Final Project Report:

Elevator Control Unit

Brandon Bernier & Christopher Graham

ECE 4140: VLSI Design and Simulation

Professor Proie

December 3, 2013

1. Introduction

The proposed project is an elevator control unit. The system will control three elevators that have the ability to travel up and down to a maximum of seven floors. The theory of operation is very similar to how real elevators work. There will be both call buttons at each floor as well as buttons inside the elevator. When a floor has been selected, an elevator will go up or down to that floor. The elevator that is closest to the floor call that has been selected will be the one that responds. The elevator system makes use of both combinational and sequential logic to make decisions and operate as desired. This project is a great example of a finite state machine that must make decisions based on the current state or position of all three elevators.

2. Overall System Design

The system will specifically allow three elevators to operate independently across seven floors in order to serve requests as efficiently as possible. Initially it was intended to have the system serve eight floors, however, 000 was not a viable input because there was no way to distinguish it from no input. Therefore, the design had to be limited to seven floors of operation. Originally, it was also desired to have the control unit keep track of multiple button presses, and sequentially progress through them. This functionality was found to require a lot of extra logic that would have ballooned the overall transistor count to a range that was felt could not fit in the given bounds of the chip. This was simplified, and the system can currently only handle a single request at a time for the buttons inside each elevator. The inputs are received by the control unit, stored, and then used to compute the best path of each of the elevators. Logic within the system will assist with determining the most efficient path for all three elevators. The system then communicates with each elevator car by outputting a 3-bit number representing the floor that the elevator must progress to.

The system is initialized with all three elevators in a 0 state, meaning the register holding its current location starts at 0. Each elevator must be called at least once to leave this output state and give an actual floor for where the elevator should be located. Technically, any elevator in the 0 state would be sitting at the 1st floor, but it will only output that once it is called to move somewhere. After this, it never again returns to the 0 state.

Another important design decision was the decision to copy and paste layouts into bigger modules instead of simply instancing them. This allowed for a lot of flexibility when hand routing because the lower-level gates were not always designed perfectly for inclusion in the bigger project. This also allowed us to change the size of the V_{DD} and GND rails to 3 contacts wide and expand all the way down a row of components. A thick V_{DD} and GNC rail ensures that they can source enough current to all of the components connected to them without any trouble. This decision made it very easy to connect multiple sub-blocks into bigger modules as well as the final overall layout. It was very simply to connect the rails with large strips of metal1 out to the pads.

Inputs and Outputs

There are 25 input and outputs for this design, excluding power, ground, and the clock. This is further broken down to 16 inputs and 9 outputs. The inputs will consist of the buttons at each floor to call the elevator and the individual floor buttons inside the elevator. The decision to design this system for 3 elevators in a building of up to 7 floors was directly related to the number of I/O pins that would be available. Each floor has a single button with which an elevator can be called. Within each elevator, there will be 7 buttons to get to each of the seven floors. This totals 21 buttons across all three elevators. The system will assume that these buttons are externally encoded to a 3-bit number. That allows us to reduce the necessary inputs from these buttons from 21 to only 9. The control unit will then have 9 output pins, 3 going to each elevator. These outputs will represent the floor that the elevator must travel to. The 3-bit number will be decoded externally to signify floors 1 through 8. This output would be fed to another system powering the actual elevator motors and is simply telling that separate module where to move each elevator.

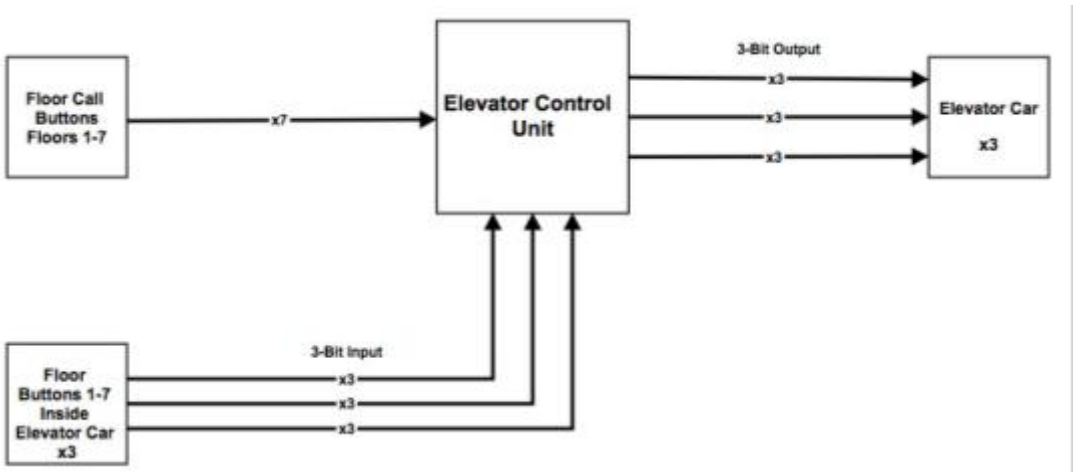


Figure 1: Context Diagram of the Elevator Control Unit Showing Inputs & Outputs

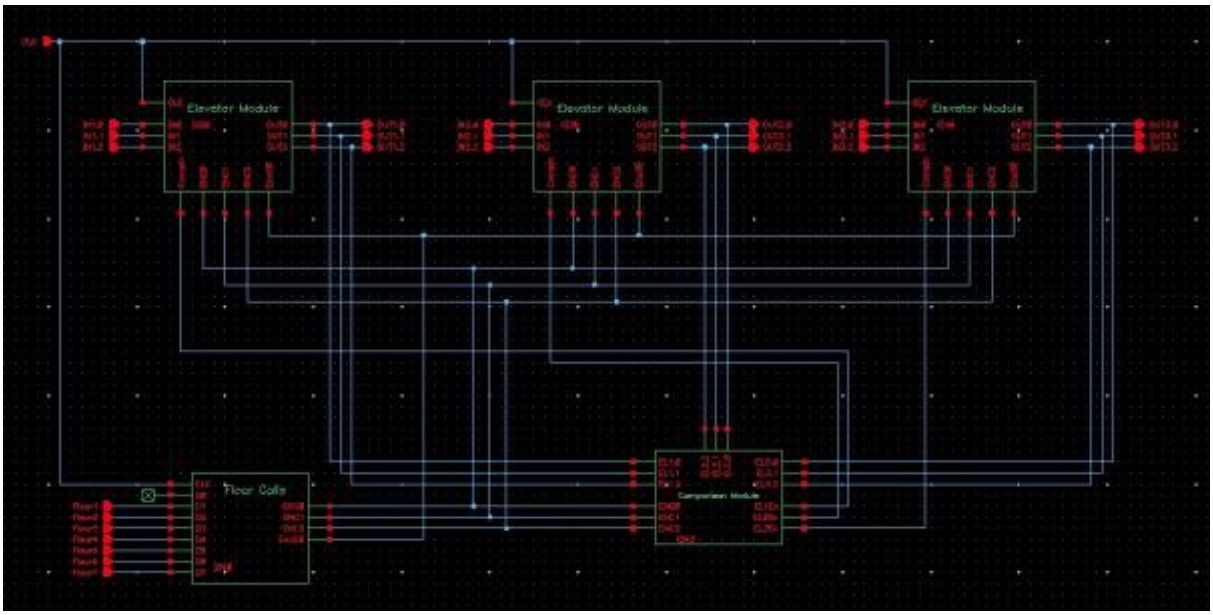


Figure 2: Top-Level Schematic of the Entire System

Area/Floor Plan

An original estimation for the total number of transistors necessary for the design brought us to between 3,000 and 3,500 transistors total. The actual final transistor count was 3764. From this rough estimation, we predicted that the design would require anywhere between 0.75mm^2 and the full 0.9mm^2 of usable area. The final dimensions of the system layout are roughly 780x760µm, for a total area of only 0.593mm^2. The final area came in much smaller than expected for this number of transistors due to a lot of work on the layouts ensuring they were as compact and efficient as possible. Even in the final design, there is a noticeable amount of blank space between many of the modules, which could have easily been avoided to condense the design down even further. The exact floor plan of the final modules within the PAD frame follows below.

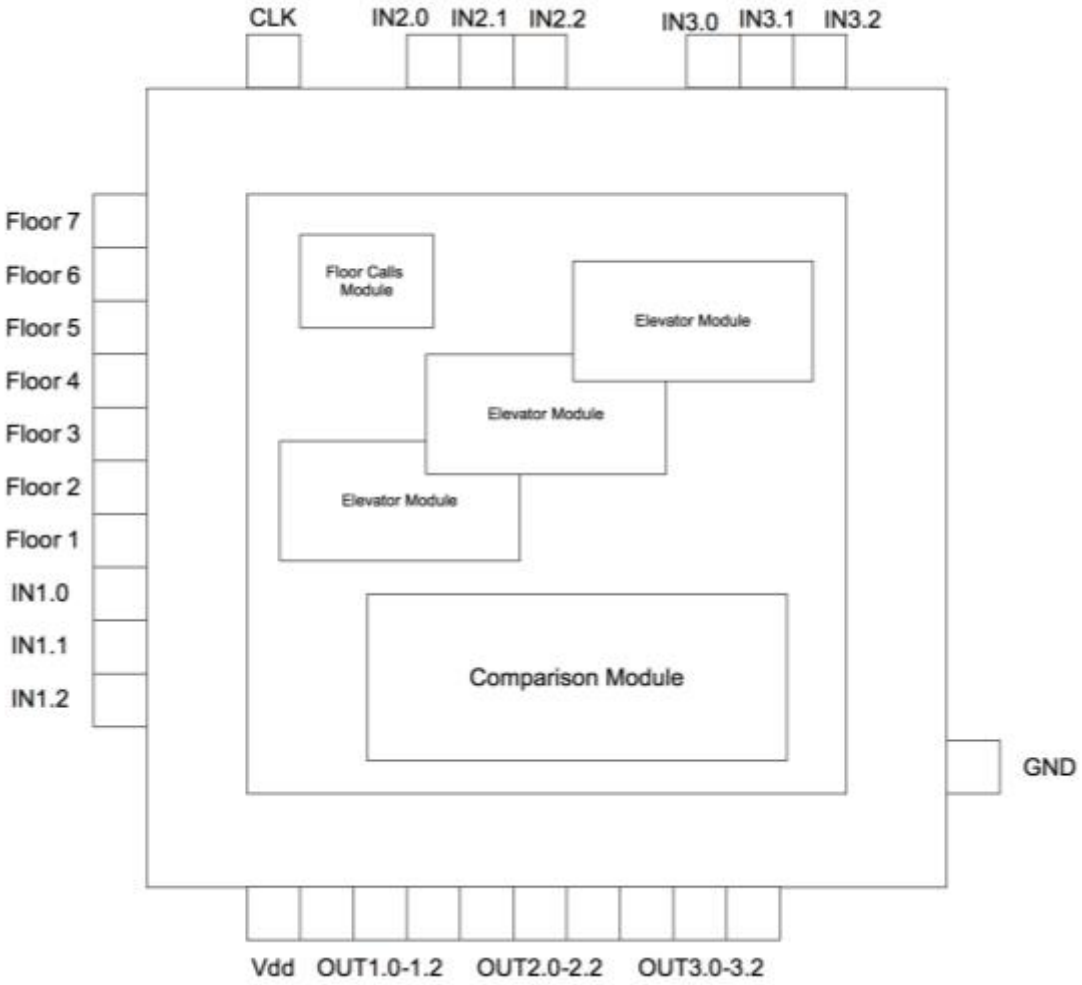


Figure 3: Accurate Layout Floor Plan Showing Major Modules

Group Breakdown

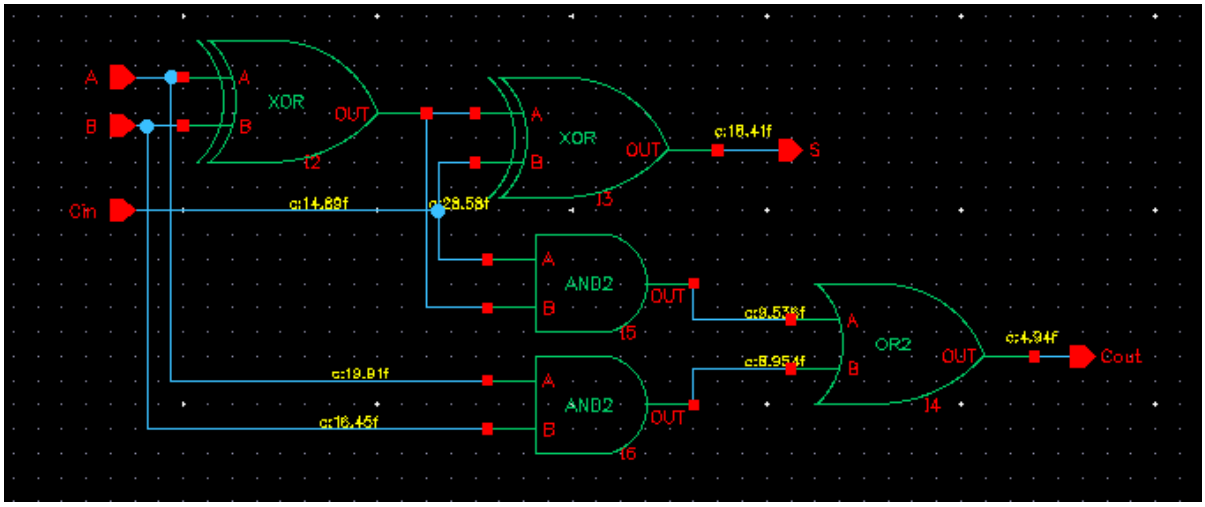
The breakdown of work for this project is as follows. Brandon focused on the necessary memory elements in the design including all of the data registers. He also be verified the layout and operation of all of the simple gates and blocks necessary for the design. Brandon was also be tasked with testing the system. Together we designed the necessary components for the overall schematic of the system. Brandon debugged the schematic to the point where we were able to have all three elevators functioning as expected. Brandon also handled the majority of the layout for the system including the major modules as well as the final overall layout, PAD frame placement, and GDS tape out. Chris focused on completion of the schematic and verification of each sub-block. He was responsible for compiling the majority of the figures and verification results of the individual modules for the final report.

3. Lower-Level Design Implementation

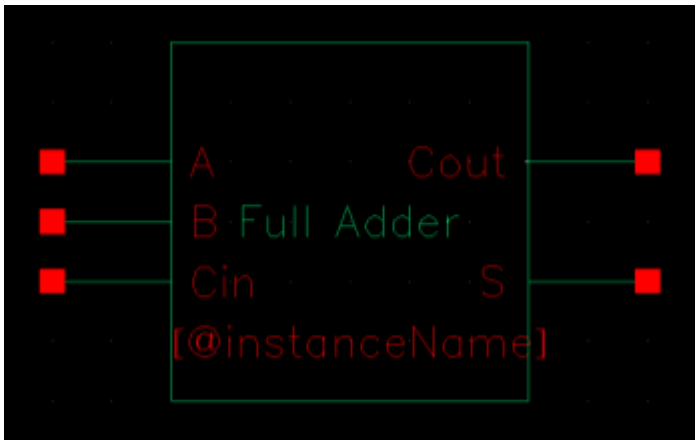
Full Adder

A one-bit full adder adds two one-bit numbers A and B. The full adder is used to create a 3-Bit full adder.

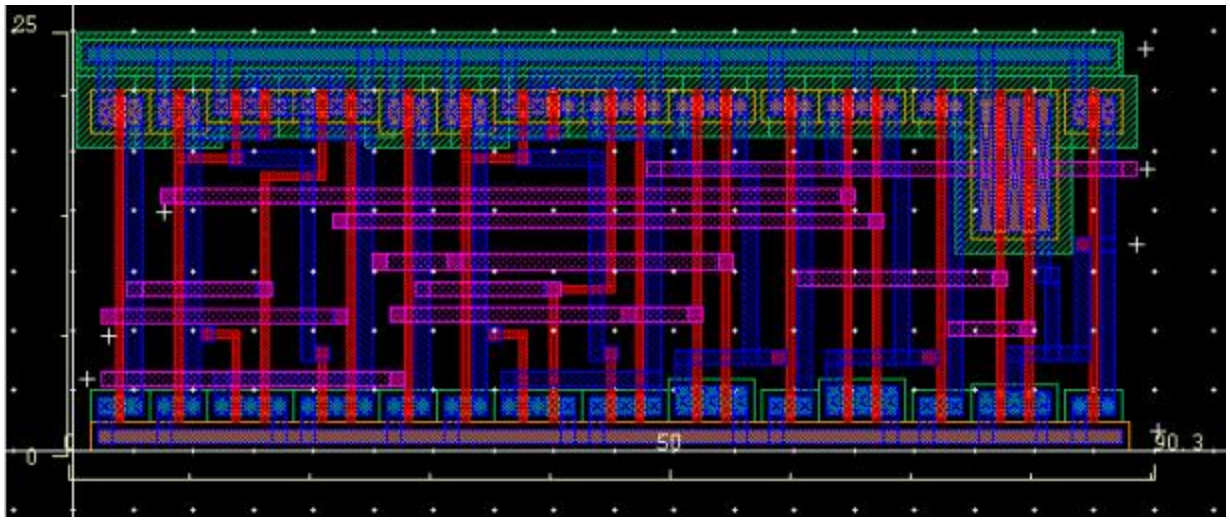
Inputs	Outputs
A	S
B	Cout
Cin	



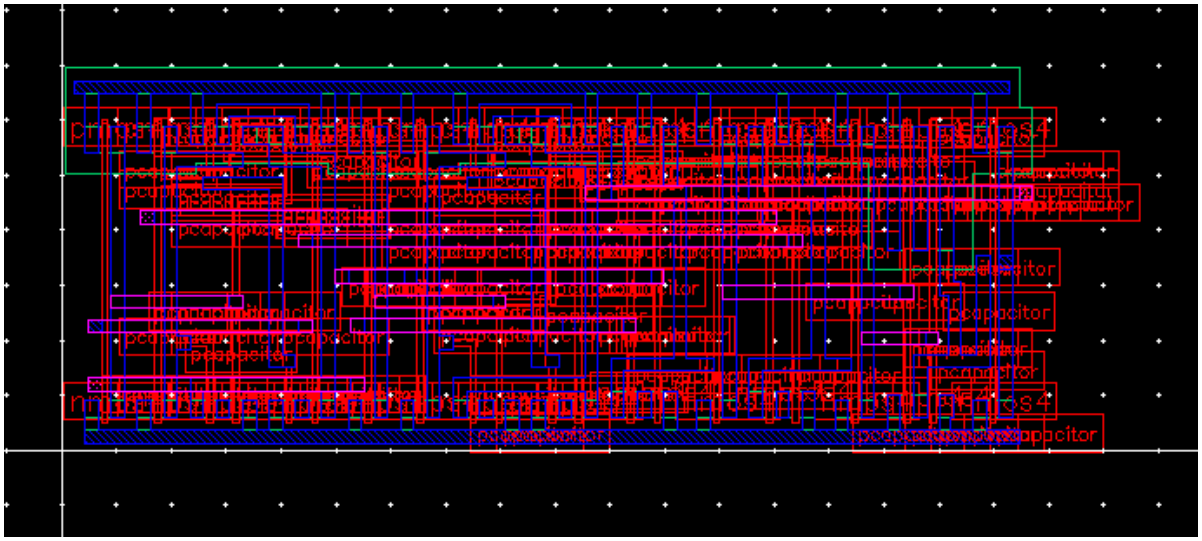
Full Adder Schematic



Full Adder Symbol

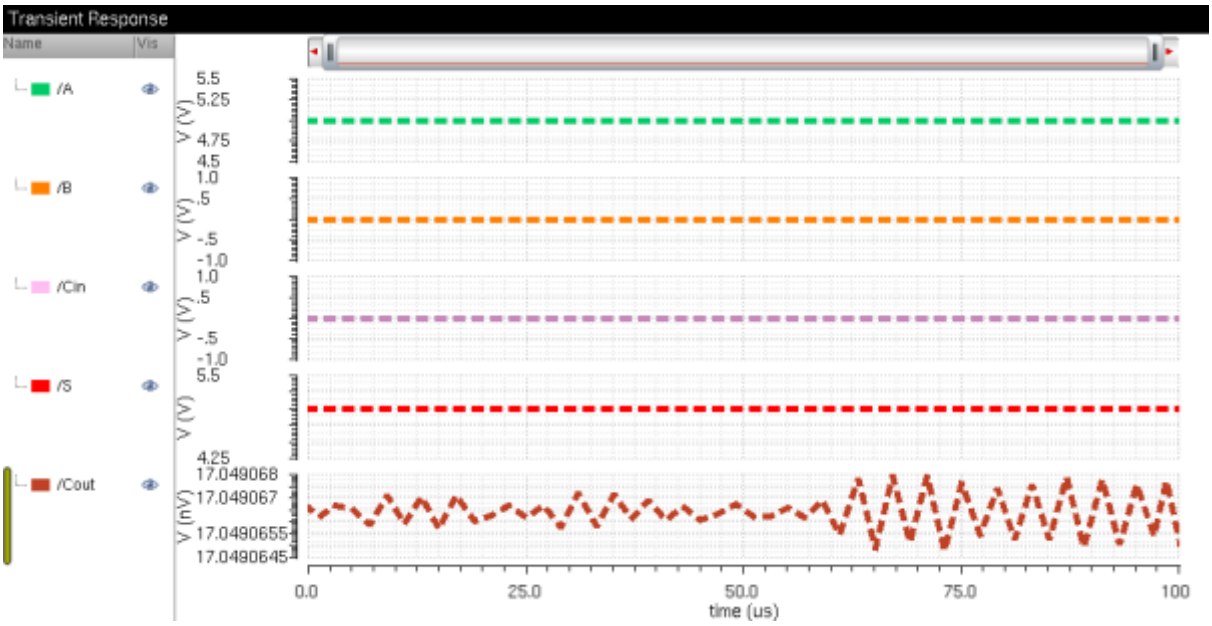


Full Adder Layout (Area ≈ 25x90µm)


Full Adder Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/seas/c
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
          26              nets
          7               terminals
          21              pmos
          21              nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
          26              nets
          7               terminals
          21              pmos
          21              nmos


    Terminal correspondence points
    N22         N6          A
    N21         N7          B
    N24         N9          Cin
    N25         N5          Cout
    N20         N2          S
    N19         N0          gnd!
    N23         N1          vdd!

Devices in the netlist but not in the rules:
          pcapacitor
Devices in the rules but not in the netlist:
          cap nfet pfet rmos4 pmos4

The net-lists match.
```

Full Adder LVS Report

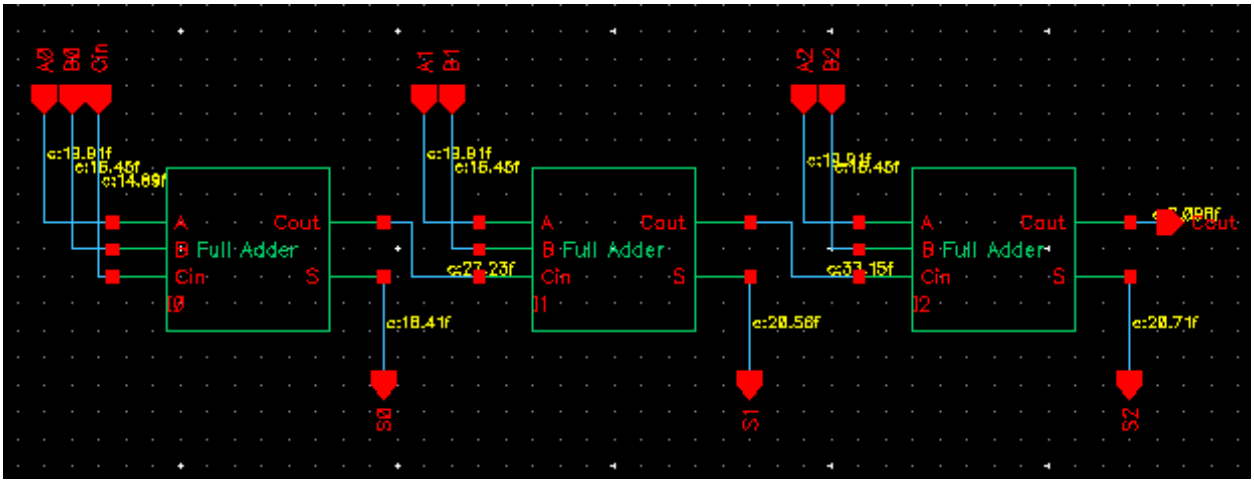


Full Adder Simulation

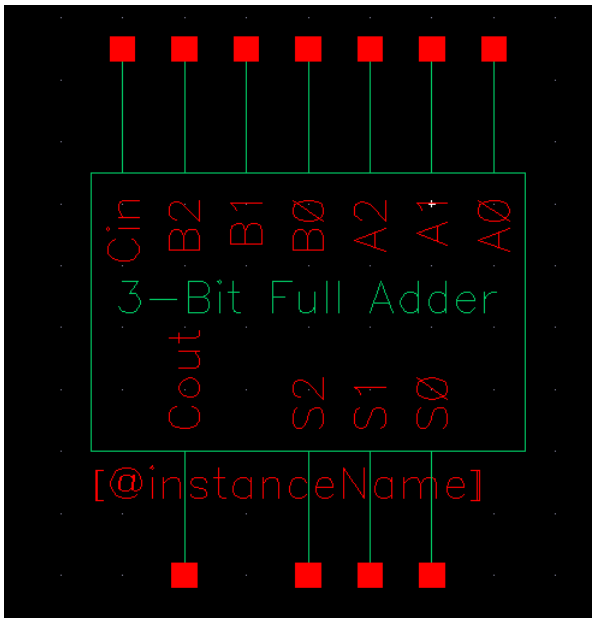
3-Bit Full Adder

The three bit full adder adds two three bit binary numbers. The three bit full adder is made from three full adders. The three bit full adder is used in the elevator control unit to do the addition in the adder subtractor module.

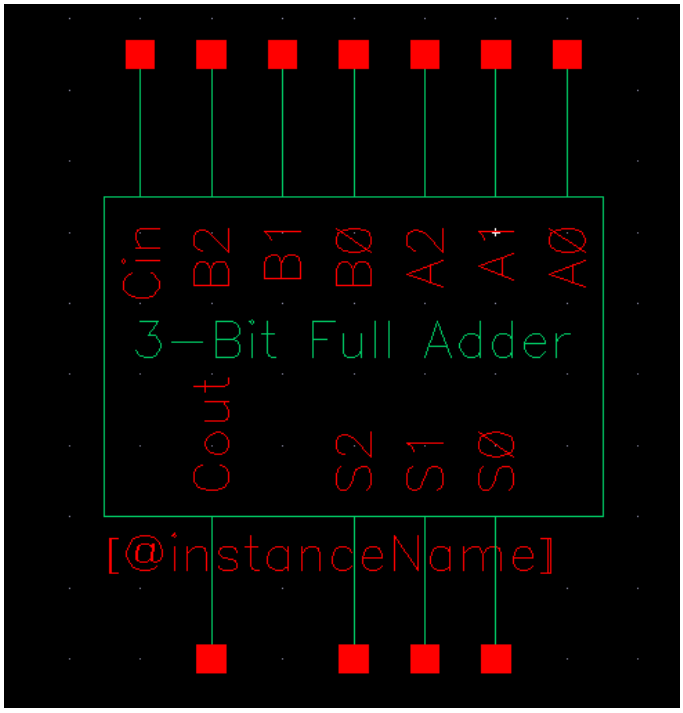
Inputs	Outputs
A0	S0
B0	S1
A1	S2
B1	Cout
A2	
B2	
Cin	



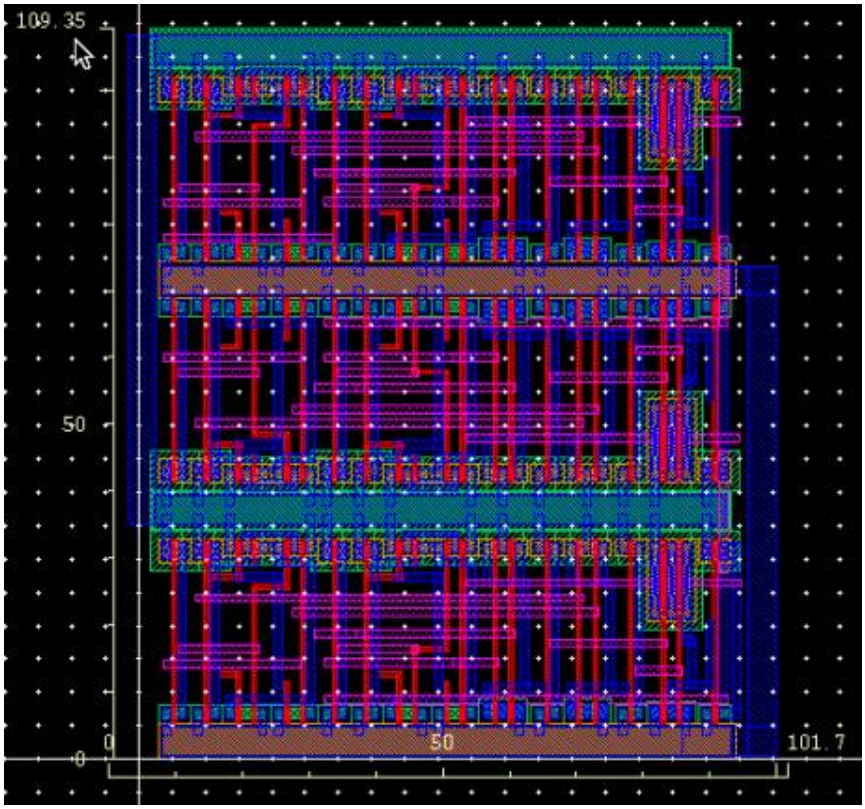
3-Bit Full Adder Schematic



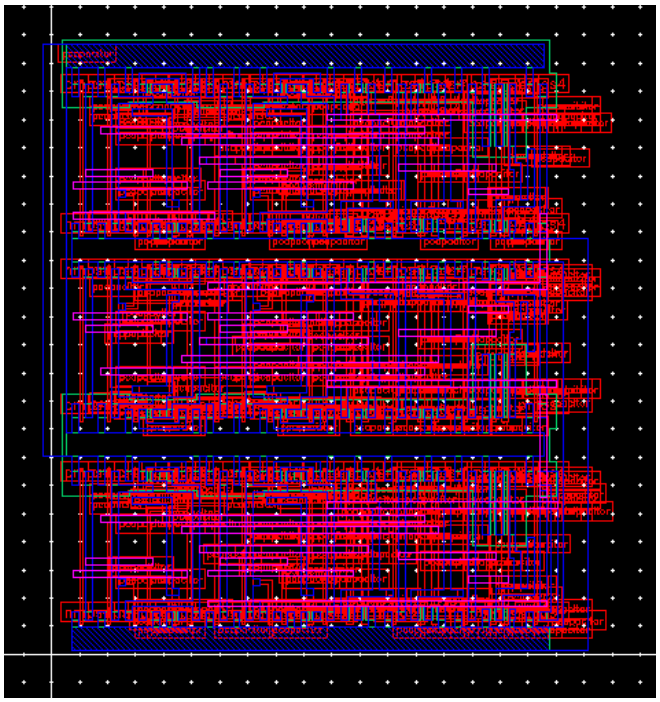
3-Bit Full Adder Symbol



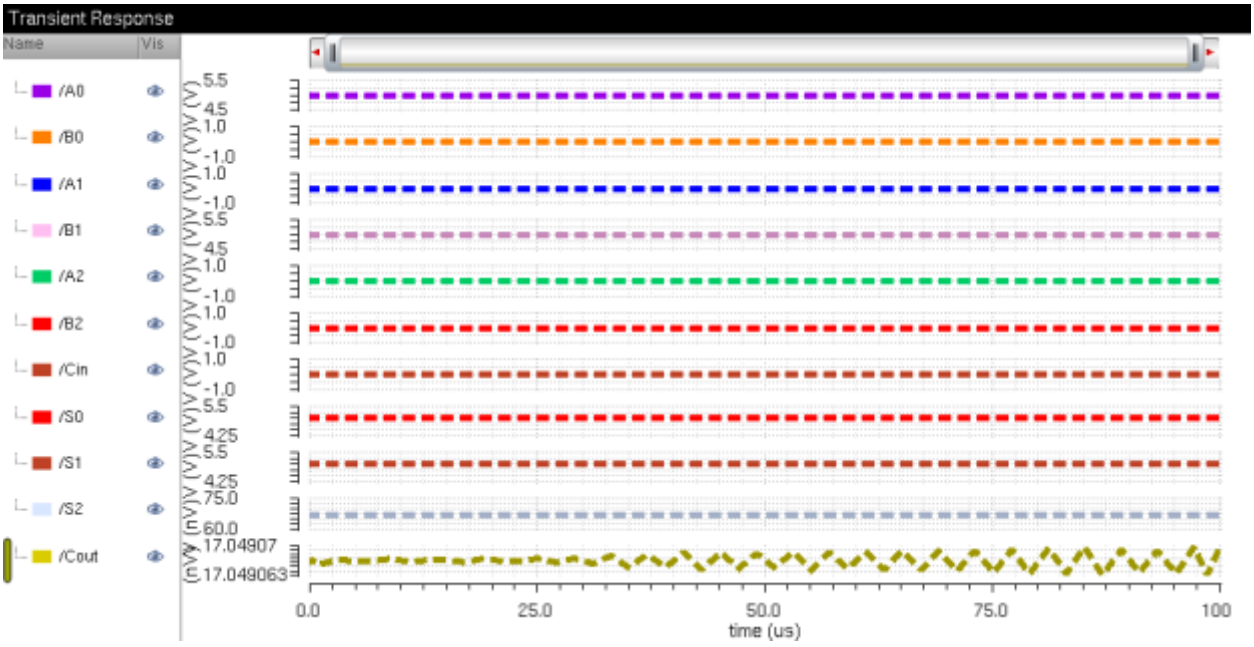
3-Bit Full Adder Layout (Area ≈ 109x102µm)



3-Bit Full Adder Extracted View



3-Bit Full Adder LVS Report

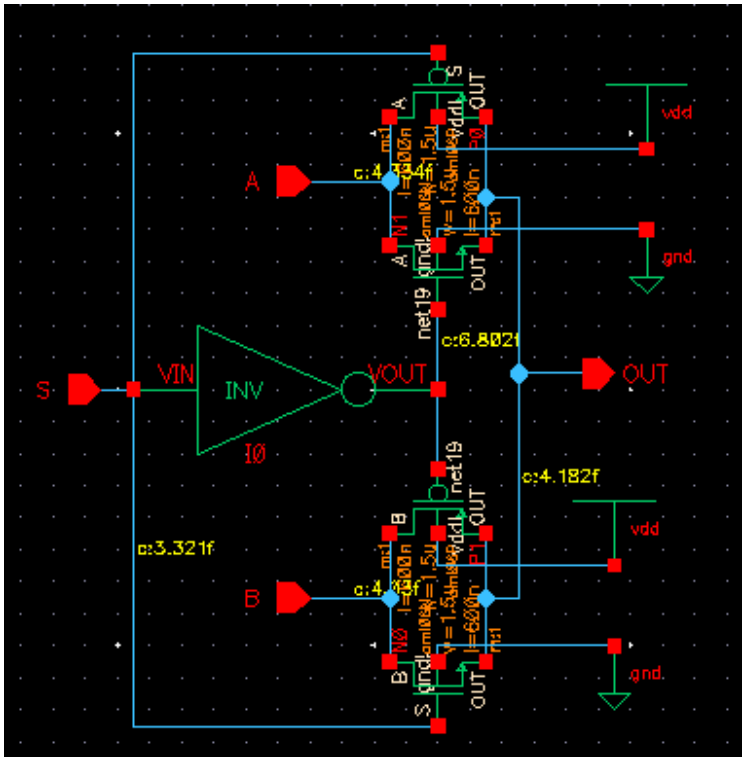


3-Bit Full Adder Simulation

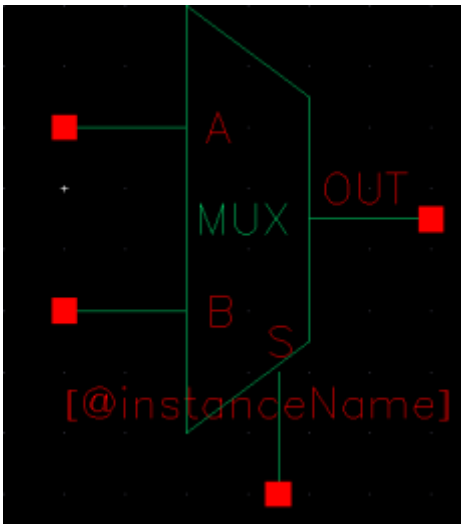
2-to-1 MUX

A multiplexer selects either input A or B to be outputted. The multiplexer is used to create a 4 to 1 multiplexer.

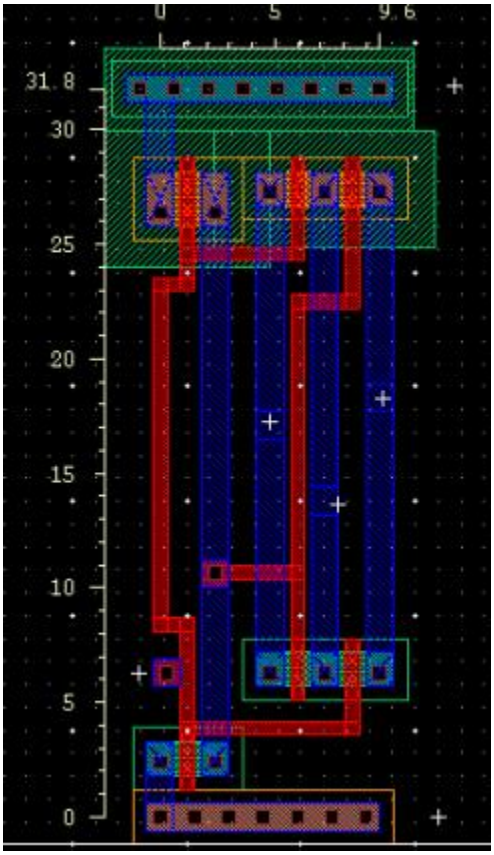
Inputs	Outputs
A	OUT
B	
S	



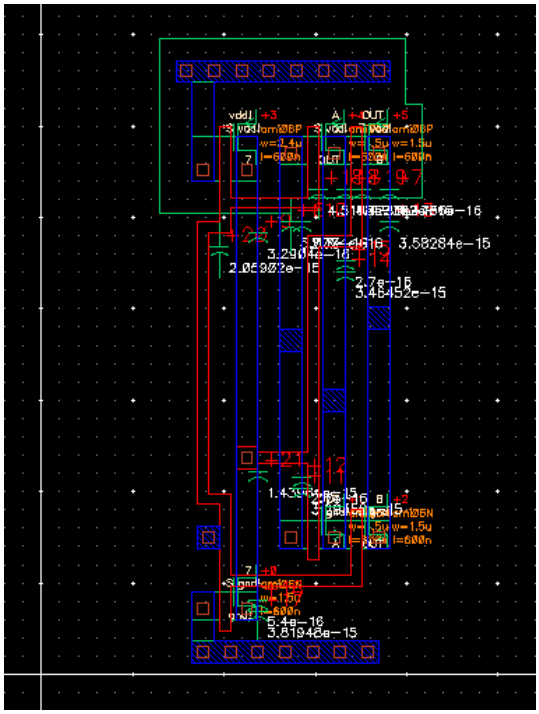
2-to-1 MUX Schematic



2-to-1 MUX Symbol

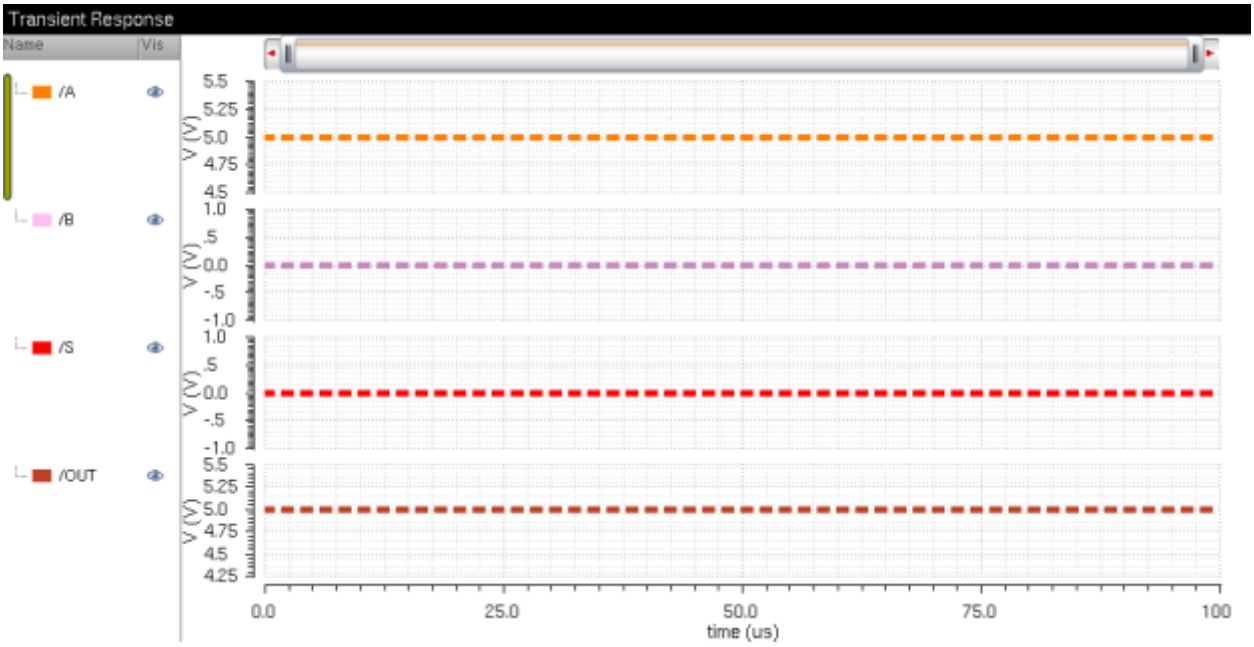


2-to-1 MUX Layout (Area ≈ 31.8x9.6µm)


2-to-1 MUX Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfd1224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/seas/
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
         7              nets
         6              terminals
         3              pmos
         3              nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
         7              nets
         6              terminals
         3              pmos
         3              nmos


    Terminal correspondence points
    N5          N2          A
    N4          N8          B
    N3          N5          OUT
    N2          N4          S
    N1          N0          gnd!
    N6          N1          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.

                                layout  schematic
                                  instances
        un-matched                 0       0
        rewired                    0       0
        size errors                0       0
        pruned                     0       0
        active                     6       6
        total                      6       6
```

2-to-1 MUX LVS Report

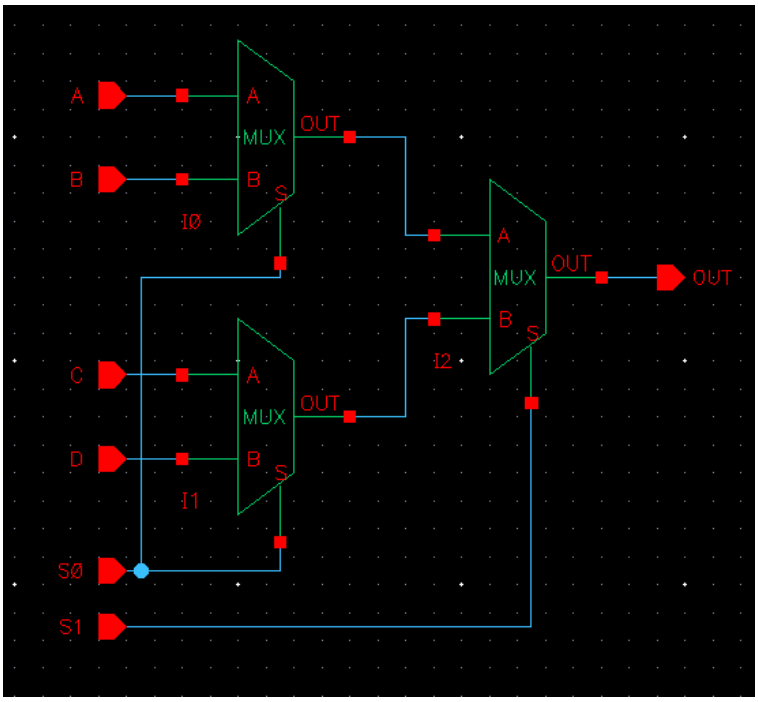


2-to-1 MUX Simulation

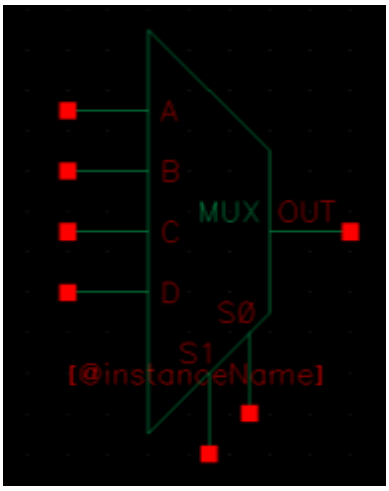
4-to-1 MUX

The 4 to 1 multiplexer consists of three 2 to 1 multiplexers. The 4 to 1 multiplexer operates the same way as a 2 to 1 multiplexer but has more inputs. The 4 to 1 multiplexer is used in the elevator control unit to create the 3 bit adder subtractor module.

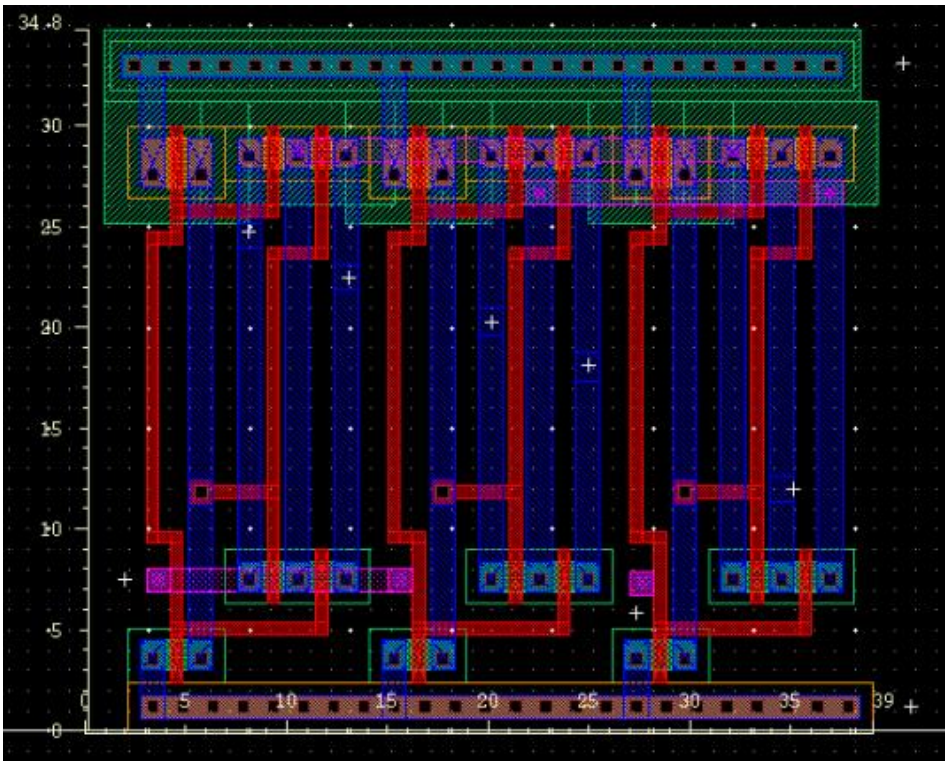
Inputs	Outputs
A	OUT
B	
C	
D	
S0	
S1	



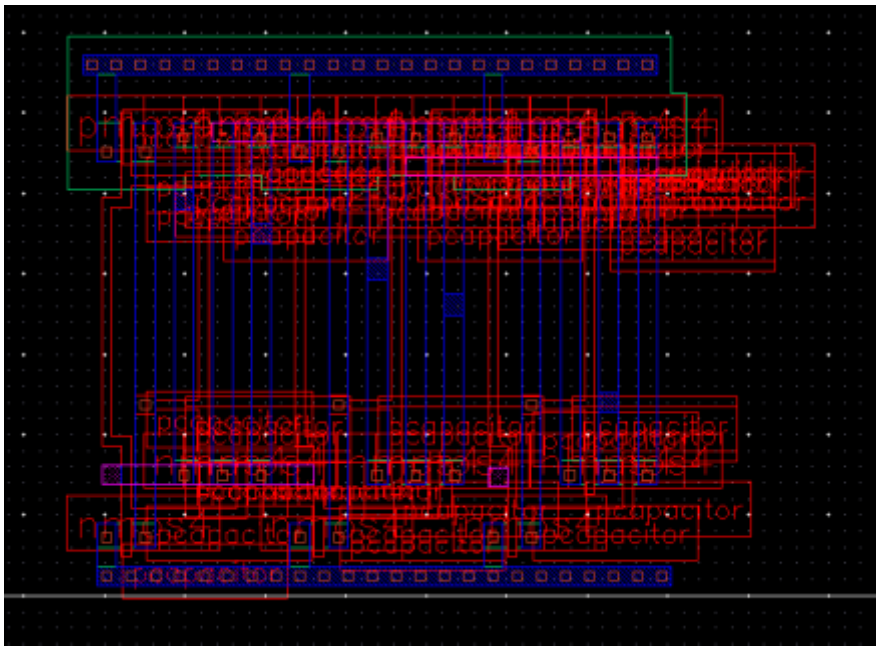
4-to-1 MUX Schematic



4-to-1 MUX Symbol

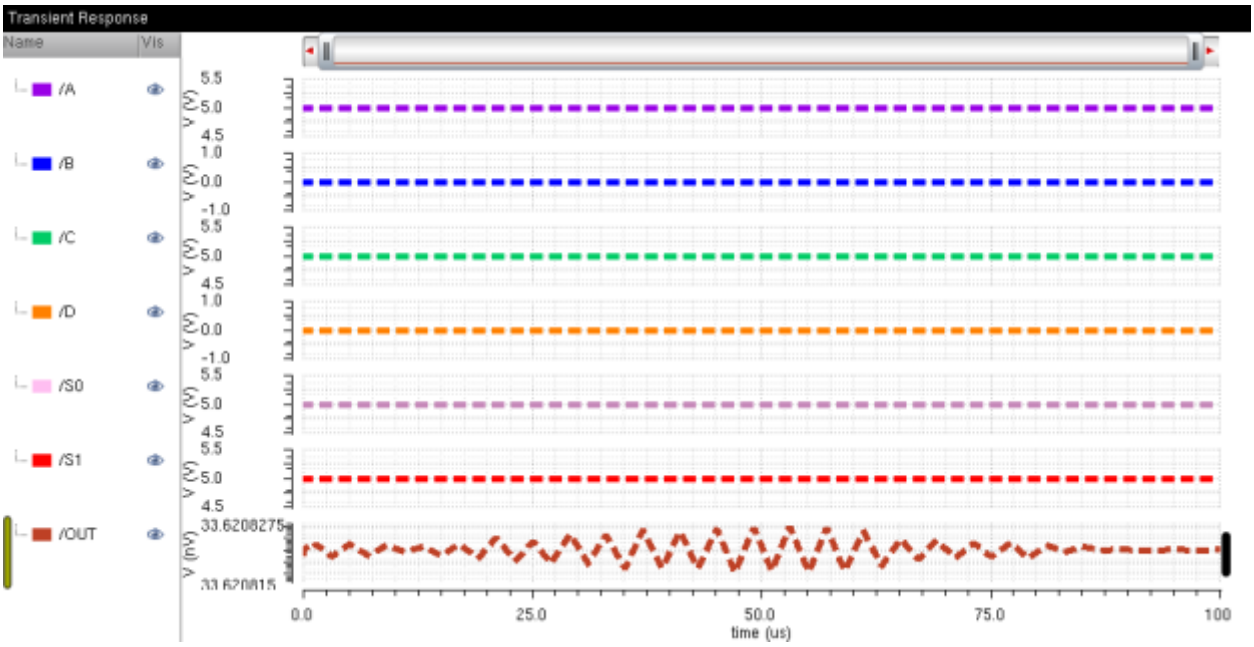


4-to-1 MUX Layout (Area ≈ 31.8x39µm)


4-to-1 MUX Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/sea
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
          14              nets
          9               terminals
          9               pmos
          9               nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
          14              nets
          9               terminals
          9               pmos
          9               nmos


    Terminal correspondence points
    N12         N6          A
    N11         N2          B
    N10         N8          C
    N9          N3          D
    N8          N5          OUT
    N6          N4          S0
    N5          N10         S1
    N7          N0          gnd!
    N13         N1          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.
```

4-to-1 MUX LVS Report

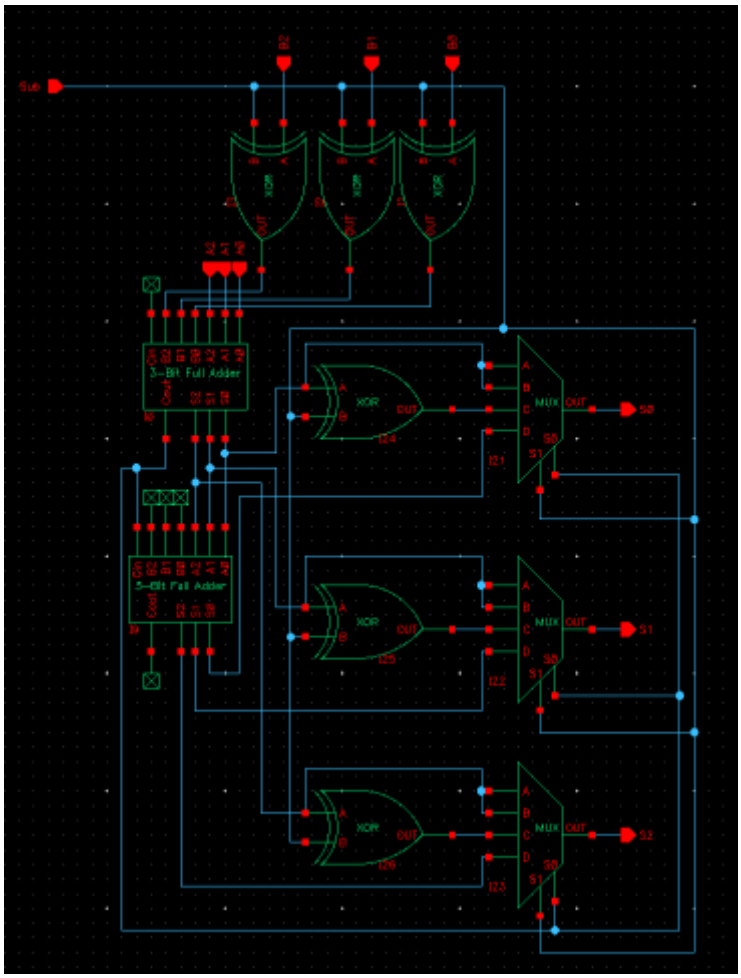


4-to-1 MUX Simulation

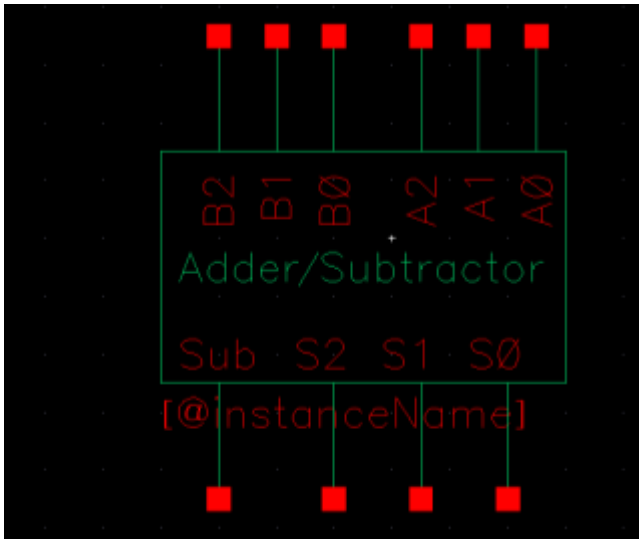
3-Bit Adder Subtractor

The 3 bit adder subtractor consists of 4 to 1 multiplexers, 3 bit adders, and xor gates. The 3 bit adder subtractor either subtracts or adds two three bit numbers depending on the input. The 3 bit adder subtractor is used in the comparison module to figure out which elevator is closest to the floor selected.

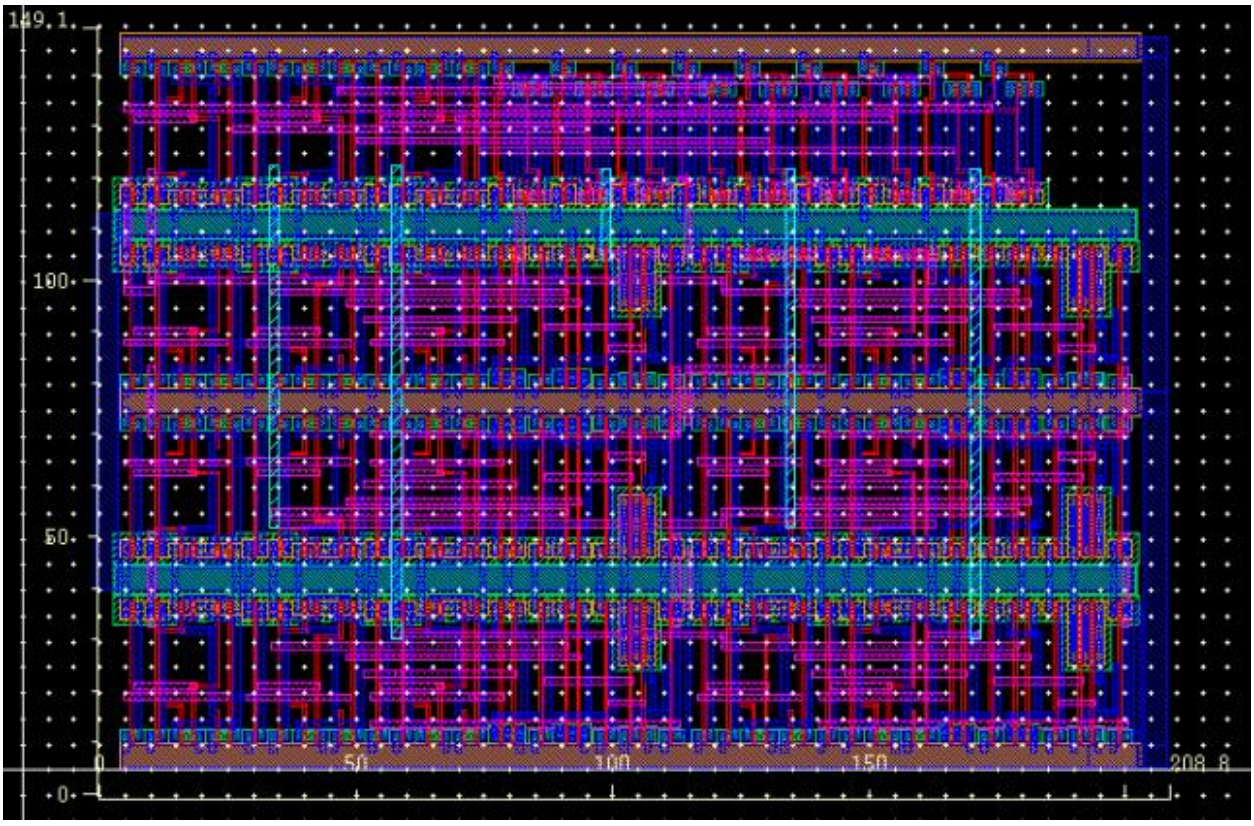
Inputs	Outputs
A0	S0
B0	S1
A1	S2
B1	
A2	
B2	
Sub	



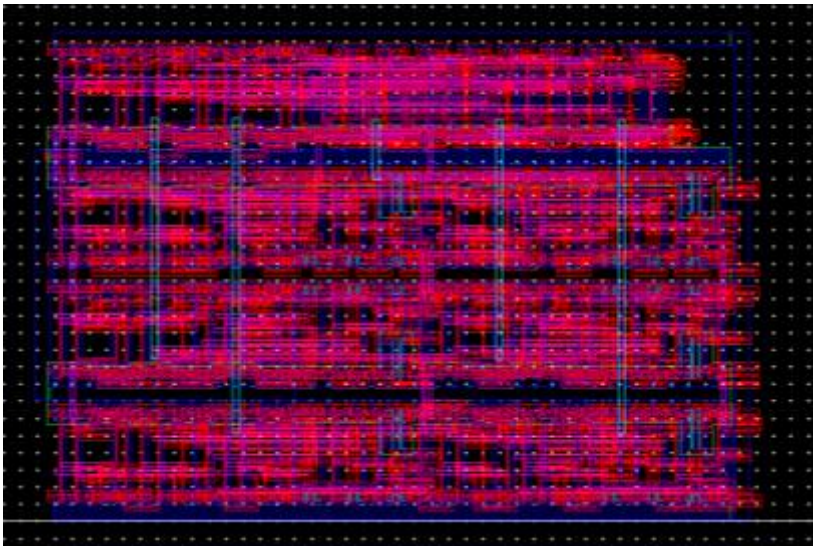
3-Bit Adder Subtractor Schematic



3-Bit Adder Subtractor Symbol



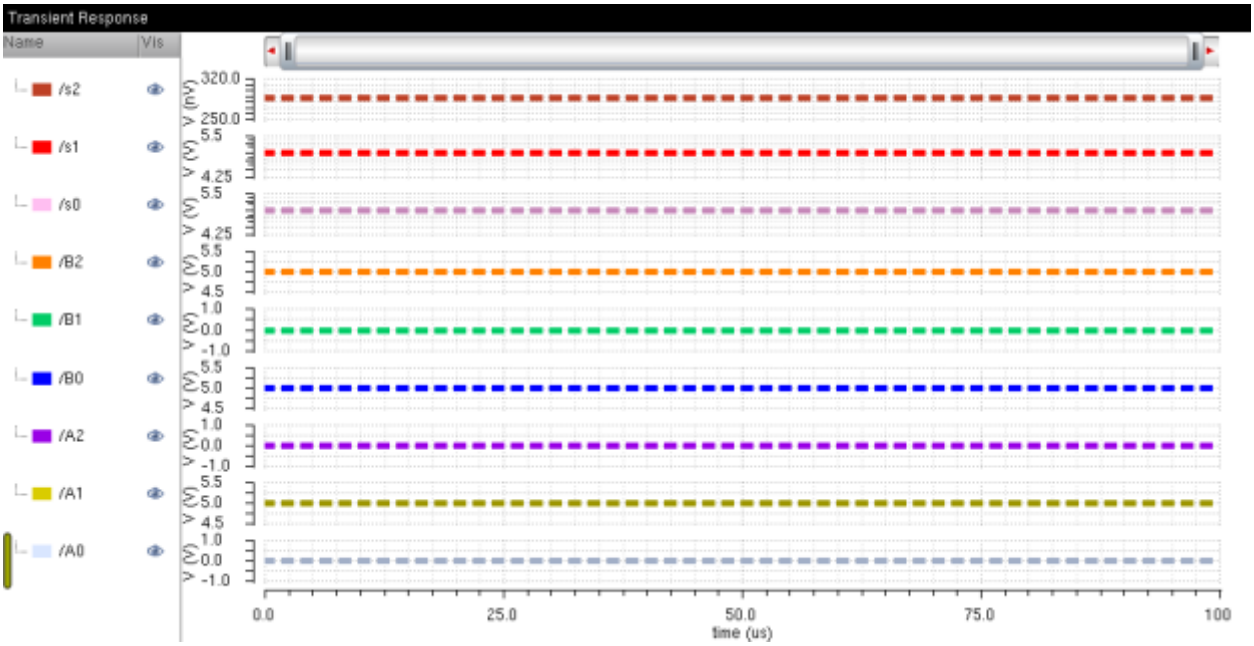
3-Bit Adder Subtractor Layout (Area ≈ 149x209µm)



3-Bit Adder Subtractor Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/se
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
         193             nets
          12             terminals
         189             pmos
         189             nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
         193             nets
          12             terminals
         189             pmos
         189             nmos


    Terminal correspondence points
    N186      N24       A0
    N185      N16       A1
    N184      N4        A2
    N192      N9        B0
    N191      N28       B1
    N190      N14       B2
    N183      N13       S0
    N182      N29       S1
    N181      N25       S2
    N189      N2        Sub
    N187      N0        gnd!
    N188      N1        vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.
```

3-Bit Adder Subtractor LVS Report

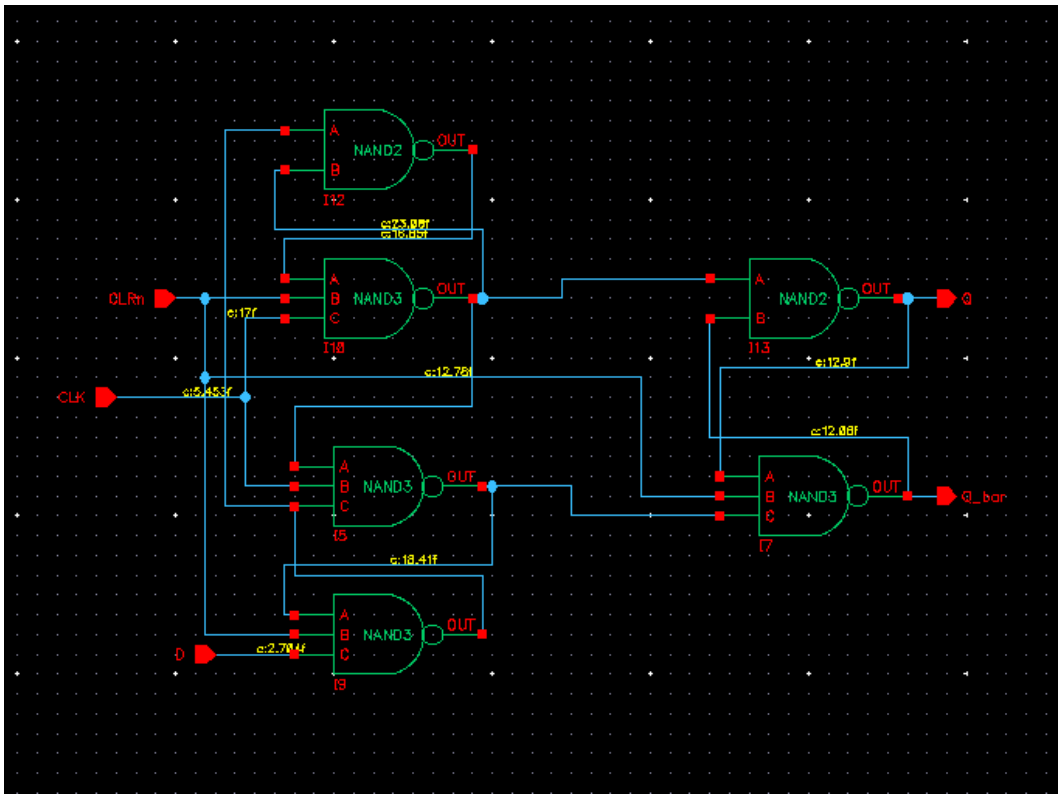


3-Bit Adder Subtractor Simulation

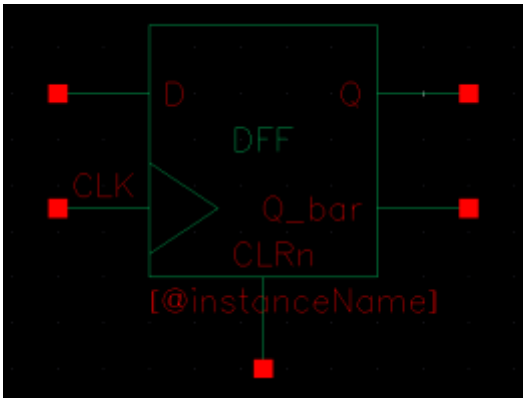
D Flip Flop

The d flip flop is used to create memory in the elevator control unit. A single d flip flop will store a data value for a clock cycle. D flip flops are cascaded together to create registers.

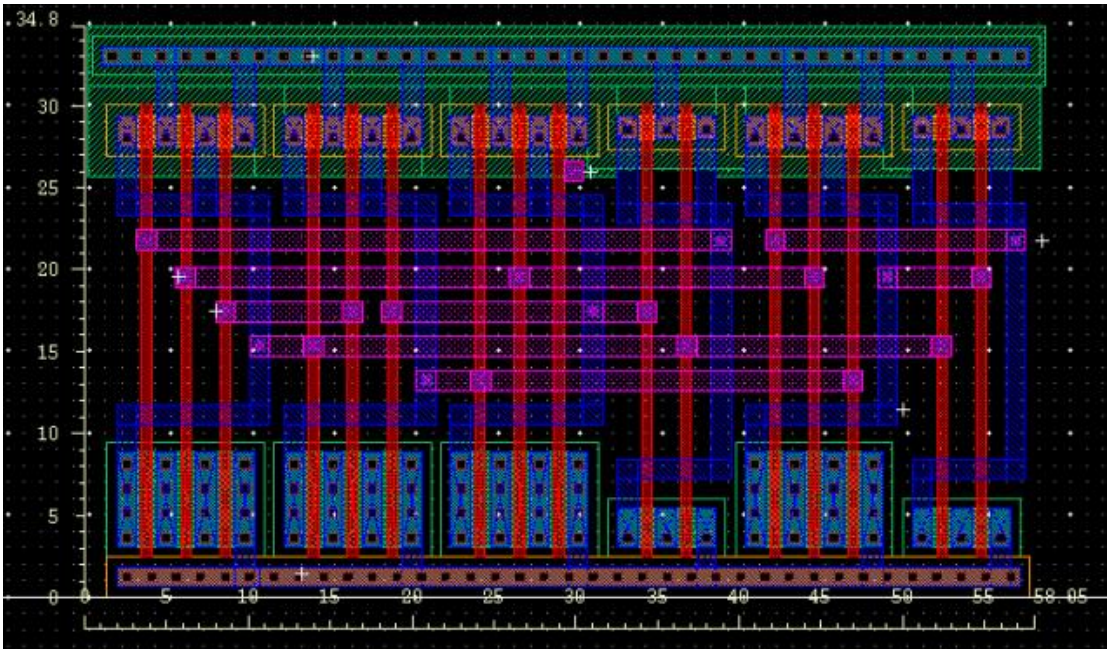
Inputs	Outputs
D	Q
CLK	Q_bar
CLRn	



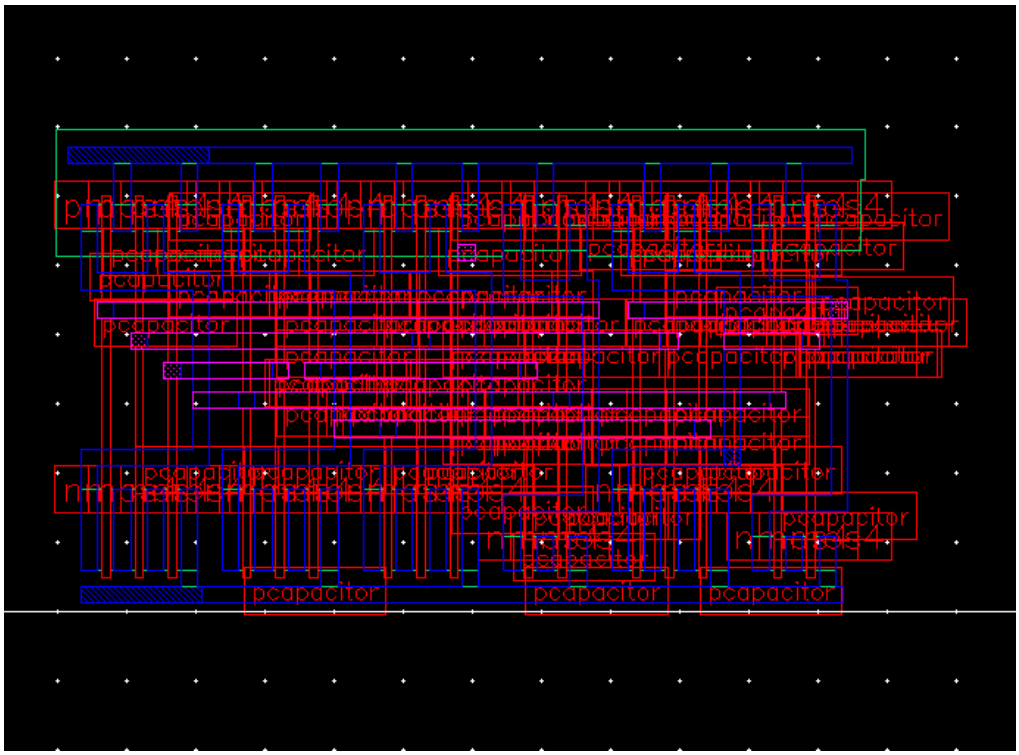
D Flip Flop Schematic



D Flip Flop Symbol



D Flip Flop Layout (Area ≈ 31.8x58µm)



D Flip Flop Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjrd1224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/s
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
        21              nets
        7               terminals
        16              pmos
        16              nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
        21              nets
        7               terminals
        16              pmos
        16              nmos


    Terminal correspondence points
    N18         N8          CLK
    N15         N3          CLRn
    N19         N9          D
    N16         N4          Q
    N17         N10         Q_bar
    N14         N0          gnd!
    N20         N1          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.
```

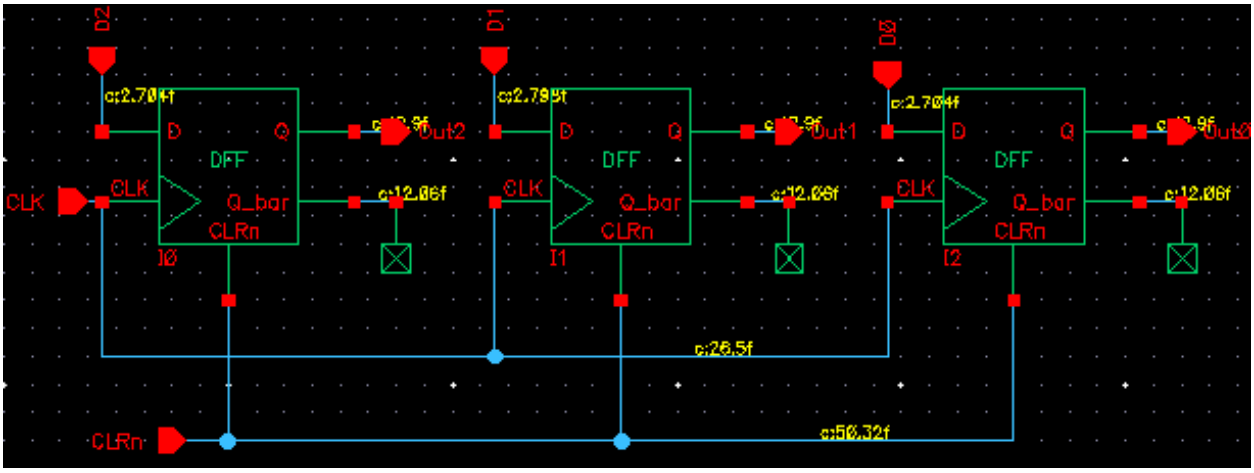
D Flip Flop LVS Report

D Flip Flop Simulation

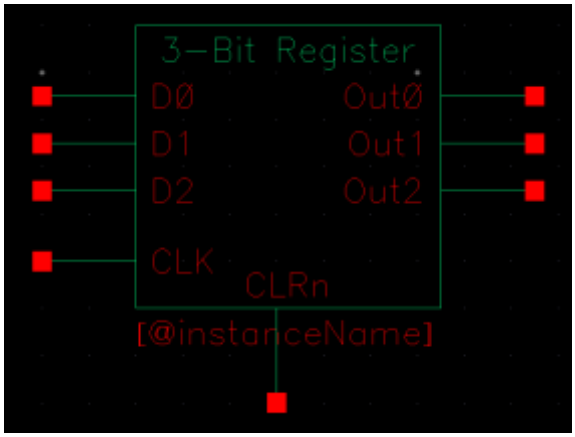
3-Bit Register

The 3 bit register consists of three d flip flops. The register stores three data values every clock cycle. The register is used in the elevator control unit to store the values of button presses.

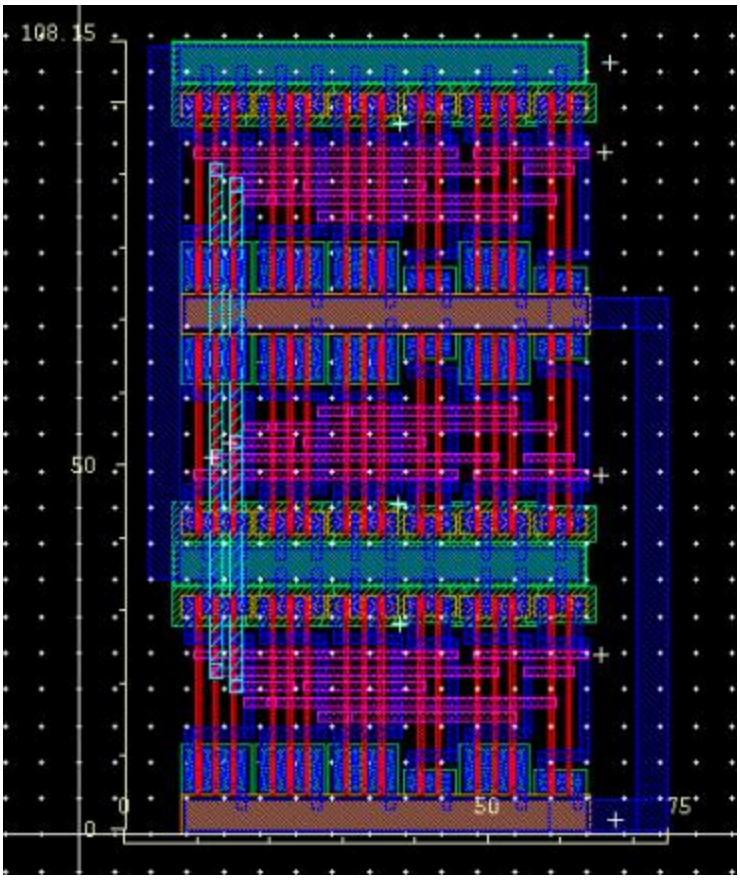
Inputs	Outputs
D0	Out0
D1	Out1
D2	Out2
CLK	
CLRn	



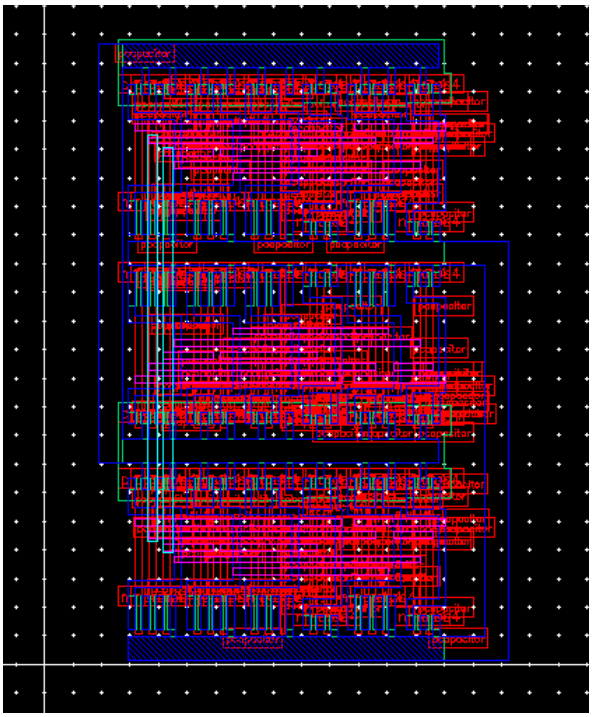
3-Bit Register Schematic



3-Bit Register Symbol

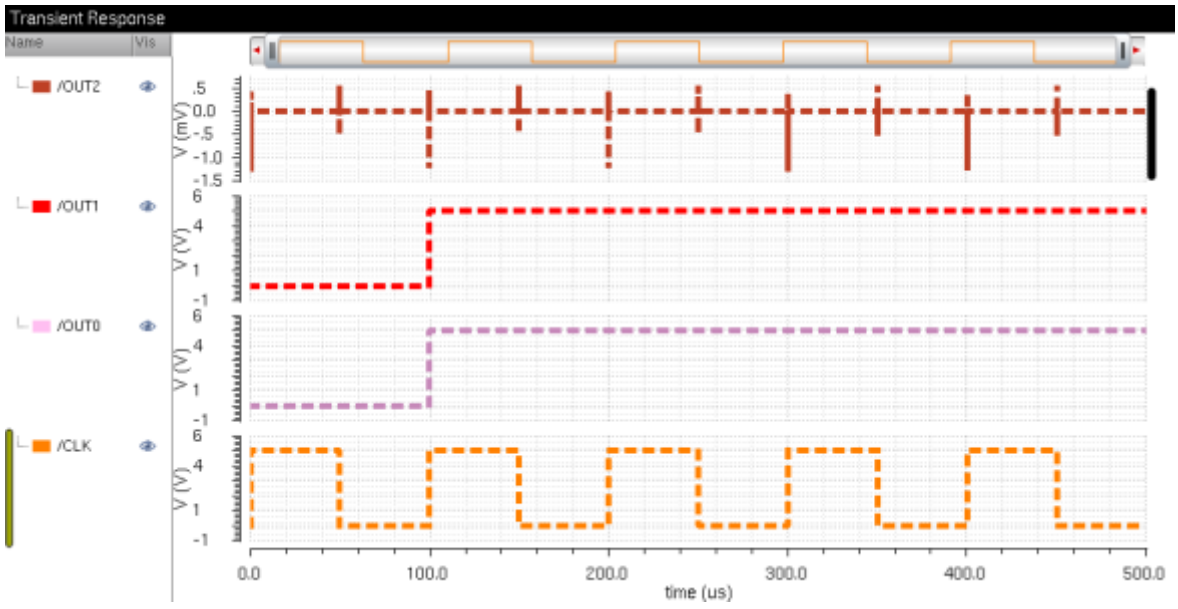


3-Bit Register Layout (Area ≈ 108x65µm)


3-Bit Register Extracted View

```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
       count
        55              nets
        10              terminals
        48              pmos
        48              nmos

    Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
       count
        55              nets
        10              terminals
        48              pmos
        48              nmos


    Terminal correspondence points
    N50         N7          CLK
    N47         N10         CLRn
    N54         N4          D0
    N53         N12         D1
    N52         N11         D2
    N49         N8          Out0
    N48         N9          Out1
    N46         N3          Out2
    N45         N0          gnd!
    N51         N1          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.
```

3-Bit Register LVS Report

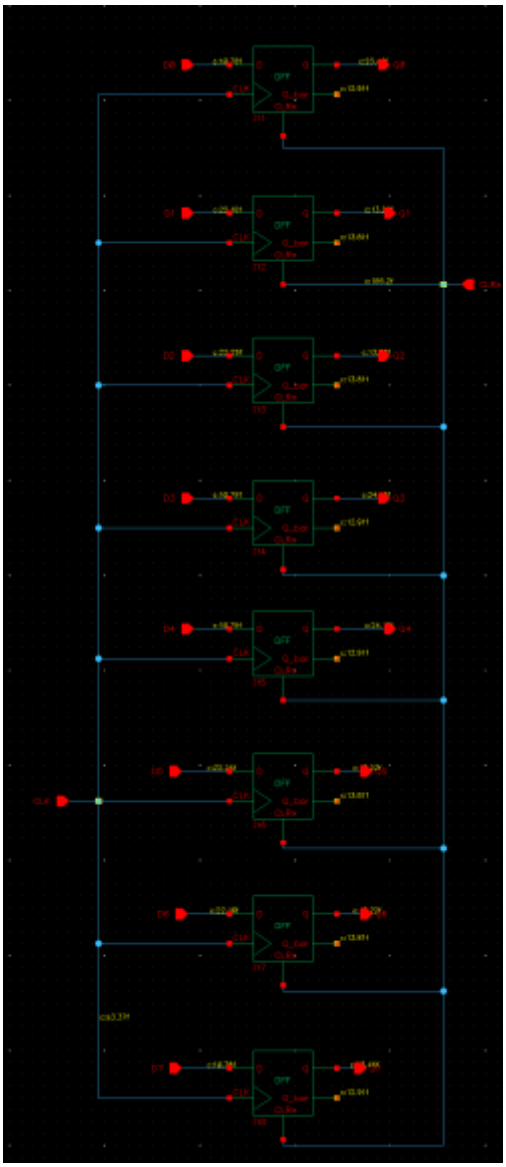


3-Bit Register Simulation

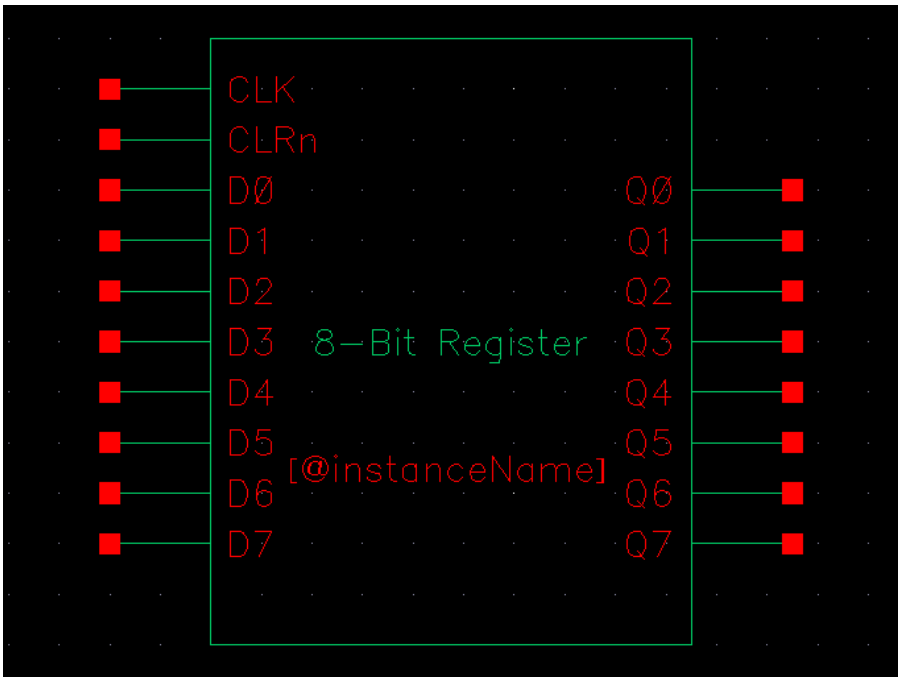
8-Bit Register

The 8 bit register is similar to the 3 bit register in terms of operation. The 8 bit register stores eight data inputs every clock cycle. The 8 bit adder is used to store the locations of each elevator.

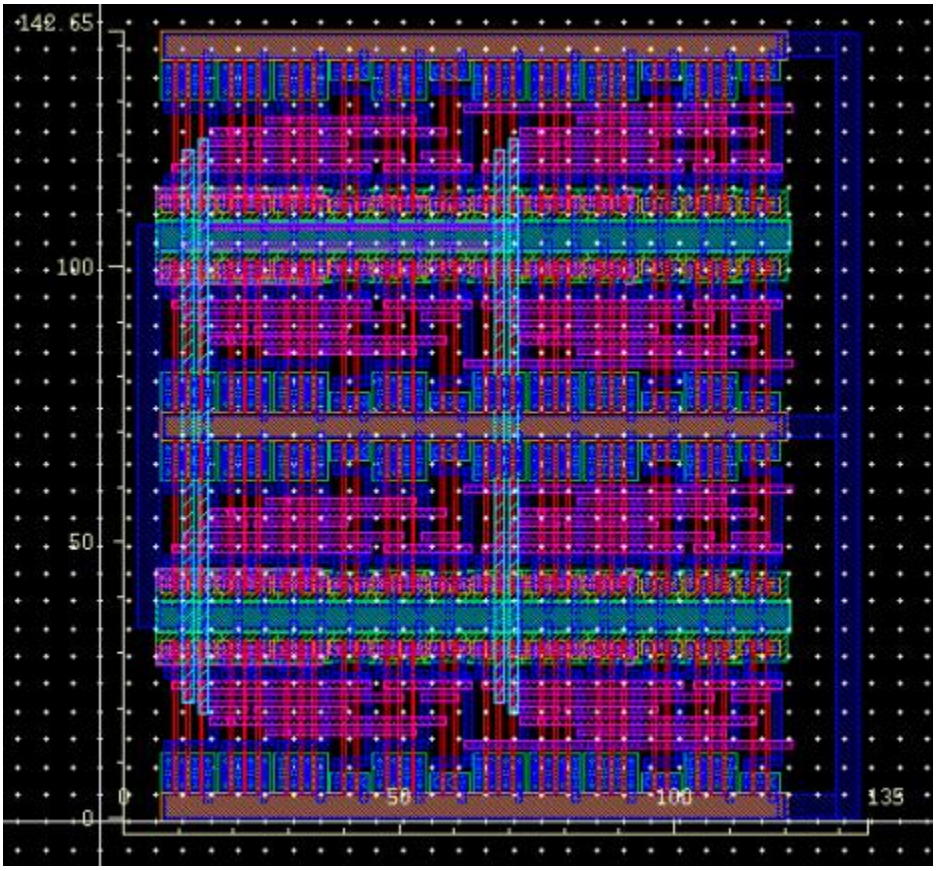
Inputs	Outputs
D0	Q0
D1	Q1
D2	Q2
D3	Q3
D4	Q4
D5	Q5
D6	Q6
D7	Q7
CLK	
CLRn	



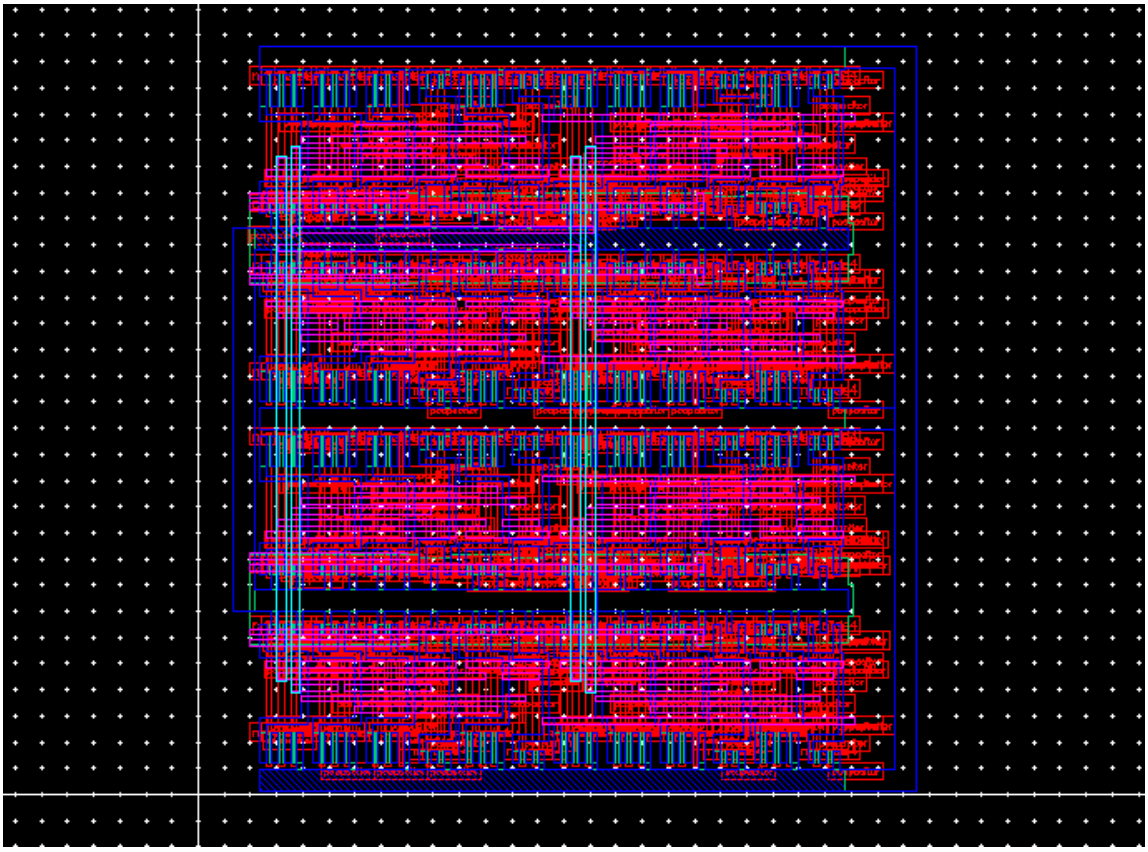
8-Bit Register Schematic



8-Bit Register Symbol



8-Bit Register Layout (Area ≈ 143x125µm)



8-Bit Register Extracted View

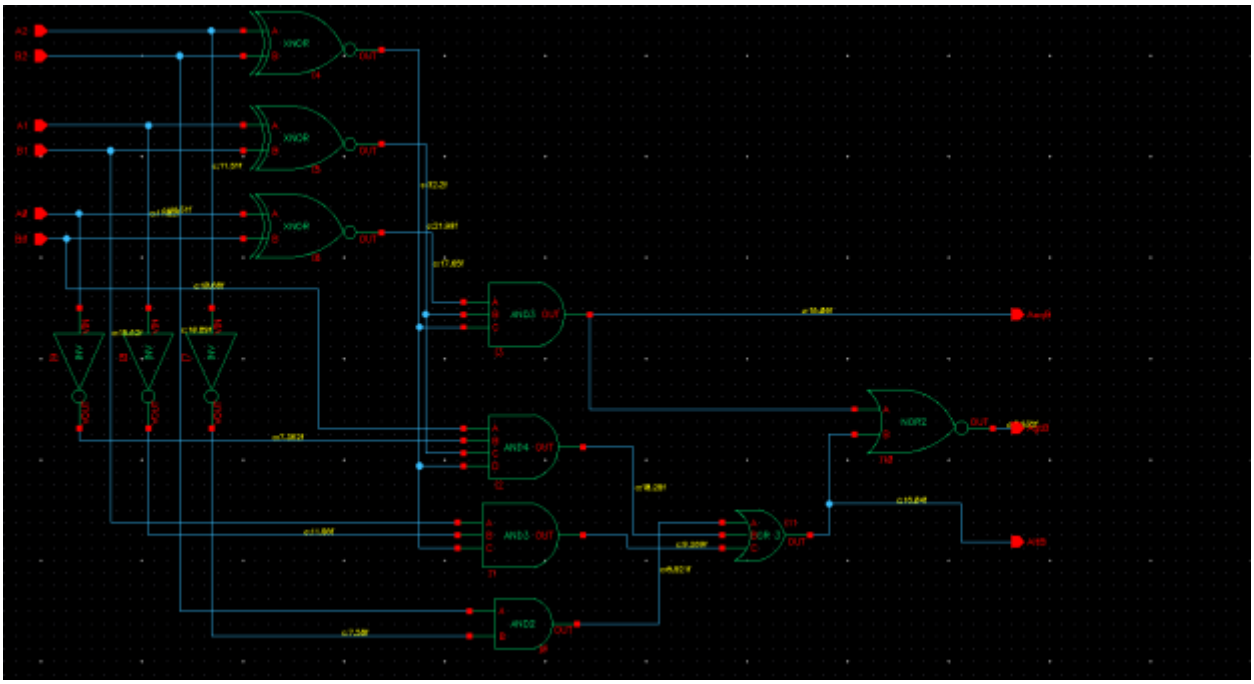
```
Net-list summary for /home/seas/crgraham/cadence/LVS/layout/netlist
   count
     140             nets
     20              terminals
     128             pmos
     128             nmos

Net-list summary for /home/seas/crgraham/cadence/LVS/schematic/netlist
   count
     140             nets
     20              terminals
     128             pmos
     128             nmos


Terminal correspondence points
N130        N16         CLK
N129        N27         CLRn
N139        N19         D0
N138        N25         D1
N137        N13         D2
N136        N23         D3
N135        N22         D4
N134        N26         D5
N133        N9          D6
N132        N7          D7
N128        N24         Q0
N127        N10         Q1
N126        N18         Q2
N125        N17         Q3
N124        N20         Q4
N123        N15         Q5
N122        N12         Q6
N121        N8          Q7
N120        N0          gnd!
N131        N1          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.
```

8-Bit Register LVS Report

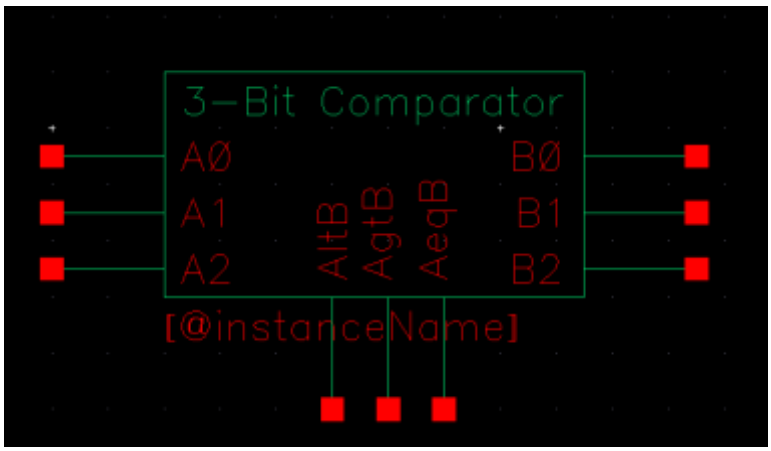
3-Bit Comparator

The 3 bit comparator is used to compare two binary numbers. The 3 bit comparator is used in the comparison module to compare the location of each elevator.

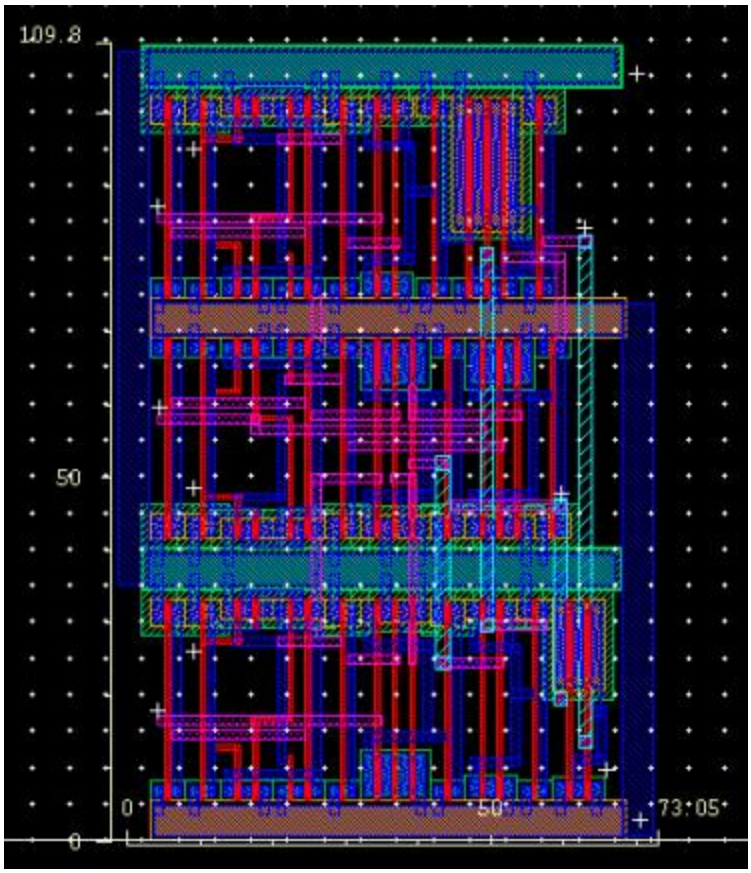
Inputs	Outputs
A0	AltB
B0	AgtB
A1	AeqB
B1	
A2	
B2	



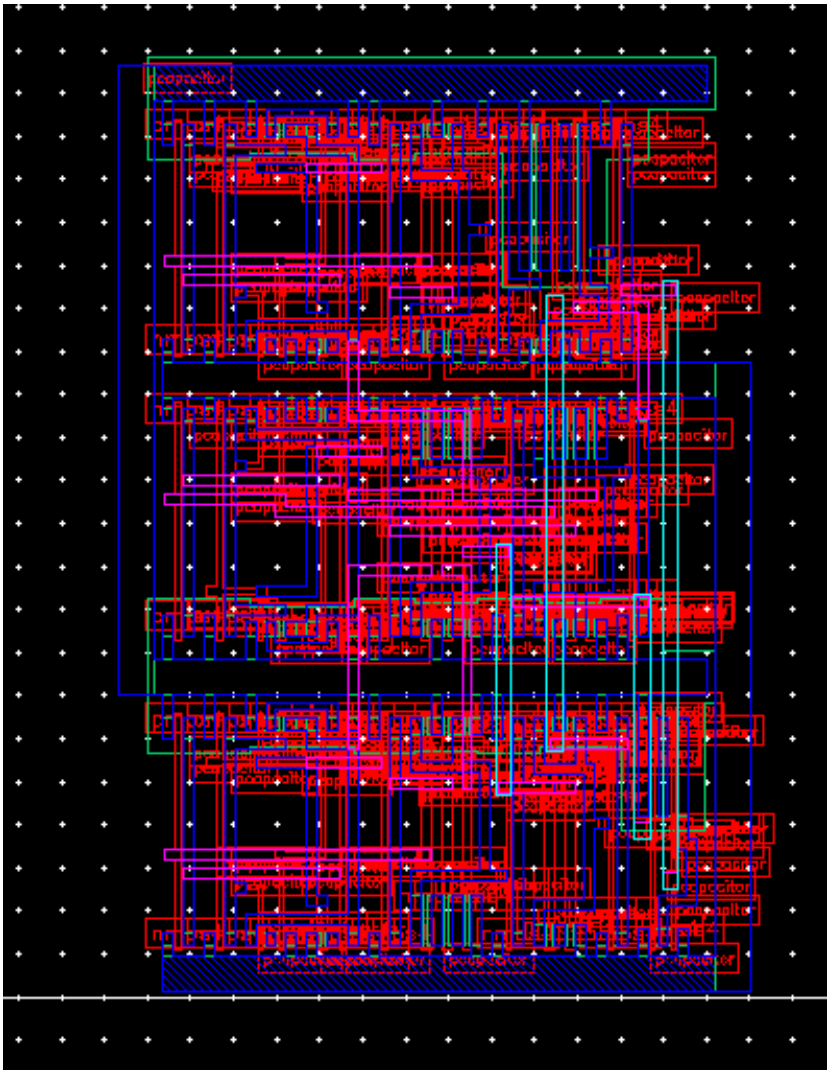
3-Bit Comparator Schematic



3-Bit Comparator Symbol



3-Bit Comparator Layout (Area ≈ 110x73µm)

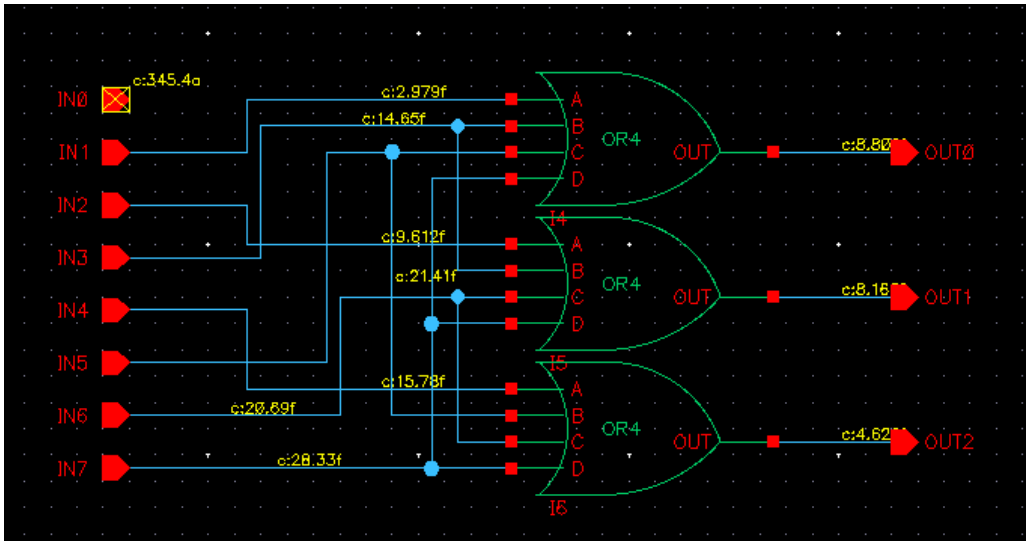


3-Bit Comparator Extracted View

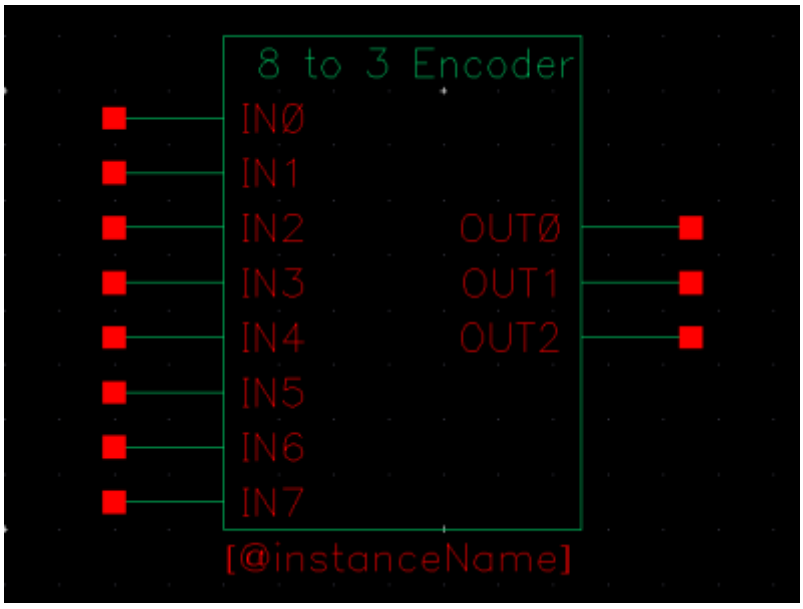
8-to-3 Encoder

The 8 to 3 encoder converts multiple inputs into a binary signal. The 8 to 3 encoder is used in the floor call module to encode the floor button presses into a binary number.

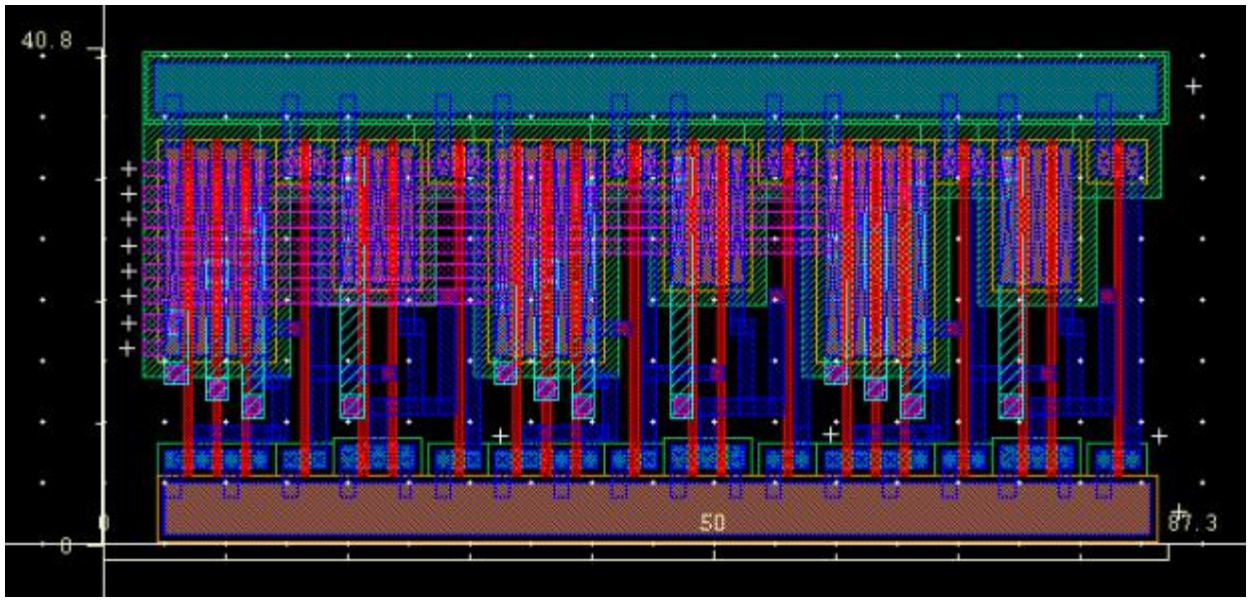
Inputs	Outputs
IN0	OUT0
IN1	OUT1
IN2	OUT2
IN3	
IN4	
IN5	
IN6	
IN7	



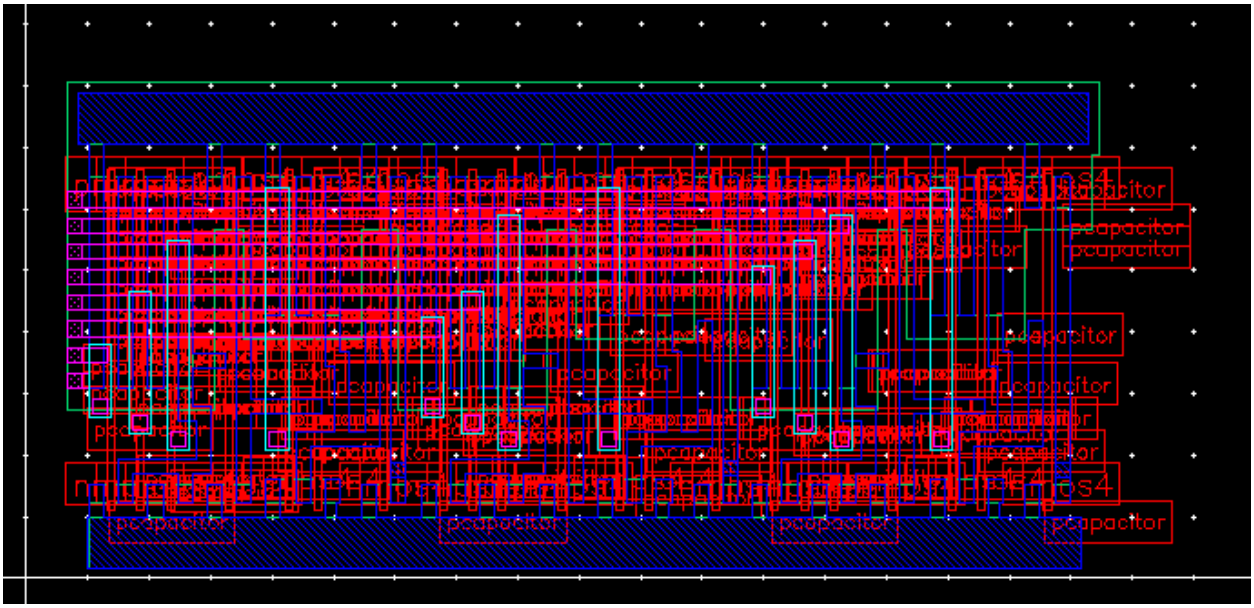
8-to-3 Encoder Schematic



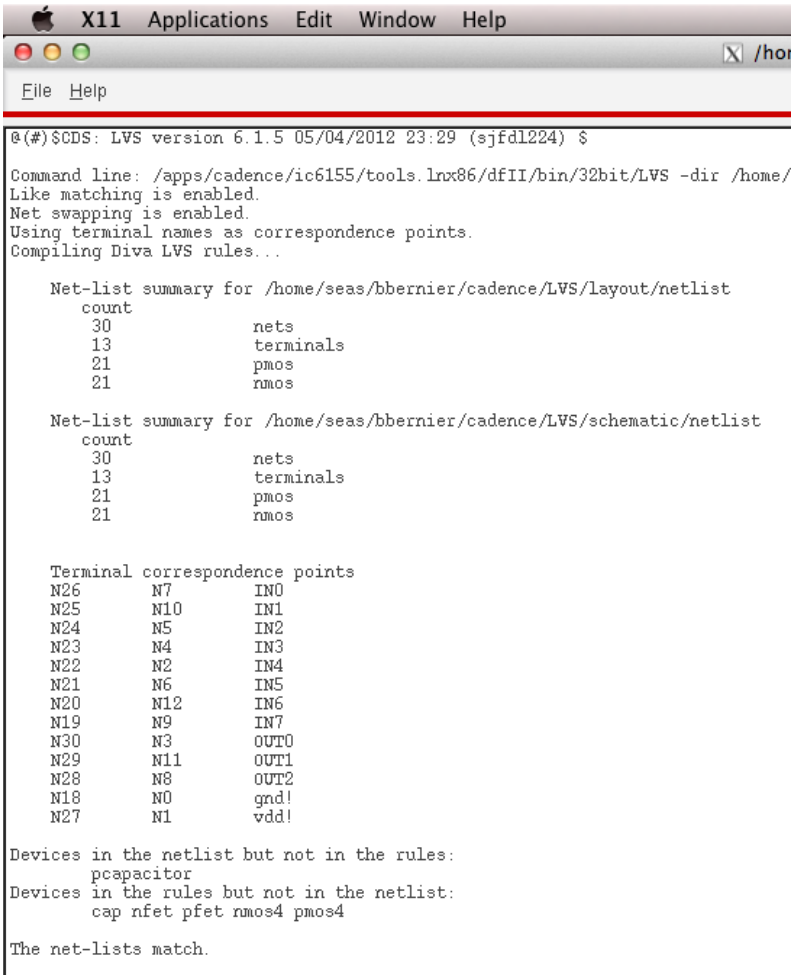
8-to-3 Encoder Symbol



8-to-3 Encoder Layout (Area ≈ 41x87µm)



8-to-3 Encoder Extracted View

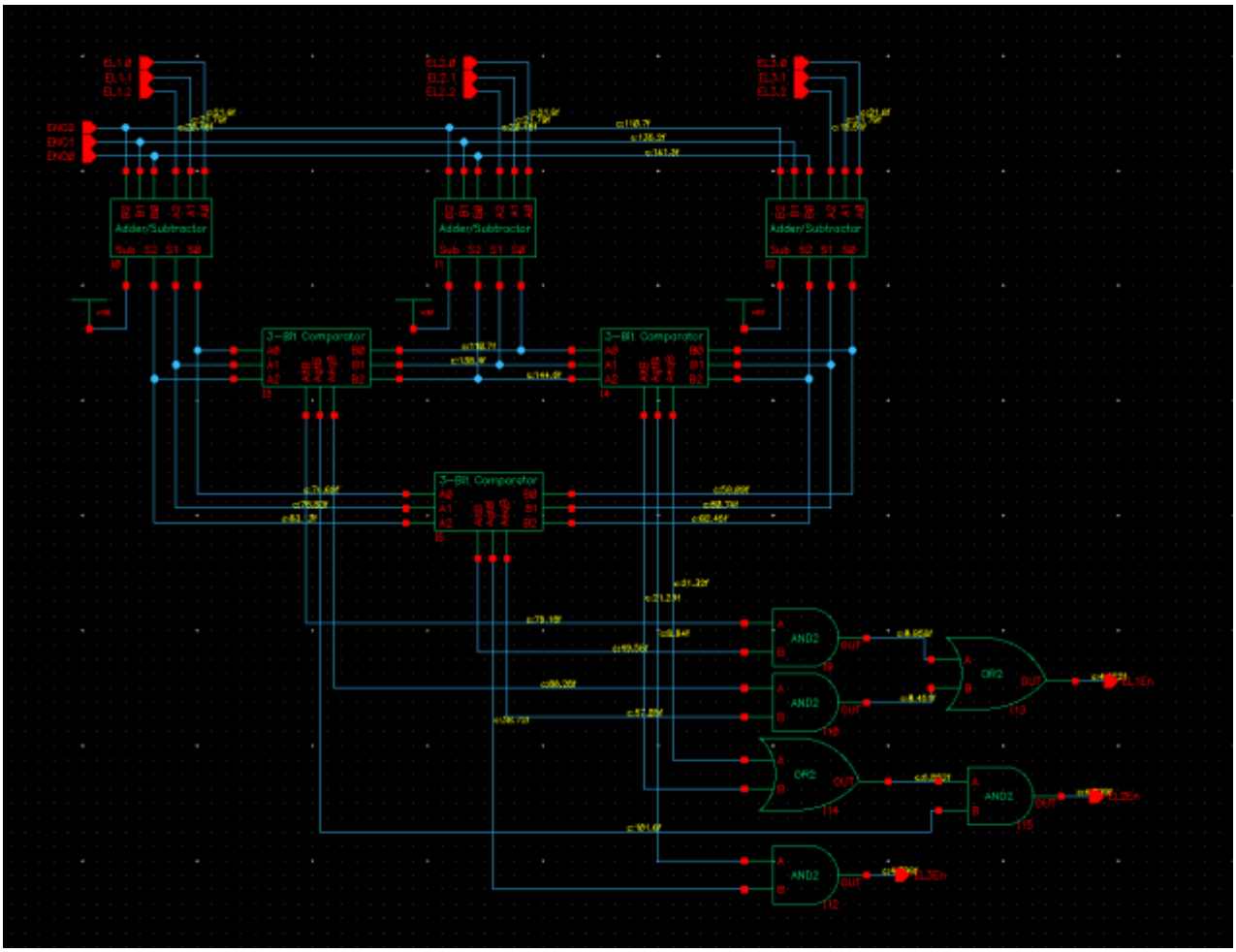


8-to-3 Encoder LVS Report

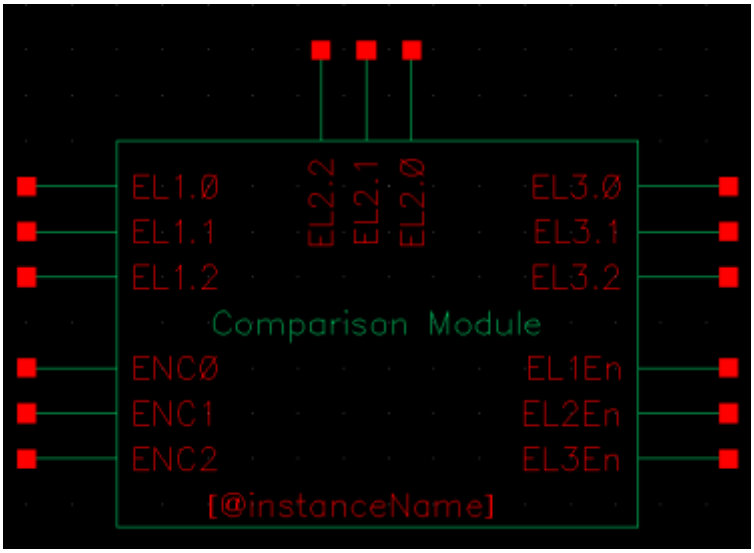
Comparison Module

The comparison module compares the location of each of the three elevators in the elevator control unit.

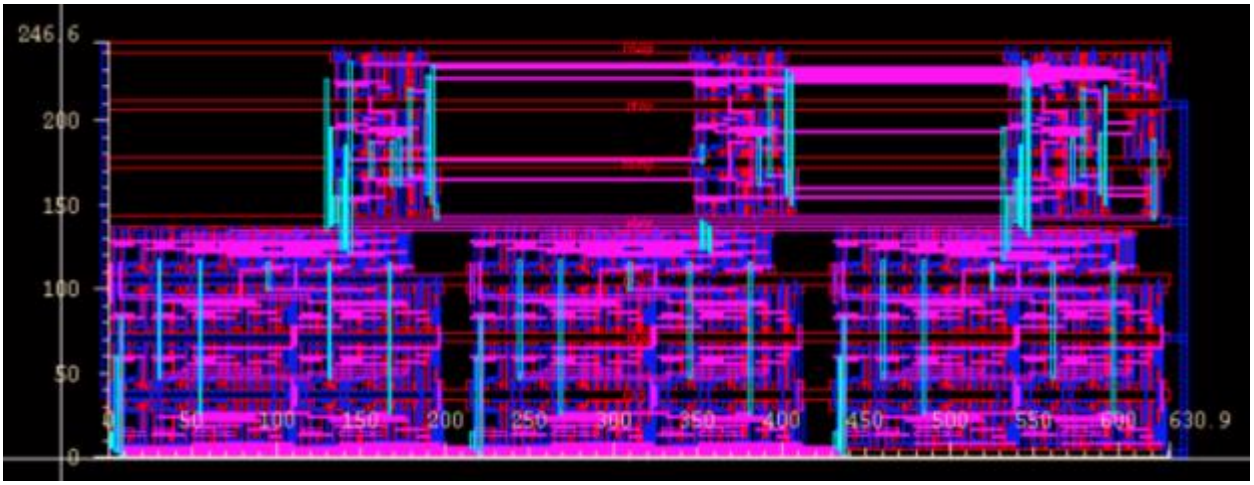
Inputs	Outputs
ENC0	EL1En
ENC1	EL2En
ENC2	EL3En
EL1.0	
EL1.1	
EL1.2	
EL2.0	
EL2.1	
EL2.2	
EL3.0	
EL3.1	
EL3.2	



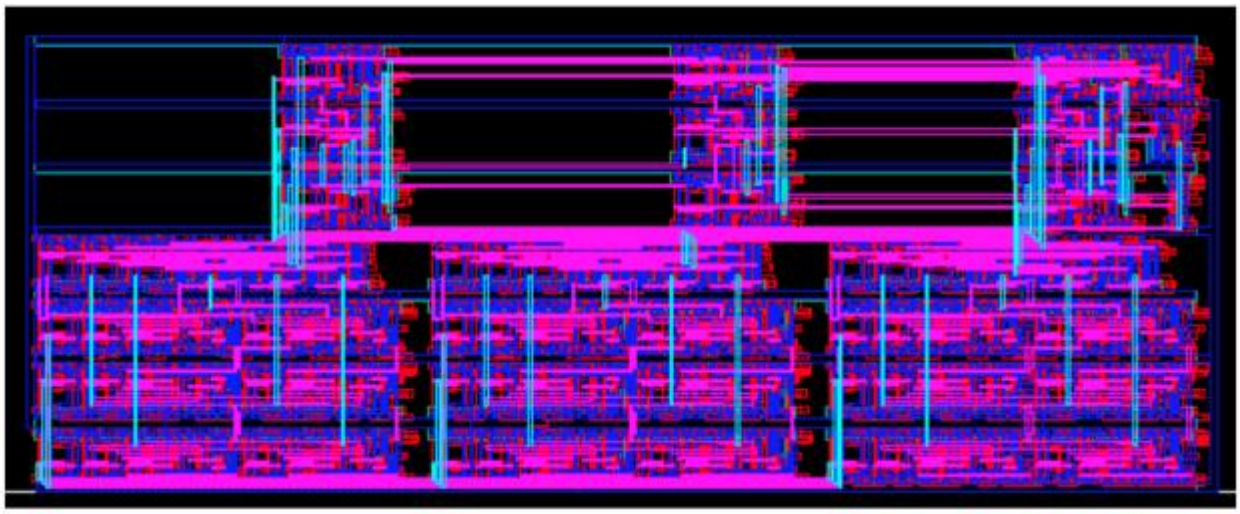
Comparison Module Schematic

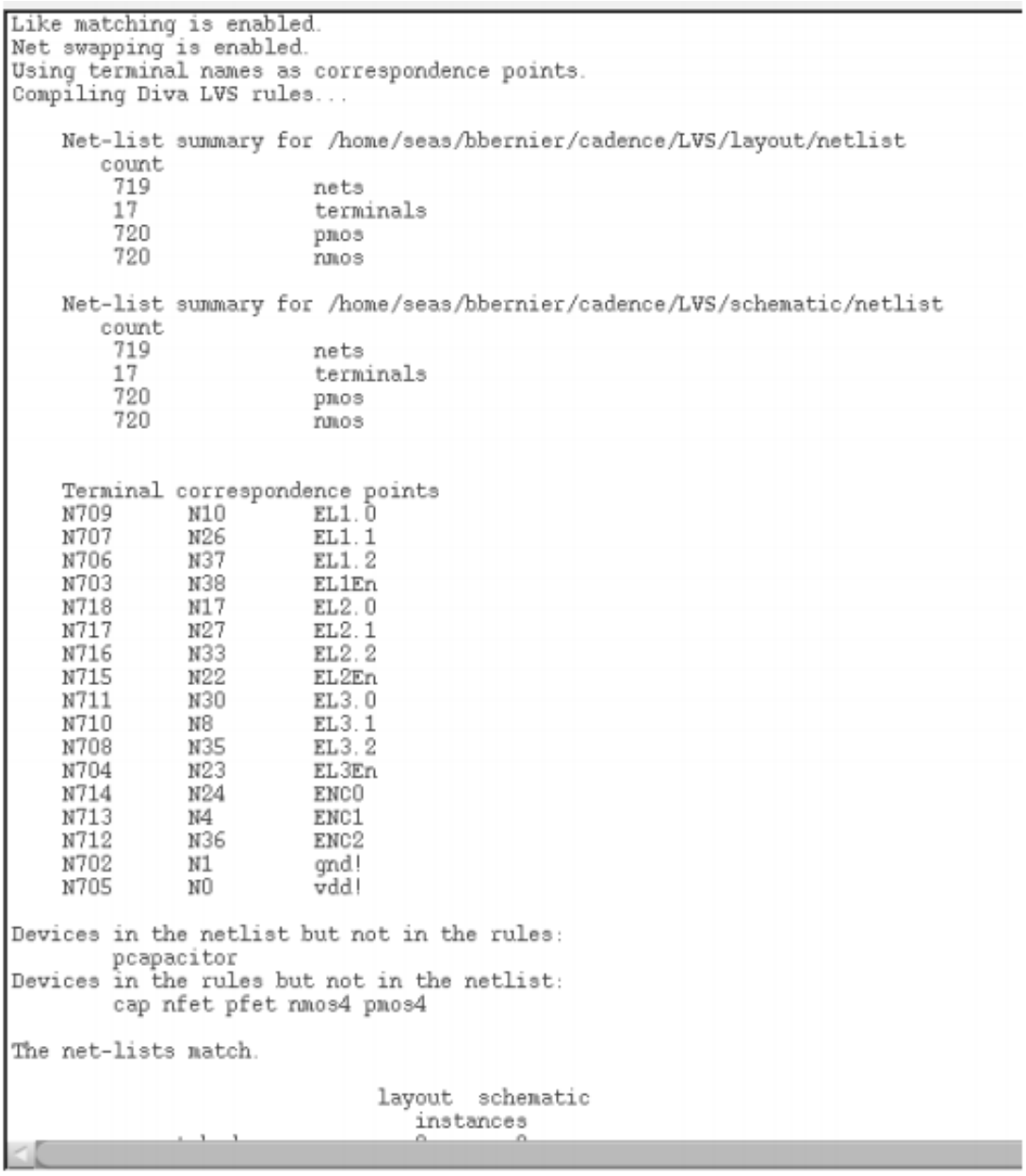


Comparison Module Symbol



Comparison Module Layout (Area ≈ 247x631µm)


Comparison Module Extracted View



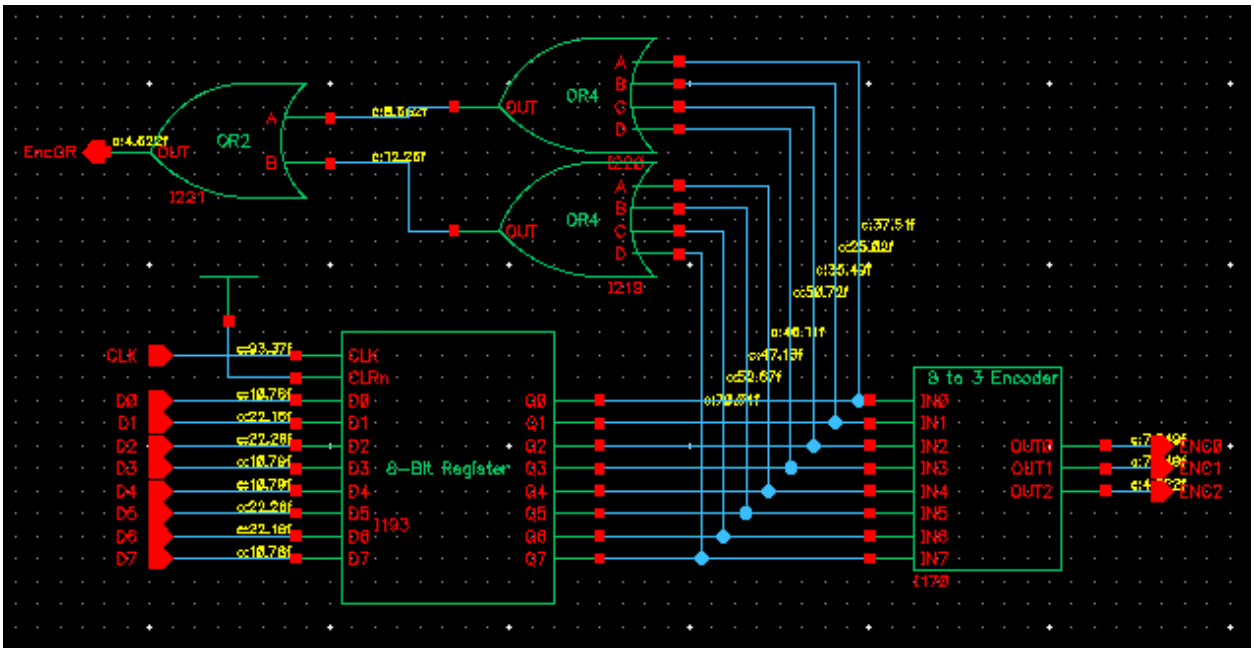
```
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/bbernier/cadence/LVS/layout/netlist
       count
         719              nets
         17               terminals
         720              pmos
         720              nmos

    Net-list summary for /home/seas/bbernier/cadence/LVS/schematic/netlist
       count
         719              nets
         17               terminals
         720              pmos
         720              nmos


    Terminal correspondence points
    N709        N10         EL1.0
    N707        N26         EL1.1
    N706        N37         EL1.2
    N703        N38         EL1En
    N718        N17         EL2.0
    N717        N27         EL2.1
    N716        N33         EL2.2
    N715        N22         EL2En
    N711        N30         EL3.0
    N710        N8          EL3.1
    N708        N35         EL3.2
    N704        N23         EL3En
    N714        N24         ENC0
    N713        N4          ENC1
    N712        N36         ENC2
    N702        N1          gnd!
    N705        N0          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.

                              layout  schematic
                                 instances
```

Comparison Module LVS Report

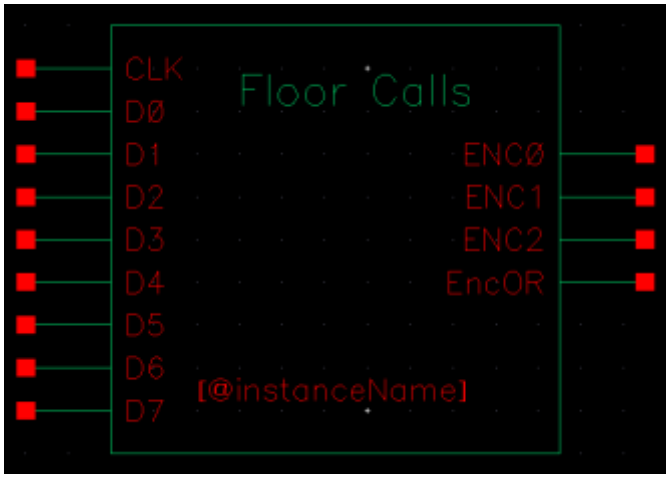
Floor Calls Module

The floor calls module takes the elevator floor button presses and converts them into a 3 bit output. The module also checks to see if a button has been pressed.

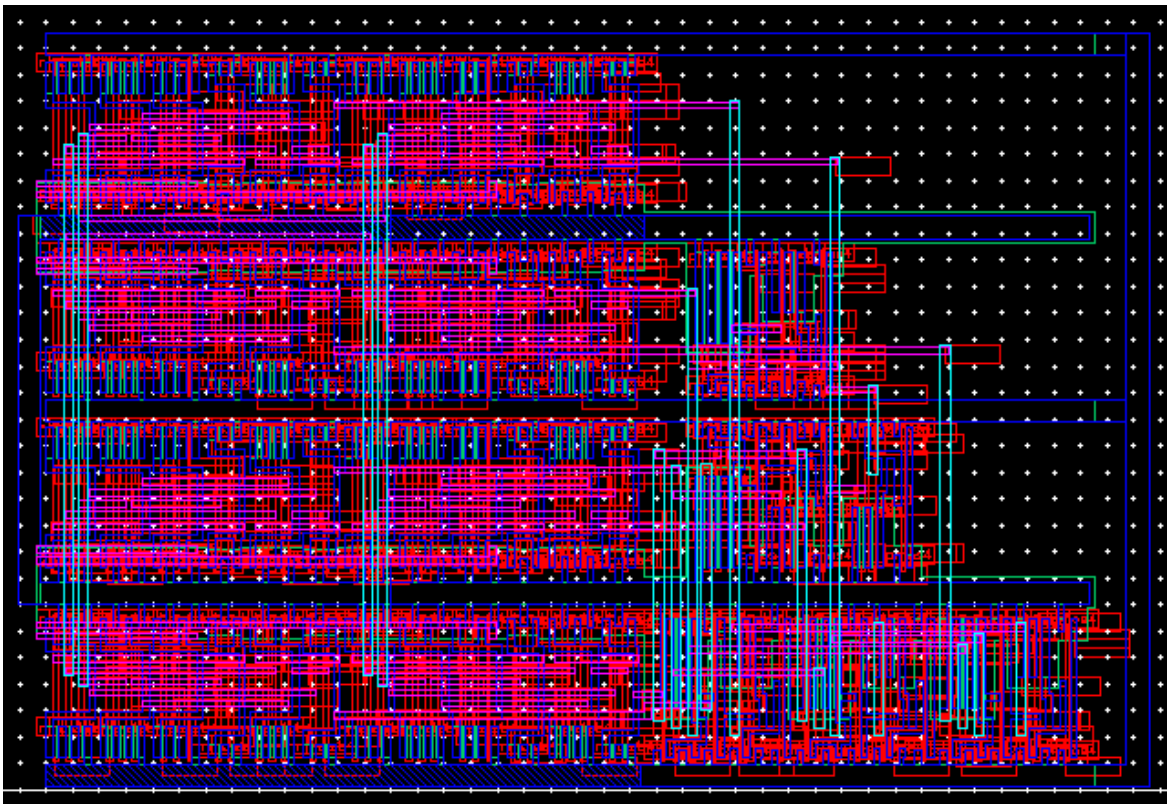
Inputs	Outputs
CLK	ENC0
D0	ENC1
D1	ENC2
D2	EncOR
D3	
D4	
D5	
D6	
D7	



Floor Calls Module Schematic



Floor Calls Module Symbol



Floor Calls Module Extracted View (Area ≈ 143x199µm)

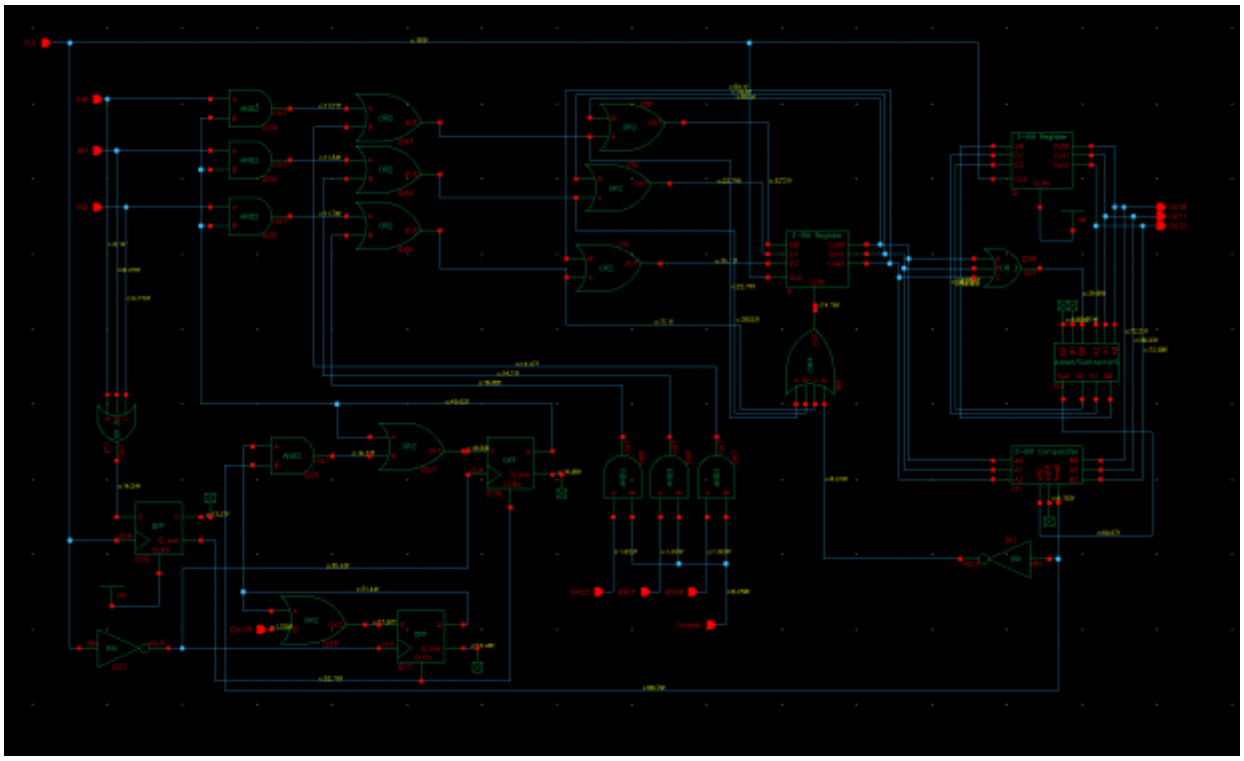
```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/seas
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/bbernier/cadence/LVS/layout/netlist
       count
         177            nets
          15            terminals
         166            pmos
         166            nmos

    Net-list summary for /home/seas/bbernier/cadence/LVS/schematic/netlist
       count
         177            nets
          15            terminals
         166            pmos
         166            nmos


    Terminal correspondence points
    N163      N10       CLK
    N176      N22       D0
    N175      N18       D1
    N174      N23       D2
    N173      N17       D3
    N172      N15       D4
    N171      N16       D5
    N170      N21       D6
    N169      N24       D7
    N168      N12       ENC0
    N167      N11       ENC1
    N166      N25       ENC2
    N165      N5        EncOR
    N162      N1        gnd!
    N164      N0        vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.

                            layout  schematic
```

Floor Calls Module LVS Report

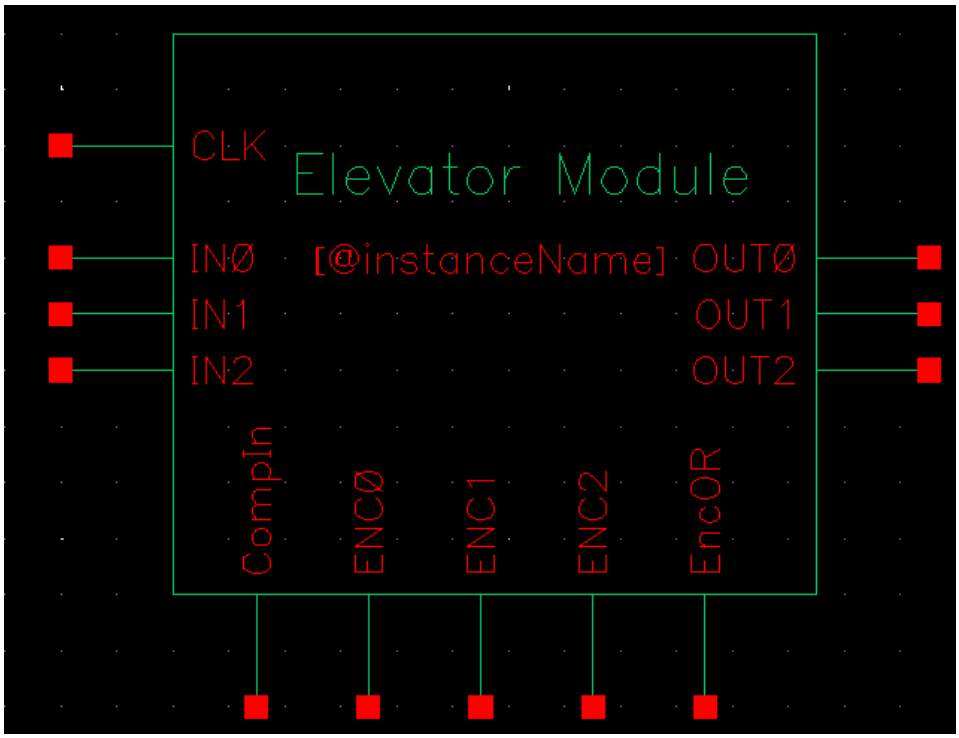
Elevator Module

The elevator module contains all the logic for the elevator to move up and down.

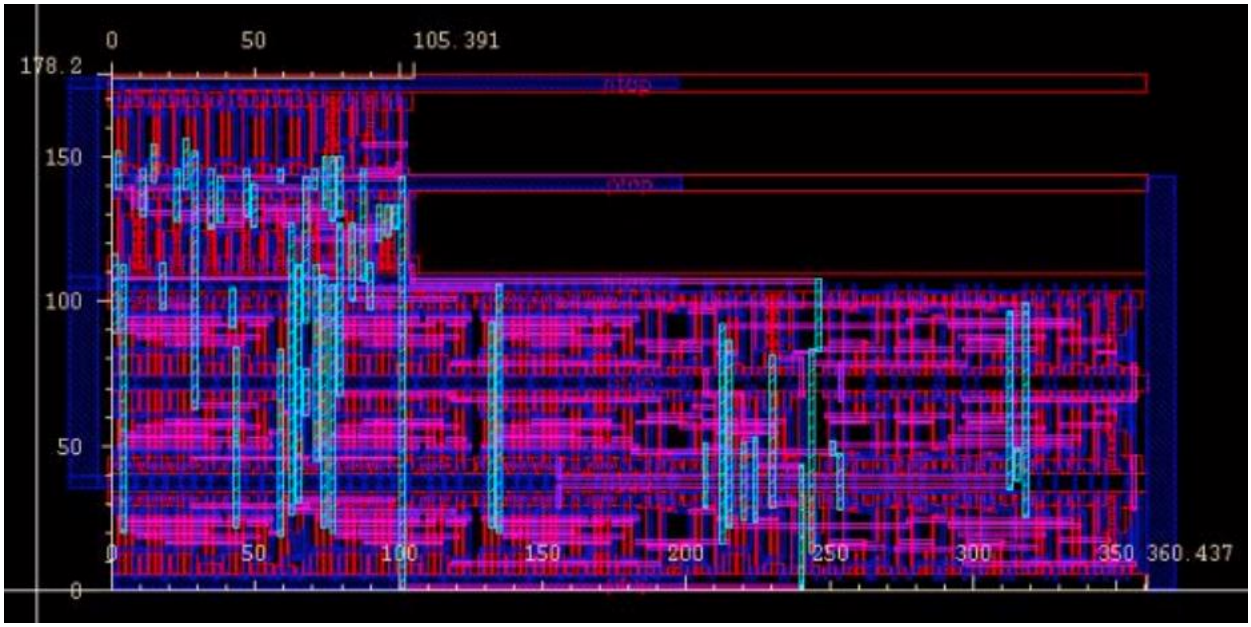
Inputs	Outputs
CLK	OUT0
IN0	OUT1
IN1	OUT2
IN2	
EncOR	
ENC0	
ENC1	
ENC2	
CompIn	



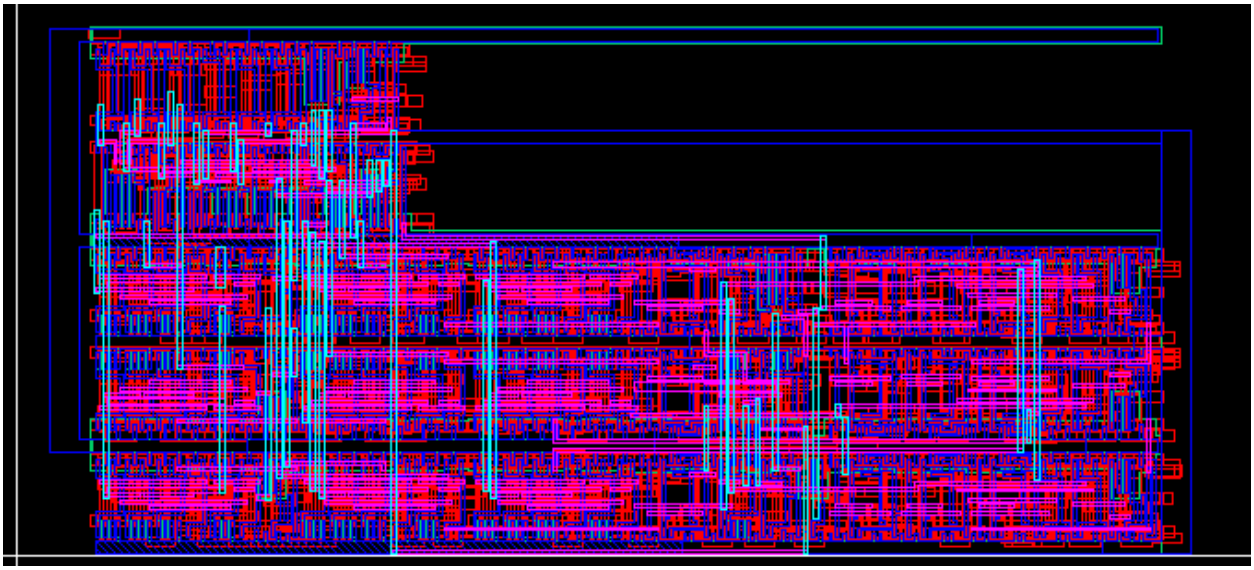
Elevator Module Schematic



Elevator Module Symbol



Elevator Module Layout (Area ≈ 178x360µm)



Elevator Module Extracted View

X11 Applications Edit Window Help

/hom

File Help

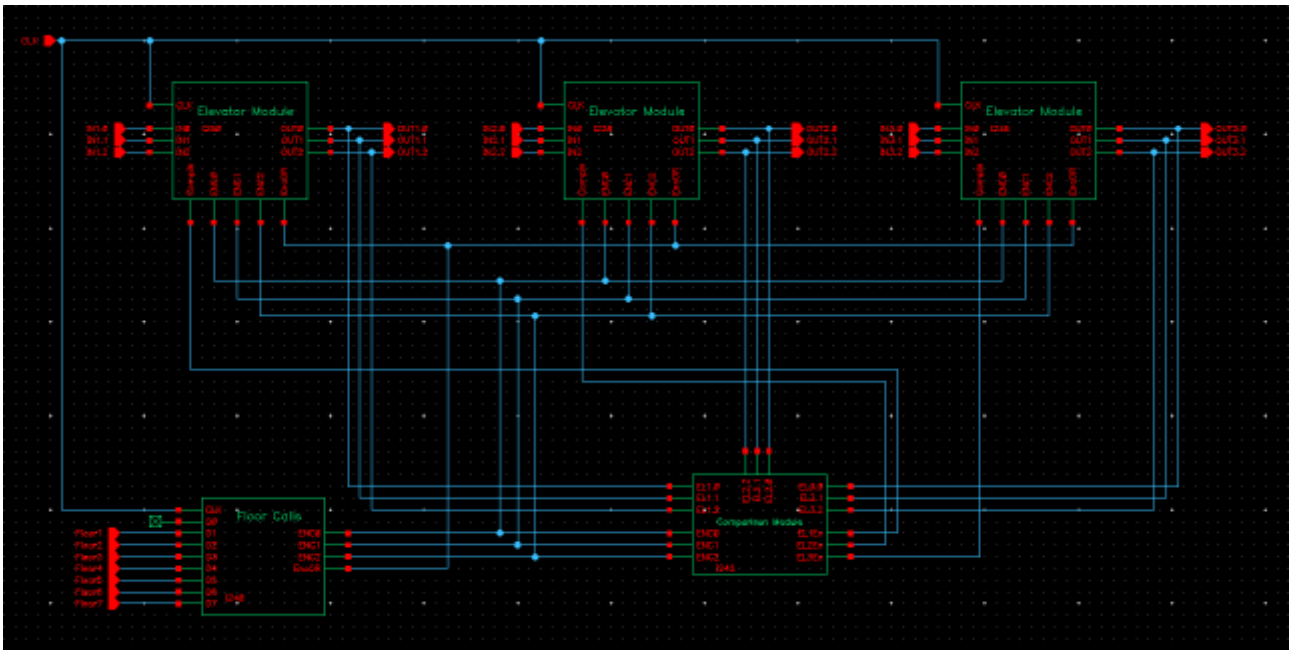
```
@(#)$CDS: LVS version 6.1.5 05/04/2012 23:29 (sjfdl224) $

Command line: /apps/cadence/ic6155/tools.lnx86/dfII/bin/32bit/LVS -dir /home/se
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/seas/bbernier/cadence/LVS/layout/netlist
       count
         345             nets
         14              terminals
         332             pmos
         332             nmos

    Net-list summary for /home/seas/bbernier/cadence/LVS/schematic/netlist
       count
         345             nets
         14              terminals
         332             pmos
         332             nmos


    Terminal correspondence points
    N333        N48         CLK
    N332        N50         CompIn
    N341        N11         ENC0
    N340        N23         ENC1
    N339        N12         ENC2
    N338        N10         EncOR
    N336        N21         IN0
    N335        N13         IN1
    N334        N20         IN2
    N344        N47         OUT0
    N343        N22         OUT1
    N342        N49         OUT2
    N331        N1          gnd!
    N337        N0          vdd!

Devices in the netlist but not in the rules:
        pcapacitor
Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.

                              layout  schematic
                                 instances
```

Elevator Module LVS Report

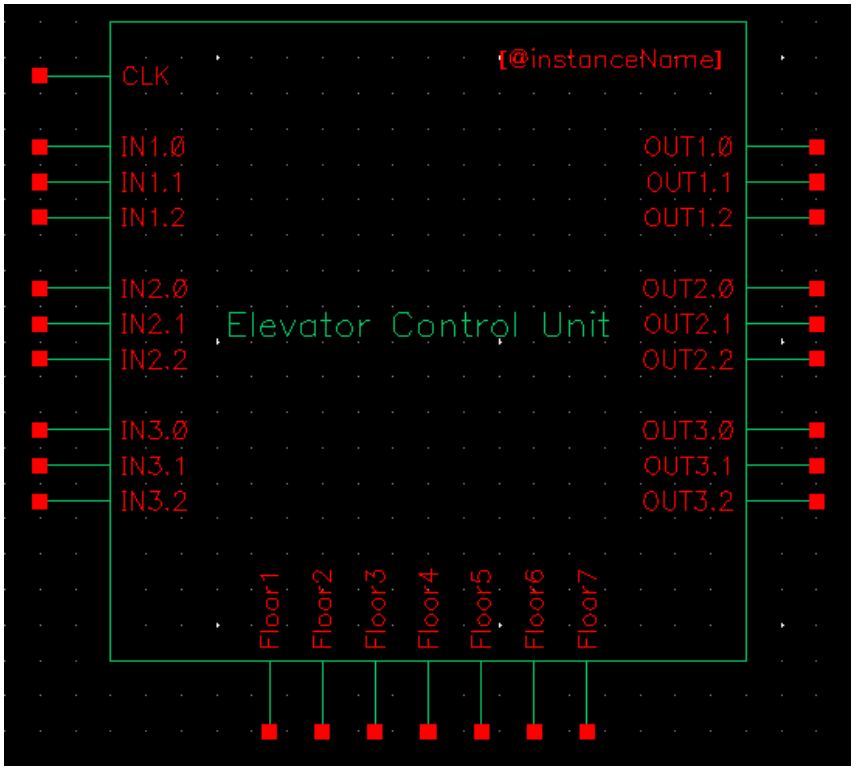
Elevator Control Unit

The elevator control unit is the final design of the system.

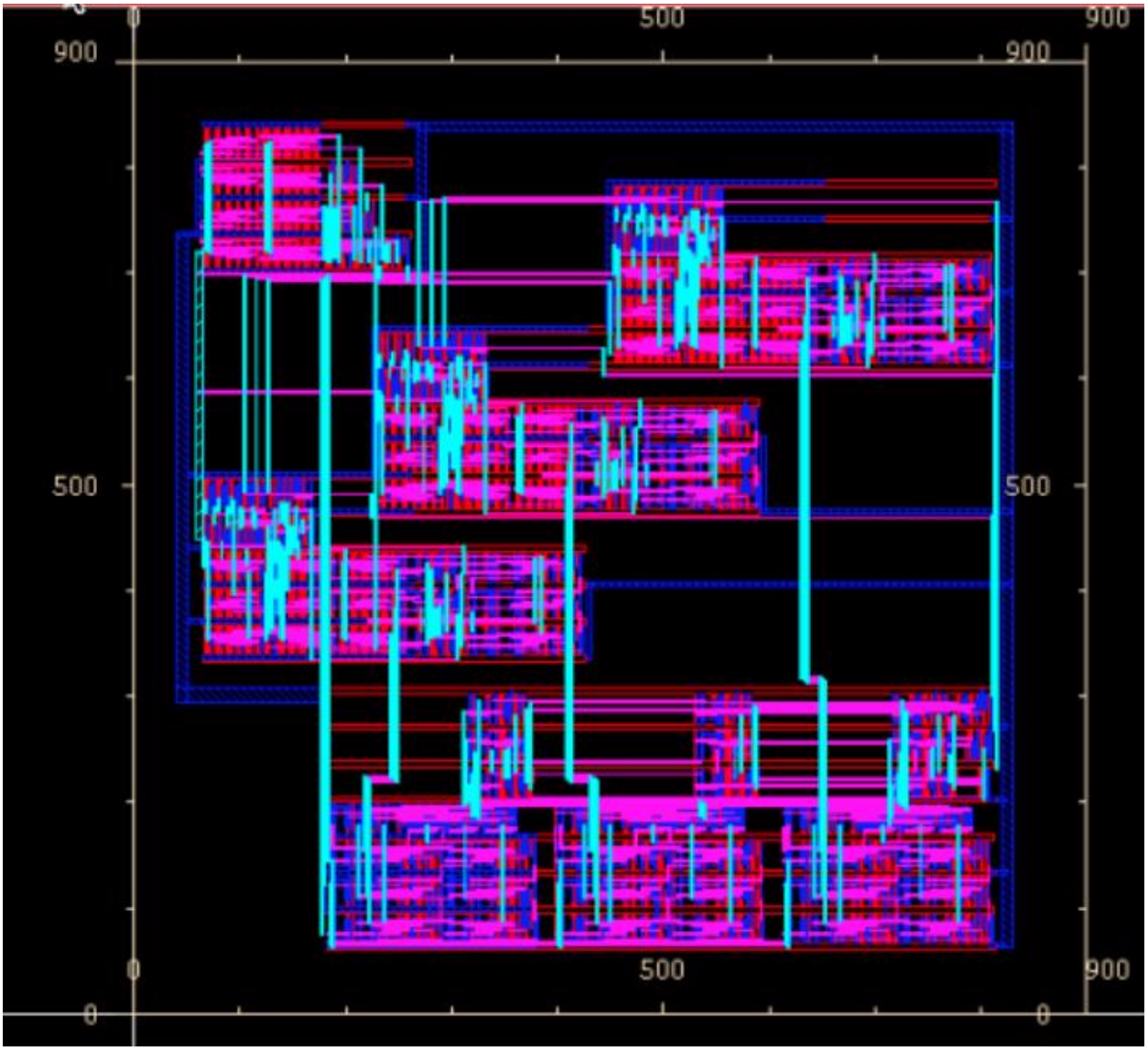
Inputs	Outputs
CLK	OUT1.0
IN1.0	OUT1.1
IN1.1	OUT1.2
IN1.2	OUT2.0
IN2.0	OUT2.1
IN2.1	OUT2.2
IN2.2	OUT3.0
IN3.0	OUT3.1
IN3.1	OUT3.2
IN3.2	
Floor1	
Floor2	
Floor3	
Floor4	
Floor5	
Floor6	
Floor7	



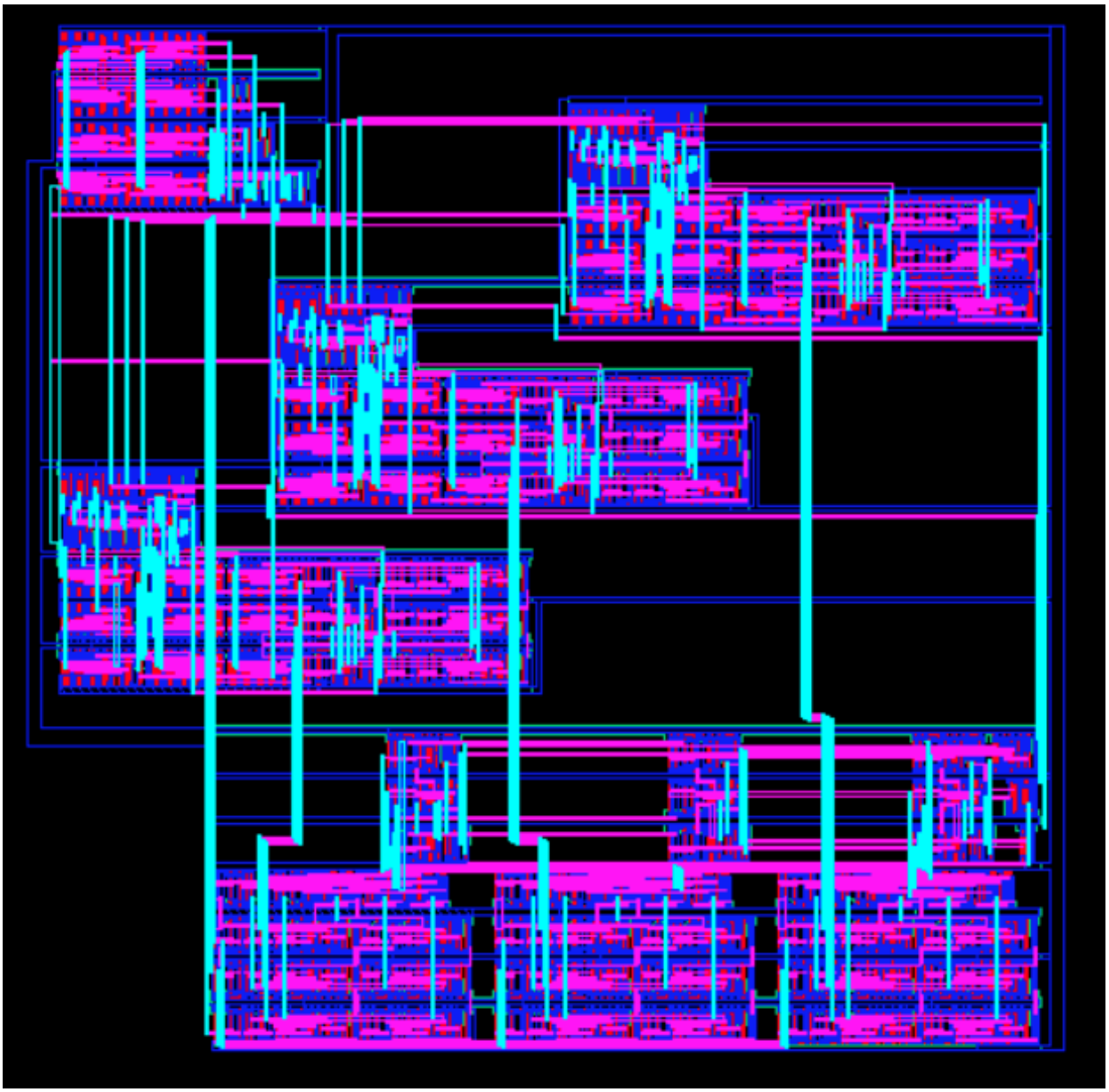
Elevator Control Unit Schematic



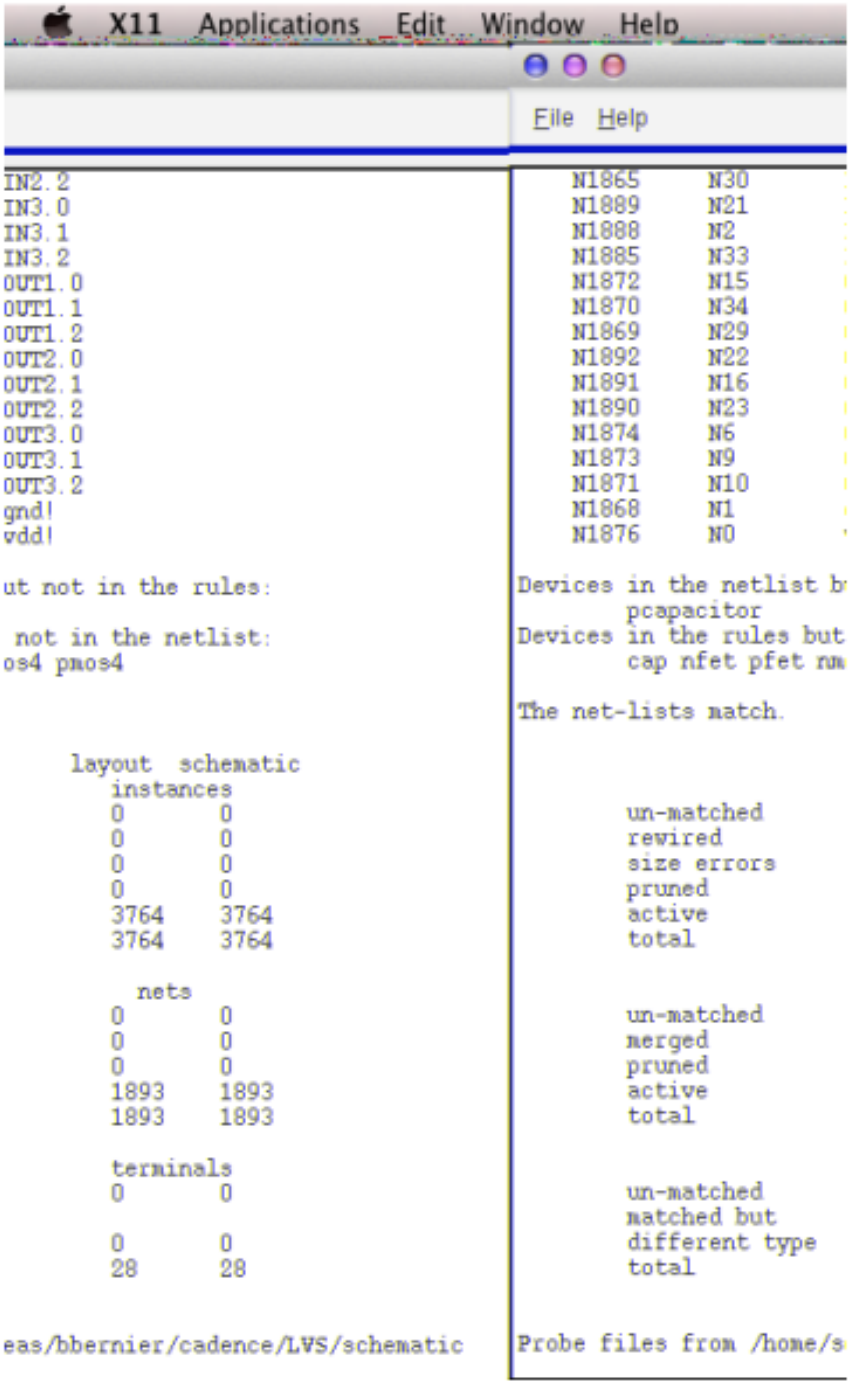
Elevator Control Unit Symbol



Elevator Control Unit Layout (Area ≈ 780x760µm)

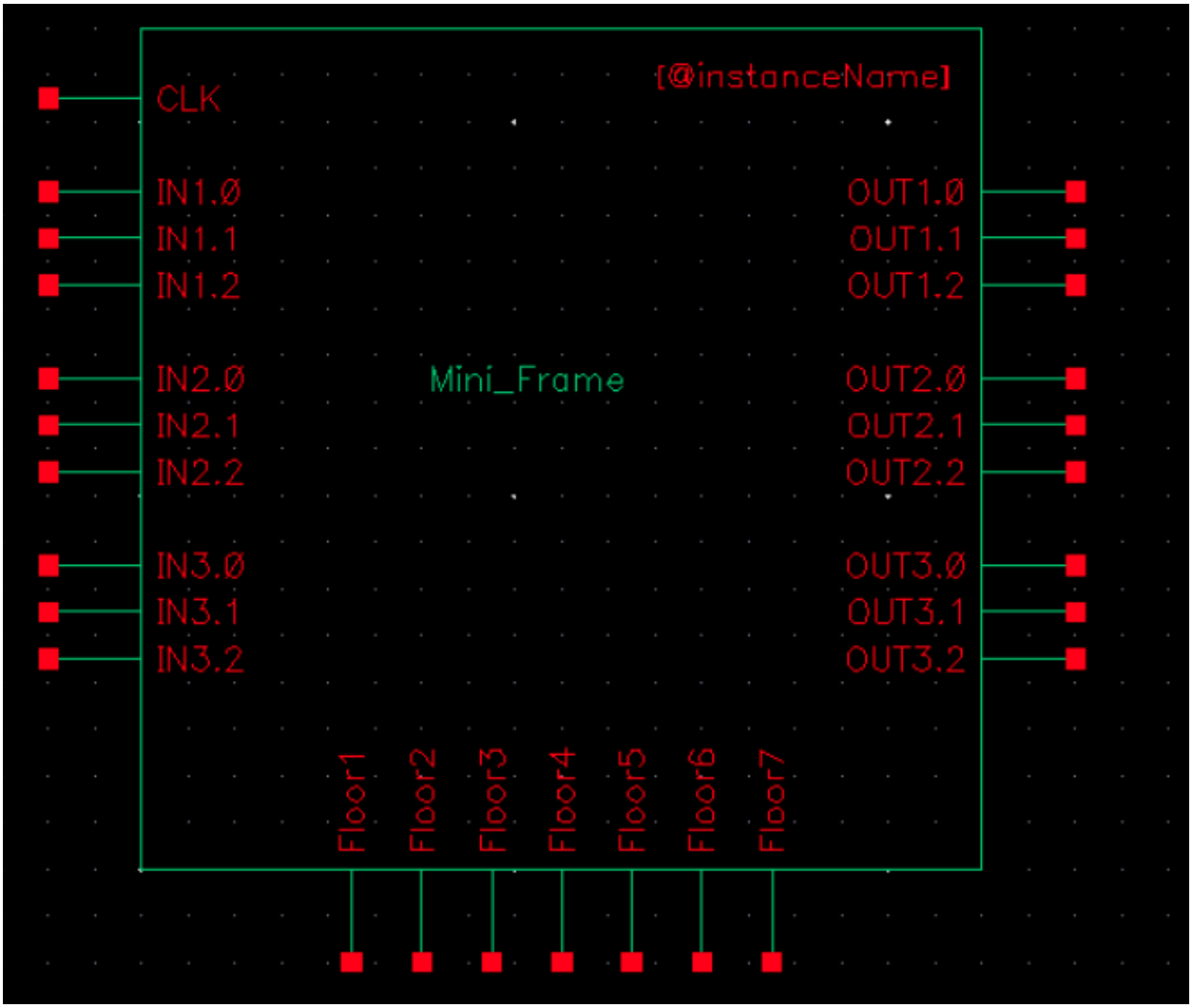


Elevator Control Unit Extracted View

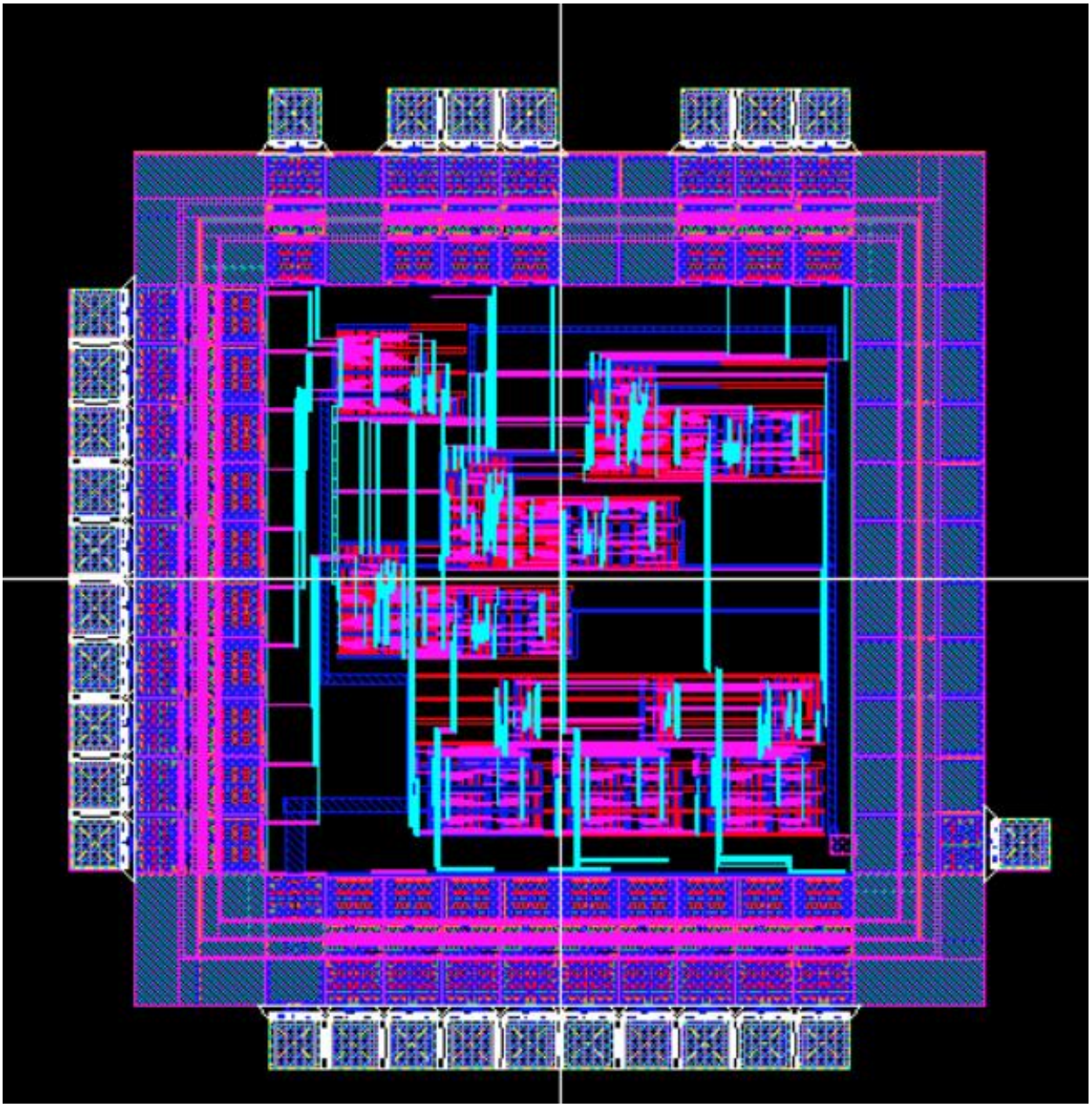


Elevator Control Unit LVS Report

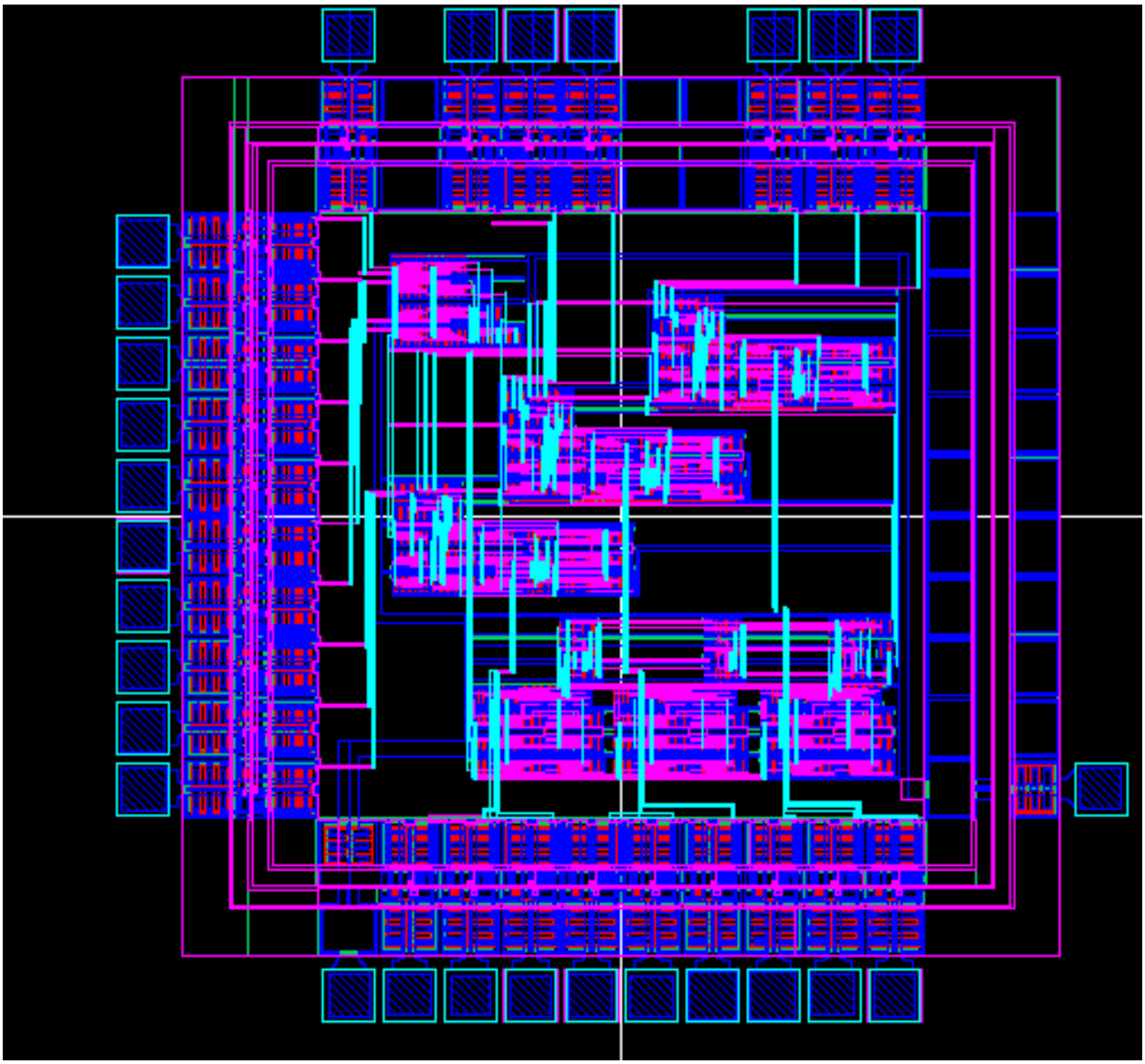
Elevator Control Unit Pad



Elevator Control Unit Pad Symbol



Elevator Control Unit Pad Layout



Elevator Control Unit Extracted View

4. Testing Procedures

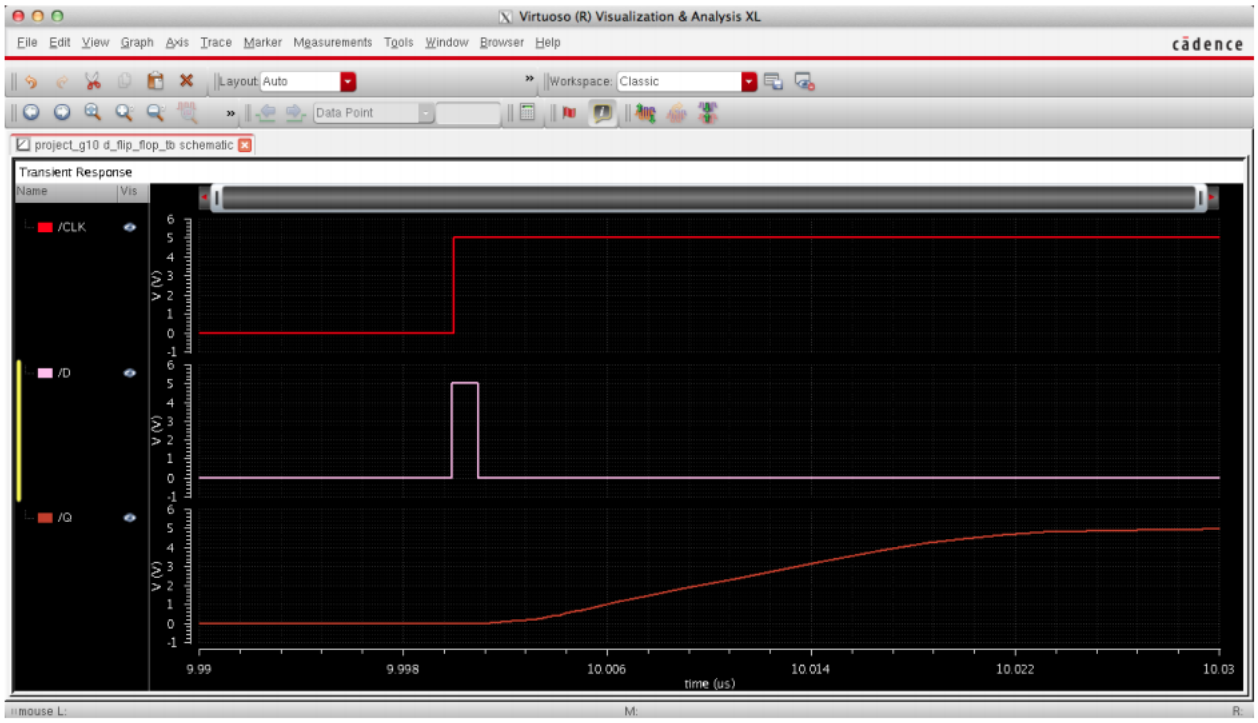
If this chip were to be fabricated, the testing procedure would be very similar to the simulations performed within Virtuoso. Inputs would need to be set up in a similar fashion and timed to recreate the simulation. There would need to be buttons capable of outputting pulses that last for at least 2 clock cycles to ensure the control unit receives the input. The outputs could be hooked up to a microcontroller capable of reading the values and converting them to decimal digits to display what floor the elevator should be on. It could also do this at a delay, because the control unit itself will change the floor every clock cycle, which would be indistinguishable to the human eye.

Obviously, the chip could not be tested exhaustively because the number of states and possible inputs is immense. It would need to be assumed that any test situation thrown at it would be representative of real world use cases and a variety of them could be tested, just not all.

5. Overall System Results

Critical System Specifications

- Total Transistor Count: 3764
- Total Area: 780x760µm = 0.5928mm^2
- Maximum Operating Frequency: ~40MHz
- Minimum Input Pulse: 2*Clock Period
- Average Power Consumption:
  - Without PAD Frame: 7.088µW
  - With PAD frame: 139.3µW
- D Flip Flop Setup/Hold Time:
  - Setup Time: 62ps
  - Hold Time: 953ps

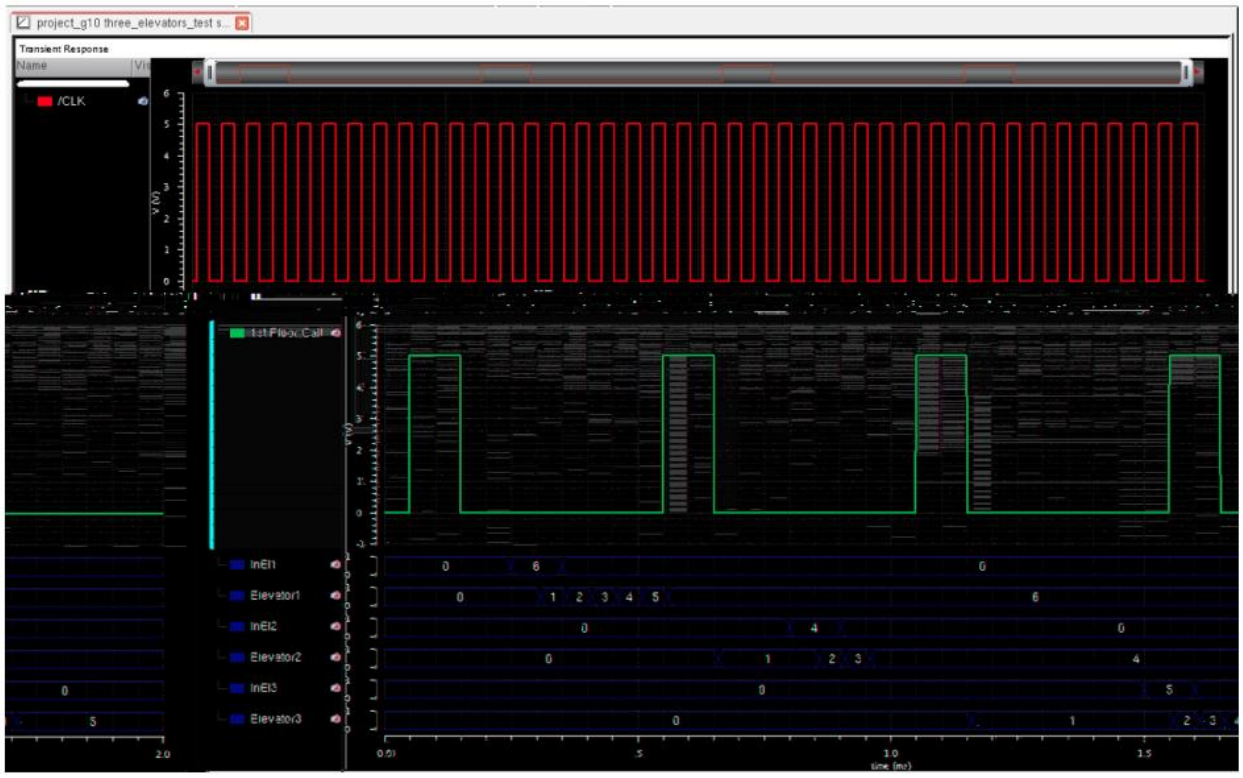


Test to Find Setup and Hold Time

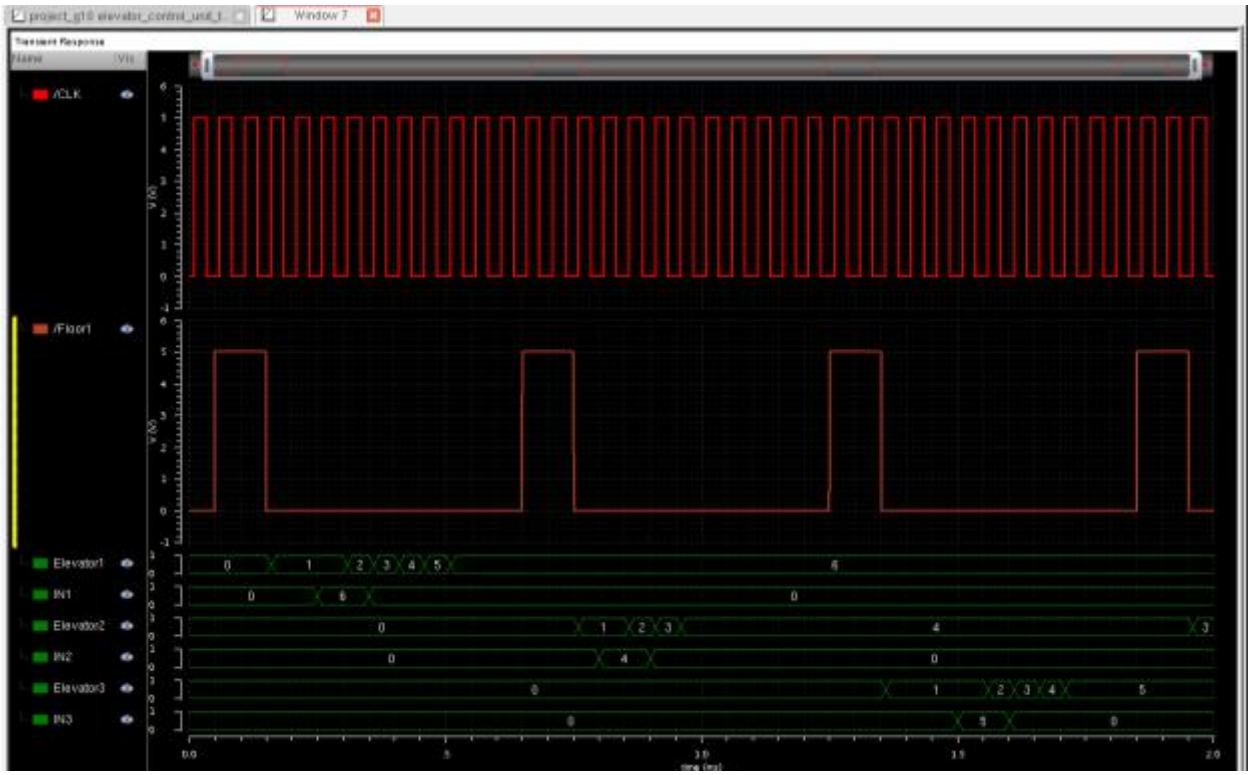
Two parametric simulations were run to find the set and hold time for the D Flip Flop. First, the start of the input pulse was varied to determine how long before the clock pulse it needed to go high to have the output follow as expected. Once the set time was determined, the length of the input pulse was varied to see how short it could be made and still get the correct output. This allowed us to compute both the set and hold times determined above.

Also important to note, the maximum operating frequency was found to be roughly 40MHz before the operation of the system began to seriously degrade. This, however, is not extremely important in the design because it simply needs to be clocked fast enough to get the input pulses from the elevator buttons. A single button press is usually longer than 10ms, so a much slower frequency would suffice. The elevators also are not moving between floors that fast, so this does not affect clock time either. All simulations were run off a 40KHz clock.

Final System Simulations

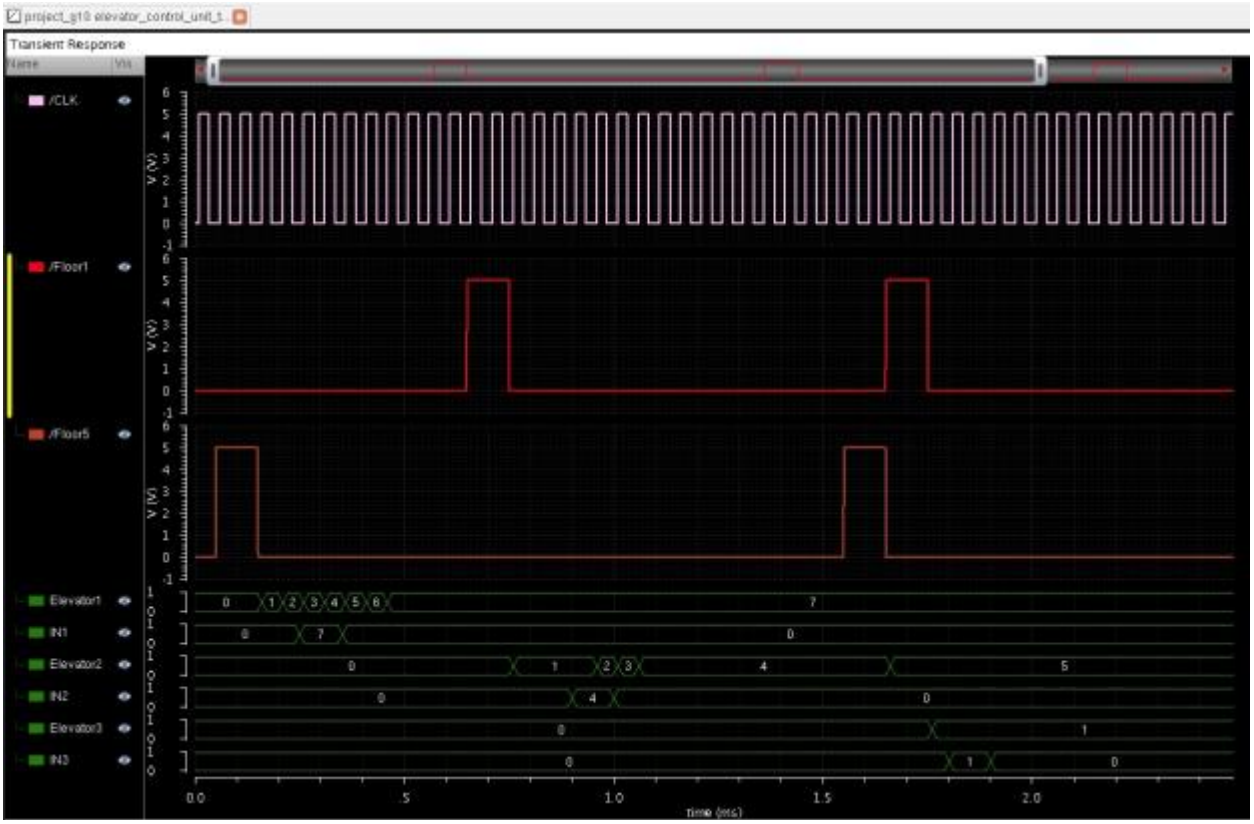


Simulation Results off Schematic View



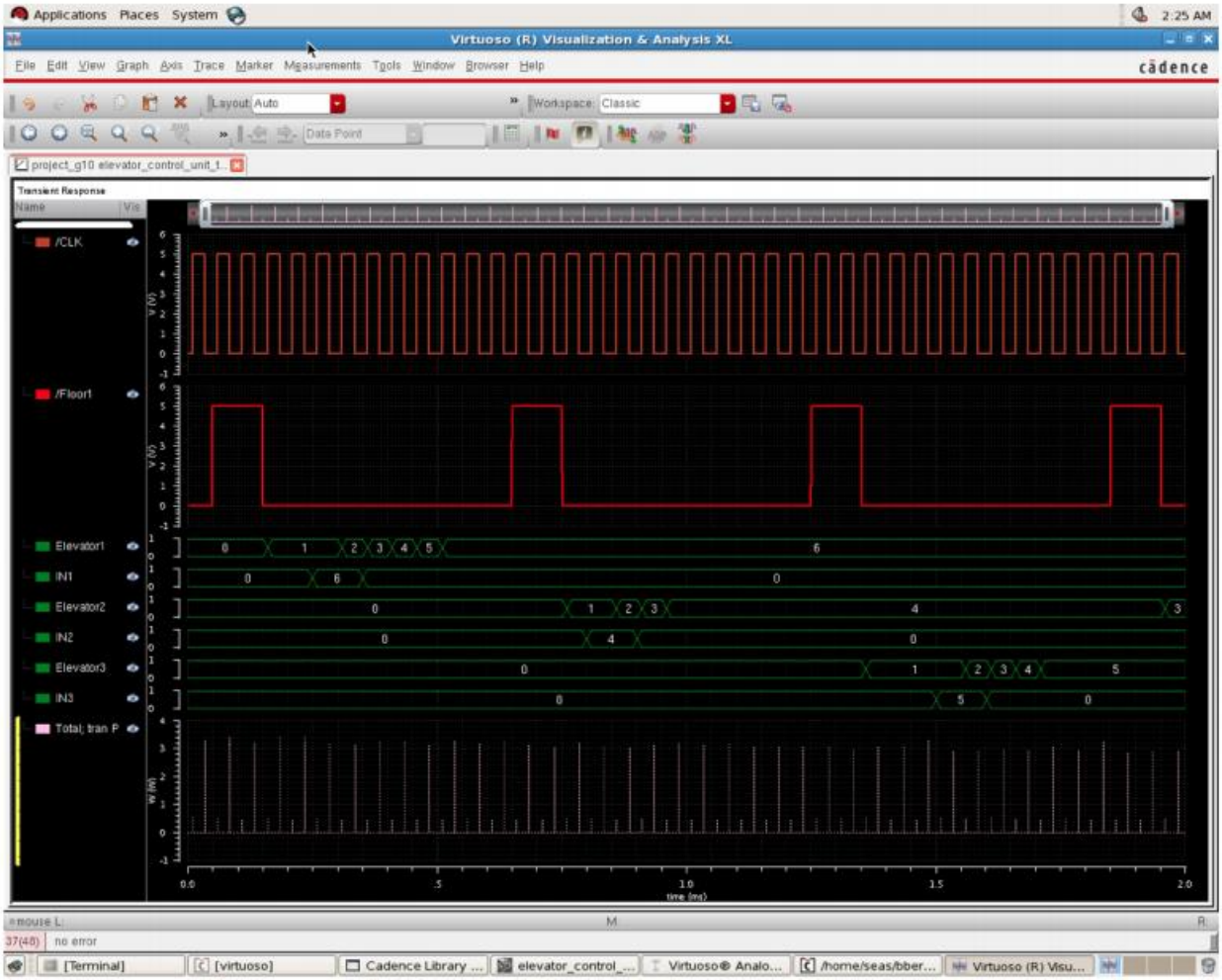
Overall Test 1 from PAD Frame

Overall Test 1 shows multiple calls for an elevator at the first floor. Each elevator starts uninitialized in the 0 state, and must then initialize its register to 1 at each call for an elevator. When all the elevators are on the same floor, it is built into the chip that Elevator 1 would take the task. Similarly, if 2 and 3 were at the same floor and closest to the call for an elevator, Elevator 2 would go instead of Elevator 3. Here, Elevator 1 responds to the first call, coming to floor 1 and holding until it receives an input from within the car to move to floor 6. It then proceeds to output 2, 3, 4, 5, and finally 6 steadily once it reaches that point. On the next call, Elevator 2 responds and is told to go to floor 4. Finally, Elevator 3 gets called to go to floor 5. At the very end, a final call at floor 1 requires the system to decide which elevator is closest, with them now being at floors 6, 4, and 5, respectively. It correctly chooses to send Elevator 2 because it is closest and that elevator begins moving down to floor 3 as can be seen at the far right of the test.

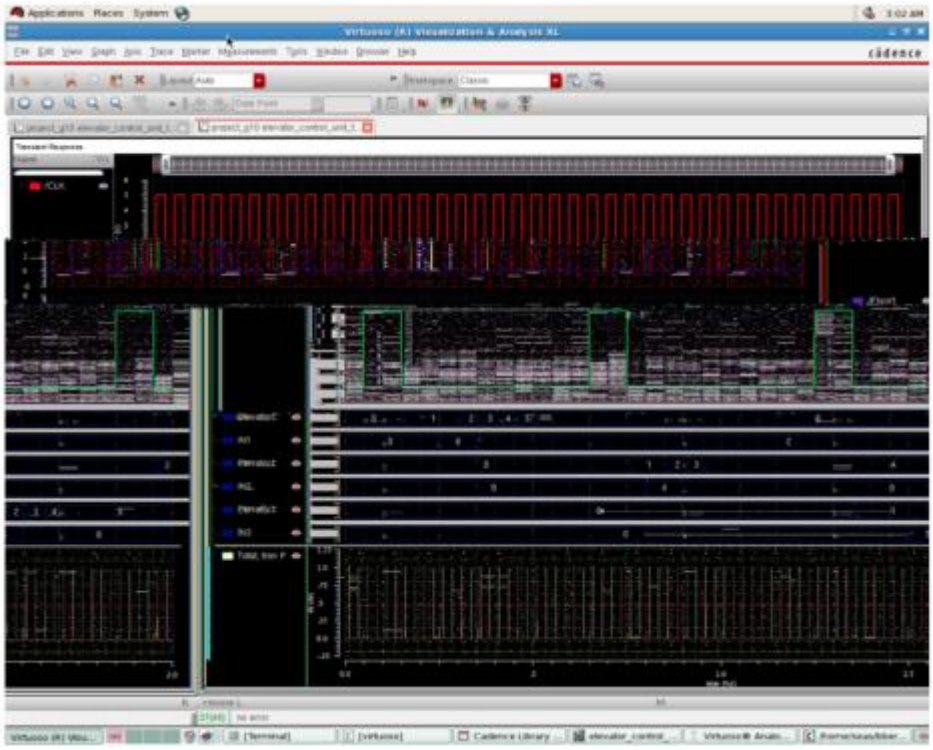


Overall Test 2 from PAD Frame

This test is similar to the previous one in that it has multiple calls, however, this time from multiple floors. The elevators respond accordingly, and it is important to note that Elevator 2 responds to both the 2nd and 3rd calls at the 1st and 5th floors, respectively. It completes a request to go to the 4th floor, then gets a call from the 5th floor because it is only 1 floor down. Elevator 3 responds to the final floor 1 call, however, floor 1 is pressed inside the elevator, so it remains there and waits on another input.



Extracted Simulation without the PAD Frame Showing Power Consumption



Extracted Simulation with PAD Frame Showing Power Consumption

In the simulations with power consumption, it was interesting that the average power found with the PAD frame was much higher than that simulated without it. This is most likely simply due to the case that the simulations performed without the PAD frame did not take into account the input capacitances as well as those on the outputs. The PAD frame simulation is obviously more realistic to what may be found in real life. Another interesting note is that without the PAD frame, the power peaks were actually close to momentary spikes at 3W, but with the PAD frame, the spikes were only 1W.

6. Conclusion

In conclusion, this project provided students with the opportunity for hands on design and layout of an ASIC. Students used the knowledge they built throughout the class lectures and lab sessions to complete a functional design and layout the entire design for fabrication. The project used industry standard tools necessary to success in the VLSI field. The specific elevator control unit project posed many challenges that were not initially taken into consideration. For one, the actual design of the schematic to fulfill the necessary logical requirements was much harder than anticipated. The control unit is dependent upon a lot of control logic, not simply chaining a lot of gates together to perform an operation. Each component had to be carefully thought out to produce the desired decision. When it came to the layout, this also proved challenging because it was a connection of many random gates as opposed to a uniform setup of many of the same blocks. Overall, the challenges were mainly overcome and the project was a success. In the future, it would be useful to add more realistic functionality to the elevator such as up and down buttons at each floor; however, under the time and area constraints for this project, the chip is pretty versatile.

7. References

[1] Chandrakasan, Anantha, Borivoje Nikolic, and Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.