1_E_r[2]),
   .X(s1_X_r), .F(s1_F),   .C1(s1_C0[4]), .S(s2_S_n[3]),
   .D(s2_D_n[3]), .E(s2_E_n[3]), .C0(s1_C0[3]) );
ac ac24(.A(s1_A_r[3]), .B(s1_D_r[3]), .C(s1_E_r[3]),
   .X(s1_X_r), .F(s1_F),   .C1(s1_C0[5]), .S(s2_S_n[4]),
   .D(s2_D_n[4]), .E(s2_E_n[4]), .C0(s1_C0[4]) );
ac ac25(.A(s1_A_r[4]), .B(s1_B_r[4]), .C(s1_C_r[4]),
   .X(s1_X_r), .F(s1_F),   .C1(s1_X_r),   .S(s2_S_n[5]),
   .D(s2_D_n[5]), .E(s2_E_n[5]), .C0(s1_C0[5]) );

/*** stage 2 ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s2_X_r   <= 1'bx;
s2_P_r   <= 3'bx;
s2_C_r   <= 3'bx;
s2_B_r   <= 3'bx;
s2_A_r   <= 6'bx;
s2_S_r   <= 5'bx;
s2_D_r   <= 5'bx;
s2_E_r   <= 5'bx;
s2_F_r   <= 2'bx;
end else begin
s2_X_r   <= s2_X_n;
s2_P_r   <= s2_P_n;
s2_C_r   <= s2_C_n;
s2_B_r   <= s2_B_n;
s2_A_r   <= s2_A_n;
s2_S_r   <= s2_S_n;
s2_D_r   <= s2_D_n;
s2_E_r   <= s2_E_n;
s2_F_r   <= s2_F_n;
end
end

always @* begin
s3_X_n        = s2_X_r            ;
s3_P_n[4:5]   = s2_P_r[4:5]          ;
```

```
s3_C_n[6:7]    = s2_C_r[6:7]          ;
s3_B_n[6:7]    = s2_B_r[6:7]          ;
s3_A_n[7:10]   = s2_A_r[7:10]         ;
s3_F_n[1:3]    = {s2_F_r[1:2], s2_C0[1]} ;
end

/* control cell of row 3 (X,P,C0 / F) */
cc cc3(s2_X_r, s2_P_r[3],  s2_C0[1],   s2_F);

/* arithmetic cells of row 3 (A,B,C,X,F,C1/S,D,E,C0) */
ac ac31(.A(s2_S_r[1]), .B(1'b0),      .C(1'b0),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[2]), .S(s3_S_n[1]),
   .D(s3_D_n[1]), .E(s3_E_n[1]), .C0(s2_C0[1]));
ac ac32(.A(s2_S_r[2]), .B(s2_D_r[1]), .C(s2_E_r[1]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[3]), .S(s3_S_n[2]),
   .D(s3_D_n[2]), .E(s3_E_n[2]), .C0(s2_C0[2]));
ac ac33(.A(s2_S_r[3]), .B(s2_D_r[2]), .C(s2_E_r[2]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[4]), .S(s3_S_n[3]),
   .D(s3_D_n[3]), .E(s3_E_n[3]), .C0(s2_C0[3]));
ac ac34(.A(s2_S_r[4]), .B(s2_D_r[3]), .C(s2_E_r[3]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[5]), .S(s3_S_n[4]),
   .D(s3_D_n[4]), .E(s3_E_n[4]), .C0(s2_C0[4]));
ac ac35(.A(s2_S_r[5]), .B(s2_D_r[4]), .C(s2_E_r[4]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[6]), .S(s3_S_n[5]),
   .D(s3_D_n[5]), .E(s3_E_n[5]), .C0(s2_C0[5]));
ac ac36(.A(s2_A_r[5]), .B(s2_D_r[5]), .C(s2_E_r[5]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_C0[7]), .S(s3_S_n[6]),
   .D(s3_D_n[6]), .E(s3_E_n[6]), .C0(s2_C0[6]));
ac ac37(.A(s2_A_r[6]), .B(s2_B_r[5]), .C(s2_C_r[5]),
   .X(s2_X_r), .F(s2_F),   .C1(s2_X_r),   .S(s3_S_n[7]),
   .D(s3_D_n[7]), .E(s3_E_n[7]), .C0(s2_C0[7]));

/*** stage 3 ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s3_X_r   <= 1'bx;
s3_P_r   <= 2'bx;
s3_C_r   <= 2'bx;
s3_B_r   <= 2'bx;
s3_A_r   <= 4'bx;
s3_S_r   <= 7'bx;
s3_D_r   <= 7'bx;
s3_E_r   <= 7'bx;
s3_F_r   <= 3'bx;
```

```
end else begin
s3_X_r   <= s3_X_n;
s3_P_r   <= s3_P_n;
s3_C_r   <= s3_C_n;
s3_B_r   <= s3_B_n;
s3_A_r   <= s3_A_n;
s3_S_r   <= s3_S_n;
s3_D_r   <= s3_D_n;
s3_E_r   <= s3_E_n;
s3_F_r   <= s3_F_n;
end
end

always @* begin
s4_X_n         = s3_X_r            ;
s4_P_n[5:5]    = s3_P_r[5:5]          ;
s4_C_n[7:7]    = s3_C_r[7:7]          ;
s4_B_n[7:7]    = s3_B_r[7:7]          ;
s4_A_n[9:10]   = s3_A_r[9:10]         ;
s4_F_n[1:4]    = {s3_F_r[1:3], s3_C0[1]} ;
end

/* control cell of row 4 (X,P,C0 / F) */
cc cc4(s3_X_r, s3_P_r[4],  s3_C0[1],   s3_F);

/* arithmetic cells of row 4 (A,B,C,X,F,C1/S,D,E,C0) */
ac ac41(.A(s3_S_r[1]), .B(1'b0),      .C(1'b0),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[2]), .S(s4_S_n[1]),
   .D(s4_D_n[1]), .E(s4_E_n[1]), .C0(s3_C0[1]));
ac ac42(.A(s3_S_r[2]), .B(s3_D_r[1]), .C(s3_E_r[1]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[3]), .S(s4_S_n[2]),
   .D(s4_D_n[2]), .E(s4_E_n[2]), .C0(s3_C0[2]));
ac ac43(.A(s3_S_r[3]), .B(s3_D_r[2]), .C(s3_E_r[2]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[4]), .S(s4_S_n[3]),
   .D(s4_D_n[3]), .E(s4_E_n[3]), .C0(s3_C0[3]));
ac ac44(.A(s3_S_r[4]), .B(s3_D_r[3]), .C(s3_E_r[3]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[5]), .S(s4_S_n[4]),
   .D(s4_D_n[4]), .E(s4_E_n[4]), .C0(s3_C0[4]));
ac ac45(.A(s3_S_r[5]), .B(s3_D_r[4]), .C(s3_E_r[4]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[6]), .S(s4_S_n[5]),
   .D(s4_D_n[5]), .E(s4_E_n[5]), .C0(s3_C0[5]));
ac ac46(.A(s3_S_r[6]), .B(s3_D_r[5]), .C(s3_E_r[5]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[7]), .S(s4_S_n[6]),
   .D(s4_D_n[6]), .E(s4_E_n[6]), .C0(s3_C0[6]));
```

```
ac ac47(.A(s3_S_r[7]), .B(s3_D_r[6]), .C(s3_E_r[6]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[8]), .S(s4_S_n[7]),
   .D(s4_D_n[7]), .E(s4_E_n[7]), .C0(s3_C0[7]));
ac ac48(.A(s3_A_r[7]), .B(s3_D_r[7]), .C(s3_E_r[7]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_C0[9]), .S(s4_S_n[8]),
   .D(s4_D_n[8]), .E(s4_E_n[8]), .C0(s3_C0[8]));
ac ac49(.A(s3_A_r[8]), .B(s3_B_r[6]), .C(s3_C_r[6]),
   .X(s3_X_r), .F(s3_F),   .C1(s3_X_r),   .S(s4_S_n[9]),
   .D(s4_D_n[9]), .E(s4_E_n[9]), .C0(s3_C0[9]));

/*** stage 4 ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s4_X_r   <= 1'bx;
s4_P_r   <= 1'bx;
s4_C_r   <= 1'bx;
s4_B_r   <= 1'bx;
s4_A_r   <= 2'bx;
s4_S_r   <= 9'bx;
s4_D_r   <= 9'bx;
s4_E_r   <= 9'bx;
s4_F_r   <= 4'bx;
end else begin
s4_X_r   <= s4_X_n;
s4_P_r   <= s4_P_n;
s4_C_r   <= s4_C_n;
s4_B_r   <= s4_B_n;
s4_A_r   <= s4_A_n;
s4_S_r   <= s4_S_n;
s4_D_r   <= s4_D_n;
s4_E_r   <= s4_E_n;
s4_F_r   <= s4_F_n;
end
end

always @* begin
s5_F_n[1:5]    = {s4_F_r[1:4], s4_C0[1]} ;
end

/* control cell of row 5 (X,P,C0 / F) */
cc cc5(s4_X_r, s4_P_r[5],  s4_C0[1],   s4_F);

/* arithmetic cells of row 5 (A,B,C,X,F,C1/S,D,E,C0) */
ac ac51(.A(s4_S_r[1]), .B(1'b0),      .C(1'b0),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[2]), .S(s5_S_n[1]),
   .D(s5_D_n[1]), .E(s5_E_n[1]), .C0(s4_C0[1]));
ac ac52(.A(s4_S_r[2]), .B(s4_D_r[1]), .C(s4_E_r[1]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[3]), .S(s5_S_n[2]),
   .D(s5_D_n[2]), .E(s5_E_n[2]), .C0(s4_C0[2]));
ac ac53(.A(s4_S_r[3]), .B(s4_D_r[2]), .C(s4_E_r[2]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[4]), .S(s5_S_n[3]),
   .D(s5_D_n[3]), .E(s5_E_n[3]), .C0(s4_C0[3]));
ac ac54(.A(s4_S_r[4]), .B(s4_D_r[3]), .C(s4_E_r[3]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[5]), .S(s5_S_n[4]),
   .D(s5_D_n[4]), .E(s5_E_n[4]), .C0(s4_C0[4]));
ac ac55(.A(s4_S_r[5]), .B(s4_D_r[4]), .C(s4_E_r[4]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[6]), .S(s5_S_n[5]),
   .D(s5_D_n[5]), .E(s5_E_n[5]), .C0(s4_C0[5]));
ac ac56(.A(s4_S_r[6]), .B(s4_D_r[5]), .C(s4_E_r[5]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[7]), .S(s5_S_n[6]),
   .D(s5_D_n[6]), .E(s5_E_n[6]), .C0(s4_C0[6]));
ac ac57(.A(s4_S_r[7]), .B(s4_D_r[6]), .C(s4_E_r[6]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[8]), .S(s5_S_n[7]),
   .D(s5_D_n[7]), .E(s5_E_n[7]), .C0(s4_C0[7]));
ac ac58(.A(s4_S_r[8]), .B(s4_D_r[7]), .C(s4_E_r[7]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[9]), .S(s5_S_n[8]),
   .D(s5_D_n[8]), .E(s5_E_n[8]), .C0(s4_C0[8]));
ac ac59(.A(s4_S_r[9]), .B(s4_D_r[8]), .C(s4_E_r[8]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[10]),   .S(s5_S_n[9]),
   .D(s5_D_n[9]), .E(s5_E_n[9]), .C0(s4_C0[9]));
ac ac5a(.A(s4_A_r[9]), .B(s4_D_r[9]), .C(s4_E_r[9]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_C0[11]),   .S(s5_S_n[10]),
   .D(s5_D_n[10]),   .E(s5_E_n[10]),   .C0(s4_C0[10]));
ac ac5b(.A(s4_A_r[10]),.B(s4_B_r[7]), .C(s4_C_r[7]),
   .X(s4_X_r), .F(s4_F),   .C1(s4_X_r),   .S(s5_S_n[11]),
   .D(s5_D_n[11]),   .E(s5_E_n[11]),   .C0(s4_C0[11]));

/*** stage 5 (output) ***/
always @(posedge clk, posedge rst) begin
if(rst) begin
s5_S_r   <= 11'bx;
s5_D_r   <= 11'bx;
s5_E_r   <= 11'bx;
s5_F_r   <= 5'bx;
end else begin
s5_S_r   <= s5_S_n;
s5_D_r   <= s5_D_n;
s5_E_r   <= s5_E_n;
```

```
s5_F_r   <= s5_F_n;
end
end

always @* begin
F = s5_F_r;
S = s5_S_r;
end
endmodule

module gpca40p
(
input wire        clk, rst,
input wire [1:0]  op,
input wire [9:0]  din0,
input wire [6:0]  din1,
output reg [10:0] dout
);
//       op[1:0]  din0[9:0]      din1[6:0]      dout[10:0]
(TBD)
//
// sq    00    P[1:5]         7'bx        S[1:11]
   (X=1'b0, B=7'b0011111, C=7'b0100000, P=din0[4:0], A=10'b0)
// sqr      01    A[1:10]        7'bx        {6'bx, F[1:5]}
   (X=1'b1, B=7'b0011111, C=7'b0100000, P=5'b0, A=din0[9:0])
// mult  10    B[1:7], C[1:7] {2'bx, P[1:5]} S[1:11]
   (X=1'b0, B=C=din0[6:0], P=din1[4:0], A=10'b0)
// div      11    A[10:1]        B[7:1], C[7:1] {7'bx, F[2:5]}
      (X=1'b1, B=C=din1[0:6], P=5'b0, A=din0[0:9])

/* input */
reg         X;
reg [1:5]   P;
reg [1:7]   B, C;
reg [1:10]  A;

/* output */
wire [1:5]  F;
wire [1:11] S;

/* operation */
reg [1:0]   op_r [0:5];

/* original circuit */
```

```
gpca inst_gpca (clk, rst,
X, P, C, B, A,
F, S);

/* operation delay */
always @(posedge clk, posedge rst) begin
if(rst) begin
op_r[0] <= 2'bx;
op_r[1] <= 2'bx;
op_r[2] <= 2'bx;
op_r[3] <= 2'bx;
op_r[4] <= 2'bx;
op_r[5] <= 2'bx;
end else begin
op_r[0] <= op;
op_r[1] <= op_r[0];
op_r[2] <= op_r[1];
op_r[3] <= op_r[2];
op_r[4] <= op_r[3];
op_r[5] <= op_r[4];
end
end

/* multiplexer (input)*/
always @* begin
case(op)
2'b00: begin
X     = 1'b0;
P[1:5]   = din0[4:0];
B[1:7]   = 7'b0011111;
C[1:7]   = 7'b0100000;
A[1:10]  = 10'b0;
end
2'b01: begin
X     = 1'b1;
P[1:5]   = 5'b0;
B[1:7]   = 7'b0011111;
C[1:7]   = 7'b0100000;
A[1:10]  = din0[9:0];
end
2'b10: begin
X     = 1'b0;
P[1:5]   = din1[4:0];
B[1:7]   = din0[6:0];
```

```
C[1:7]   = din0[6:0];
A[1:10]  = 10'b0;
end
2'b11: begin
X     = 1'b1;
P[1:5]   = 5'b0;
B[1:7]   =  {din1[0],  din1[1],  din1[2],  din1[3],  din1[4],
din1[5], din1[6]};
C[1:7]   =  {din1[0],  din1[1],  din1[2],  din1[3],  din1[4],
din1[5], din1[6]};
A[1:10]  =  {din0[0],  din0[1],  din0[2],  din0[3],  din0[4],
din0[5], din0[6], din0[7], din0[8], din0[9]};
end
default: begin
X     = 1'bx;
P[1:5]   = 5'bx;
B[1:7]   = 7'bx;
C[1:7]   = 7'bx;
A[1:10]  = 10'bx;
end
endcase
end

/* multiplexer (output)*/
always @* begin
case(op_r[5])
2'b00: begin
dout[10:0] = S[1:11];
end
2'b01: begin
dout[10:0] = {6'b0, F[1:5]};
end
2'b10: begin
dout[10:0] = S[1:11];
end
2'b11: begin
dout[10:0] = {7'b0, F[5], F[4], F[3], F[2]};
end
default: begin
dout[10:0] = 11'bx;
end
endcase
end
endmodule
```

APPENDIX D Script for Cadence Encounter RTL Compiler (rc.cmd)

```
set_attribute hdl_search_path {./}
set_attribute lib_search_path {./}
set_attribute library [list
/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_4.lib]
set_attribute information_level 6
set_attribute ungroup true
set_attribute write_vlog_unconnected_port_style none
read_hdl -v2001 simple.v
elaborate simple
synthesize -to_mapped
write_hdl -mapped > nl.v
q
```

APPENDIX E Script for Cadence Encounter (encounter.cmd)

```
set_global                      _enable_mmmc_by_default_flow
$CTE::mmmc_default
suppressMessage ENCEXT-2799
win
set conf_qxconf_file NULL
set conf_qxlib_file NULL
set defHierChar /
set init_gnd_net gnd!
set init_pwr_net vdd!
set init_mmmc_file mmmc.tcl
set                                             init_lef_file
/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_4.lef
set lsgOCPGainMult 1.000000
set init_verilog nl.v
set init_top_cell simple
create_rc_corner    -name typical \
-preRoute_res {1.0} \
-preRoute_cap {1.0} \
-preRoute_clkres {0.0} \
-preRoute_clkcap {0.0} \
-postRoute_res {1.0} \
-postRoute_cap {1.0} \
-postRoute_xcap {1.0} \
-postRoute_clkres {0.0} \
-postRoute_clkcap {0.0}
create_library_set       -name       typical       -timing
{/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_4.lib}
create_constraint_mode -name typical -sdc_files {typical.sdc}
create_delay_corner  -name  typical  -library_set  {typical}
-rc_corner {typical}
create_analysis_view -name typical -constraint_mode {typical}
-delay_corner {typical}
init_design


floorPlan -site core -r 1 0.4 30 30 30 30
saveDesign floorplan.enc

set sprCreateIeStripeNets {}
set sprCreateIeStripeLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeSpacing 4.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeNets {}
set sprCreateIeStripeLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeSpacing 4.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingNets {}
set sprCreateIeRingLayers {}
set sprCreateIeRingWidth 1.0
set sprCreateIeRingSpacing 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
addRing -center 1 -stacked_via_top_layer metal3 -around core
-jog_distance  1.5  -threshold  1.5  -nets  {gnd!  vdd!}
-stacked_via_bottom_layer metal1 -layer {bottom metal1 top
metal1 right metal2 left metal2} -width 9 -spacing 1.8 -offset
1.5
set sprCreateIeStripeNets {}
set sprCreateIeStripeLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeSpacing 4.0
set sprCreateIeStripeThreshold 1.0
addStripe         -block_ring_top_layer_limit         metal3
-max_same_layer_jog_length 5.0 -snap_wire_center_to_grid Grid
-padcore_ring_bottom_layer_limit metal1 -set_to_set_distance
100              -stacked_via_top_layer               metal3
-padcore_ring_top_layer_limit    metal3    -spacing    0.9
-merge_stripes_value        1.5        -layer        metal2
-block_ring_bottom_layer_limit metal1 -width 1.5 -nets {gnd!
vdd!} -stacked_via_bottom_layer metal1
saveDesign power.enc

globalNetConnect vdd! -type tiehi -module {}
globalNetConnect gnd! -type tielo -module {}

sroute  -connect  {  blockPin  padPin  padRing  corePin  }
-layerChangeRange   {   metal1   metal3   }   -blockPinTarget
{  nearestRingStripe  nearestTarget  }  -padPinPortConnect
{ allPort oneGeom } -checkAlignedSecondaryPin 1 -blockPin
useLef  -allowJogging  1  -crossoverViaBottomLayer  metal1
-allowLayerChange     1     -targetViaTopLayer     metal3
-crossoverViaTopLayer  metal3  -targetViaBottomLayer  metal1
-nets { gnd! vdd! }
```

```
setPlaceMode -fp false
placeDesign -prePlaceOpt

setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -preCTS
createClockTreeSpec -file Clock.ctstch
clockDesign  -specFile  Clock.ctstch  -outDir  clock_report
-fixedInstBeforeCTS
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS

setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithTimingDriven 1
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -drouteStartIteration default
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven true
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail

setDelayCalMode -engine aae -SIAware false
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute

getFillerMode -quiet
addFiller -cell FILL8 FILL4 FILL2 FILL -prefix FILLER

verifyConnectivity -type all -error 1000 -warning 50

setVerifyGeometryMode  -area  {  0  0  0  0  }  -minWidth  true
-minSpacing  true  -minArea  true  -sameNet  true  -short  true
-overlap false -offRGrid false -offMGrid true -mergedMGridCheck
true -minHole true -implantCheck true -minimumCut true -minStep
true -viaEnclosure true -antenna false -insuffMetalOverlap true
-pinInBlkg  false  -diffCellViol  true  -sameCellViol  false
-padFillerCellsOverlap   true   -routingBlkgPinOverlap   true
-routingCellBlkgOverlap    true    -regRoutingOnly    false
-stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
false -maxWidth true -maxNonPrefLength -1 -error 1000 -warning
50
verifyGeometry
```

```
saveDesign final.enc
defOut -floorplan -netlist -routing simple.def
saveNetlist -flat -replaceTieConnection nlopt.v
# then one may need to add 1'b0/0'b0 manually
```

APPENDIX F Multi-Mode Multi-Corner Script (mmmc.tcl)

```
create_rc_corner    -name typical \
-preRoute_res {1.0} \
-preRoute_cap {1.0} \
-preRoute_clkres {0.0} \
-preRoute_clkcap {0.0} \
-postRoute_res {1.0} \
-postRoute_cap {1.0} \
-postRoute_xcap {1.0} \
-postRoute_clkres {0.0} \
-postRoute_clkcap {0.0}
create_library_set       -name       typical       -timing
{/opt/cds/lib/UofU_Digital_v1_2/UofU_Digital_v1_4.lib}
create_constraint_mode -name typical -sdc_files {typical.sdc}
create_delay_corner  -name  typical  -library_set  {typical}
-rc_corner {typical}
create_analysis_view -name typical -constraint_mode {typical}
-delay_corner {typical}
set_analysis_view -setup {typical} -hold {typical}
```

APPENDIX G Synopsys Design Constraints (typical.sdc)

```
set sdc_version 1.6
create_clock [get_ports clk] -period 100 -waveform {0 50}
```

REFERENCES

[1] Kamal, A.K et al., "A generalized pipeline array," *IEEE Trans. Comput.*, vol. 23, pp. 533-536, May. 1974.

[2] J Bhaskar, *A Verilog HDL Primer*, 3rd ed. Allentown, PA: Star Galaxy Publishing, Jan. 2005.

[3] Kai Hwang, *Computer Arithmetic: Principles, Architecture and Design*. Hoboken, NJ: John Wiley & Sons, Apr. 1979.

[4] Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, 1st ed. New York City, NY: McGraw-Hill, Dec. 1992.

[5] Peter M. Kogge, *The Architecture of Pipelined Computers*. Boca Raton, FL: CRC Press, Jan. 1981.

[6] Agrawal, Dharma P., "High-speed arithmetic arrays," *IEEE Trans. Comput.*, vol. 28, pp. 215-224, Mar. 1979.

[7] Agrawal, Dharma P., "Optimum array-like structures for high-speed arithmetic," in *IEEE 3rd. Symp. on Comput. Arithmetic (ARITH)*, Dallas, TX, Nov. 1975, pp. 208-219.

[8] Partha P. Banerjee and Arif Ghafoor, “Design of a pipelined optical binary processor”, *Appl. Optics*, vol. 27, 1988, pp. 4766-4770.

[9] Harpreet Singh et al., "On simulation and design implementation of generalized pipeline cellular array ," in *Int. Conf. on Inform. Sci., Electron. and Elect. Eng. (ISEEE)*, Sapporo, Apr. 2014, pp. 1761-1765.

[10] Na Gong, "TM-RF: aging-aware power-efficient register file design for modern

microprocessors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, pp. 1196-1209, Jun. 2015.

[11] Hassan Rabah et al., "FPGA implementation of orthogonal matching pursuit for compressive sensing reconstruction," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, pp. 2209-2220, Sep. 2015.

[12] Goel, S., "Design of robust, energy-efficient full adders for deep-submicrometer design using hybrid-CMOS logic style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, pp. 1309-1321, Jan. 2007.

[13] Montoye, R.K., "An 18 ns 56-bit multiply-adder circuit," in *IEEE Int. Solid-State Circuits Conf (ISSCC)*, San Francisco, CA, Feb. 1990, pp. 46-47.

[14] *SN74ACT8847 64-Bit Floating Point Unit*, Texas Instruments, Dallas, TX.

[15] Stephen Brown, Zvonko Vranesic, *Fundamentals of Digital Logic with VHDL Design*, 3rd ed. New York: McGraw-Hill, Apr. 2008.

[16] Stephen Brown, Zvonko Vranesic, *Fundamentals of Digital Logic With Verilog Design*, 3rd ed. New York: Mcgraw-Hill, Feb. 2013.

[17] Neil Weste, David Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Boston: Addison-Wesley, 2011.

[18] Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic, *Digital Integrated Circuits*, 2nd ed. Upper Saddle River: Prentice Hall, Jan. 2003.

[19] *Cadence Help User Guide, Product Version 2.0,* Cadence Design Systems, Inc., San Jose, CA, 2012.

[20] *Cadence Application Infrastructure User Guide, Product Version 6.1.5*,

Cadence Design Systems, Inc., San Jose, CA, 2012.

[21] Paul Franzon, Rhett Davis, "NCSU electronic design automation (EDA) tutorial," Dept. Comput. Elect. Eng., North Carolina State Univ., Raleigh, NC, 2012.

[22] Nabil Abu-Khade, "From Verilog to MOSIS," Dept. Comput. Elect. Eng., Wayne State Univ., Detroit, MI, Nov. 2004.

[23] William Gibb, Bowei Zhang, "Cadence tutorial," Dept. Comput. Elect. Eng., Washington, DC, Spring 2011.

[24] Erik Brunvand, *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*, 1st ed. Boston, MA: Addison-Wesley, Feb. 2009.

[25] Ronald W. Williams, "An undergraduate VLSI CMOS circuit design laboratory," *IEEE Trans. Edu.*, *IEEE Trans. Edu.*, vol. 34, no 1, Feb. 1991.

[26] Mark S. Nixon, "On a programmable approach to introducing digital design," *IEEE Trans. Edu.*, vol. 40, no. 3, Aug. 1997.

[27] Etienne Sicard, "A VLSI design system for teaching introduction to microelectronics," *IEEE Trans. Edu.*, vol. 35, no. 4, Nov. 1992.

[28] Chyi-Shyong Lee et al., "A project-based laboratory for learning embedded system design with industry support," *IEEE Trans. Edu.*, vol. 53, no. 2, May 2010.

[29] Jesús Manuel Gómez-de-Gabriel et al., "Mobile robot lab project to introduce engineering students to fault diagnosis in mechatronic systems," *IEEE Trans. Edu.*, vol. 58, no. 3, Aug. 2015..

[30] *MOSIS FAQs: MOSIS Educational Program (MEP)*, the MOSIS Service, Marina del Rey, CA, Nov. 2015.

[31] *Command Reference for Encounter RTL Compiler, Product Version 12.1*, Cadence Design Systems, Inc., San Jose, CA, Nov. 2012.

[32] Yudi Xie, Harpreet Singh, "The VLSI implementation of digital design projects," Dept. Comput. Elect. Eng., Wayne State Univ., Detroit, MI, Nov. 2015 [Online]. Available: http://ece.eng.wayne.edu/~singhweb/

[33] *EDI System Text Command Reference, Product Version 12.0*, Cadence Design Systems, Inc., San Jose, CA, 2012.

[34] *EDI System Menu Reference, Product Version 12.0*, Cadence Design Systems, Inc., San Jose, CA, 2012.

[35] *MOSIS Scalable CMOS (SCMOS) Design Rules*, *Revision 8.00*, The MOSIS Service, Marina del Rey, CA, May. 2009.

[36] *Virtuoso Layout Suite XL User Guide*, *Product Version 6.1.5*, Cadence Design Systems, Inc., San Jose, CA, May. 2012.

[37] *How To Submit A Design To MOSIS*, The MOSIS Service, Marina del Rey, CA, Nov. 2015.

[38] Yier Jin, "Taping out your chip," Dept. Comput. Elect. Eng., Univ. of Central Florida, Orlando, FL, May. 2014.

[39] Kenney, J. F. and Keeping, E. S. (1962) "Linear Regression and Correlation." Ch. 15 in *Mathematics of Statistics*, Pt. 1, 3rd ed. Princeton, NJ: Van Nostrand, pp. 252-285.

[40] *ModelSim User's Manual, Software Version 10.1c*, Mentor Graphics Corporation, Wilsonville, OR, 2012.

[41] S. Brown, “FPGA architectural research: A survey,” *IEEE Des. Test. Comput.*, vol. 13, no. 4, pp. 9–15, Winter 1996.

[42] *Quartus II Handbook Version 13.1*, Altera Corporation, San Jose, CA, Nov. 2013.

[43] Mohamed Ibrahim, "How pipelining improves cpu performance," Digital Internals, Feb. 2009.

[44] Hennessy, Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Philadelphia, PA: Elsevier, Sep. 2011.

[45] *3164/3364 64-Bit Floating-Point Data Path Unit*, Weitek Corporation, Berlin, Germany.

ABSTRACT

THE DESIGN AND VLSI IMPLEMENTATION OF DIGITAL ARITHMATIC PROCESSORS -
A CASE STUDY OF A GENERALIZED PIPELINE CELLULAR ARRAY

by

YUDI XIE

December 2015

Advisor: Dr. Harpreet Singh

Major: Computer Engineering

Degree: Master of Science

A generalized pipeline array appeared in IEEE transaction in 1974. The array appeared in a few textbooks on computer arithmetic. From time to time, a number of papers appeared which reflected the modifications of this array. The objective of this thesis is to present the design and VLSI implementation of this array, which can add, subtract, multiply, divide, square and square root of binary numbers. In this thesis, we suggest a step-by-step procedure by which the design can be sent to MOSIS and to get the fabricated chip back. The array has been extended from 5 rows to 7 rows so that the extended operations can be performed. In particular, a procedure is developed by which the design and the implementation methodologies are suitable for 40 pin and 500 nm technologies. An algorithm has been developed by which one can predict and advance the maximum size and performance of the array. In addition, to increase data processing throughput, the extension of pipelining is conducted based on the original design. It is hoped that the design and implementation done here will go a long way in the development of advanced processors.

AUTOBIOGRAPHICAL STATEMENT

Yudi Xie is pursuing M.Sc of Computer Engineering at Wayne State University, Detroit, MI.

He worked in the university labs for various projects, including embedded system design, FPGA and VLSI. He also worked as the Graduate Teaching Assistant for Digital Logic course at Wayne State University. His current field of interest includes computer microarchitecture, FPGA and VLSI and system programming.