SAN JOSE STATE UNIVERSITY
Electrical Engineering Department

Tale of an IC Design Engineer

IC DESIGN GROUP SAN JOSE STATE UNIVERSITY

HDB3 Encoder/Decoder circuit

Daniel R. Hicks
MSEE Candidate
Cadence Design Systems, Inc. Intern
Electrical Engineering, SJSU
One Washington Square
San Jose, CA 95192-0084

PL

Table of Contents

Chapter

1

Purpose:

The report that follows will document the design of an Integrated Circuit, highlighting the process and evolution of the project, from inception to fabrication. Design is be implemented in the AMI06 0.5-micron process and all architecture and simulation is done using Cadence Design Suite 4.46. Fabrication is done by the MOSIS service. The chip is tested for performance and functionality using an Agilent 1672G Logic Analyzer. The report emphasizes all aspects of the design process including changes, setbacks, scheduling, and management involvement.

This report is built around documentation of the design from Dan Hicks' project notebook, Dr. Parent's project journal, and Prashanta Lal's memoirs of the project. These will all be interwoven as the body of this paper order to shed light on as many aspects of the IC design process as possible. To this end, the journal entries of Dr. Parent and Prashanta Lal will be indicated whenever they are used in this document to distinguish them from the main flow of the report.

Dr. Parent's Journal

The purpose of this journal is document a simple full custom IC design project from the beginning to end in order that students new to the field of IC design might learn about the IC design process vicariously before they embark on a project for themselves. The idea that since student time is in short supply, students should have a design reference showing all the pitfalls of the design process so they do not waste time making mistakes common among novices. (Is it better to make mistakes common among experts?)

The major parts of this diary are:

- The original specification of the circuit
- The original timeline proposed
- A scanned appendix of the raw log book of the lead design engineer
- A journal documenting the major parts of the design process. Each section will be written from the viewpoint of the technical manager, lead engineer, and design engineer. The major steps documented are:
    - Writing the proposal
    - Setting up the design environment

    - Writing the specification
    - Logic design
    - Verilog logic verification
    - Hand calculations
    - Transient analysis
    - Layout
    - Final verification
    - Tape out
    - Testing finished product
- Proper documentation of the finalized circuits

The diary includes the cyclic nature of design and has not been sanitized from any warts. If we made a mistake or went down a blind alley we mention it-although we do not dwell on it.

Background:

Digital encoding defines the way data bits are represented on the physical communication line. HDB3 stands for 'High Density Bipolar 3' which is a variation of AMI 'Alternate Mark Inversion' that allows at most 3 consecutive 0 voltage codes to be transmitted. Standard HDB3 is one of the most complicated methods of encoding digital data, which is used in telephone services. The advantages of this complexity are:

1. Small bandwidth in order to allow many signals to be transmitted on a given communication channel.

2. Low DC level as signals with high levels are attenuated more and it is more useful to transmit a signal over large distances.

3. Many changes in the voltage in order to allow synchronization between the transmitter and the receiver.

4. Non-polarized signal so that passing it on a twisted pair will not affect the signal.

What is this project? :

This project utilizes a variation on HDB3 that is not bipolar but instead uses two transmission lines to achieve similar levels of bandwidth and synchronization while

adding robustness to the error-checking capability of HDB3 with the second line. This variation was chosen so that a purely digital circuit could be used to implement the HDB3 design, utilizing a single power supply.

What do we want the circuit to do? :

adfg

How fast can the design operate? :

This encoder/decoder circuit is designed to run at 250MHz. As it turns out, however, the pad frame only allows for speeds of about 33MHz. Since high-speed IO pads were not available for the 0.5 micron design, and there was not sufficient space in the pad to place a PLL, the circuit is required to run at a slower clock rate.

Project Timeline:

The initial timeframe for this project was about 4 months of design and then submission for fabrication.

Dr. Parent's Journal

Even though Dan started later in the summer I still felt that we would make our tape out date of September.

The timetable was the most victimized part of this whole project. There were many setbacks that resulted in the project time frame being delayed, most of which can now be accounted for and avoided in future projects.

Chapter

2

Design Flow:

Defining the project and making sure that the most important aspects have been considered, insures that the design starts out on good footing and will be less likely to run into problems. Having a clear design flow in mind will save time and avoid unnecessary revisions.

Project starting point:

1. Choosing the Architecture and Design specifications
    - Logic diagrams
    - Initial paper design
2. Schematic Design
    - Transistor sizing
3. Test Bench
4. Layout
    - Choosing Cell Height
    - Stick Diagram Layout
    - Design rules check (DRC)
5. LVS
6. Test Bench
    - Compare to design specifications and Schematic performance
7. Adjust layout

- Resize and modify transistors and layout.
- Repeat steps 2 – 8 until all design specifications are met.

8. Send to MOSIS.
9. Test physical device.
10. Evaluate and document the project
    - Recommendations for future changes or project direction

Chapter

3

Architecture and Design Specifications:

The Spec

First, the specifications for the project must be established.

Dr. Parent's Journal

Writing the specification was somewhat hard in that we had no boundary conditions. We were making up our own circuit to do what we wanted. I felt that we should try to push the limits of the technology so that we would encounter any problem that a student might encounter. We choose the telcom test bench as a medium complex circuit that maybe others or we could use to start networking research. I though it would be not too hard because the vhdl code for these items was not very complex. (I had worked on this type of design at Transwitch for an ATM test bench.) I felt that I was able to give a rough idea to Dan and he was able to turn it into a detailed specification. This took time, but I was still working on the design environment so it did not slow the project down. After this task, Dan and Prashanta would do the majority of work.

Here is the Specification that we came up with:

The HDB3 encoder/decoder will consist of two main blocks that all have an asynchronous reset and will operate on the rising edge of the clock signal. The Encoder splits a serial bit stream into two bit streams allowing for error detection in the transmitted signal at the Decoder. Also consecutive zeroes are broken into a special pattern that would allow for recovering encoded clock data if the Encoder were equipped with a phase-locked-loop or tuned circuit.

The HDB3 Encoder takes a one-bit wide data stream either from an on-chip pseudo-random-bit-stream-generator or from an off-chip signal 'ser_data'. It converts the input stream into a two-bit wide data stream (TX0 and TX1) in the following manner:

1. It checks for four consecutive '0's ("0000") and for the next four clock cycles, outputs a special pattern.
   TX0 1001
   TX1 0110
2. TX0 and TX1 alternate as to which is assigned a '1' from the input stream. Each signal line will not have consecutive '1's two or more clock cycles in a row (except for the special pattern mode).

Inside the Encoder, serial data comes into a shift register of four D-flip-flops. After four clock cycles, the data in the shift register is evaluated to determine if it represents a string of four zeroes (a special state). If it is four zeroes, then the Special Code Enabler enables the counter, which generates the Special Code using an XNOR. If the data in the shift register does not trigger the Special Code, the data in Q3 goes into the Line Selector, which places signal '1's onto the lines Tx_Data0 and Tx_data1 in an alternating manner.

The HDB3 Decoder decodes an HDB3 signal back into a serial bit stream while checking for coding and transmission errors:

1. Consecutive '1's on RX0 and RX1 (except for special pattern).
2. Four or more '0's in a row on RX0 or RX1.
3. If RX0 and RX1 are both '1' simultaneously

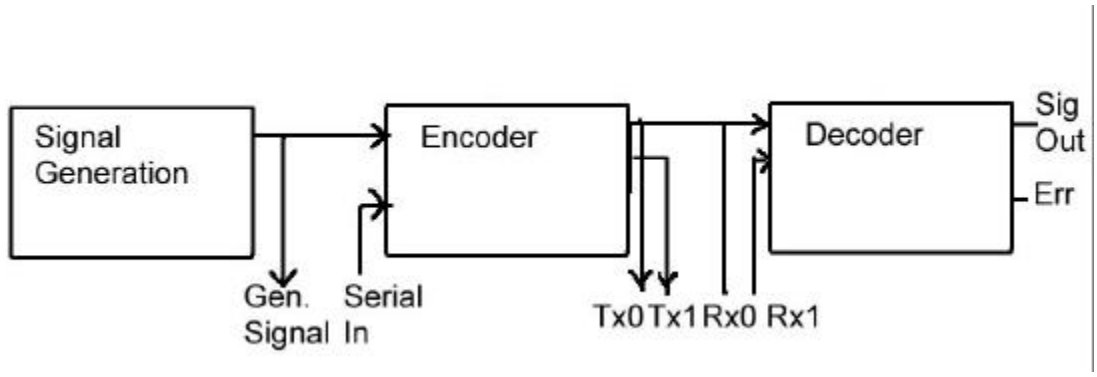
'Error' is set to 1 if any of these errors occur.

Logic design

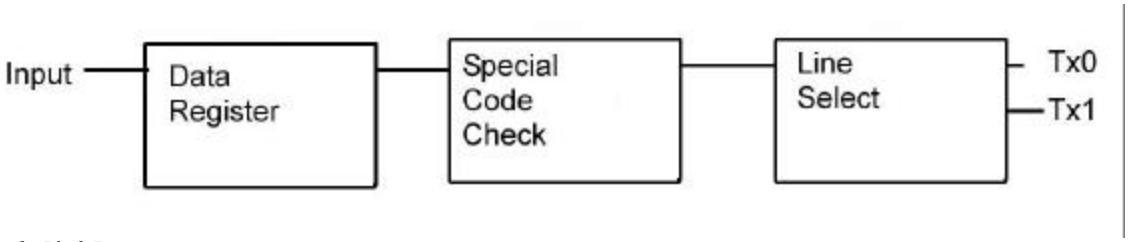
For most digital designs, breaking the logic into chunks that can be implemented with a Finite State Machine (FSM) is the most efficient way to design. This method guarantees the minimum logic functions for each block that minimizes power consumption and complexity. On the down side, FSM design is abstract and may be difficult to understand. It is also mechanistic (input the variables and turn the crank) so it is prone to designer error.

Another design option is the Functional Decomposition approach. Functional Decomposition utilizes modular design making it easier to understand. Since each block will be thought out individually, the designer has an intimate knowledge of how their circuit works. The downside of this method is that the circuit may be of less than optimal size. Also, assembling the full design from parts may allow errors into the design.

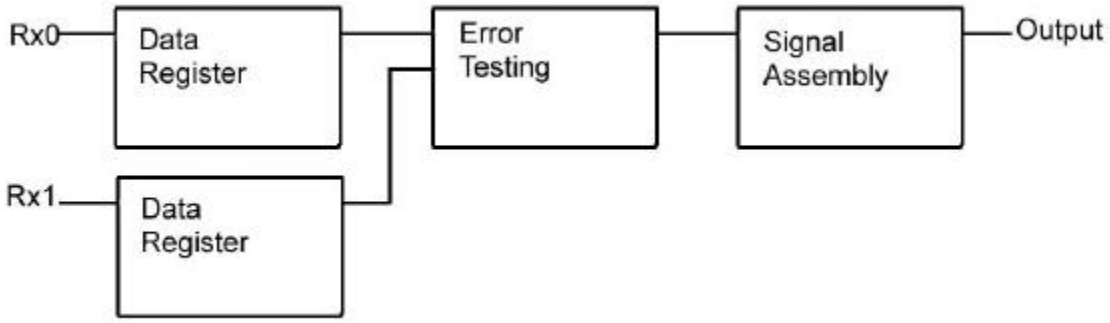
After playing around with the FSM method for a while, I came to the conclusion that I could do this design more successfully if it was broken into blocks I could easily understand and thus opted for the Functional Decomposition methodology. Along these lines here are the block diagrams for the HDB3 project:



HDB3 Block Diagram



Encoder Block Diagram



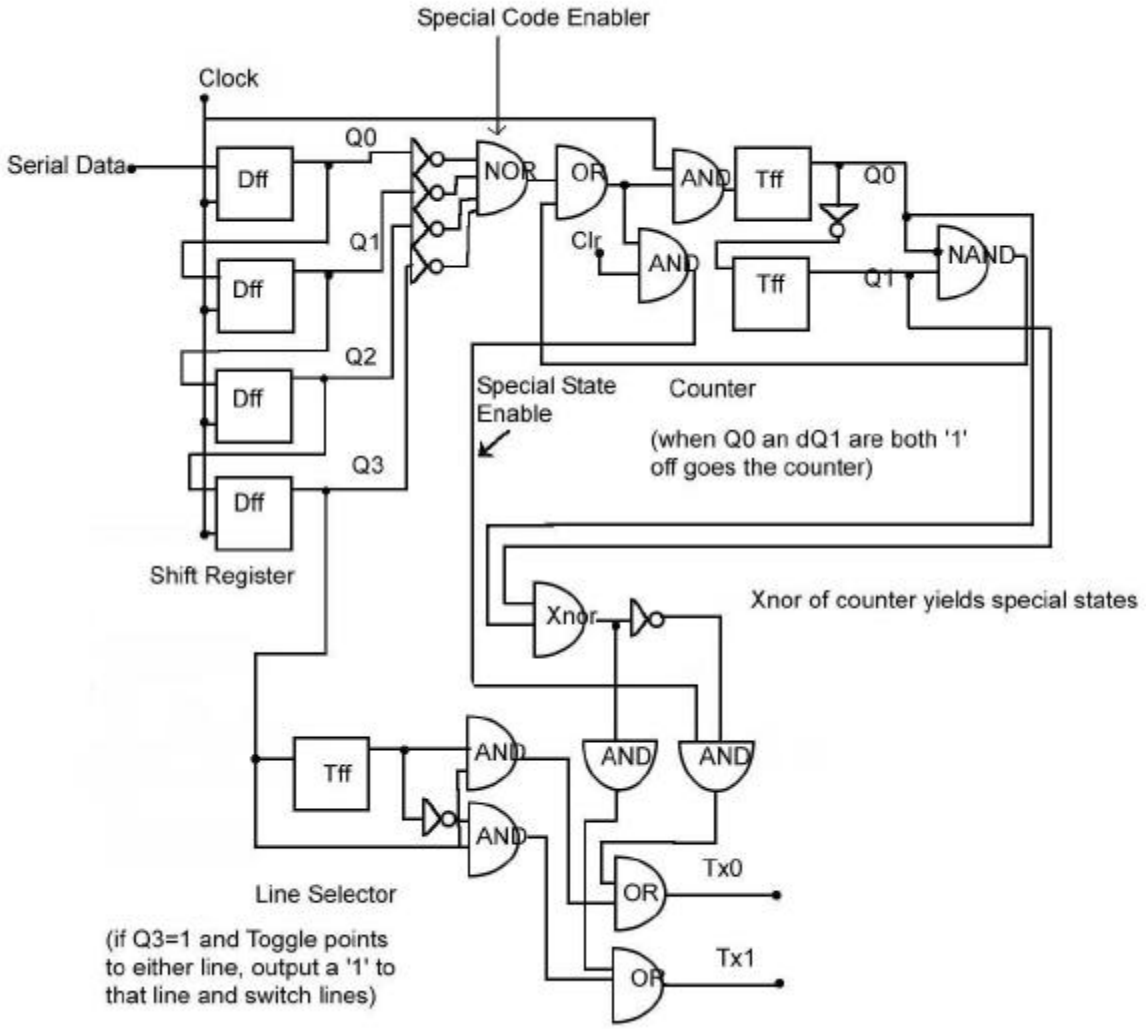
Decoder Block Diagram

From these block diagrams I began to fill in the details of each block, trying to re-use as much circuitry as possible in order to simplify the design.

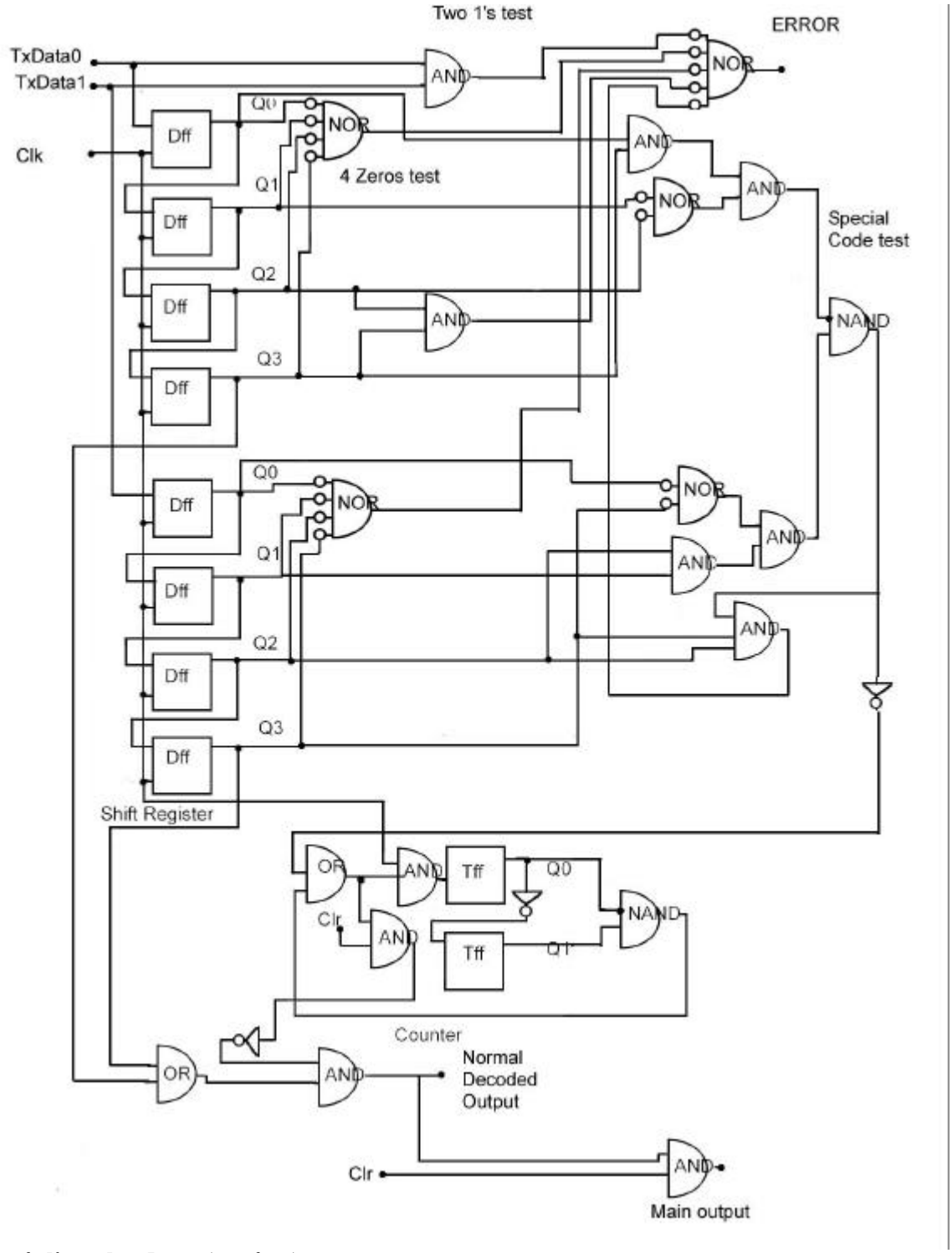
The data registers are simply D-flip flops. So once I create a DFF I can stamp it down four times thus creating a 4-bit shift register. Then the shift register can be stamped down once for the Encoder and twice for the Decoder blocks. For the Special Code Checker, there needs to be a counter in the Encoder to keep track of how many bits

in a row were ‘0’ in case of a special state. This counter could also be used in the Decoder to output the reconstructed data stream when a special code in encountered on the inputs. There needs to be some way to toggle between Tx0 and Tx1 since the input stream is output in an alternating manner out of the Encoder, so I will need a T-FF for that. This can be used again in the Decoder to reconstruct the data from the Rx0 and Rx1 inputs. Of course the cells for NAND, NOR, XNOR, etc. can also be designed once and then used wherever necessary.

Initial Paper Design



Encoder Logic Diagram



Decoder Planning Logic Diagram (paper design).

The above diagrams are cleaned-up versions of the ones in my notebook. They are the result of breaking each block of the project into smaller parts and coming up with simple equations about how they should work. This paper design represents the first attempt at planning the schematic diagrams. It has not been tested in any way, but

provides a good starting point, which attempts to realize the design and make it perform the correct functions.

Schematic Design

From the paper design, the list of logic cells is made. Starting with the humble Inverter, it is decided that a circuit with symmetric propagation delay will be used as the basis of all logic. Since the original circuit was specified to run at 500 MhZ, the sizing was done accordingly. 500 MhZ means we get 2 ns per clock pulse. The Inverter must be completely switched and settled within this period to avoid clocking errors. If we factor in 10% of safety margin that means the inverter has a 1.8 ns window. A symmetrically sized Inverter with Wp = 2.7u and Wn =1.5u yielded a rise time of 643ps. This may be an acceptable amount of time in a single Inverter, but if the longest signal path passes through several Inverters or even more likely, through several slower gates such as 3-input NANDs, each device switching this slow would cause a delay longer than one clock cycle.

The result of a signal running through a long path of gates would be data arriving out of sync with other parts of the circuit. By increasing the sizes of Wn and Wp, current can be drawn more quickly, thus speeding up the device.

Dr. Parent's Journal

I was not sure if they used my new hand calculation method or just used the parameter analyzer function in Affirma. The students who used it in class saved time, but it was a harder model to implement.

Through simulation an ideal width of Wp = 23.55 and Wn = 13.05 were found. This produced a rise time of 102ps and a fall time of 99ps respectively, which allows enough time for a clocked signal to propagate through the longest path of five gates. The drawback of this sizing is the area it will use up in layout and the power dissipation that will result from the extra current draw.

In retrospect, the consideration for circuit speed should have included the effects of the Pad Frame. The pads themselves are limited to a speed of about 33MHz due to the way they were designed. Unfortunately, there are no high-speed I/O pads for the 0.5micron "Minframe" that comes from the MOSIS website. If I still wanted my circuit to run at 500 MHz, then I would have needed to design a clocking circuit inside the chip to produce this clock rate. At 33MHz, the clock pulse allows 30ns per cycle, which means I could have used minimally sized logic devices and still have made the timing requirements. This would have resulted in a much smaller layout, and significantly lower power consumption.

All of the logic gates were designed based upon the Inverter equivalent. Virtuoso schematic editor was used to create the logic gates, and larger circuits such as D-Flip Flops were then made. But, before doing any further optimization, and certainly before doing ANY layout, the circuit components were tested to see if they were logically correct and functioning in the way that they are supposed to.

Logic Verification:

The Cadence design tool that we use has a test feature called NCVerilog. This feature can be used to quickly verify that the logical function of a block works the way it should. NCVerilog ignores timing issues such as gate delay and races, verifying only that the logic elements change at the clock edge.

Dr. Parent's Journal

Things proceeded smoothly until the beginning of the semester. Dan and Prashanta got a reputation as cadence tool experts and found that it was hard to do work in the lab. Unfortunately it was too late in the semester when I found out about this and there was no way to get them lab space. The only other glitch was the verilog simulator required a special test bench to run PRBS type circuits, which I had to solve. At this time I was busy trying to manage the unix system and 500 users. I also ran a short course to debug my tutorial and to try to get other professors involved in IC design.

Schematics of the component cells for the Encoder were designed. After Dr. Parent got the NCVerilog simulator working, some of these circuits were then tested, and the output in the waveform viewer clearly showed that I had missed a few things. This began the process of adjusting the schematic and re-testing it in NCVerilog until it functioned properly, (i.e. a Shift register that shifts, a Counter that counts, a Line selector that selects, etc.). Once the components were verified, they were tested together in the Encoder block. Again, adjustments had to be made to the schematic and the whole thing retested repeatedly in NCVerilog until it came out right.

The main advantage of first building the parts and then verifying the logic this way is the timesavings. Affirma Analog simulations of the circuits at this stage would have taken much longer and revealed the logical problems in a more difficult manner to interpret. Since the NCVerilog ignores timing issues, we did not have to look at any delays or mysterious voltage levels in the circuits. Either the output is a '0' or a '1', and it either happens at a clock edge or it doesn't. At this stage of the project, it is not important to know how well the circuit works, only that it works at all.

Engineering Change Order

While the logic verification process was going on, the first major error encountered caused the Decoder to improperly decode the transmitted data from the Encoder. Figuring out where the problem was localized took some time, since I had to go step-by-step through the design until I found the portion that was creating the error. I narrowed it down to be the Line-selector in the Encoder, since anything that was properly encoded was properly decoded. Therefore, the only conclusion was that the Encoder was doing something wrong. This is where it got tricky, since the Line-selector was operating according to the specifications of the project, how could it not be functioning properly?

It soon became evident that the wording of the specification resulted in a functionality problem that did not allow the circuit to work as intended. The specification causing trouble dealt with the way the Decoder detected errors in the transmission. The original spec read:

"***More than four '0's in a row on Rx_Data0 or Rx_Data1 is an error***".

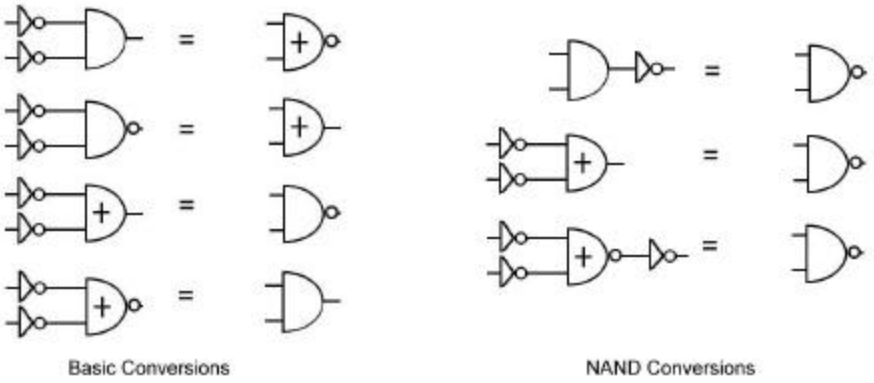
Taken literally, this means that four '0's is not an error, but five '0's is. Since we really wanted four '0's to be an error, I brought this to the attention of the project manager, Dr. Parent. The result of this was a change to our original specification. The new spec reads:

"***Four or more '0's in a row on Rx_Data0 and Rx_Data1 is an error***"

This subtle shift of wording ends up drastically changing the way the Line-selector is designed. Following this 'Change Order' the Line-selector was redesigned to accommodate the new spec and the resulting circuit worked properly.

Logic Conversion:

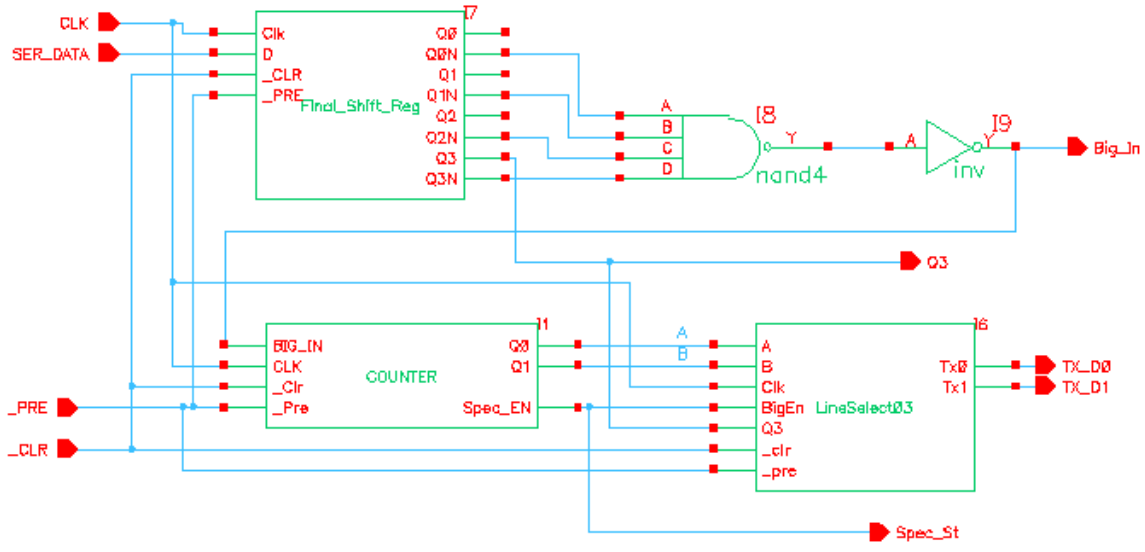
Notice that this circuit utilizes ANDs, NANDs, ORs, NORs, INVs, and an XNOR. Logically, this circuit is easy to follow from one logic function to the next to see what the signals are doing. While this is a great method for design, it utilizes logic functions that require more area and/or are slower than they could be. The fastest logic function to implement in CMOS is the NAND gate. So, to optimize the design, the next step would be to convert as many of the other logic functions into NANDs as possible.



Logic conversion diagram

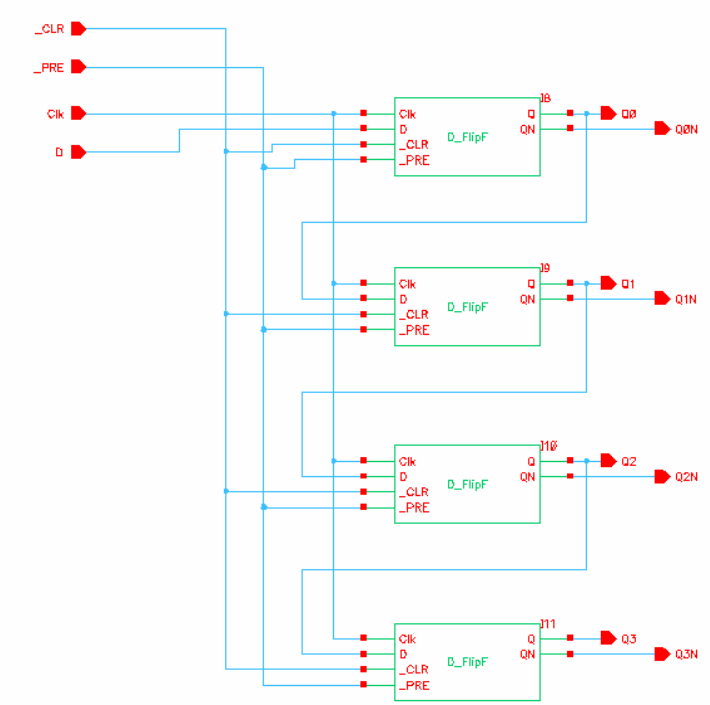
By careful use of inverters, most of your logic can be transformed into NANDs. I use the term most, because sometimes the conversion will result in the use of more gates instead of fewer gates (remember, the goal is always to use less logic rather than more so that propagation delay and power dissipation can me minimized, not to mention making the layout easier and smaller). This conversion process is best done on paper so that you can easily make changes if a conversion does not make the circuitry better. Once we were satisfied that the circuit had been improved by converting the gates into NANDS, it was time to run the whole thing through NCVerilog again to make sure we had not made mistakes. Sure enough there were several. And the process of altering the design and testing continued, until the circuit again functioned as planned.

CMOS logic circuit schematic

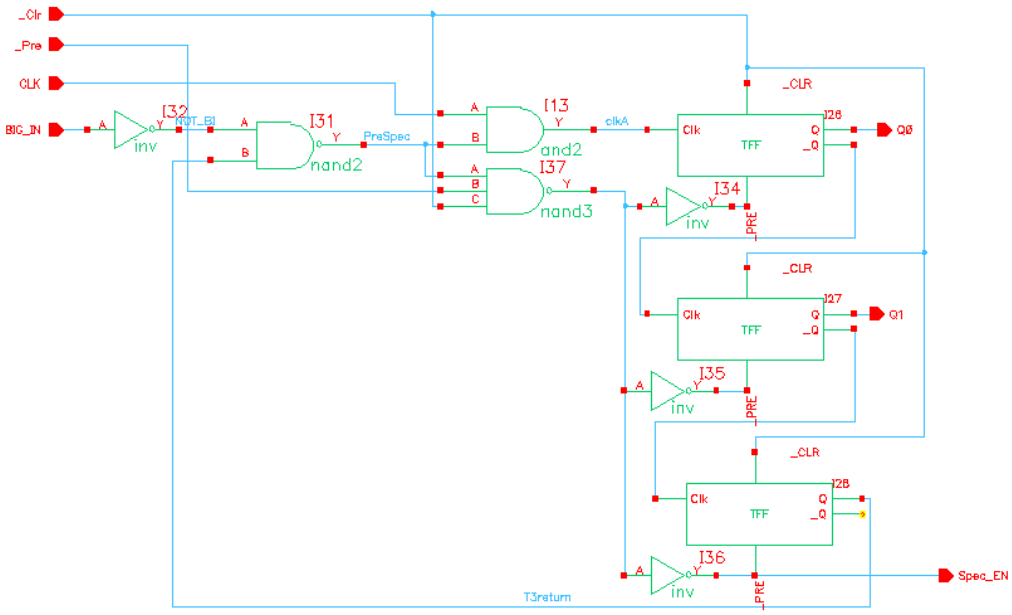


Encoder Schematic

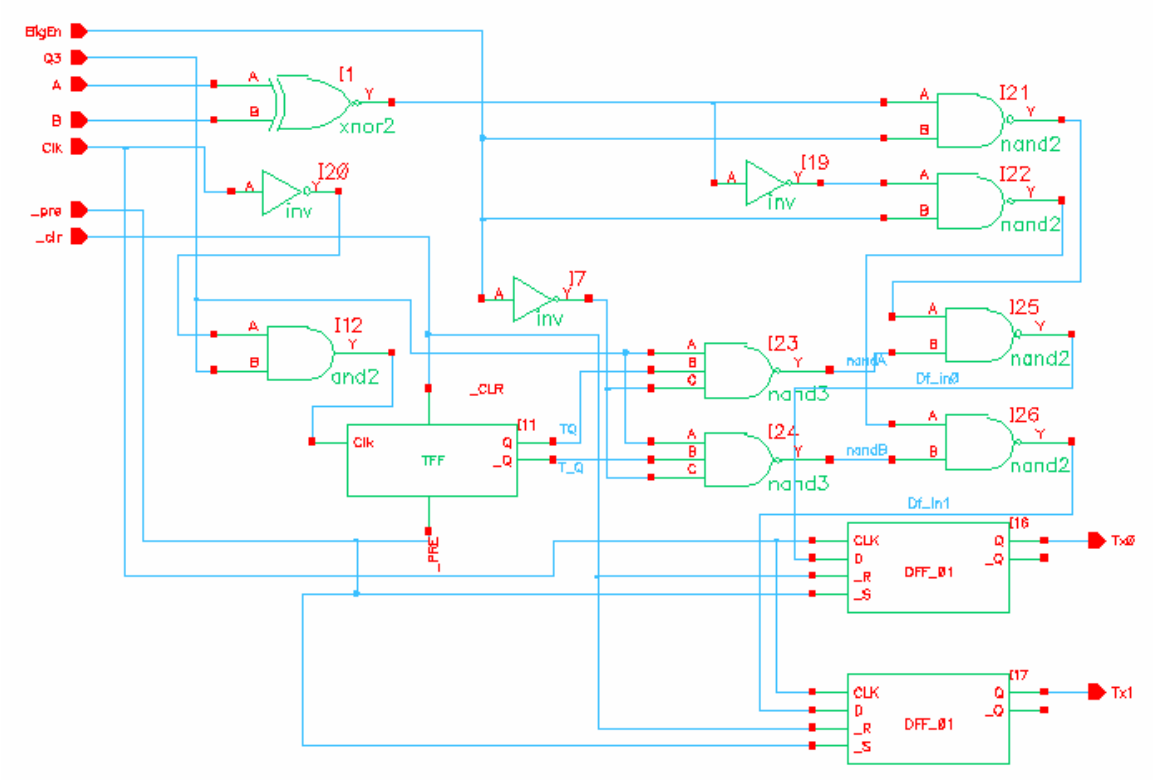
In the final schematics for the Encoder, one can see the differences between the original paper circuit and the resulting circuit that was modified first in terms of logic verification and second in terms of logic conversion to NANDs. The following schematics illustrate the contents of the Encoder sub-cells:



Final_Shift_Reg Schematic



Counter Schematic



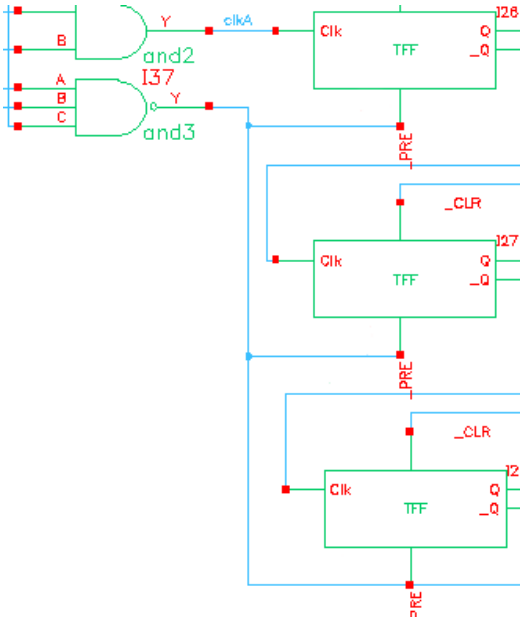
Line_Select03 Schematic

Test Bench

Affirma Analog test benches were created to test the schematics after they had been verified in NC Verilog and were functioning logically. The Analog simulations will show the effects of capacitance related to transistor sizing and therefore clock skew, signal delays and setup-and-hold violations will become evident. These simulations approximate how the circuit will run once it is fabricated. This means that sizing of logic gates and adding or shifting delay loads within the circuit may be necessary to make critical paths meet timing. The analog simulations take much longer to run as the circuit becomes larger, so it becomes critical not to make little mistakes like mistyping a transistor width or a capacitor value. This stage can be extremely frustrating since a schematic that ran perfectly in NC Verilog can sometimes fail to work at all in an analog test bench.

There is no point in going on to layout until the schematics are 100% correct. If you end up having to make a change to the schematics once you are doing layout, you may have wasted many days of simulation. This is what happened with the design of the Counter circuit. An error caused by switching the _PRE signal did not show up until after layout had been completed because the schematic was not sufficiently tested. These things happen as the complexity of the circuit goes up.

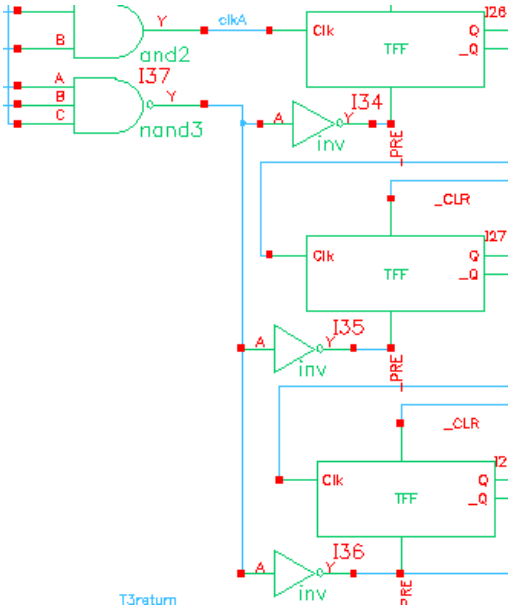
The Counter Problem:



Area of COUNTER having problem

Control signal for the _PRE line was not triggering the counter to start when it switched from low to high…. It took one clock period and then the Counter began. Upon inspection of the output simulation using the extracted view, I could see a long rise time on the _PRE line. I figured this was caused by a capacitive load of 3 TFFs connected to the AND3 which could not drive them fast enough.

The Fix:



Area of COUNTER after fix

Replacing an AND3 with an NAND3 reduced the capacitive load. The inversion was completed by adding 3 inverters at the inputs to the _PRE on the three TFFs. Capacitive load was reduced and the individual _PRE signals rose quickly enough to allow proper operation of the counter assembly.

Finding this problem and then identifying its cause took two days. It then took another day to repair and re-test the schematic. It took 45 minutes to rearrange the layout to match the schematic and another couple of hours to re-test the circuit and finally get a proper LVS and good waveforms. If I had tested the schematic of the Counter while embedded in the Encoder earlier, then I could have found this problem and saved myself the extra time to repair everything.

The test benches that were created to test the schematics will be used again to test the extracted version of the layout.

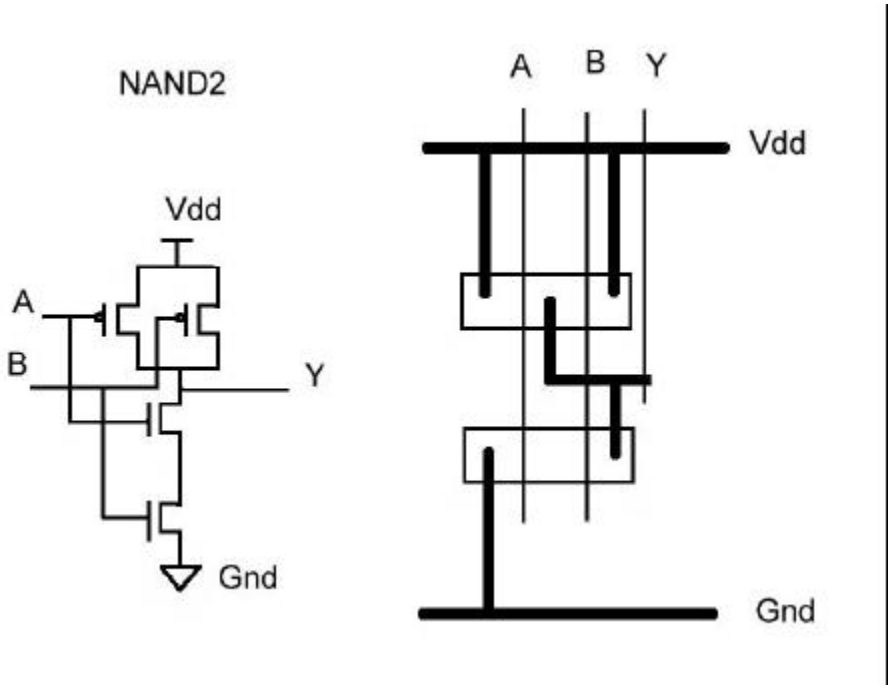
Chapter

4

Floor Plan and Layout:

CMOS logic circuit layout:

The first step of layout is floor planning.



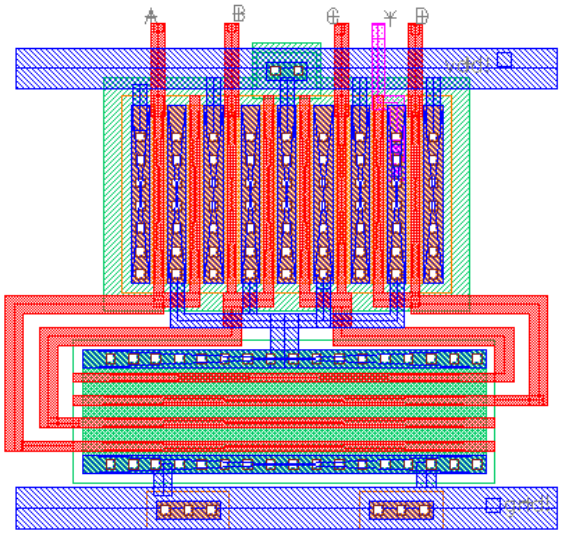
Stick Diagram

Going from a schematic to a stick diagram, the placement for the inputs and outputs was decided. By having access to both inputs and outputs on the top of the cell, channels of signal routing could be placed between rows in the transistor layout.

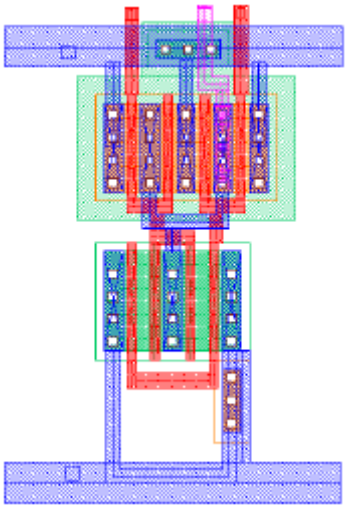
Cell Height

The layout was designed with a standard cell basis. To do this, a cell height that is consistent between cells needs to be chosen. The first step was making an inverter as the

basic layout of a cell. In looking at the schematic the largest logic element is a 4-input NAND. The goal of good layout is to produce a cell that conserves space while attempting an aspect ratio (length divided by height) of about 1. The NAND4 being the largest element needed the largest layout. By fingering the transistors, the height of a cell can be reduced and made wider at the same time. Combinations of 'multiple' and 'finger' were used to make the transistors fit in the desired cell height. Once this had been done, this cell height was used as the standard for all the rest of the logic elements including the Inverter, which was resized to accommodate the larger spacing between the VDD and GND rails.



NAND4 Layout



NAND2 Layout

The well and substrate taps were placed under the Vdd and Gnd rails to conserve space in some instances. Having sufficient taps reduces noise and ensures that the wells and substrate remain at their appropriate voltage levels. Long runs of polysilicon are avoided whenever possible since poly is highly resistive, and long poly lines increase susceptibility to circuit damage during manufacture due to 'radio rules'. In the case of the NAND4, long lines of poly were increased in width to increase conductivity.

DRC

Layout continued with the design and testing of all the basic logic blocks. Design Rules Check (DRC) insures that the layout conforms to foundry rules for placement of cells and the many layers, contacts, and interconnects within the layout design. In this way, a design is modified to work optimally with a given fabrication technology.

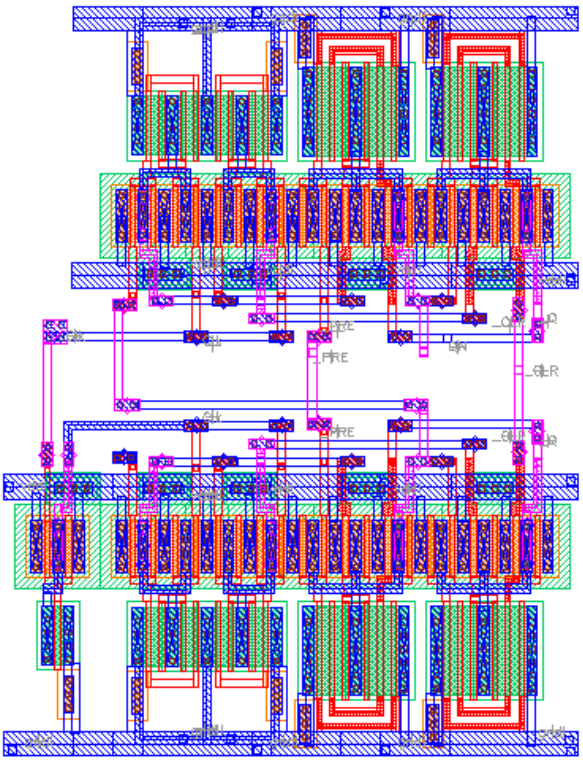
LVS

By running Layout vs. Schematic (LVS), each block was compared to its schematic counterpart, and any differences in functionality were indicated. These differences were then corrected so that a layout would pass LVS.

Verification

Transient tests run in the Affirma Analog environment, allowed direct comparison between performance of the schematic and extracted layout designs. Even though the sizes of the transistors may be the same in the layout and schematic, the performance will vary depending on capacitances that are extracted from the layout. Adjustments to cell layout and or transistor sizing were made to get the performance from each logic element that was desired. Once confident that all of these blocks worked properly and were of correct cell design, larger blocks were constructed using these basic blocks.

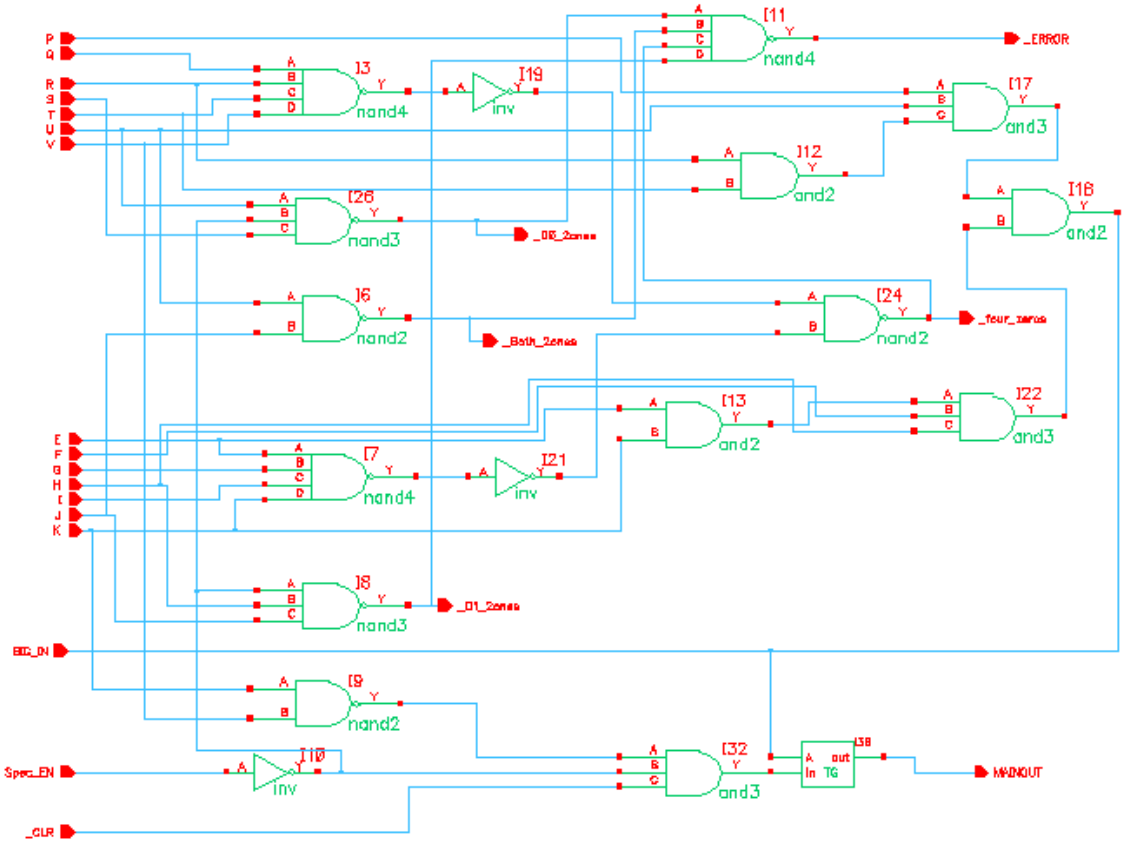
A D-flip-flop was made from NAND3s, NAND2s, and an Inverter. By flipping and rotating the layout cells, a larger cell can be made which attempts to achieve and aspect ratio of 1. Notice the space between the transistors that has been used for signal routing, as well as the Vdd and Gnd lines that form when cells are simply stamped down next to each other. When contacting between metal signal lines or from metal to poly, it is best to use a double contact instead of a single one, in order to increase reliability and reduce interconnect resistances.



D-flip-flop Layout

Sticky Layout Issues

One of the most challenging cells to layout in this design was the Decoder_parts block.

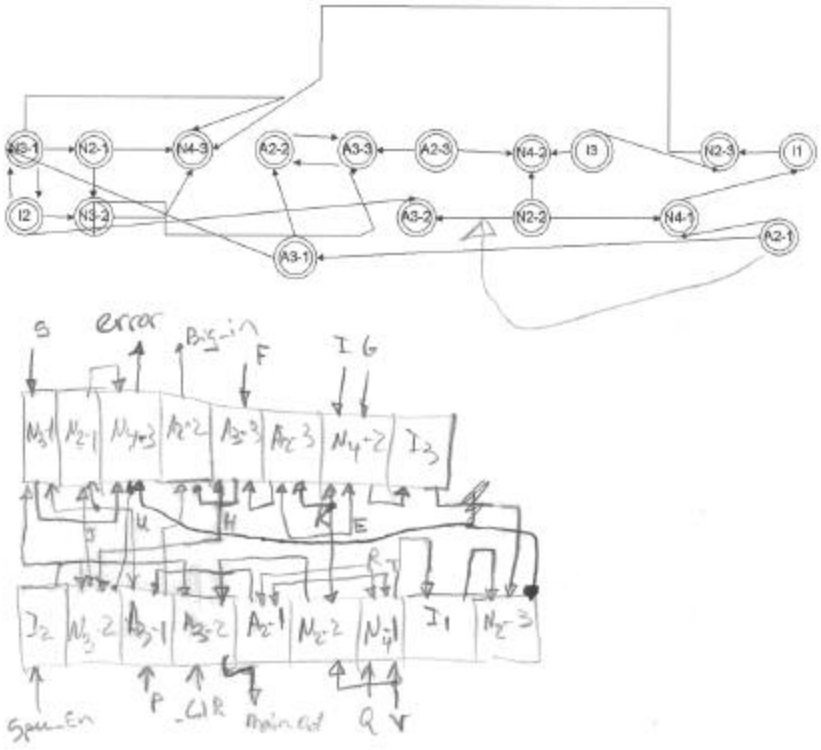


Decoder_parts Schematic.

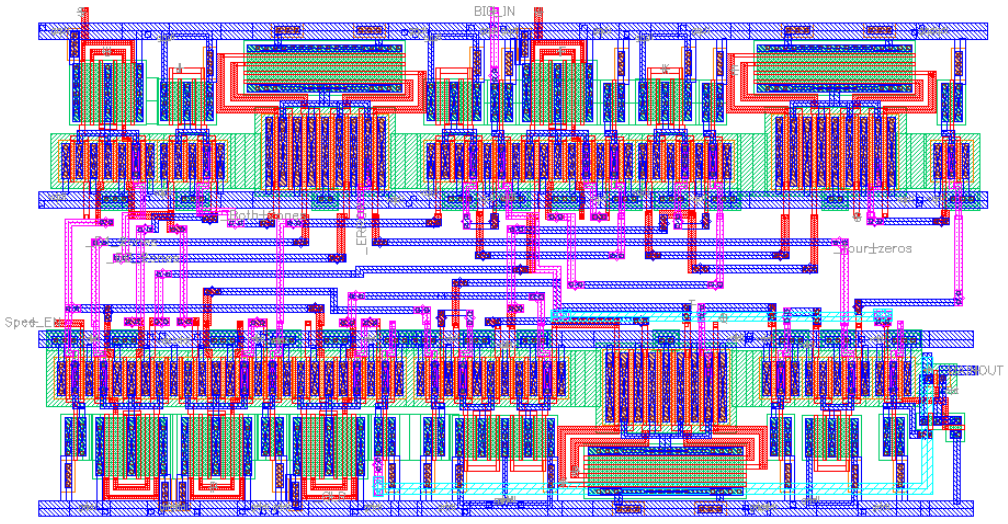
This circuit has multiple signal lines running all over the place. When trying to layout the logic blocks in a way that minimizes crossovers and wire-lengths, the standard stick-diagram method is quickly overwhelmed by input fan-in and output fan-out. Hand optimization of the 17 logic blocks and their 39 input nets and 7 output nets is a problem. Optimization can be done for inputs, for outputs, for longest path, for least crossovers, and the list goes on…..! So what to do?

Graph theory to the rescue. In theory, any complex system can be modeled algorithmically by defining nodes. Those nodes can then be ordered which will result in some optimal solution. Unfortunately, information overload occurred before I could figure out a way to make this mess into an equation. With some outside help, it came to my attention that a program called VISIO has a graphical system for optimizing nodes. After figuring out how to use VISIO, this schematic was represented by a mess of labeled circles and arrows indicating the direction of a signal connection. With several false starts, the program optimized the jumble according to a least-crossings scheme. It looked

much better than when I had tried it by hand, but it was not perfect since the jumble occupied a space that did not directly translate into the layout that I was hoping for. Adjustments were then made by hand to put the 17 circles into two parallel rows with all of the "off-cell" inputs on the outside of the layout and most of the "in-cell" inputs and outputs on the inside of the layout.



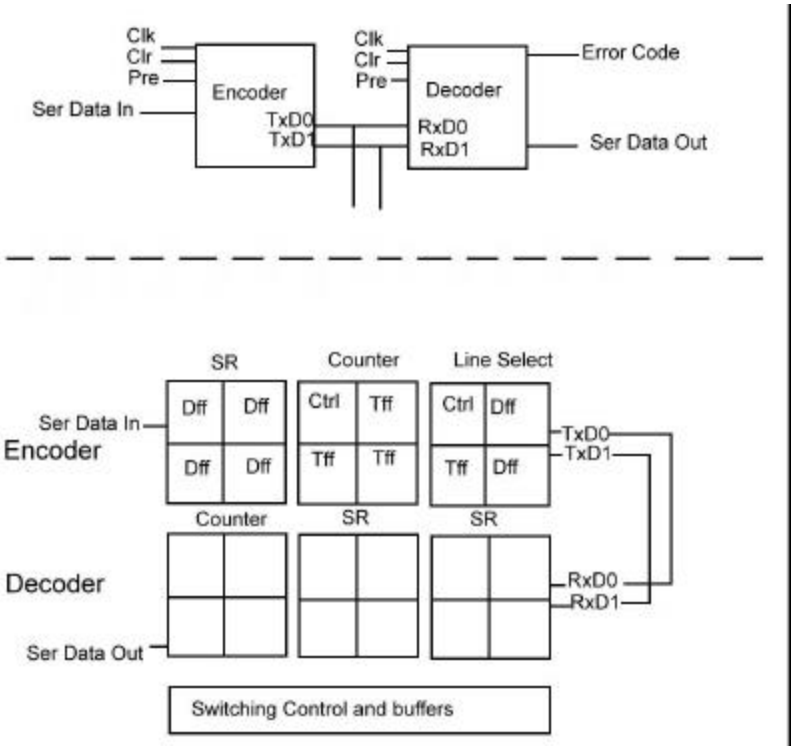
VISIO diagram and sketch of resulting final layout



Final Decoder_parts Layout

Floor Planning

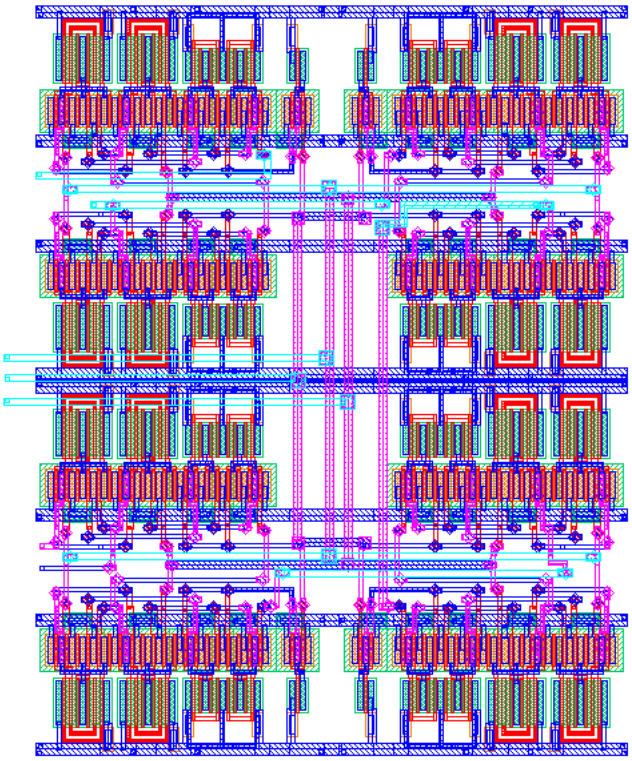
The larger blocks also need to conform to a specific planning layout. The overall circuit was first Floor Planned on paper as a guide for the architecture.



Floor Plan Diagrams

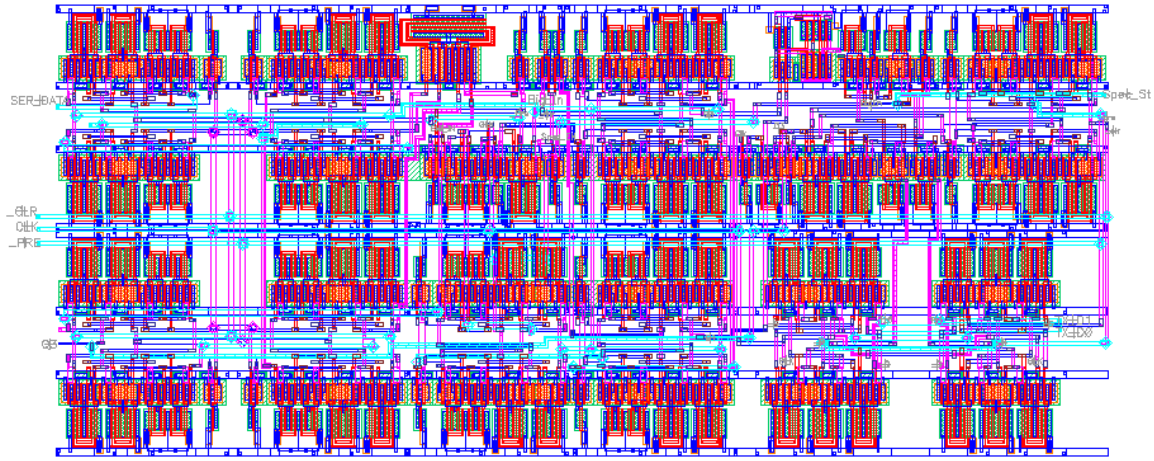
The first goal of floor planning is to make sure there is enough room in the padframe for the core design to fit and connect to the pads. Once this has been figured out, floor planning is used to control the flow of data through the design. This is important for minimizing signal line length and to minimize area consumption. It is also critical to establish clock-trees in timing sensitive circuits, so that the effects of clock-skew are minimized.

Large blocks are assembled from component cells. By flipping and rotating the component cells, a symmetric layout occurs that attempts to maintain an aspect ratio of 1. This can be seen in the layout of the 4-bit Shift Register which was made from 4 DFFs.



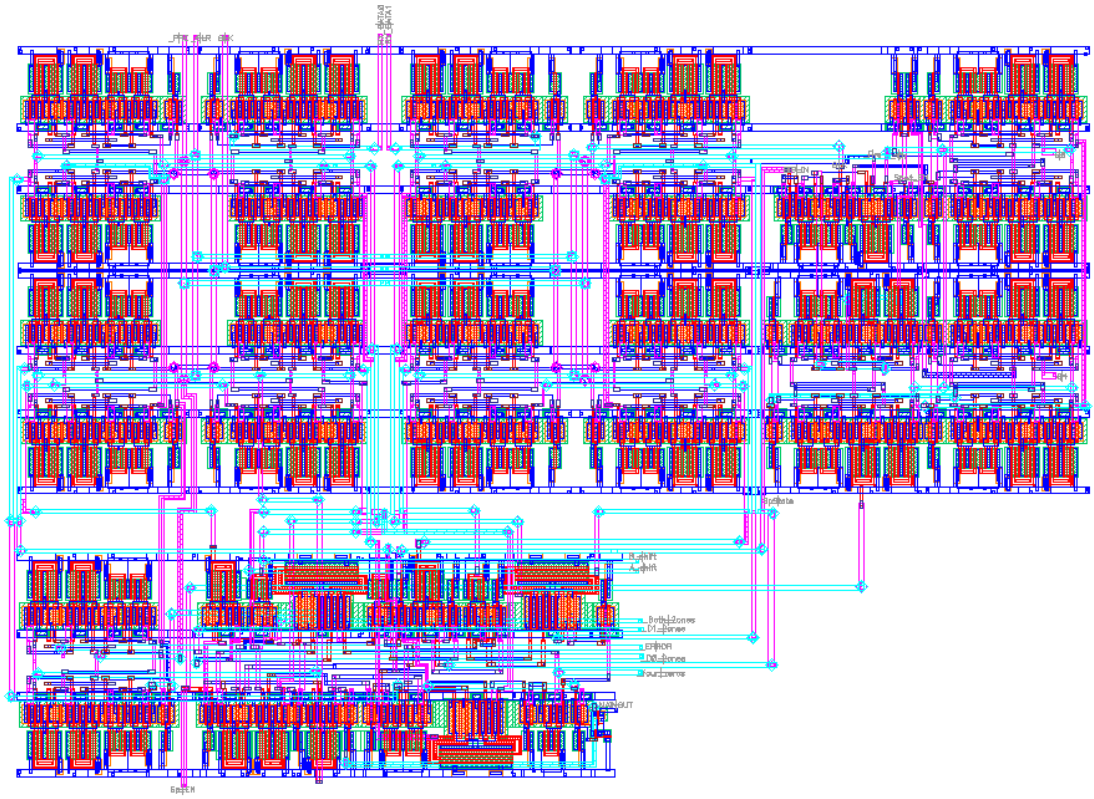
Shift Register Layout

The entire Encoder was assembled from a Shift Register, a Counter and the Line Selector.



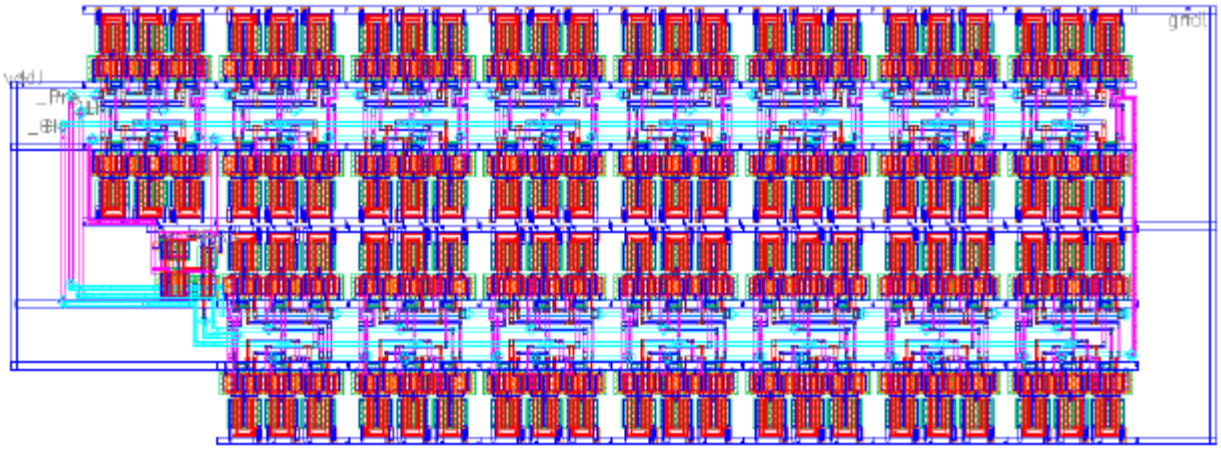
Encoder Layout

Similarly, the Decoder was constructed from two Shift Registers, a Counter and the circuitry used to create the error codes and complete the decoding.



Decoder Layout

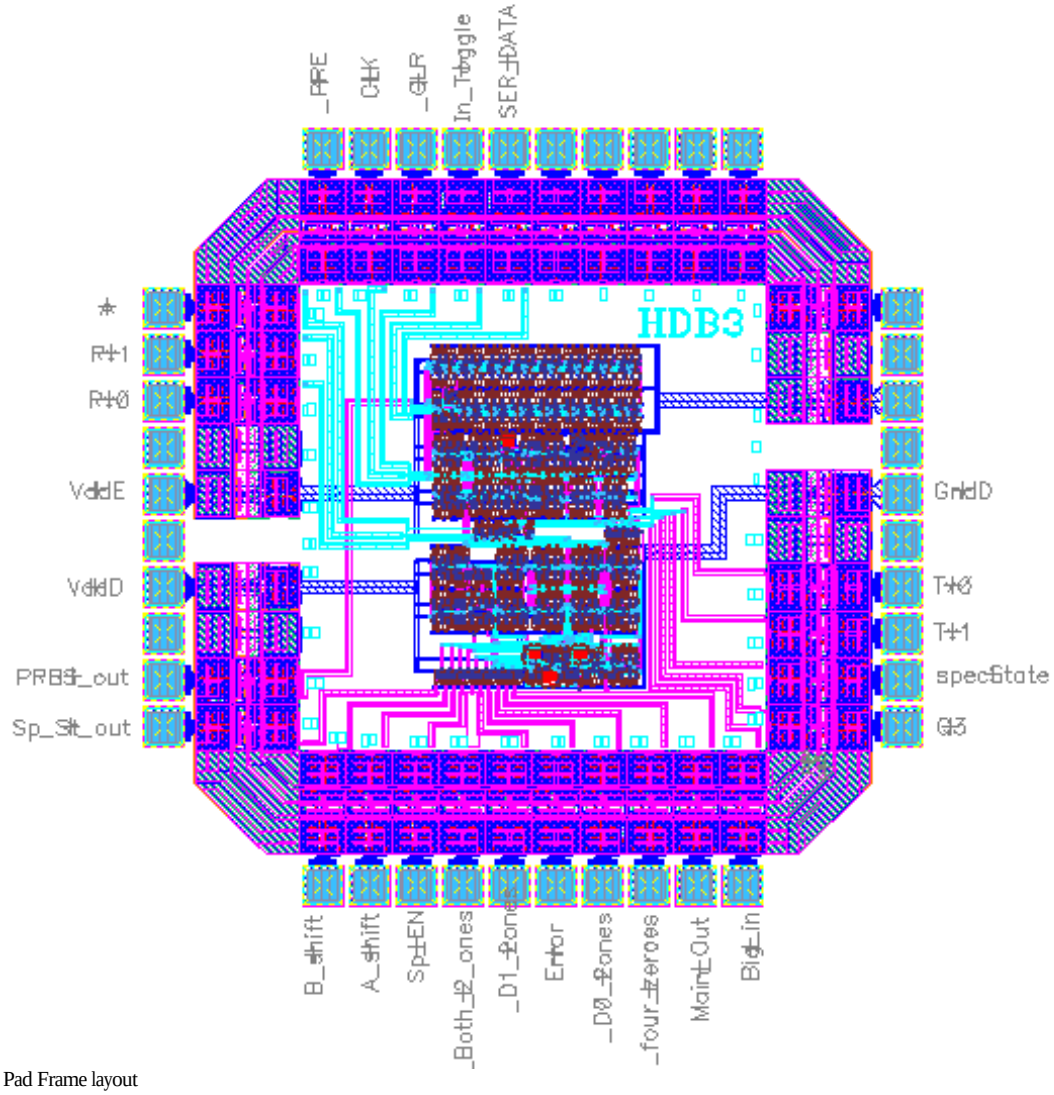
In order to test the circuit, a Pseudo-Random Bit Stream Generator was designed to be part of the completed design. This circuit will produce a serial bit-stream that will act as the input signal to the Encoder/Decoder circuit. The PRBG layout was made to match-up with the rest of the circuit.



Pseudo-Random Bit-Stream Generator

The final result of all this careful floor planning and layout is the assembled circuit, which was tested in the same test bench that was used on the schematic view in order to

compare the output and verify that the circuit functioned properly. After this, the complete design was placed into a pad frame, which was downloaded from the MOSIS web site for this technology size. Again, the whole circuit was tested in a test bench, but this time, the clock rate had to be reduced to 33MHz in order for the signal to pass through the pads without being distorted.



Pad Frame layout

The pads in this pad frame were arranged to accommodate the inputs and outputs of the circuit. Starting with the 'Minframe', input and output pads were substituted where they were needed. A separate Vdd and Gnd were created for the Encoder and Decoder portions and the pad frame is physically broken (two pads were left out) to electrically isolate one portion from another.

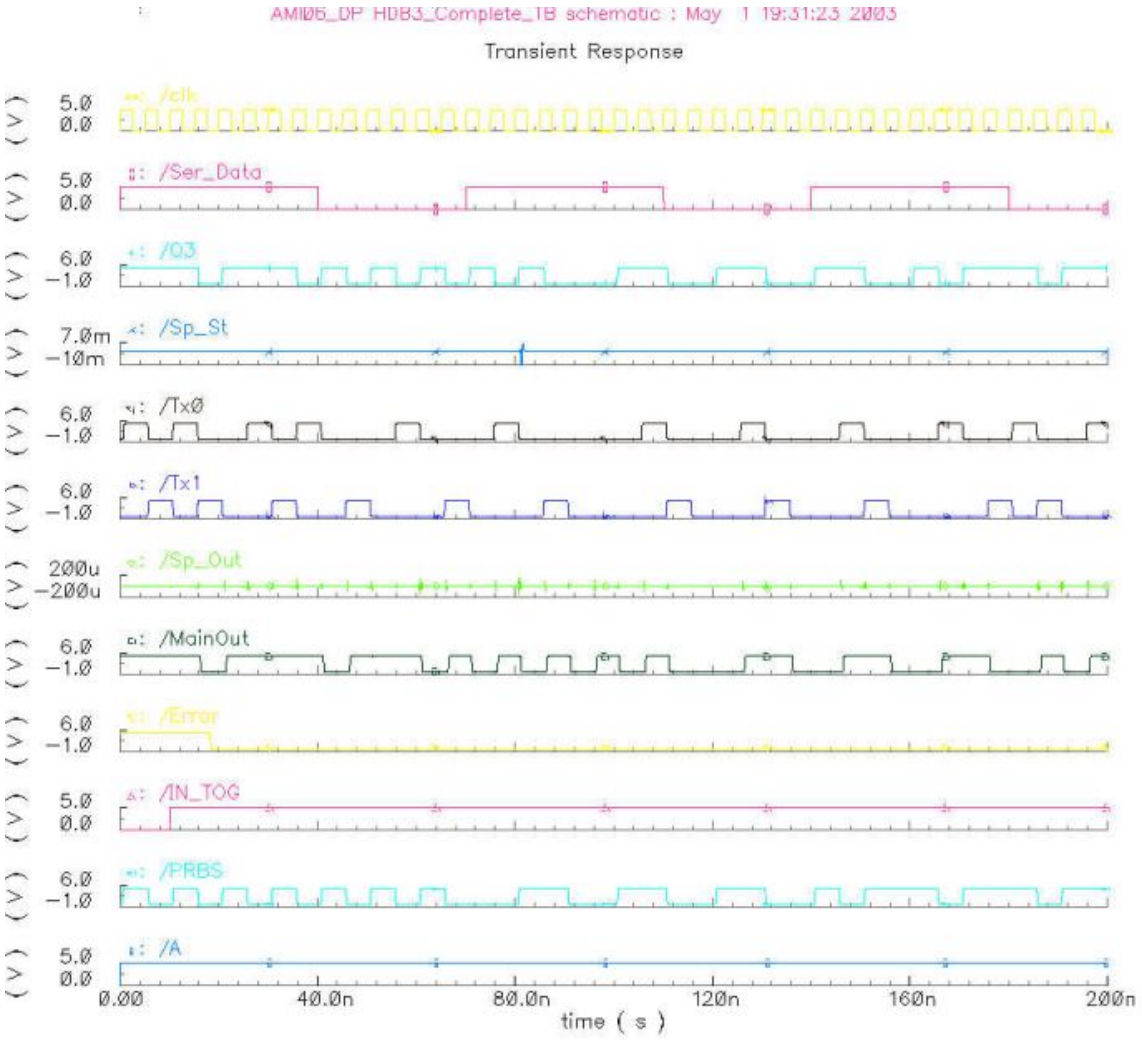
Chapter

5

Putting it all together:

Circuit Simulation

At every stage of design, the individual cells were simulated to insure that they functioned properly. That way, when a larger block is composed of smaller blocks, the only errors that can occur exist in the top-most level of the design. This systematic verification makes troubleshooting a much less difficult task. Another key benefit of the cell-based design is that if a modification is needed due to an architecture adjustment or an engineering change order, that change will automatically apply to every circuit containing the updated element. Most of the changes that were made to this circuit in the process of design were exposed through simulation. Simulations of the final circuit highlight the functionality of this design.

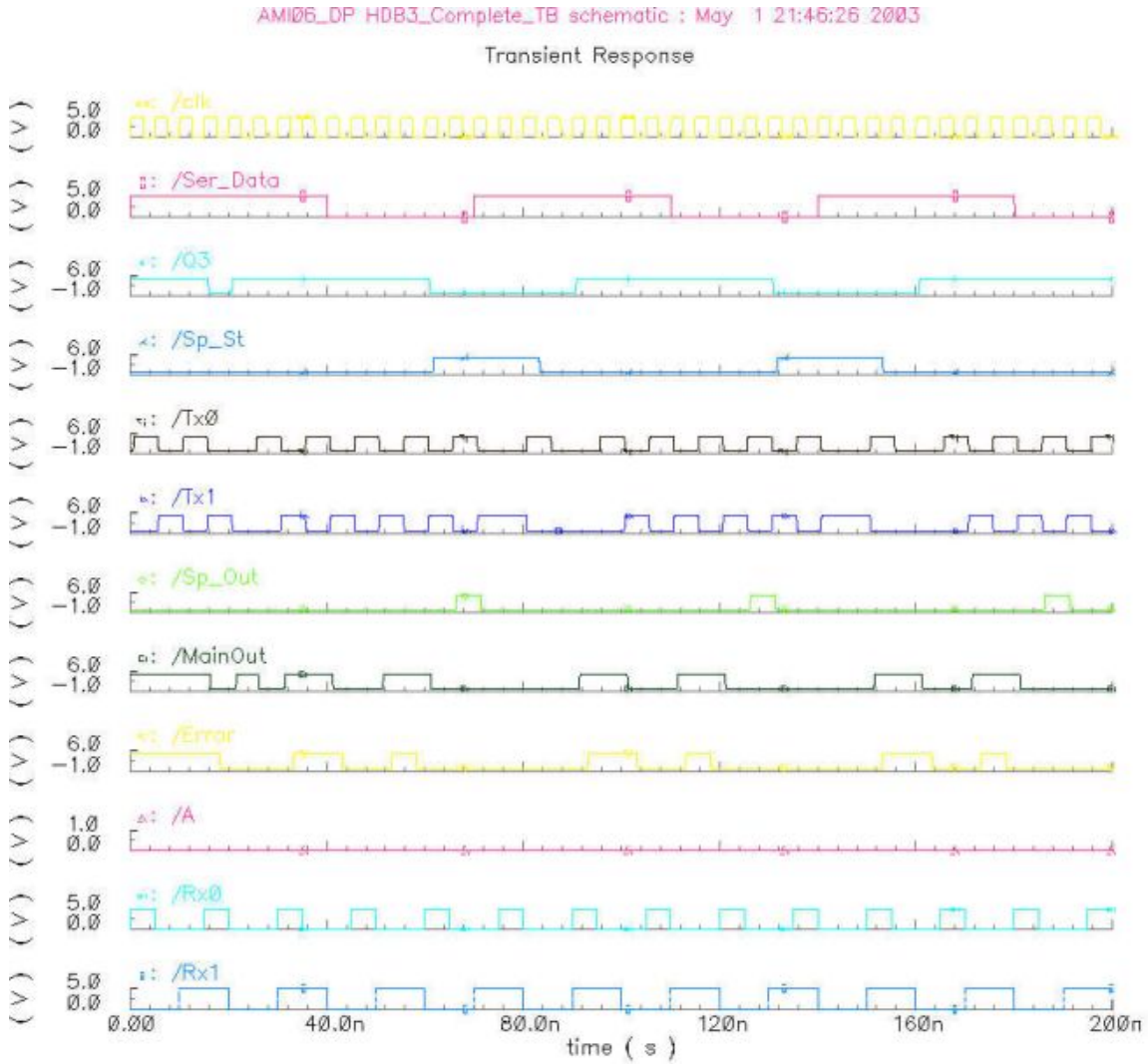


Waveform of HDB3 Circuit using PRBG

The above transient test demonstrates the circuit operating with the PRBG enabled as the serial input source. There is a nine clock-pulse delay from serial input into the Encoder to the output at the Decoder, but the signal is reproduced without error.

MainOut represents the signal after it has been reassembled in the Decoder and Error shows there were no badly transmitted bits.

The next transient demonstrates how the Decoder handles errors. The signals Rx0 and Rx1 were purposely made with violations and switched to the input of the Encoder. The errors are non-alternating ‘1’s and simultaneous ‘1’s on both lines. The Error signal registers the violations as the two signals are decoded, and of course the MainOut would represent an erroneous signal.



Waveform of HDB3 Circuit Showing Errors in the Transmitted Signal

Chapter

6

MOSIS:

Submission

Test and report

Upon receiving the fabricated chips from MOSIS, it was time to test. MOSIS provided five 40-pin DIP packaged chips. To test these chips, it was necessary to connect them to the Agilent 1672G Logic Analyzer that we have available. A test board consisting of a 40-pin socket wired to two rows of extra pins was available. This test board is easily constructed and the extra pins wired to the DIP socket make connection of the logic analyzer probes much easier since stimulus and sensing probes can be connected to the same chip pin.

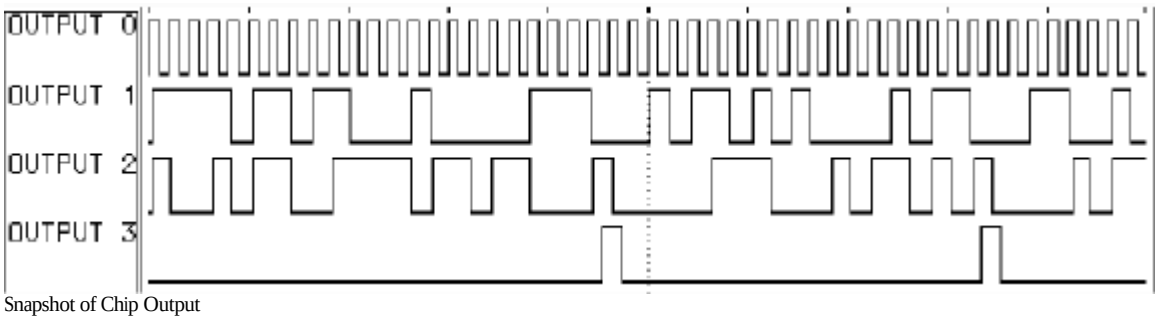
(photo of test board)

After spending two days doing the tutorial and exercises for the Logic Analyzer, I was familiar enough with it to begin connecting my project. As I was connecting the chip, I created a pinout sheet to keep track of which probes went to which pins (See Appendix C). This was invaluable in troubleshooting crossed wires and made changing the test configurations very simple. Once the chip was connected and waveforms appeared on the analyzer, I was ready to examine the functionality of the chip.

From simulation, I knew that I would not be able to run the chip much faster than about 30MHz without distorting the input clock to the point that would cause logical errors within the chip. **Test1** consisted of displaying the Clock, the pseudorandom bit stream (PRBG) input, the Main output and the Error indicator. This first test was kept simple to just make sure the chip was working at all and allowed any probe connection problems to be sorted out. At 30MHz the Main output signal was an exact reconstruction

of the PRBG input. The chip worked! Just to see how fast I could run it without errors, I then increased the clock rate to 100 MHz. The error indicator was now showing that errors had occurred at almost every transition of the clock. I then decreased the clock rate to 50MHz and noticed many fewer errors. This verified that the chip was indeed limited in the speed of operation, which agreed with the simulation.

Switching the clock rate back to 33MHz I began to shift back and forth on the logic analyzer display to see other portions of the test output. I was surprised to find that an error indication had occurred that I did not notice before. Thinking that this was perhaps just a random error, I ran the test stimulus again. After scrolling around the displayed waveform, I found another error. The error was not a fluke. In order to see more of the test, I increased the amount of waveform data that was displayed onscreen, which revealed that I had missed these important details on my first investigations.



Snapshot of Chip Output

Perhaps I had been too hasty in deciding my chip worked perfectly as designed. I slowed the clock rate to 25Mhz and viewed the output. Still the same numbers of error indicators were showing (about one every 70 clock pulses or so). I then slowed the clock rate down to 20Mhz and also to 10Mhz, each time noting that the error indications were the same. This seemed to me that the cause of the error was not clock-rate dependant, and also that is was a very consistent error that occurred in similar rates.

I began looking very closely at the areas of the test output containing the error indicator. I was curious what error had occurred under normal operating conditions. By checking the PRBG input and comparing it to the Main output it became clear to me that the output waveform matched the input exactly even though an error indication was present. This seemed very strange, and demanded a closer look.

Test2 was created to examine the error condition that was noted in Test1. In Test2, all of the pins on the chip, which indicated the types of internal errors that the Decoder portion of the chip detects, were probed. When this HDB3 codec was designed, I wanted to be able to verify what went wrong assuming that there was an error. If I had not done this, then no further information about the functionality of the chip could have been extracted. Now began a serious investigation.

As mentioned previously, the Decoder determines errors based upon a few simple rules: Both received lines (Rx0 and Rx1) can't be '1' at the same time (Both_2ones error); Rx0 can't have two '1's in a row (D0_2ones error) and the same for Rx1 except in

the case of a ‘Special State’ (D1_2ones error); and there can’t be four or more ‘0’s received on either line (Four_zeros error).

The error that was indicated was the D0_2ones error. I checked the transmitted waveform Tx0 and it shows

(more analysis to go here)

Chapter 7

Appendix:

[A] HDB3 chip test configuration

Using Agilent 1672G Logic Analyzer

Stimulus Pod Pinout

Probe#	Chip pin function
0	VDD
1	_CLR
2	_PRE
3	in_tog (manual input (0)/PRBG input(1))
4	A (Encoder/Decoder separator switch – 0=separate, 1=connected)
5	Rx0
6	Rx1
7	SerData

Detection Pod Pinout

Probe#	Chip pin function
Pod 1	
0	Clk
1	PRBS_out
2	Main_out
3	Error
4	Q3
5	Tx0
6	Tx1
7	Spec_en
8	Ashift
9	Bshift
10	BigIn
11	Both_2ones
12	D0_2ones
13	D1_2ones
14	Four_zeros
15	A
Pod 2	
16(0)	Rx0
17(1)	Rx1
18(2)	In_Tog
19(3)	SerData
20(4)	_Clr
21(5)	_Pre

Test Configurations:

Test1	Basic – PRBG input
Test2	Everything – PRBG input
Test3	In_Tog PRBG – off Manual input – on
Test4	A switch Encoder separated from Decoder. (a) Manual decoder input on Rx0 and Rx1. (b) Encoder output through pins Tx0 and Tx1 connected via wire to Decoder input pins Rx0 and Rx1.
Test5	Reset / Preset test.

[B] Diary of an IC Engineer (Dr. Parent):

This purpose of this journal is document a simple full custom IC design project from the beginning to end in order that students new to the field of IC design might learn about the IC design process vicariously before they embark on a project for themselves. The idea that since student time is in short supply, students should have a design reference showing all the pitfalls of the design process so they do not waste time making mistakes common among novices. (Is it better to make mistakes common among experts?)

The major parts of this diary are:

- The original specification of the circuit
- The original timeline proposed
- A scanned appendix of the raw log book of the lead design engineer
- A journal documenting the major parts of the design process. Each section will be written from the viewpoint of the technical manager, lead engineer, and design engineer. The major steps documented are:
    - Writing the proposal
    - Setting up the design environment
    - Writing the specification
    - Logic design
    - Verilog logic verification
    - Hand calculations
    - Transient analysis
    - Layout
    - Final verification
    - Tape out
    - Testing finished product
- Proper documentation of the finalized circuits

The diary includes the cyclic nature of design and has not been sanitized from any warts. If we made a mistake or went down a blind alley we mention it-although we do not dwell on it.

The proposal:
Technical manager:

Essentially I wrote the proposal to document a simple IC design so students might have a resource to help them quickly learn IC design. The factors determining this approach was the seemingly lack of this kind of material in the literature, and the large

student impact this kind of work could have not only at SJSU, but at other institutions. At the time I had not lead a team though an IC design process that lead to a tested IC, so I could be considered a novice in some respects myself. Many textbooks give the advice to look at your company's old designs before starting your project, and since at the time SJSU had not designed, fabricated and tested an IC in at least 5 years I thought that this was a good project to build the base technical infrastructure required to conduct research in the area of IC design.

I came up with an aggressive timeline to have the IC taped out by the end of the summer. Since I had no prior experience I just assigned IC design tasks to be done by certain times by gut feeling. As I said it was an aggressive timeline, with no real data to support whether it was possible or not. I also left no "fudge factor time" in case things went wrong[1]. (I did not pad my time line.) The reason for this was that I really wanted this grant to be funded by Cadence Design Systems, and I did not want them to think, that I thought the grant was a form of university welfare or taxation. The people from Cadence never said this, but I had never dealt with a company before and I had been a "fly-on-the-wall" during some of their other university program meetings in which I felt other faculty did not appreciate the enormity of Cadence's gift.

The proposal was funded, but with a reduction in intern support. Although I would have liked to have more support, I realized that my infrastructure was not set up to manage 5 people. I still had to write the specification and set up a design environment for the project. I would have had a lot of people sitting around waiting for me to set things up. Fortunately the proposal called for other types of documentation to be created so I had plenty of work that I could delegate when the first intern (Prashanta Lal) reported for work.

Setting up the design environment

Even though the AMI06 tech files were in place I felt I needed to come up with a more accurate method to do hand calculations[2]. The methods in the books tended to give answers that were far away from what the sprectre answer was in terms of propagation delay (In general over 20%). I felt that this was worth it because it can save a lot of simulation time on the project, and for all students doing similar projects.

After some talks with Morris Jones I realize that I could just extract the A and B constants from the measure spice results. The problem was that the extracted results would work well if the propagation delay was the same. If I tried to design for slower propagation delays the hand calculations would predict a much larger than necessary width for a propagation delay. The errors were still too large. I did not understand the physics of it, but I just linearized the A constant and the ratio between Wn and Wp for propagation delays from .1ns to .5ns. I also reformatted the equations so that Wn would

[1] This of course goes against any project management theory meaning, I should have known better.

[2] Even though one can run parameterized variables in the simulator, these still take a long time to run.

be a function of the drain capacitances. I tried to break the model, but the worst error I found was 12% error at .250 ns pro delay.

I added this to the tutorial along with make the design flow very integrated with every step in the process. I wanted to a sample design that pushed the technology so that the students would have a complete example if at least how to do something that the minimum sized inverter. I also wrote the tutorial to show the aspects of design rather than just learn the tool. This idea came out of a focus group with students of mine based on there experiences in EE166. I only rewrote the tutorial after I could not fine one on the web. The WPI tutorial was the best at the time, but our tutorial surpasses it because we show how to actually design the inverter to a specification, rather than just stamp down a minimum sized inverter.

Unfortunately I did not have these models ready for Dan and Prashanta to use when they needed them and so getting the circuits to conform to the timing specification took more time than it should have.

I also wanted to write a tutorial on how to use the schematic capture version of verilog. Since we were going to do circuits that were more complex logically, I felt that using an hdl to get the logic right, before spice simulations was wise. This was ready in time for Dan and Prashanta.

I was able to keep Prashanta busy with other tasks while I was doing these things that I felt only I could do, or that given the time constraints, I could do them fastest. Since Dan was hired later in the summer I did not have to worry about him not having any real tasks to do, because he was in the research phase of the project.

Even though Dan started later in the summer, I still felt that we would make our tape out date of semptember.

Writing the specification

Writing the specification was somewhat hard in that we had no boundary conditions. We were making up our own circuit to do what we wanted. I felt that we should try to push the limits of the technology so that we would encounter any problem that a student might encounter. We choose the telcom test bench as a medium complex circuit that maybe others or we could use to start networking research. I though it would be not too hard because the vhdl code for these items was not very complex. (I had worked on this type of design at Transwitch for an ATM test bench.) I felt that I was able to give a rough idea to Dan and he was able to turn it into a detailed specification. This took time, but I was still working on the design environment so it did not slow the project down. After this task, Dan and Prashanta would do the majority of work.

Logic design

Things proceeded smoothly until the beginning of the semester. Dan and Prashanta got a reputation as cadence tool experts and found that it was hard to do work in the lab. Unfortunately it was too late in the semester when I found out about this and there was no way to get them lab space. The only other glitch was the verilog simulator required a special test bench to run PRBS type circuits which I had to solve. At this time I was busy trying to manage the unix system and 500 users. I also ran a short course to debug my tutorial and to try to get other professors involved in IC design.

Verilog logic verification

No problems from my end in this segment. Dan and Prashanta were doing the work and did not need much superevision.

Hand calculations

I was not sure if they used my new hand calculation method or just used the parameter analyzer function in affirma. The students who used it in class saved time, but it was a harder model to implement.

Transient analysis

The project did slow down at this point due to D and P having a hard time concentrating in lab. Also they could not work full time.

Layout

Disaster struck. We had a data loss and we did not have back ups (The tapes were bad.) I now have backup to CDROM but it was too little to late. Not everything was lost, but Dan had to go back and check each circuit and sub circuit.

Some of the layouts could not be minimized using euler path. Eric had found a method, but I did not like it because it depended on the experience of the user. He did quite a bit of research and actually found a viso plug in that would do an exhaustive search of layout optimizations. Although we were behind schedule, it was well worth it because simpler layouts will save time later in the process. I am not sure if this will be useful in the intro ic course, but it does look good for complex circuits.

The project is behind and it was due to just more than loss of data, although this did cost at least a month. I think I underestimated the complexity of the project, and I needed to take a more active role with design reviews during last semester. I was able to finally get lab space for D and P , and once the computers work we can isolate them from the general population.

Final verification

Tape out

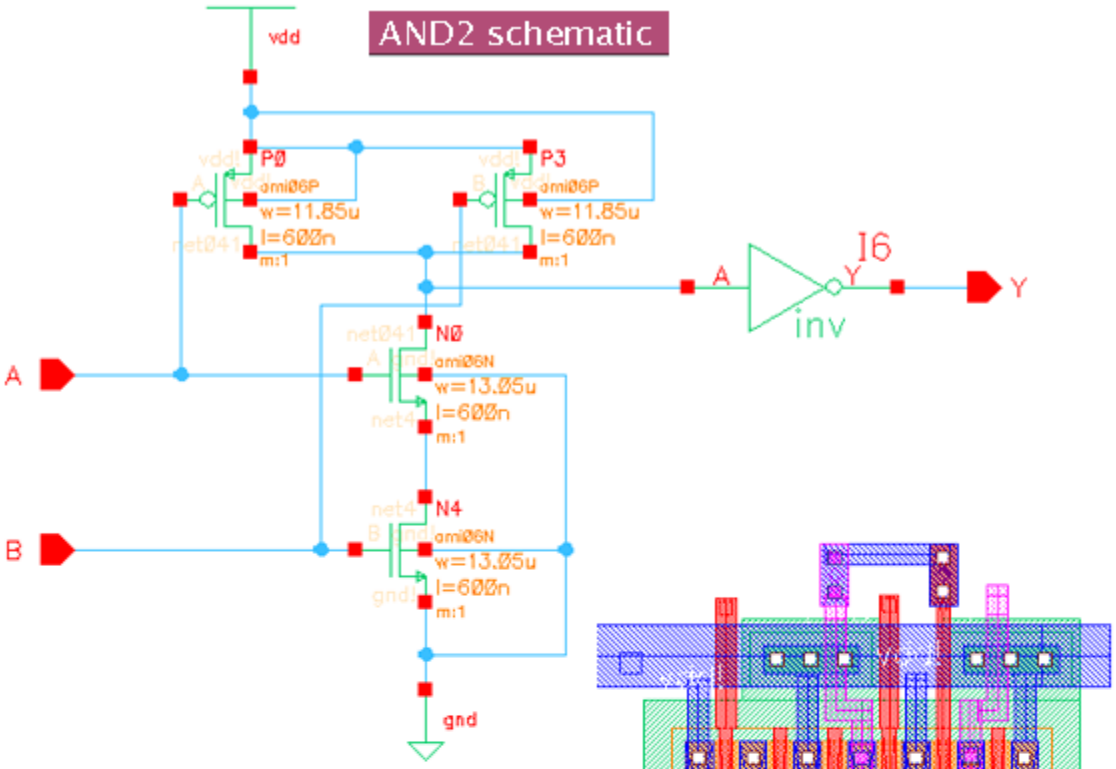
Testing finished product

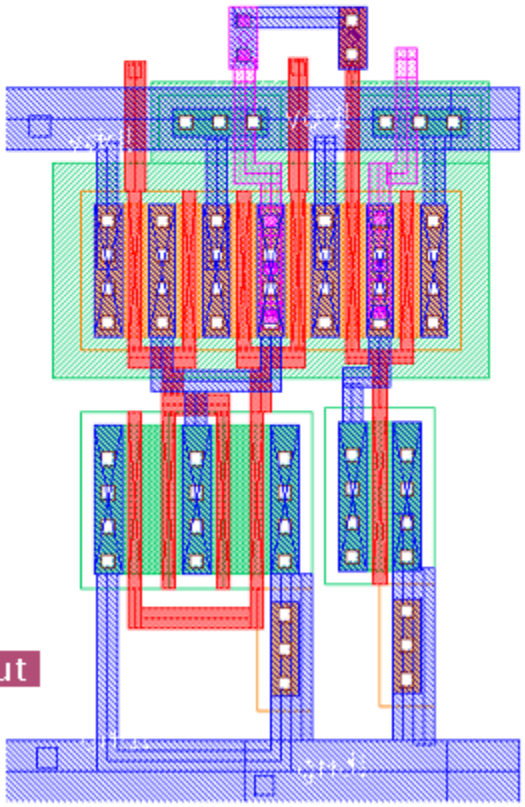
[C] Cell Documentation

Project Cells:

AND2
AND3
Counter
Decoder_parts
Decoder_test
DFF_01
D_FlipF
D_FlipF_a
DLATCH
ENCODER
Final_Shift_Reg
HDB3_Complete
INV_thin
INV_XNOR2
2Inv_buffer
Line_Select03
NAND2_thin
NAND3
NAND4
PRBG_new_D
Padframe_05_test
Stat_Switch
Stim_Switch
TFF
TG
XNOR2

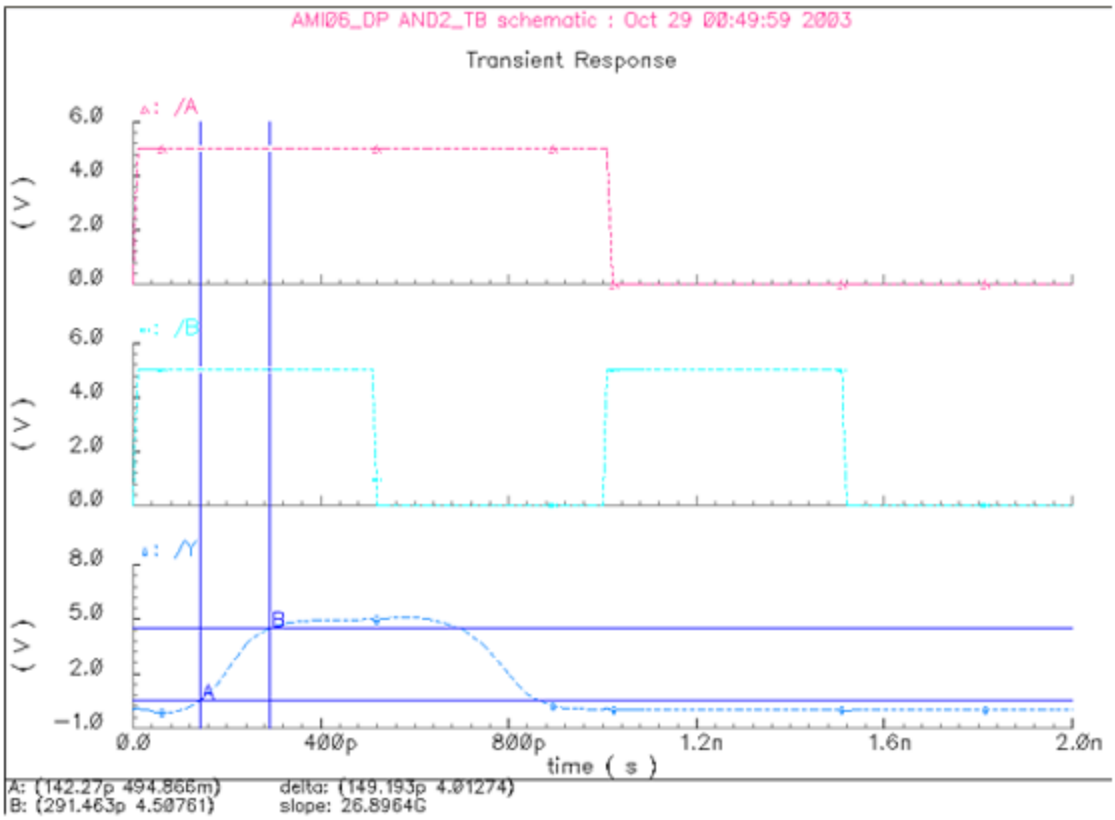
AND2



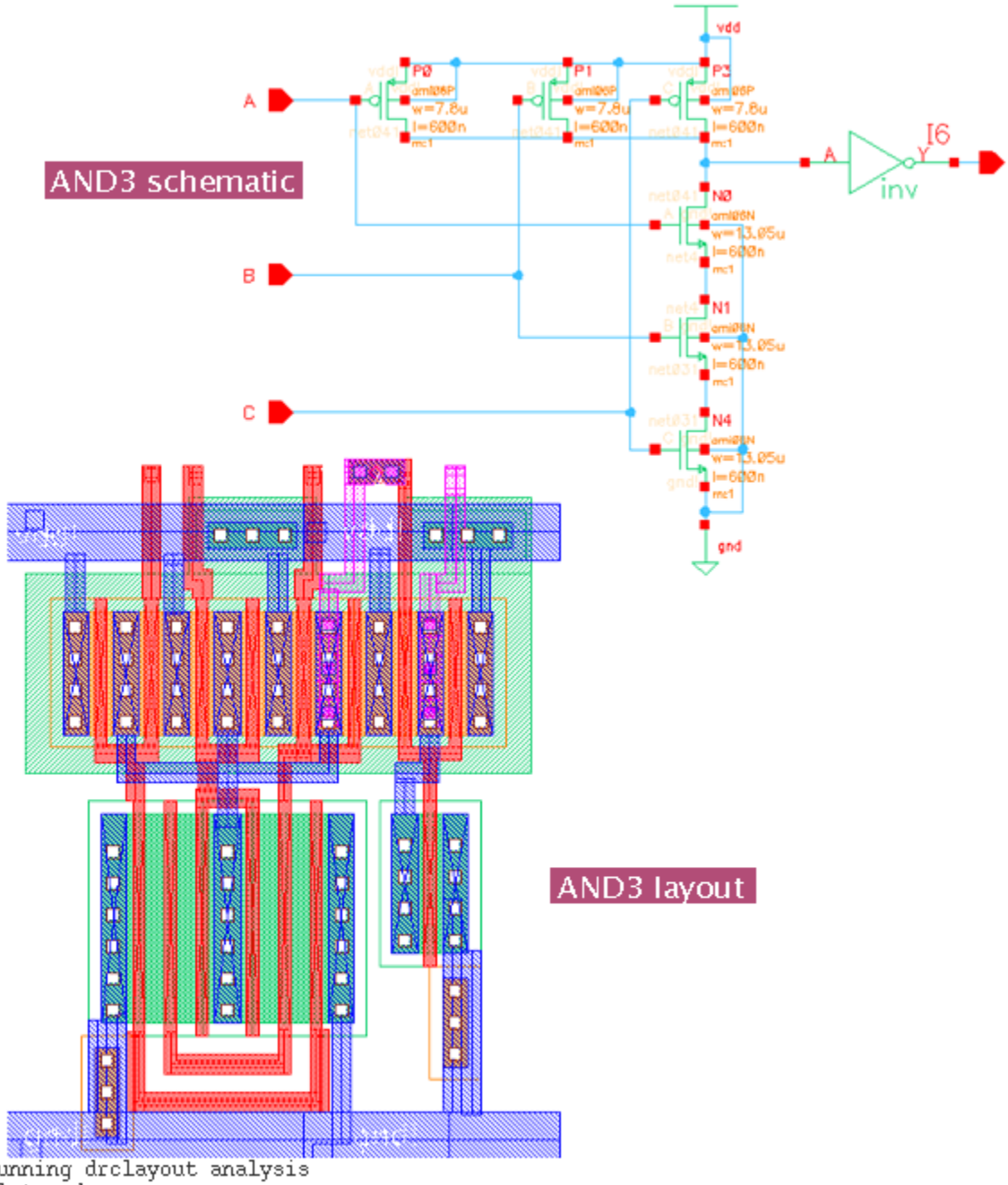



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Oct 28 15:37:24 2003
    completed ....Tue Oct 28 15:37:28 2003
    CPU TIME = 00:00:01  TOTAL TIME = 00:00:04
*********   Summary of rule violation for cell "AND2 layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
           8               nets
           0               terminals
           6               pmos
           5               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
           7               nets
           5               terminals
           3               pmos
           3               nmos


The net-lists match.
```

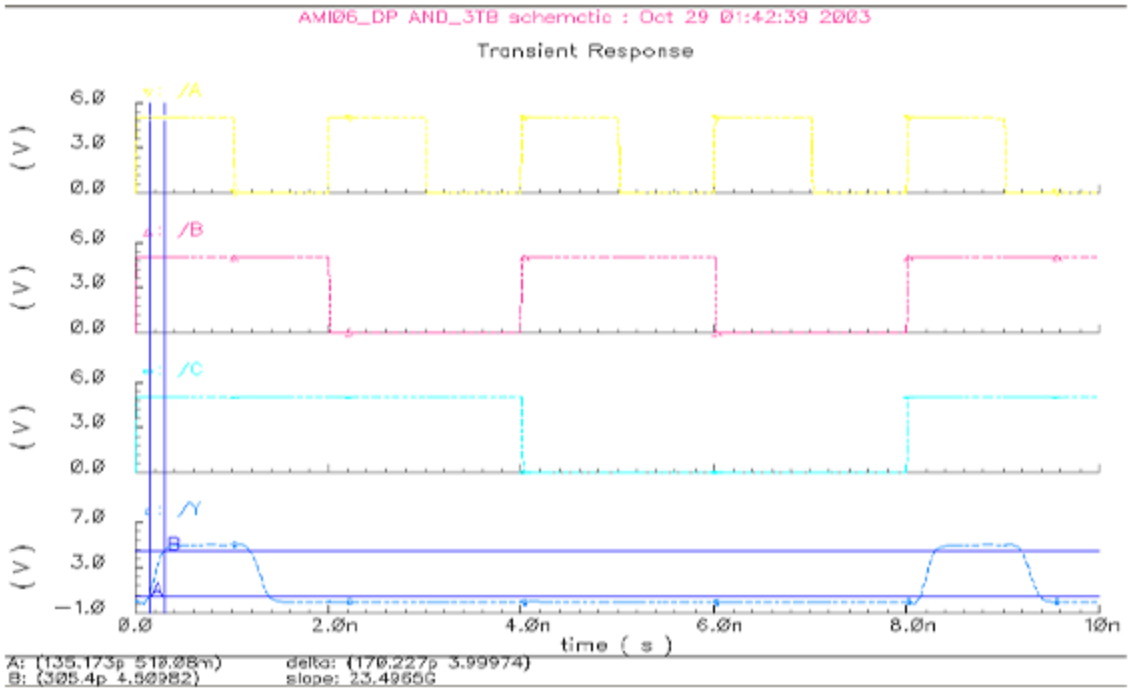


AND3

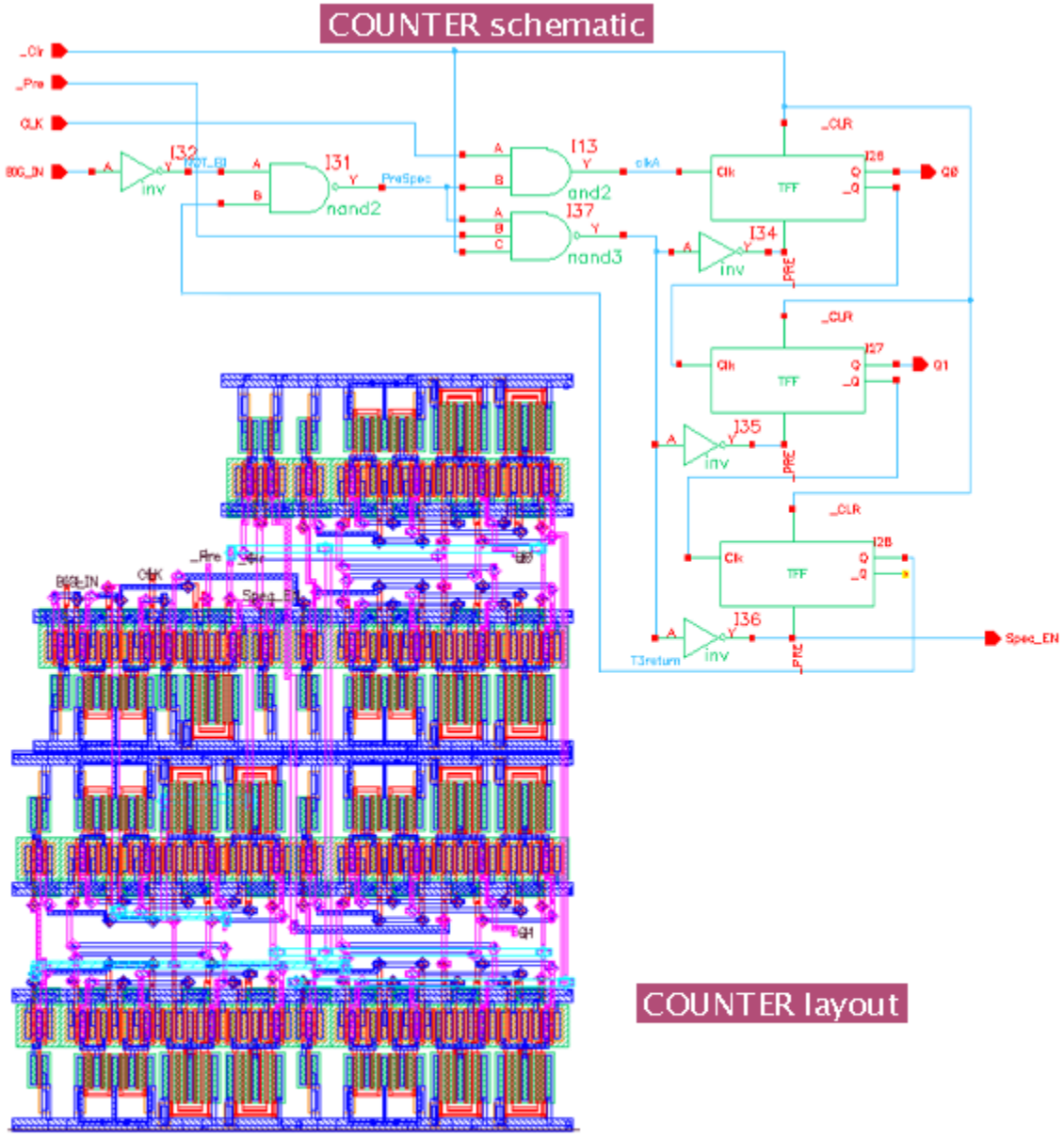



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 01:33:53 2003
    completed ....Wed Oct 29 01:33:54 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "AND3 layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        11              nets
        6               terminals
        8               pmos
        7               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        9               nets
        6               terminals
        4               pmos
        4               nmos


    Terminal correspondence points
          1     A
          2     B
          3     C
          4     Y
          5     gnd!
          6     vdd!

The net-lists match.
```

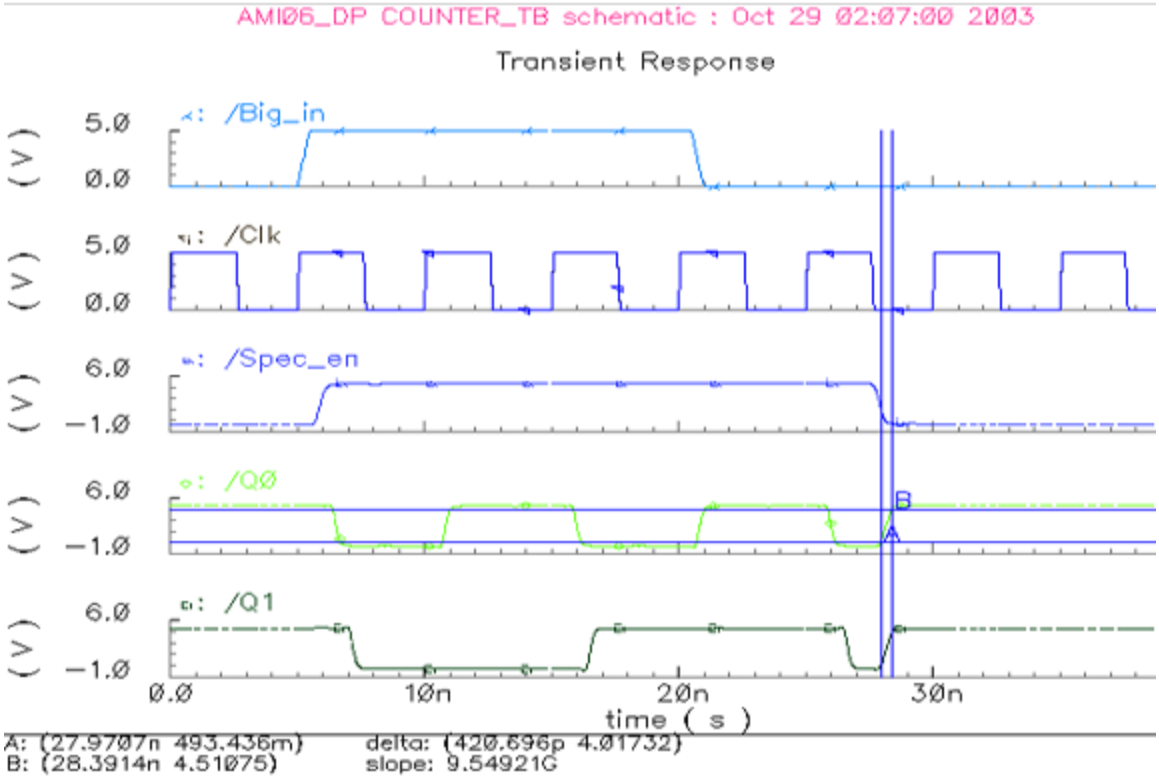


COUNTER

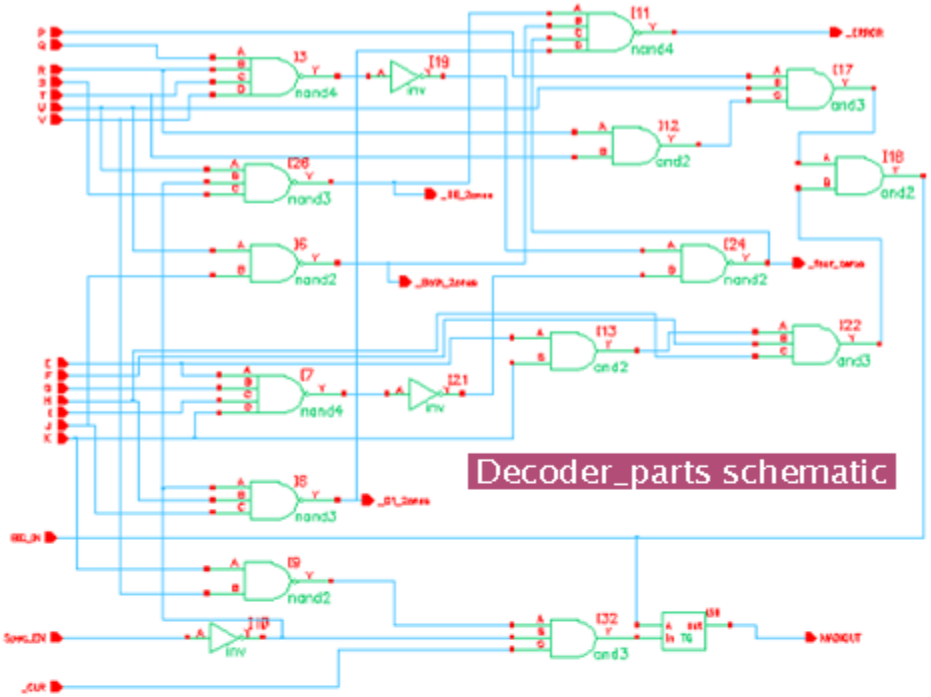


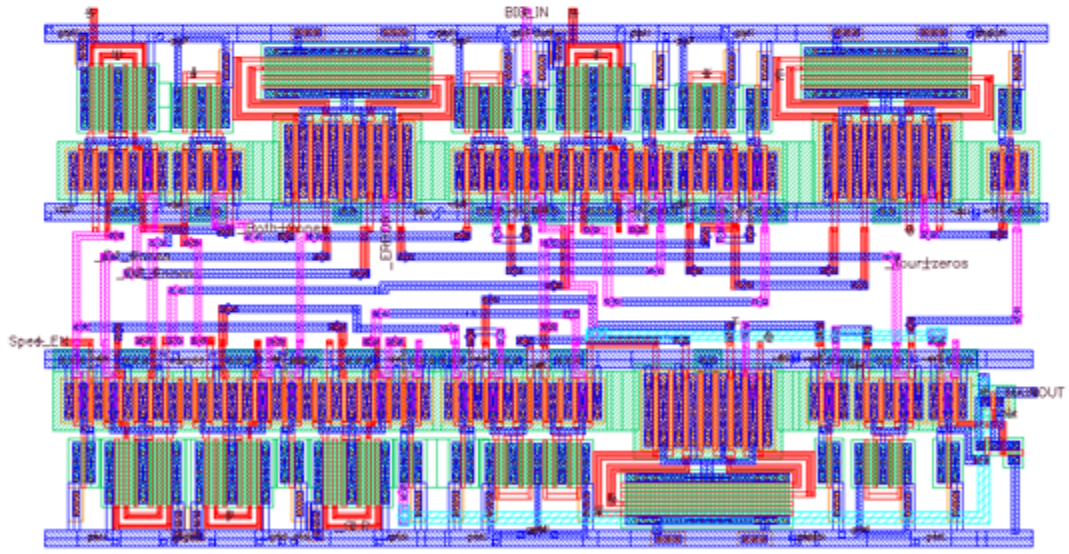

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 01:47:50 2003
    completed ....Wed Oct 29 01:47:57 2003
    CPU TIME = 00:00:01  TOTAL TIME = 00:00:07
*********   Summary of rule violation for cell "COUNTER layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         124             nets
         9               terminals
         156             pmos
         145             nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         84              nets
         9               terminals
         78              pmos
         78              nmos


    Terminal correspondence points
           1     BIG_IN
           2     CLK
           3     Q0
           4     Q1
           5     Spec_EN
           6     _Clr
           7     _Pre
           8     gnd!
           9     vdd!

The net-lists match.
```



Decoder_parts

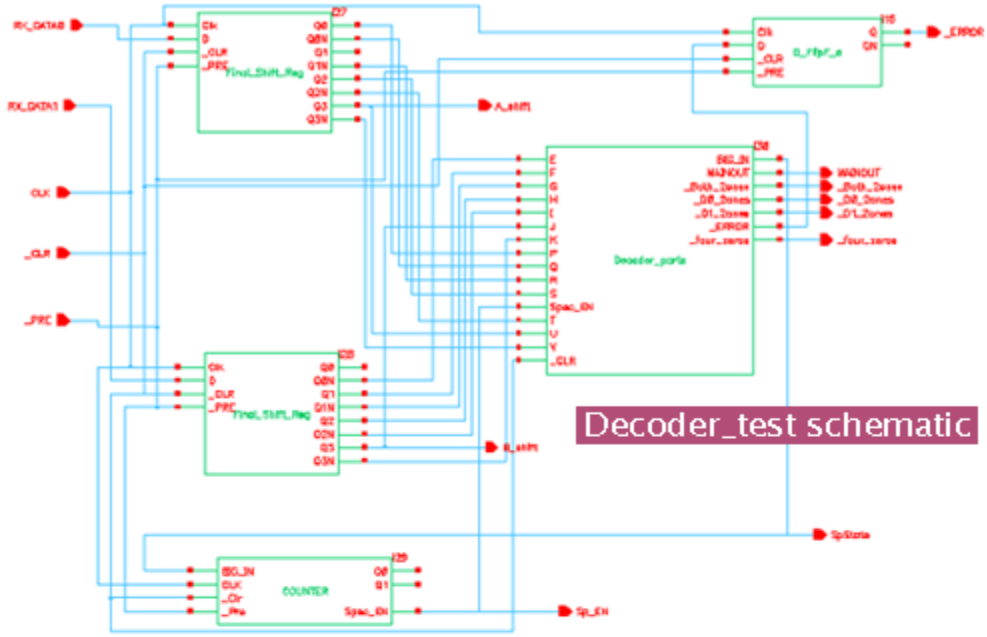


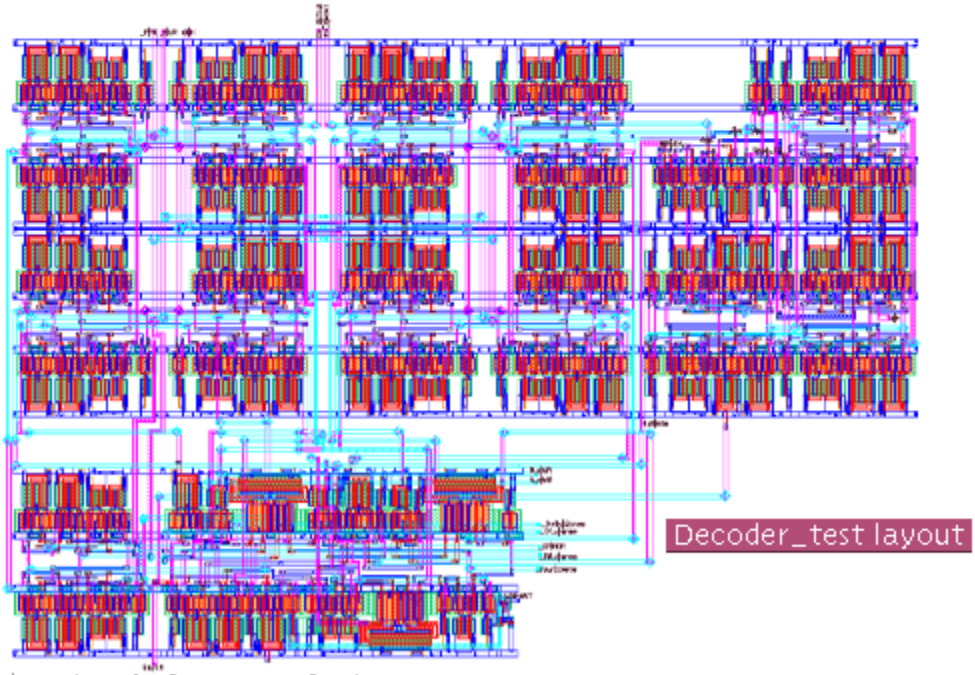



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 18:33:02 2003
    completed ....Wed Oct 29 18:33:07 2003
    CPU TIME = 00:00:01  TOTAL TIME = 00:00:05
*********   Summary of rule violation for cell "Decoder_parts layout"   *********
   Total errors found: 0
```

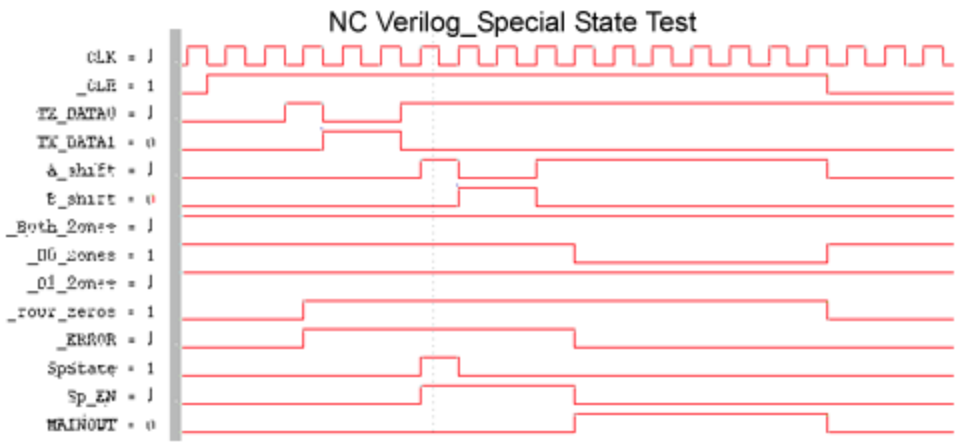
```
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        84              nets
        25              terminals
        99              pmos
        78              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        68              nets
        25              terminals
        50              pmos
        51              nmos


    Terminal correspondence points
          1     BIG_IN
          2     E
          3     F
          4     G
          5     H
          6     I
          7     J
          8     K
          9     MAINOUT
         10     P
         11     Q
         12     R
         13     S
         14     Spec_EN
         15     T
         16     U
         17     V
         18     _Both_2ones
         19     _CLR
         20     _D0_2ones
         21     _D1_2ones
         22     _ERROR
         23     _four_zeros
         24     gnd!
         25     vdd!

The net-lists match.
```

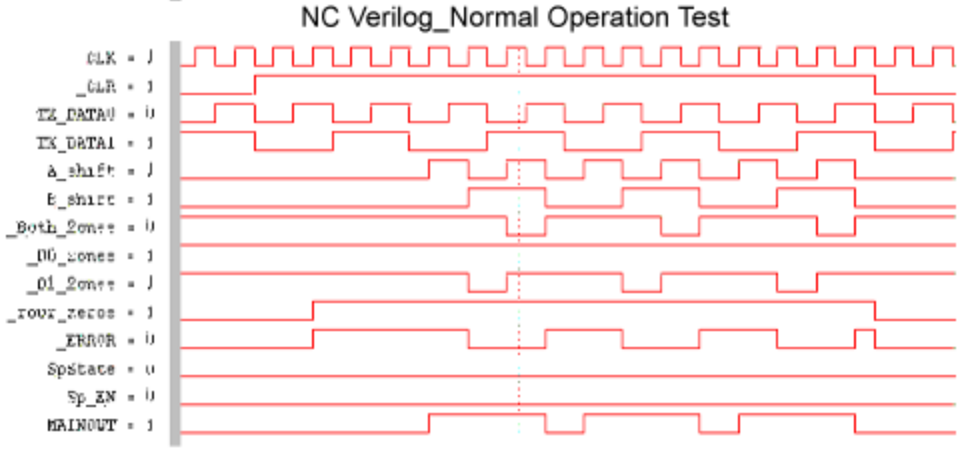
Decoder_Test



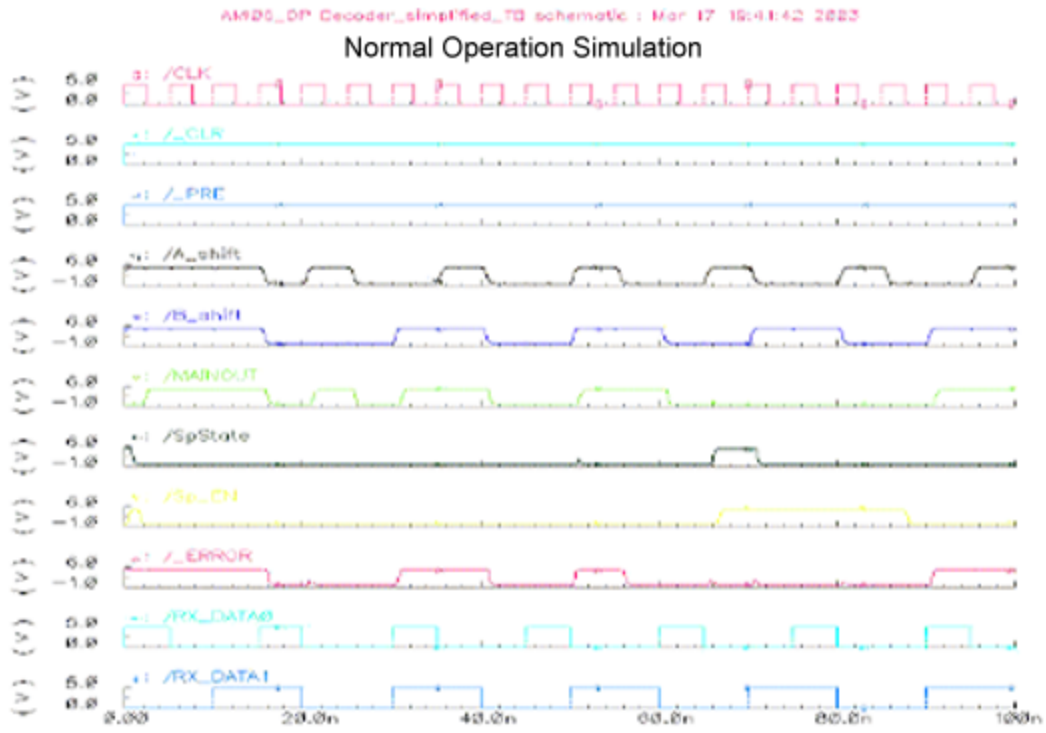


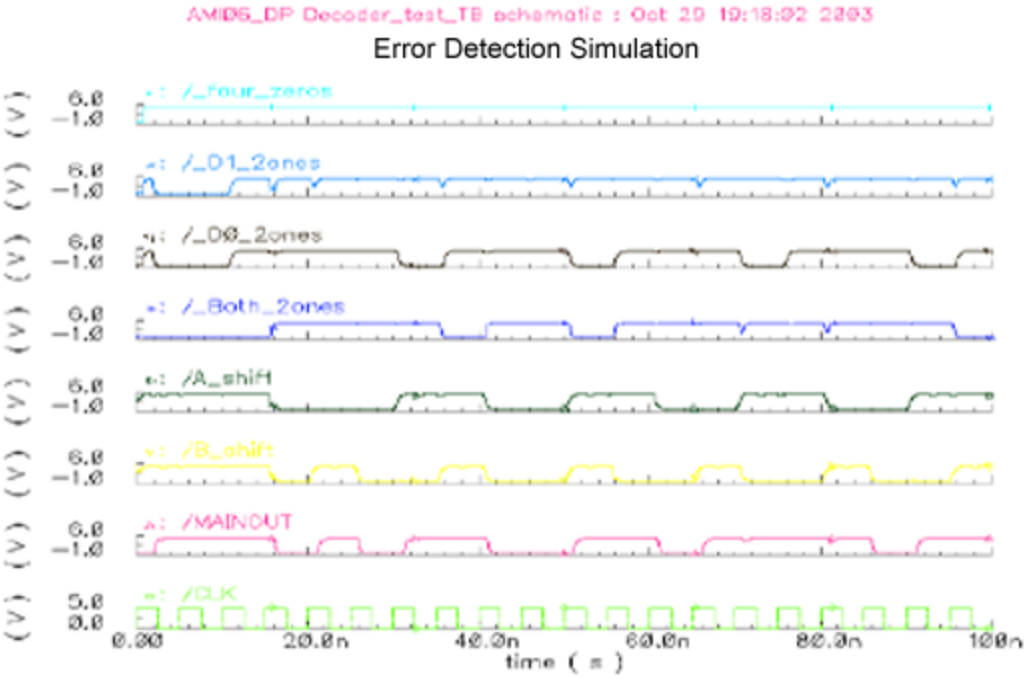

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Nov 12 14:02:31 2003
    completed ....Wed Nov 12 14:03:08 2003
    CPU TIME = 00:00:10  TOTAL TIME = 00:00:37
*********   Summary of rule violation for cell "Decoder_test layout"   *********
```



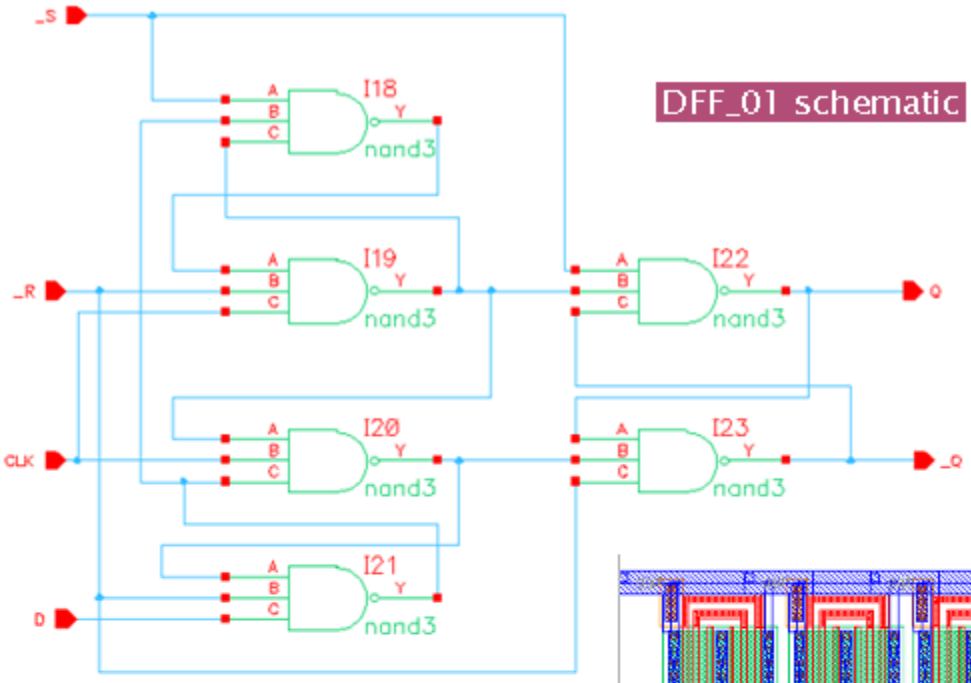



```
Net-list summary for /home/eecad37/cell/LVS/layout/netlist
   count
     488            nets
      17            terminals
     633            pmos
     592            nmos

Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
   count
     324            nets
      17            terminals
     317            pmos
     318            nmos


Terminal correspondence points
       1     A_shift
       2     B_shift
       3     CLK
       4     MAINOUT
       5     RX_DATA0
       6     RX_DATA1
       7     SpState
       8     Sp_EN
       9     _Both_2ones
      10     _CLR
      11     _D0_2ones
      12     _D1_2ones
      13     _ERROR
      14     _PRE
      15     _four_zeros
      16     gnd!
      17     vdd!

The net-lists match.
```

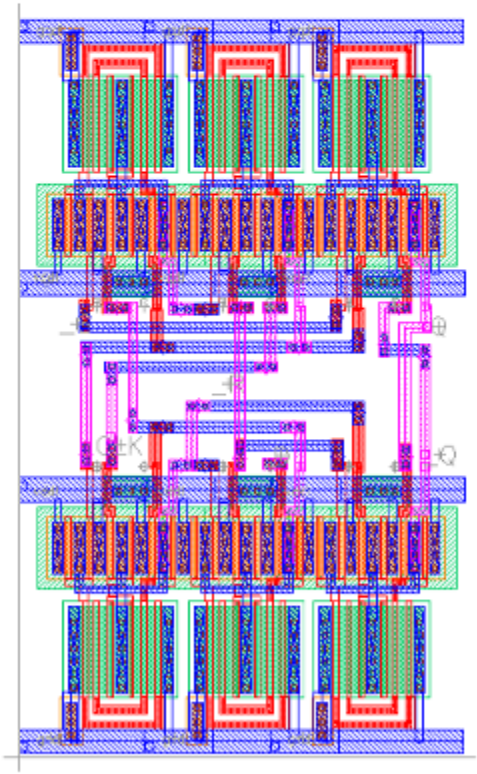

AMI05_DP Decoder_simplified_TB schematic : Mar 17 15:41:42 2023
Normal Operation Simulation
/CLK
/_CLR
/_PRE
/A_shift
/B_shift
/MAINOUT
/SpState
/Sp_CN
/_ERROR
/RX_DATA0
/RX_DATA1
6.0
0.0
-1.0
0.00
20.0n
40.0n
60.0n
80.0n
100n


AMI05_DP Decoder_test_TB schematic : Oct 20 19:18:02 2003
Error Detection Simulation
/_four_zeros
/_D1_2ones
/_D0_2ones
/_Both_2ones
/A_shift
/B_shift
/MAINOUT
/CLK
6.0
0.0
-1.0
0.00
20.0n
40.0n
60.0n
80.0n
100n
time (s)

DFF_01



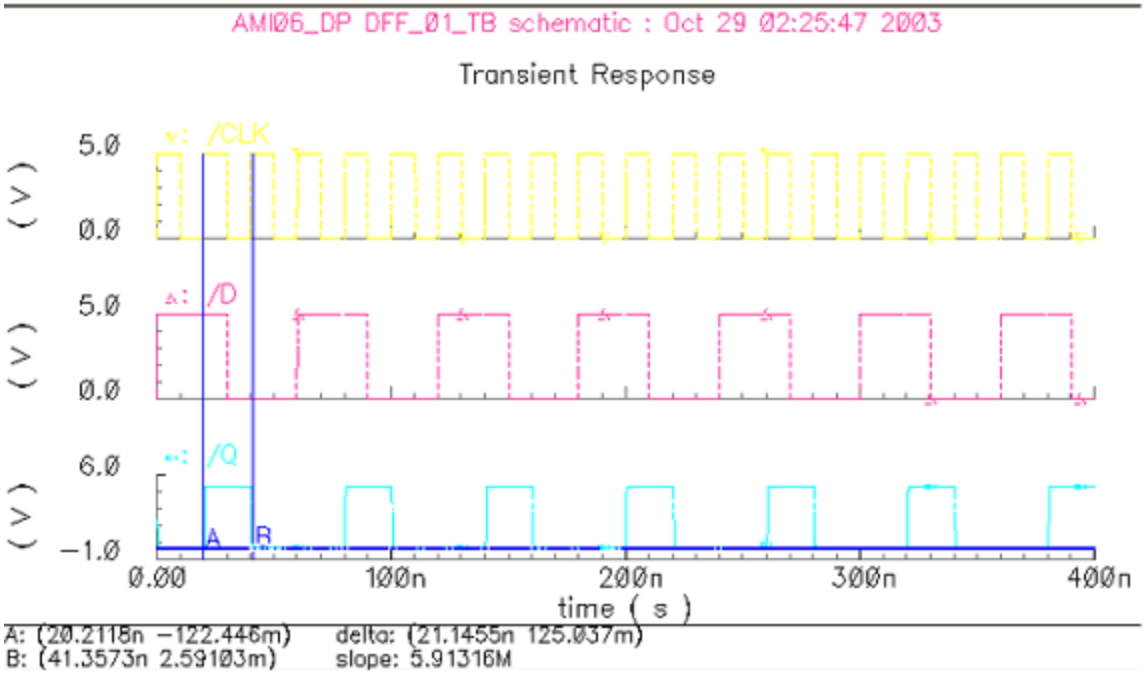
DFF_01 schematic



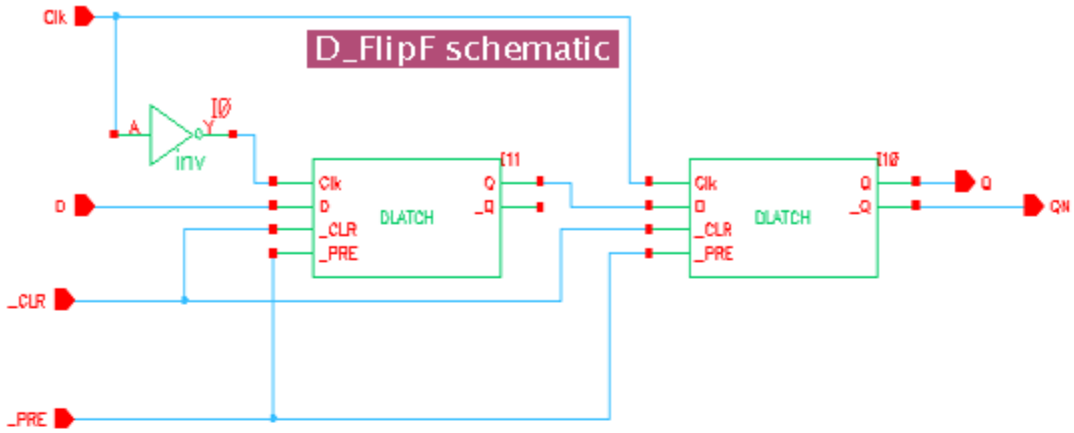
DFF_01 layout

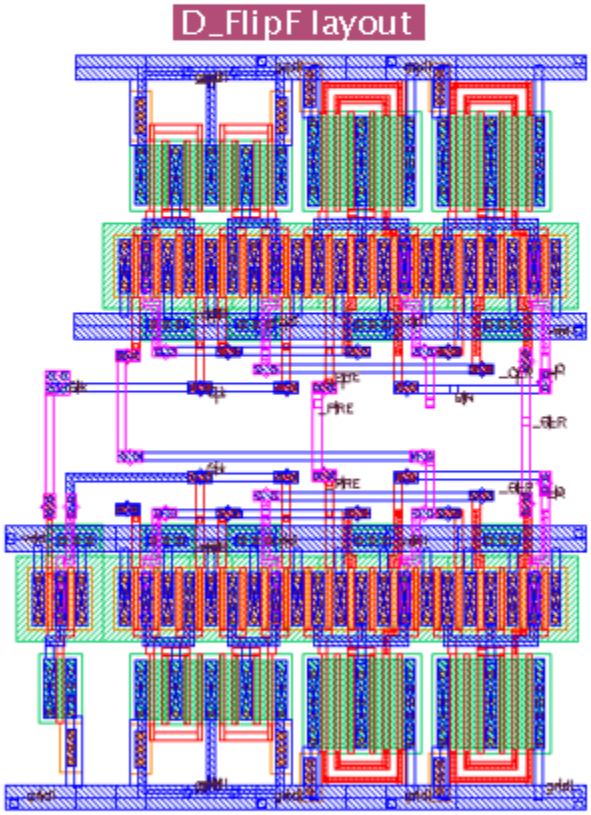
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 02:18:29 2003
    completed ....Wed Oct 29 02:18:30 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "DFF_01 layout"   *********
   Total errors found: 0
```

```
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         36               nets
         8                terminals
         36               pmos
         36               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         24               nets
         8                terminals
         18               pmos
         18               nmos


    Terminal correspondence points
           1      CLK
           2      D
           3      Q
           4      _Q
           5      _R
           6      _S
           7      gnd!
           8      vdd!

The net-lists match.
```



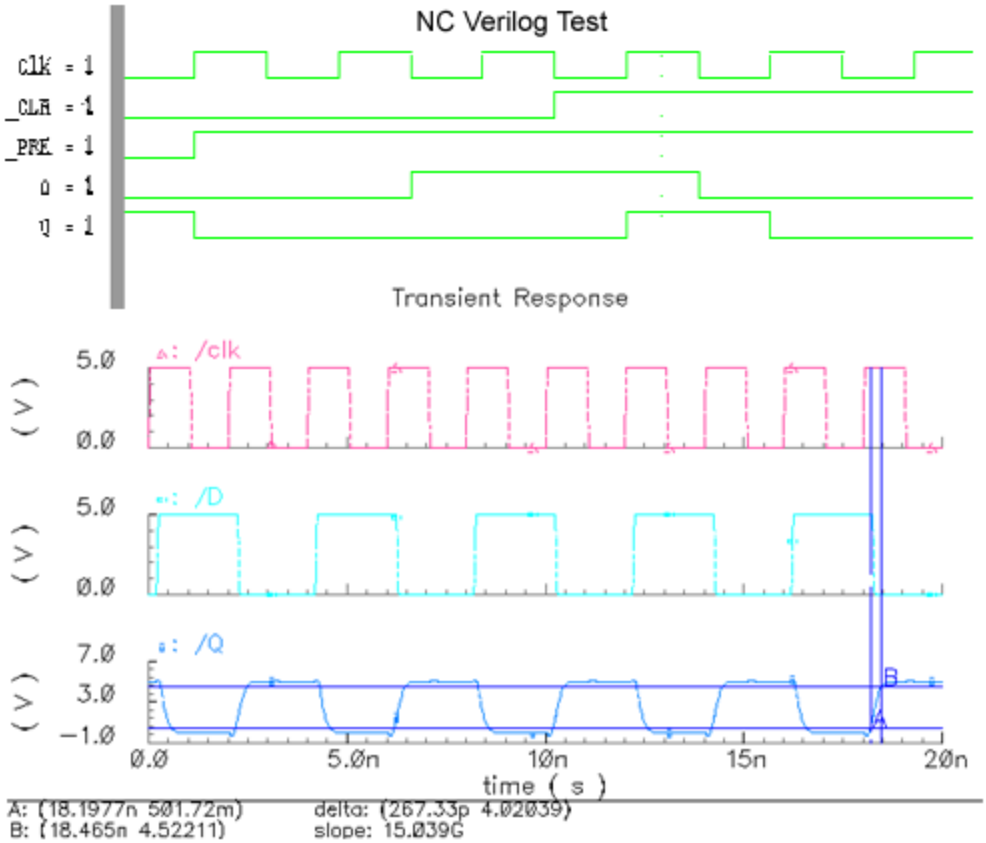
D_FlipF





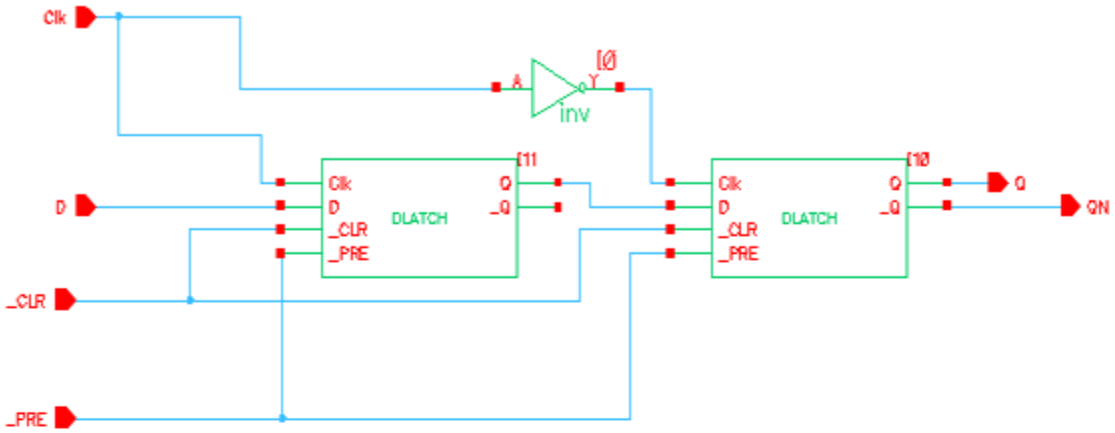

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 03:10:58 2003
    completed ....Wed Oct 29 03:10:59 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "D_FlipF layout"   *********
   Total errors found: 0
```

```
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        39              nets
        8               terminals
        42              pmos
        41              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netl
       count
        27              nets
        8               terminals
        21              pmos
        21              nmos


    Terminal correspondence points
          1     Clk
          2     D
          3     Q
          4     QN
          5     _CLR
          6     _PRE
          7     gnd!
          8     vdd!

The net-lists match.
```

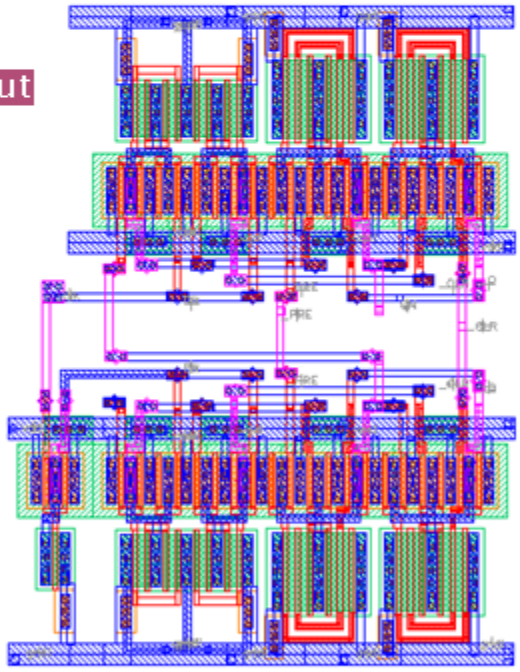


D_FlipF_a

D_FlipF_a schematic

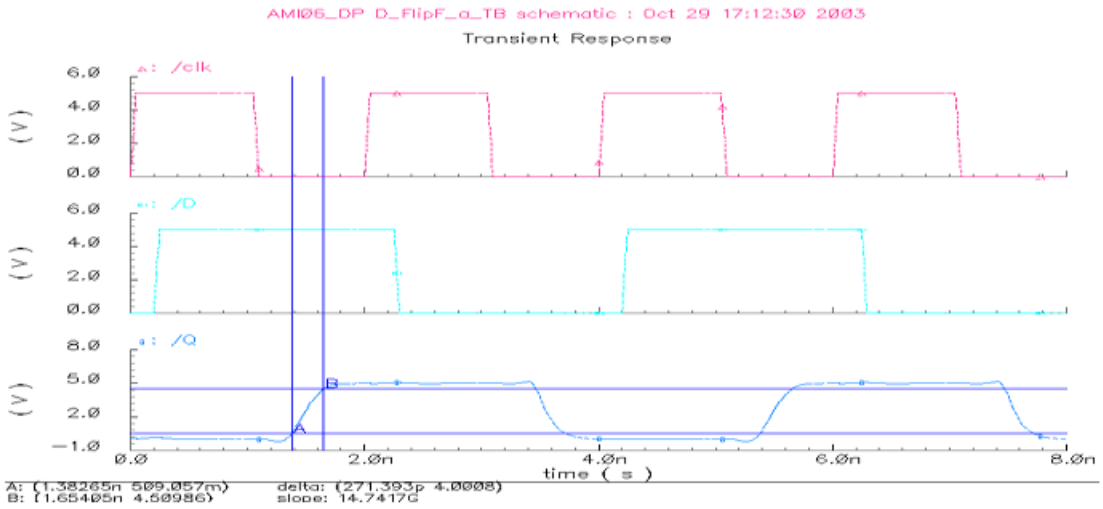


D_FlipF layout



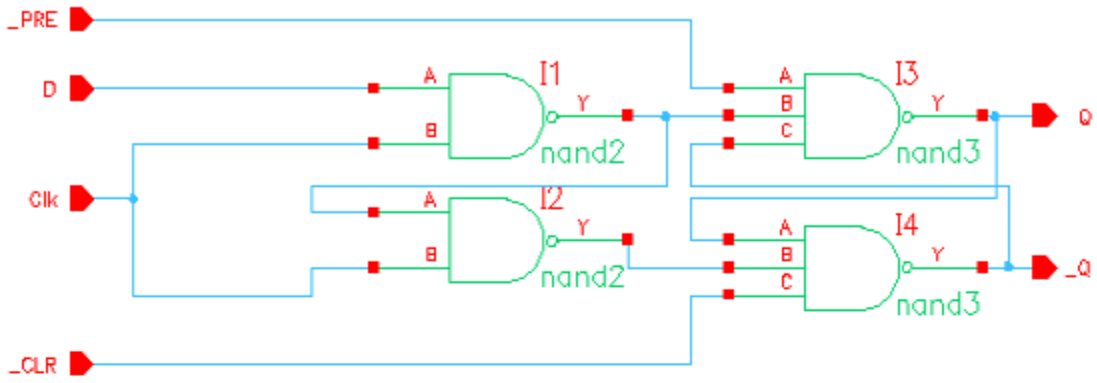
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Nov 12 16:54:07 2003
    completed ....Wed Nov 12 16:54:09 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:02
*********   Summary of rule violation for cell "D_FlipF_a layout"   *********
   Total errors found: 0
```

```
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        39              nets
        8               terminals
        42              pmos
        41              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        27              nets
        8               terminals
        21              pmos
        21              nmos


    Terminal correspondence points
          1     Clk
          2     D
          3     Q
          4     QN
          5     _CLR
          6     _PRE
          7     gnd!
          8     vdd!

The net-lists match.
```

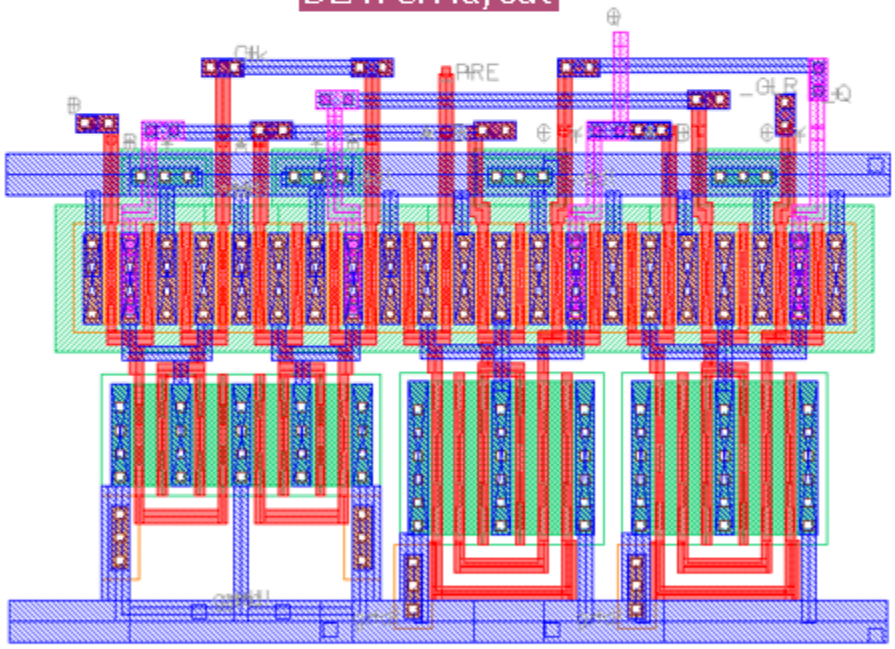


DLATCH

DLATCH schematic

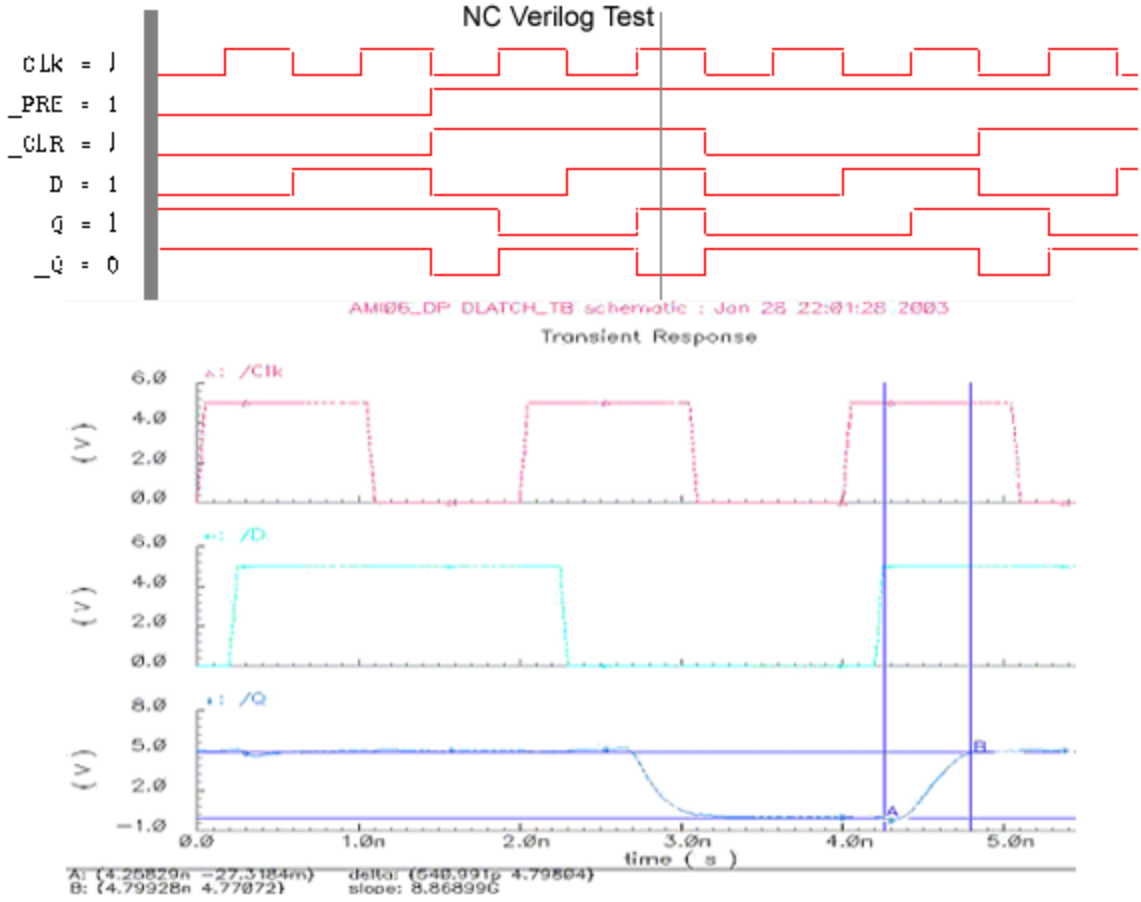


DLATCH layout



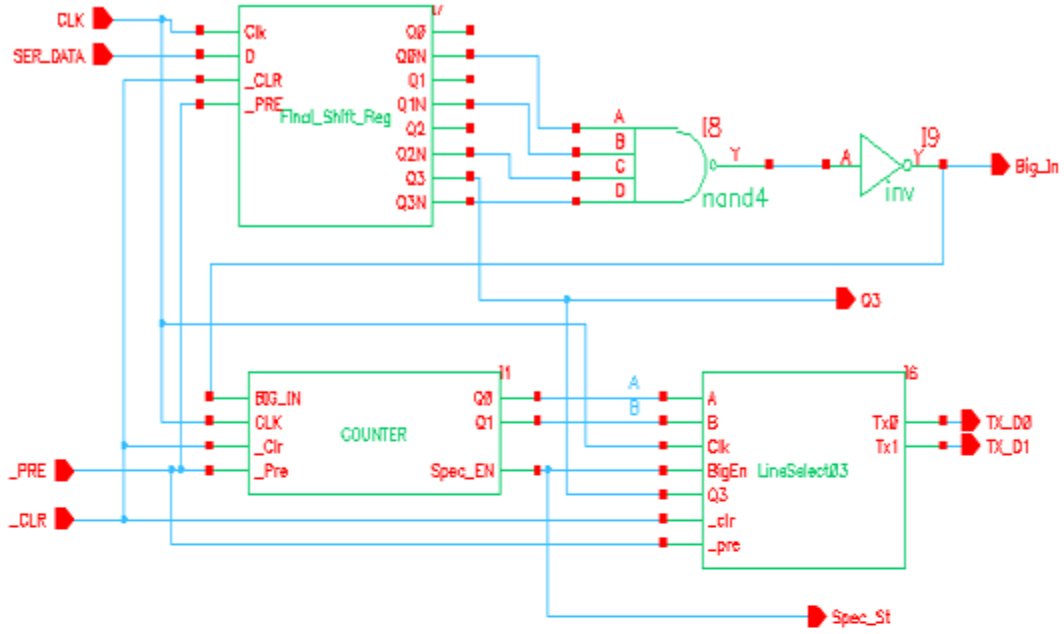

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 03:00:59 2003
    completed ....Wed Oct 29 03:01:00 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "DLATCH layout"   *********
   Total errors found: 0
```

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         22              nets
         8               terminals
         20              pmos
         20              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         16              nets
         8               terminals
         10              pmos
         10              nmos


    Terminal correspondence points
          1     Clk
          2     D
          3     Q
          4     _CLR
          5     _PRE
          6     _Q
          7     gnd!
          8     vdd!

The net-lists match.
```

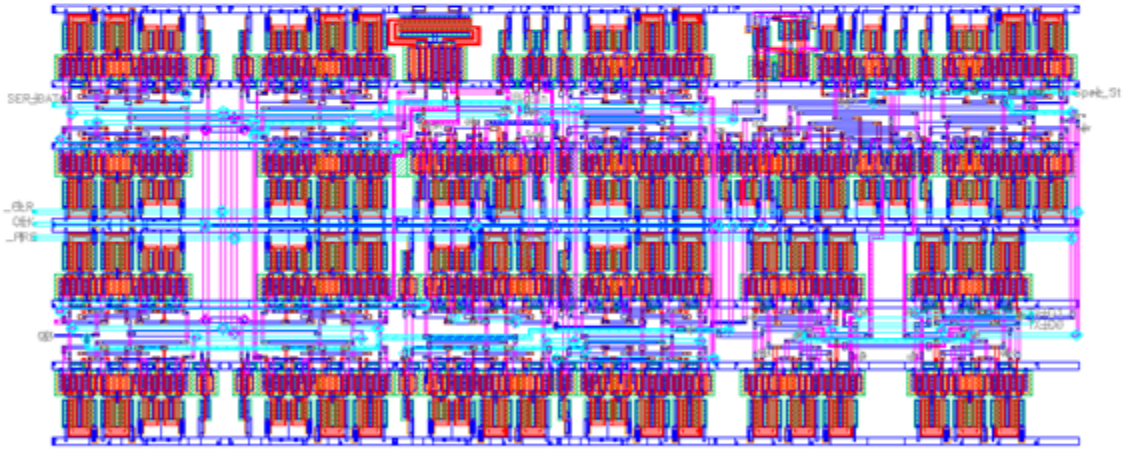


ENCODER

ENCODER schematic

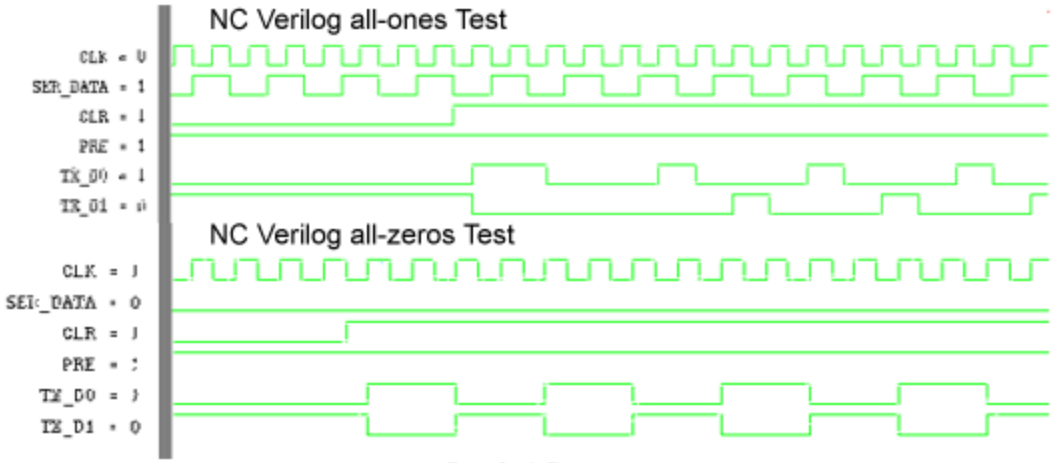


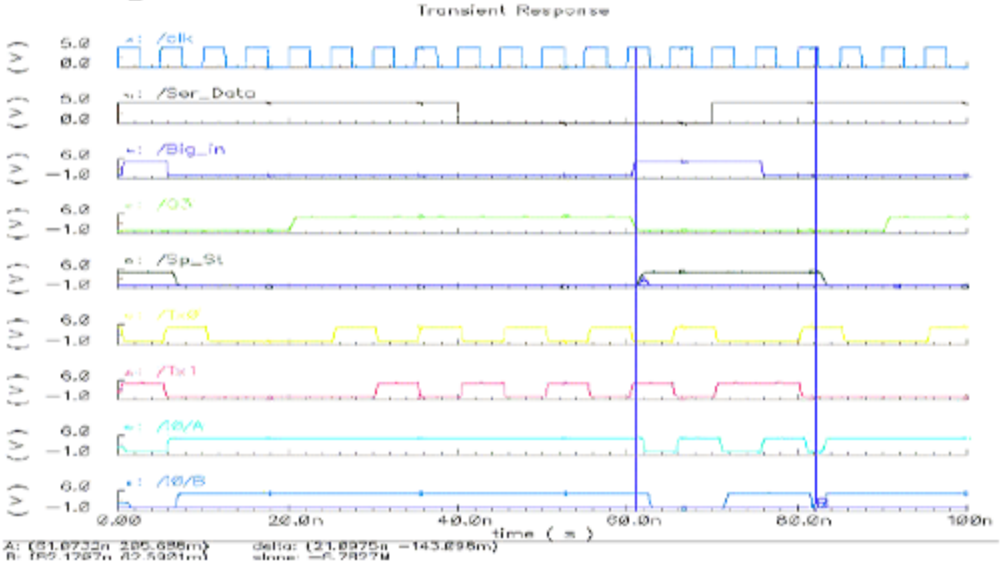
ENCODER layout



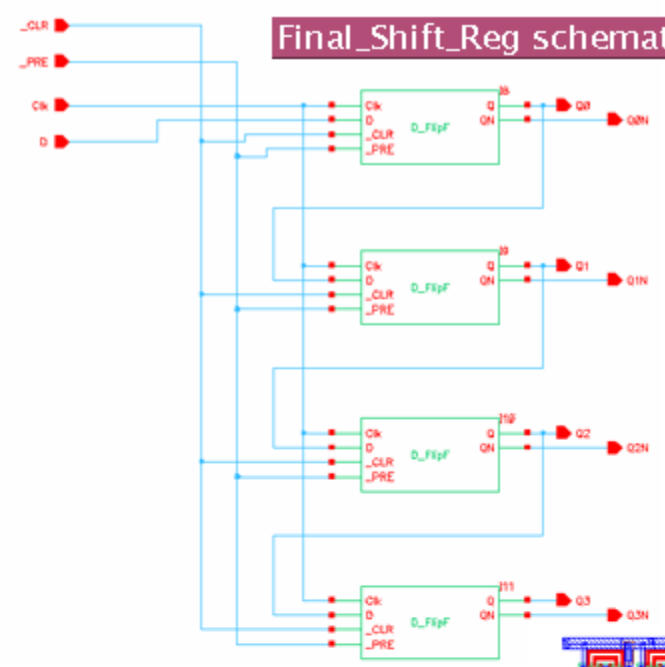
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 19:50:27 2003
    completed ....Wed Oct 29 19:51:15 2003
    CPU TIME = 00:00:05  TOTAL TIME = 00:00:48
*********   Summary of rule violation for cell "ENCODER layout"   *********
   Total errors found: 0
```

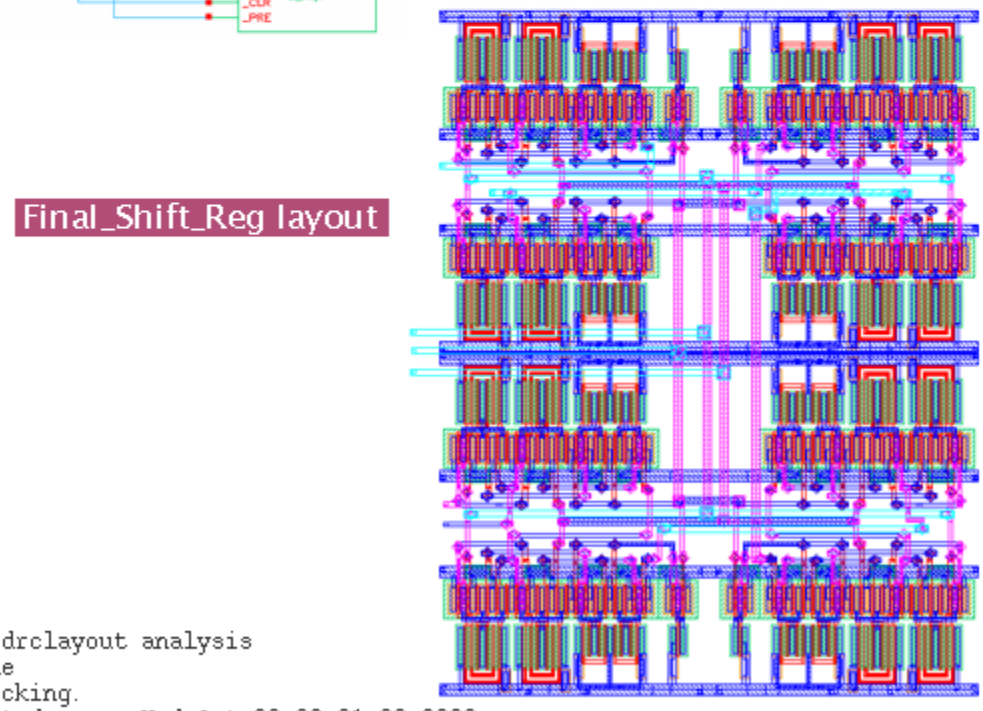
```
Net-list summary for /home/eecad37/cell/LVS/layout/netlist
   count
     391            nets
      11            terminals
     496            pmos
     470            nmos

Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
   count
     258            nets
      11            terminals
     251            pmos
     251            nmos


Terminal correspondence points
       1     Big_In
       2     CLK
       3     Q3
       4     SER_DATA
       5     Spec_St
       6     TX_D0
       7     TX_D1
       8     _CLR
       9     _PRE
      10     gnd!
      11     vdd!

The net-lists match.
```





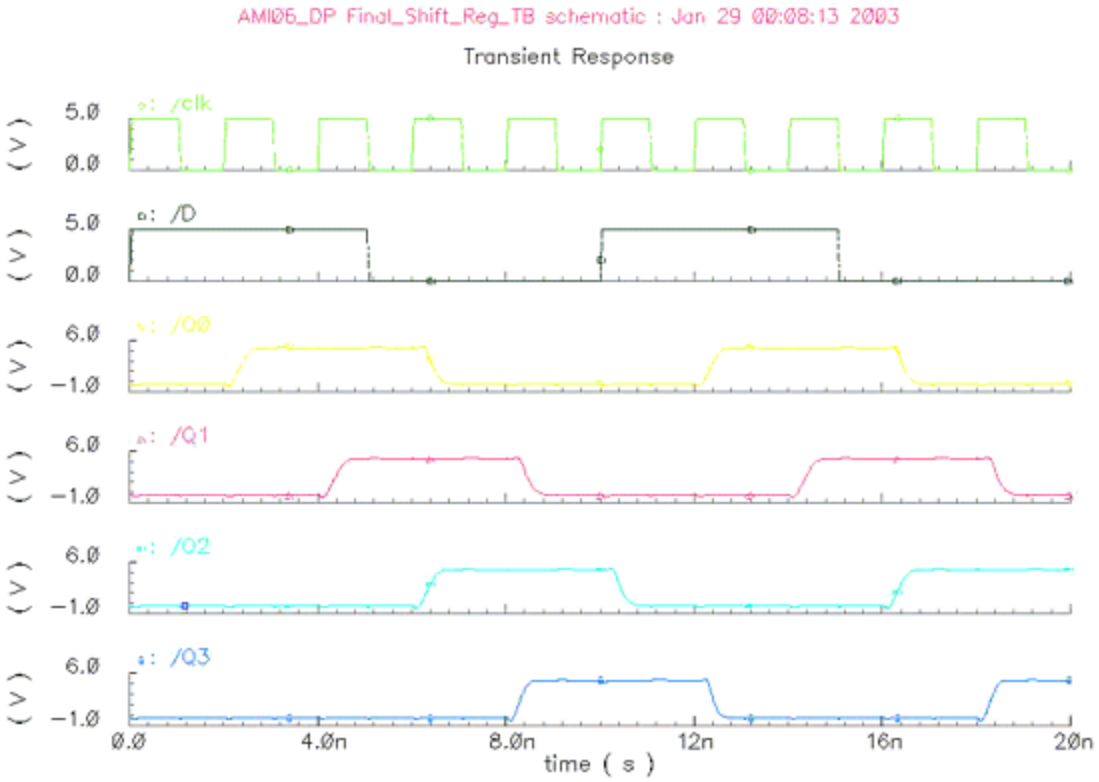
Final_Shift_Reg



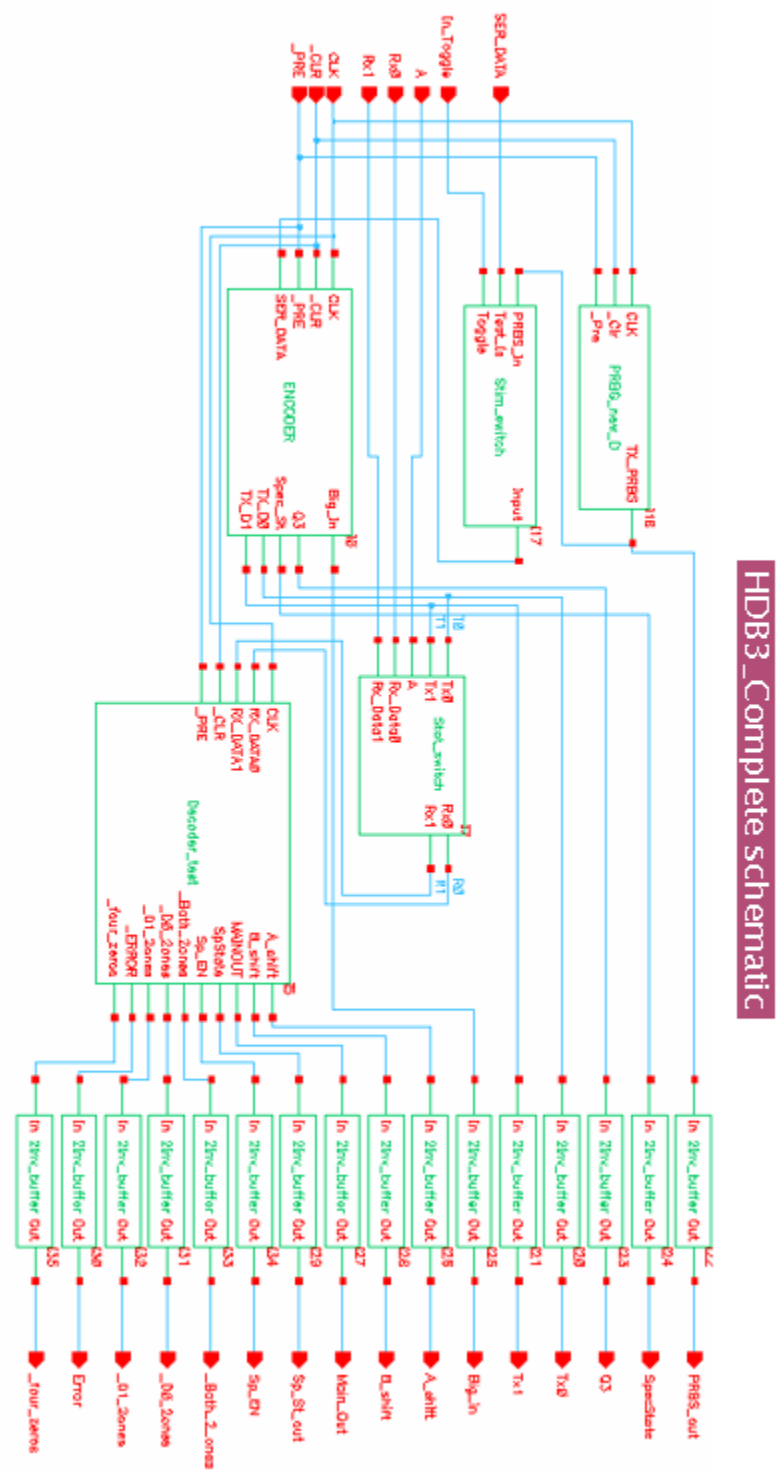



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 20:01:33 2003
    completed ....Wed Oct 29 20:01:43 2003
    CPU TIME = 00:00:01  TOTAL TIME = 00:00:10
*********   Summary of rule violation for cell "Final_Shift_Reg layout"   *******
   Total errors found: 0
```

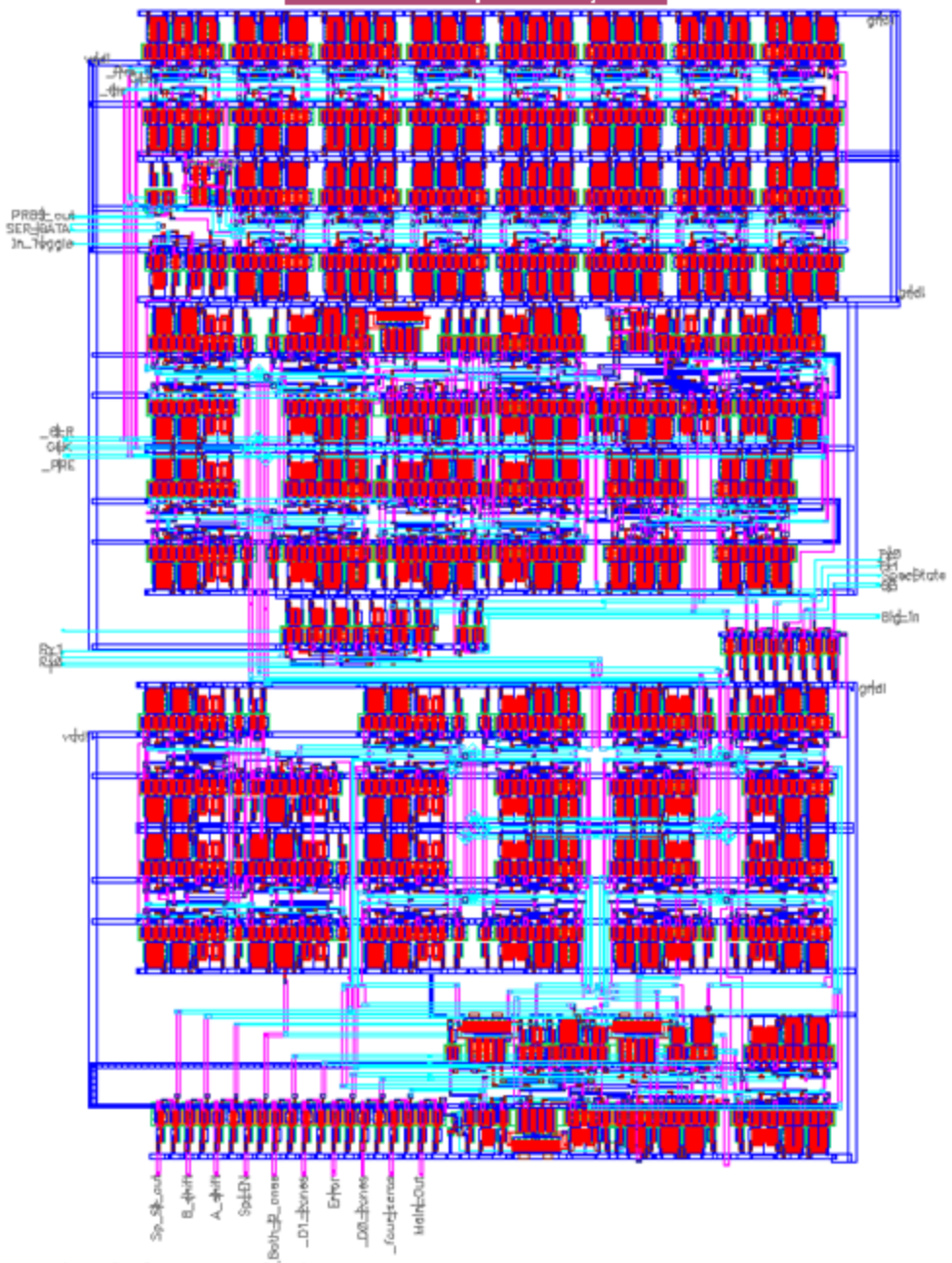
```
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         138             nets
         14              terminals
         168             pnos
         164             nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         90              nets
         14              terminals
         84              pnos
         84              nmos


    Terminal correspondence points
           1     Clk
           2     D
           3     Q0
           4     Q0N
           5     Q1
           6     Q1N
           7     Q2
           8     Q2N
           9     Q3
          10     Q3N
          11     _CLR
          12     _PRE
          13     gnd!
          14     vdd!

The net-lists match.
```



HDB3_Complete



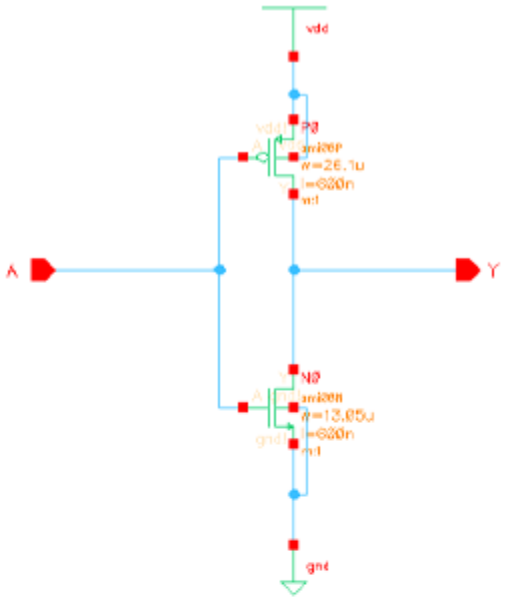
HDB3_Complete layout



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 20:31:51 2003
    completed ....Wed Oct 29 20:33:46 2003
    CPU TIME = 00:00:21  TOTAL TIME = 00:01:55
*********   Summary of rule violation for cell "HDB3_Complete layout"   *********
   Total errors found: 0
```

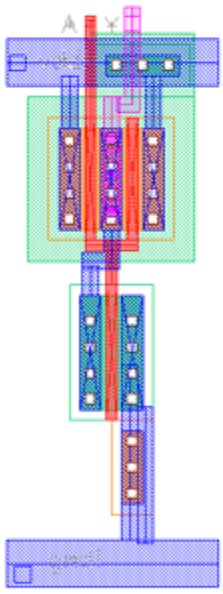
MORE HDB3 Complete images to go here

INV_thin



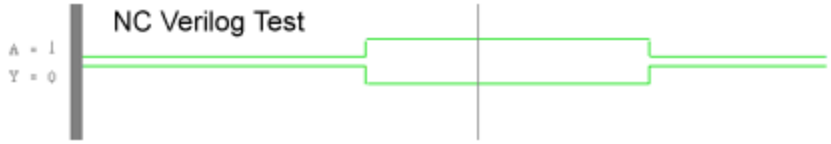
INV_thin schematic

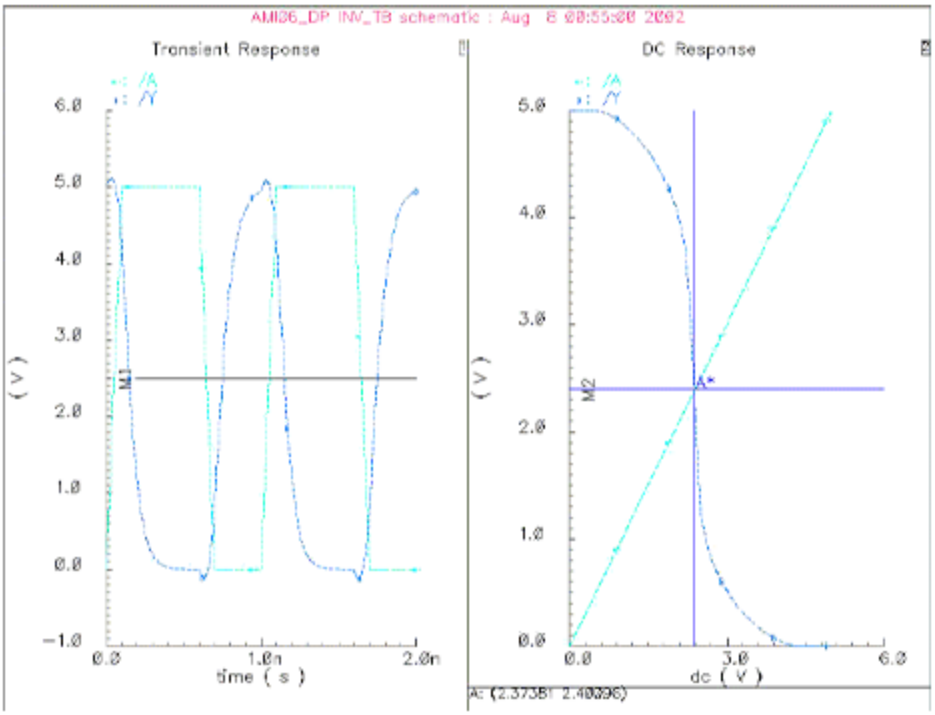
INV_thin Layout



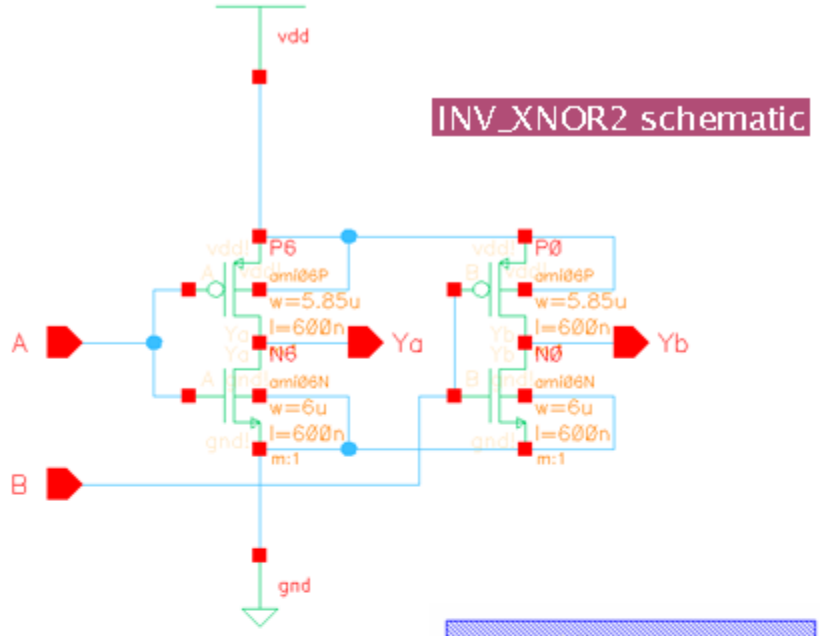
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 21:03:24 2003
    completed ....Wed Oct 29 21:03:25 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "INV_thin layout"   *********
   Total errors found: 0
```

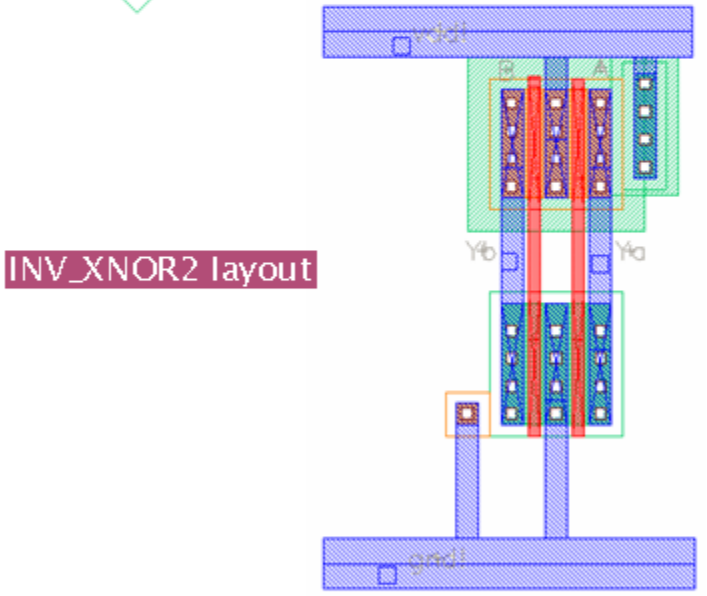
```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         4               nets
         4               terminals
         2               pmos
         1               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         4               nets
         4               terminals
         1               pmos
         1               nmos


    Terminal correspondence points
          1     A
          2     Y
          3     gnd!
          4     vdd!

The net-lists match.
```





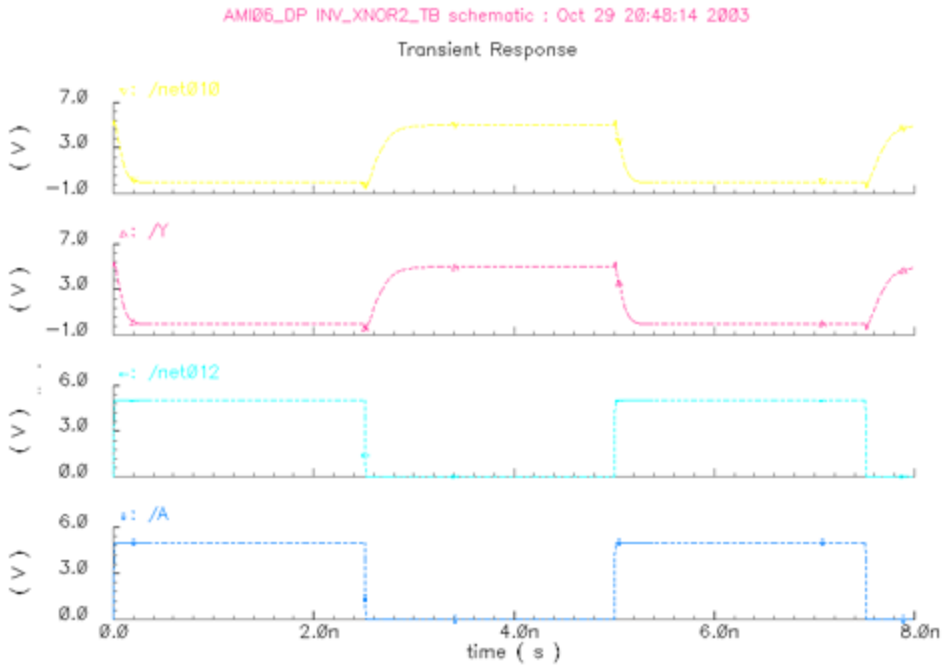
INV_XNOR2





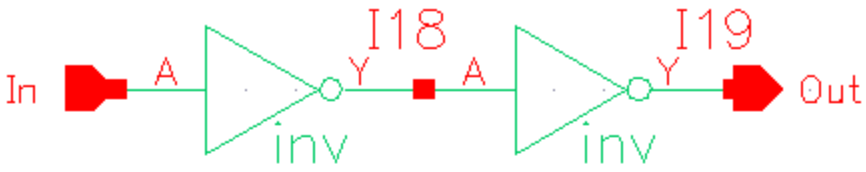

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 20:54:53 2003
    completed ....Wed Oct 29 20:54:54 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "INV_XNOR2 layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
           6               nets
           6               terminals
           2               pmos
           2               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
           6               nets
           6               terminals
           2               pmos
           2               nmos


    Terminal correspondence points
          1     A
          2     B
          3     Ya
          4     Yb
          5     gnd!
          6     vdd!

The net-lists match.
```

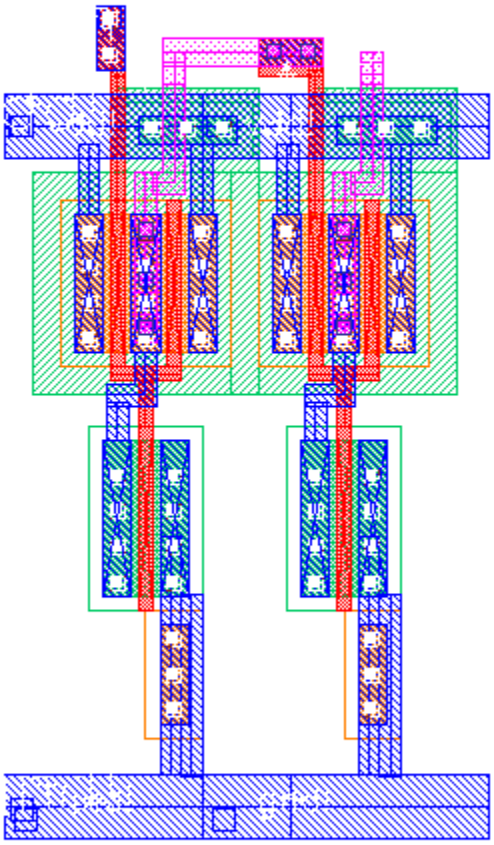


2Inv_buffer

2Inv_buffer schematic

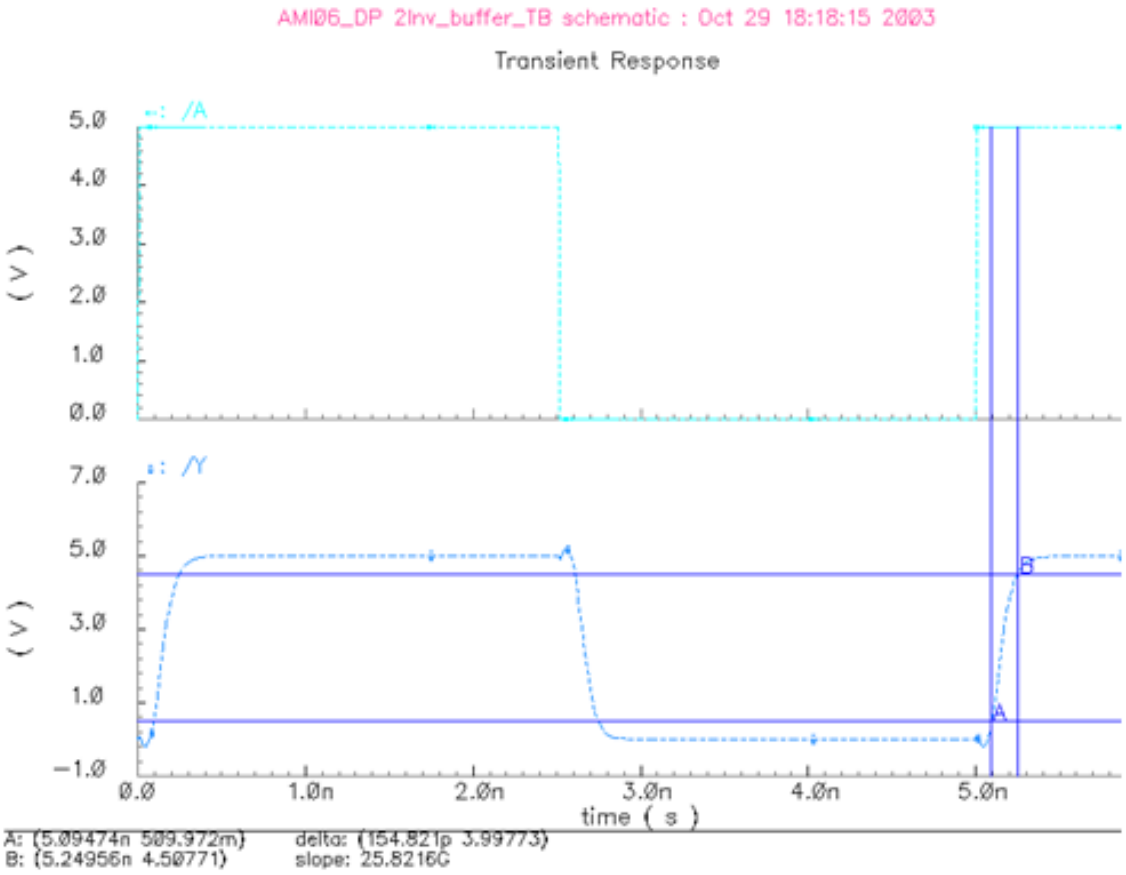


2Inv_buffer layout

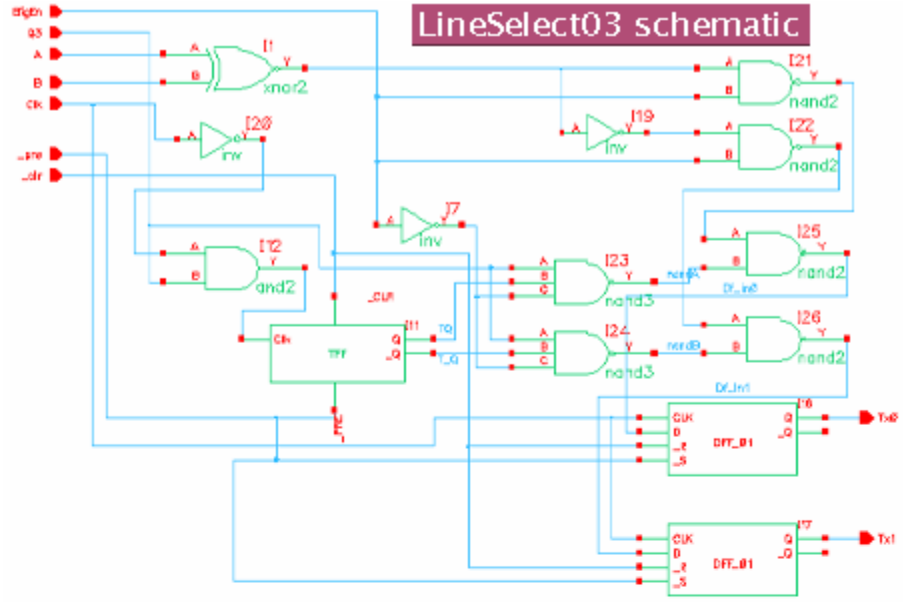


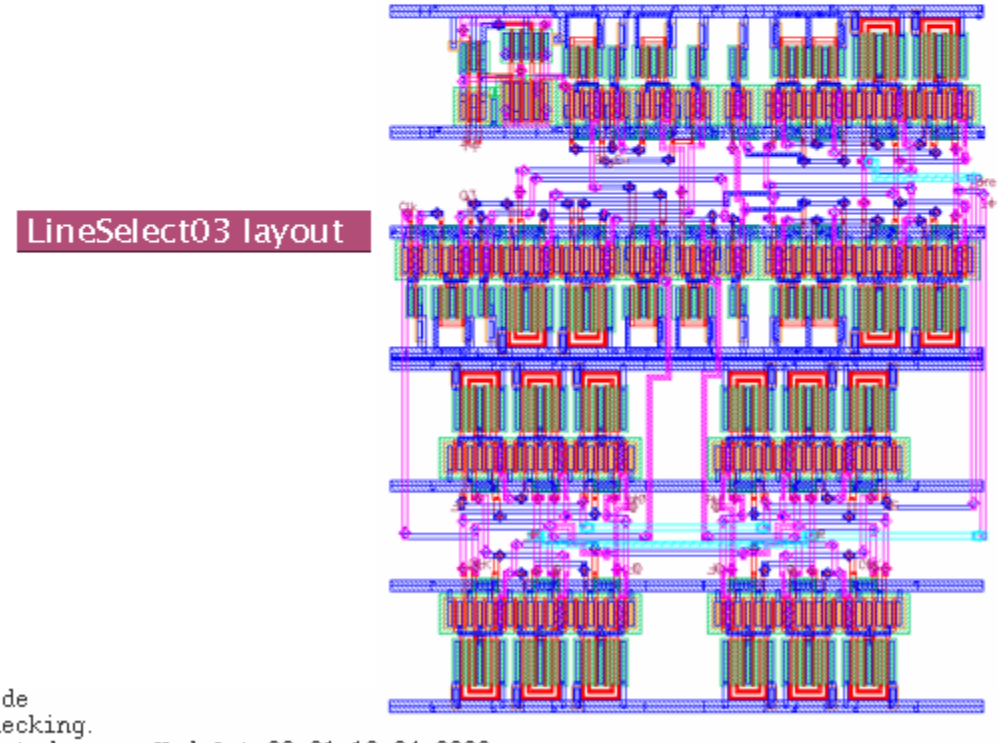
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Wed Oct 29 18:12:35 2003
    completed ....Wed Oct 29 18:12:36 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "2Inv_buffer layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
           5               nets
           4               terminals
           4               pmos
           2               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
           5               nets
           4               terminals
           2               pmos
           2               nmos


    Terminal correspondence points
          1     In
          2     Out
          3     gnd!
          4     vdd!

The net-lists match.
```



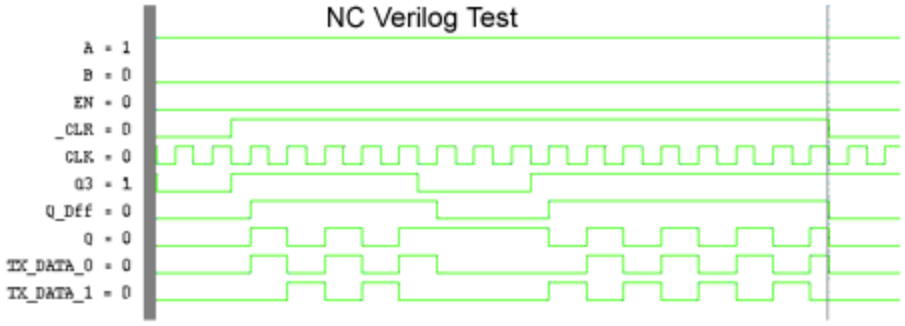
Line_Select03

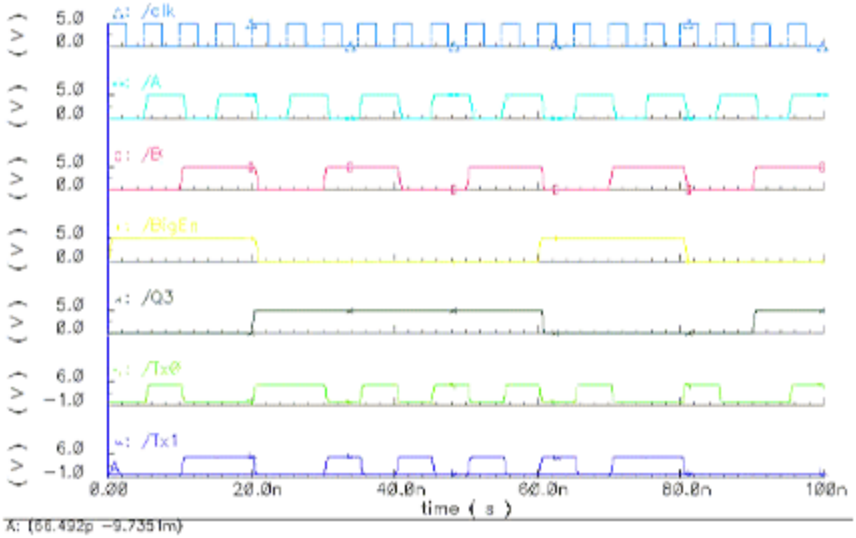




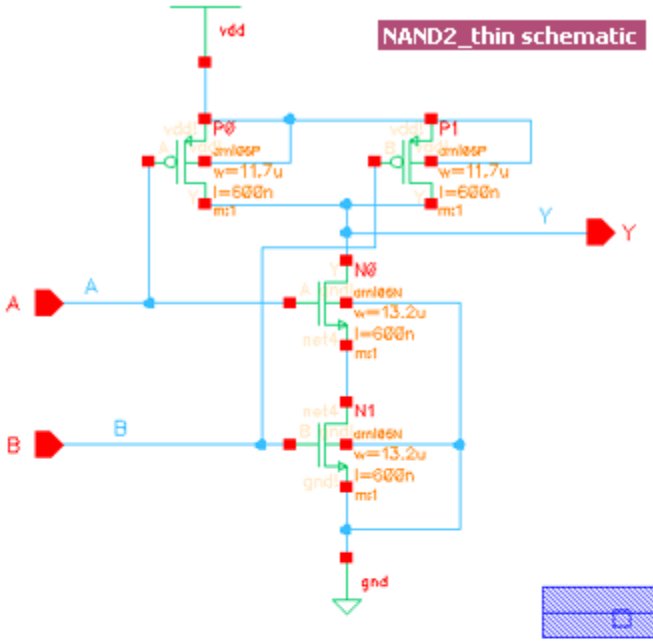

```
Flat mode
Full checking.
DRC started.......Wed Oct 29 21:18:34 2003
    completed ....Wed Oct 29 21:18:40 2003
    CPU TIME = 00:00:01  TOTAL TIME = 00:00:06
*********   Summary of rule violation for cell "LineSelect03 layout"   *********
   Total errors found: 0
```

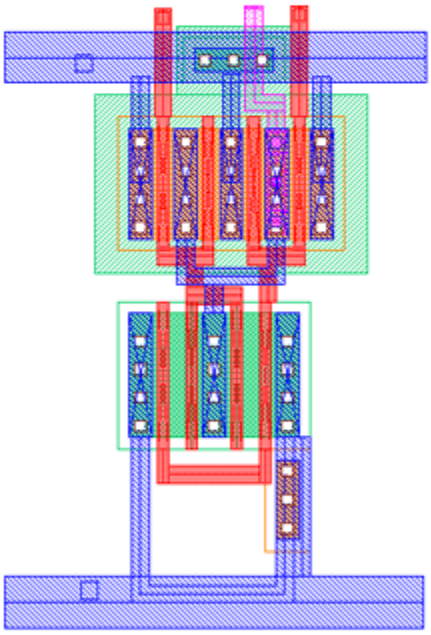
```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         139            nets
          11            terminals
         162            pmos
         156            nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
          94            nets
          11            terminals
          84            pmos
          84            nmos


    Terminal correspondence points
           1     A
           2     B
           3     BigEn
           4     Clk
           5     Q3
           6     Tx0
           7     Tx1
           8     _clr
           9     _pre
          10     gnd!
          11     vdd!

The net-lists match.
```





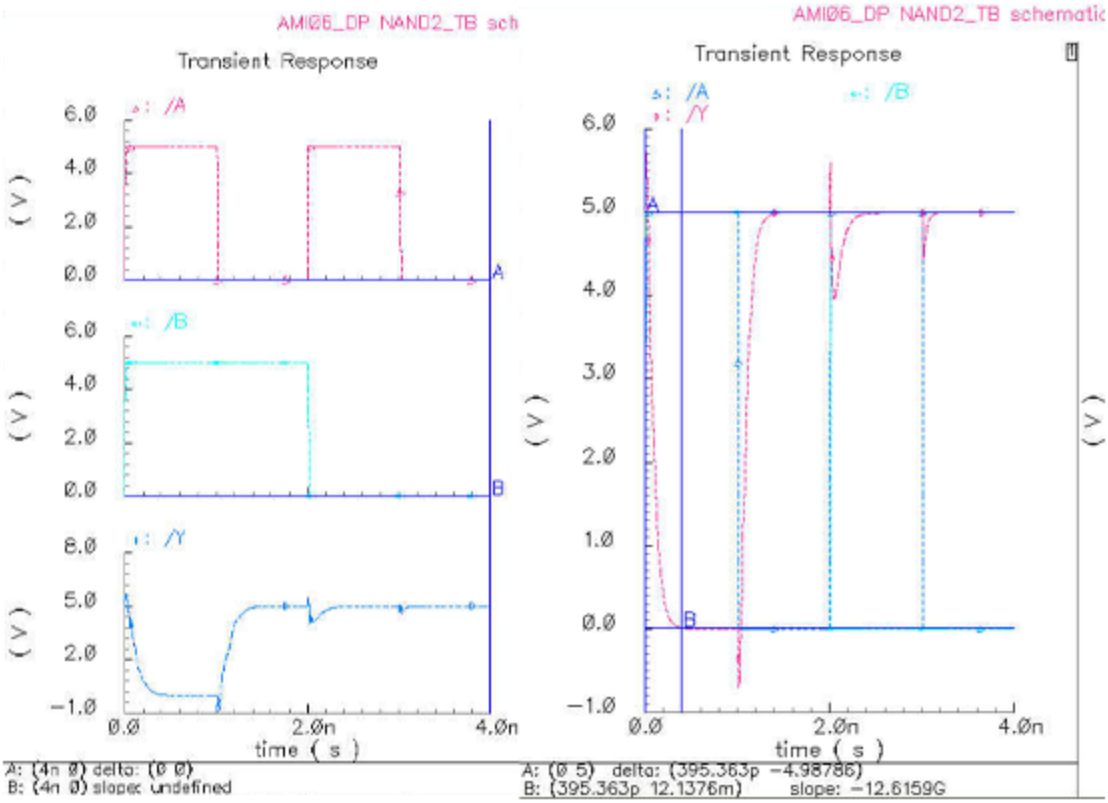
NAND2_thin



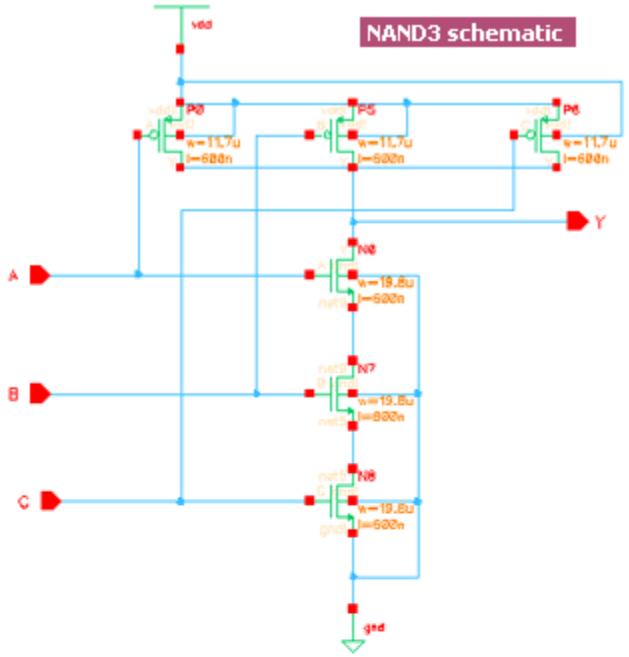



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 15:23:57 2003
    completed ....Tue Nov  4 15:24:00 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:03
*********   Summary of rule violation for cell "NAND2_thin layout"   *********
   Total errors found: 0
```

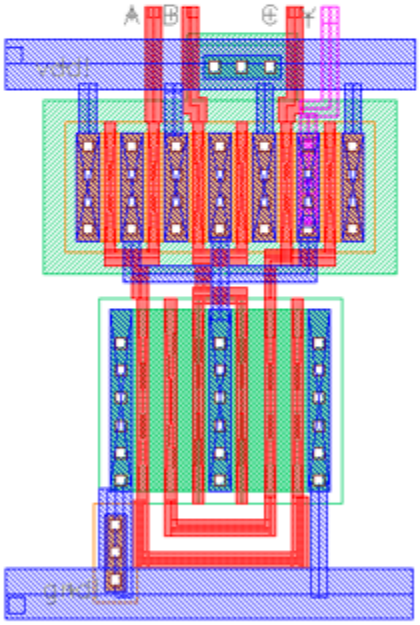
```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         7              nets
         5              terminals
         4              pmos
         4              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         6              nets
         5              terminals
         2              pmos
         2              nmos


    Terminal correspondence points
          1     A
          2     B
          3     Y
          4     gnd!
          5     vdd!

The net-lists match.
```



NAND3

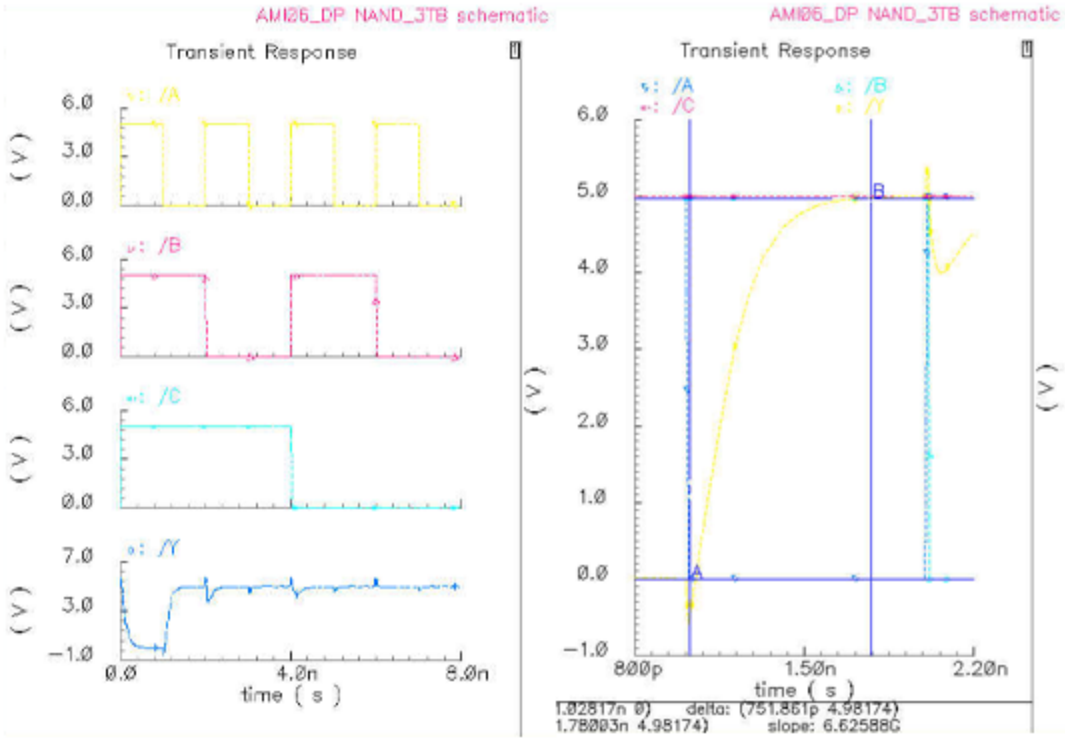


NAND3 layout

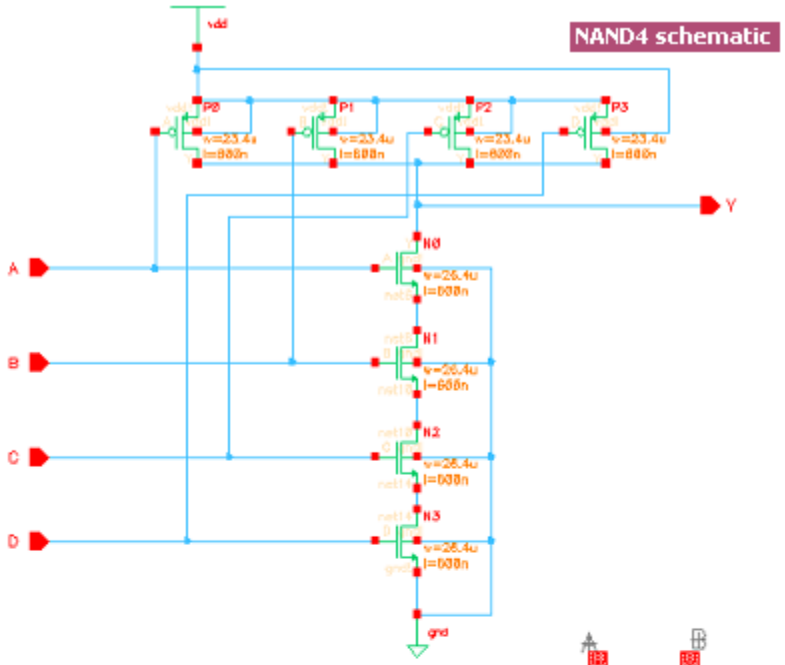


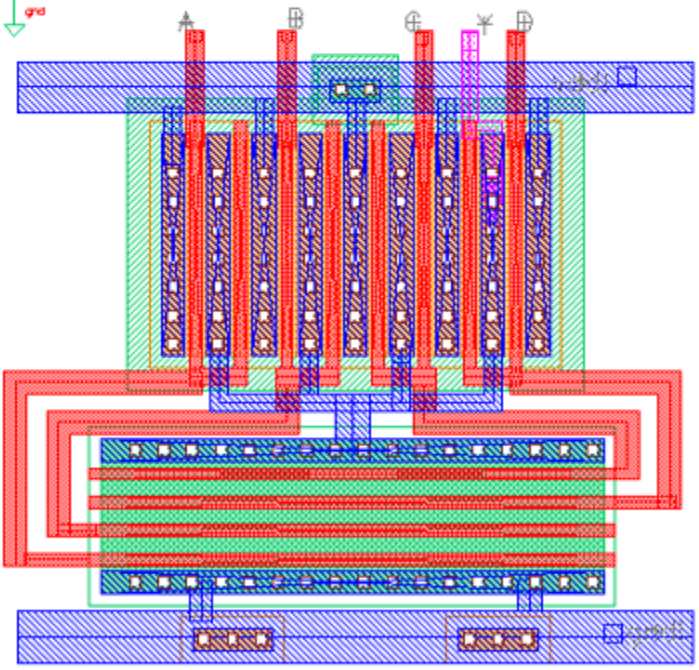

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 15:41:15 2003
    completed ....Tue Nov  4 15:41:15 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violation for cell "NAND3 layout"   *********
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        10              nets
        6               terminals
        6               pmos
        6               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        8               nets
        6               terminals
        3               pmos
        3               nmos


    Terminal correspondence points
          1     A
          2     B
          3     C
          4     Y
          5     gnd!
          6     vdd!

The net-lists match.
```



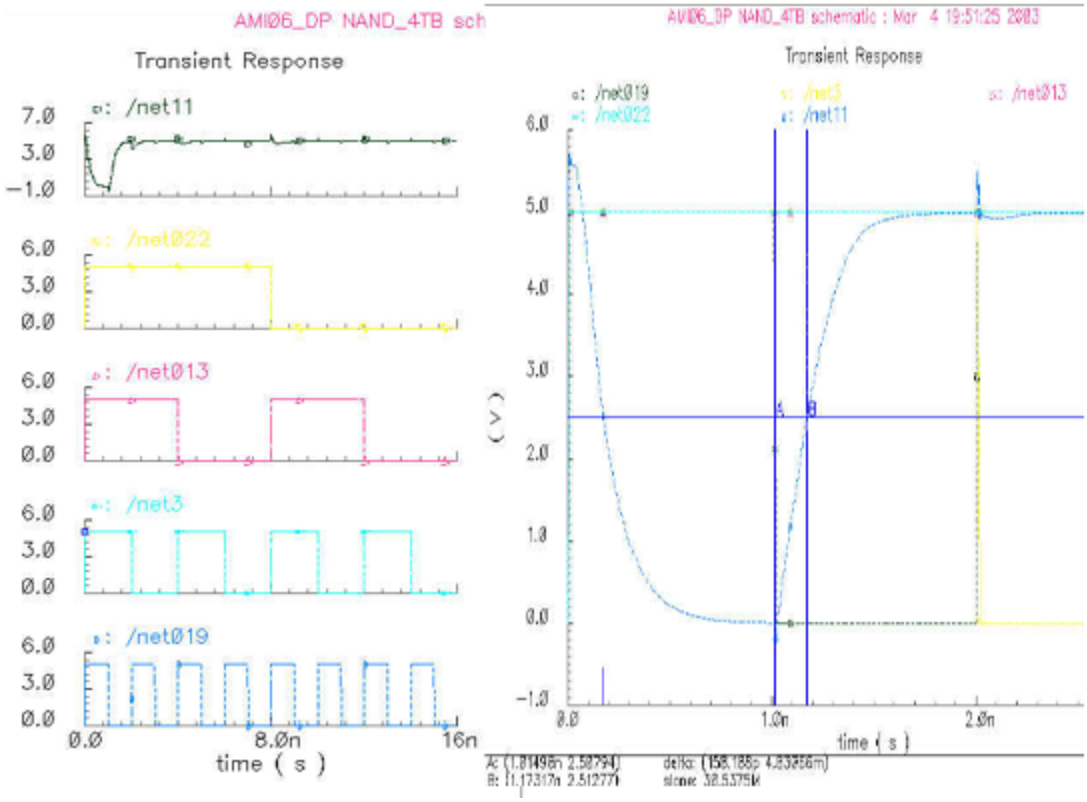
NAND4





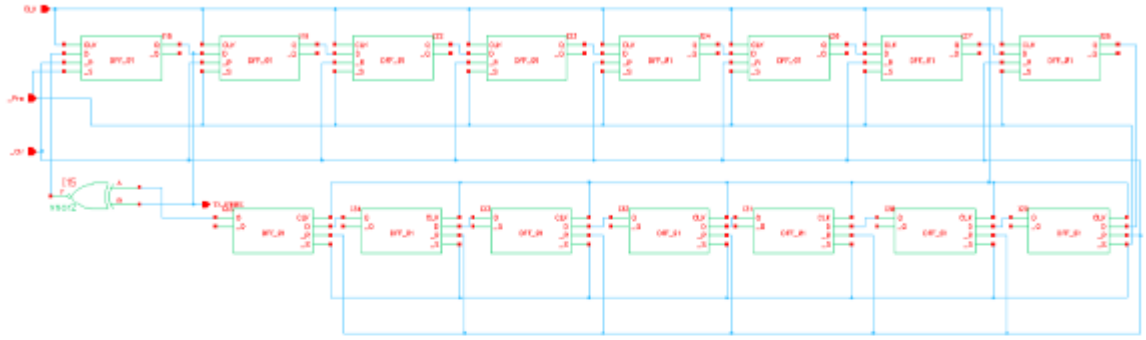

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 15:53:10 2003
    completed ....Tue Nov  4 15:53:10 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violation for cell "NAND4 layout"   *******
   Total errors found: 0
```

```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        10              nets
        7               terminals
        8               pmos
        4               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        10              nets
        7               terminals
        4               pmos
        4               nmos


    Terminal correspondence points
          1     A
          2     B
          3     C
          4     D
          5     Y
          6     gnd!
          7     vdd!

The net-lists match.
```

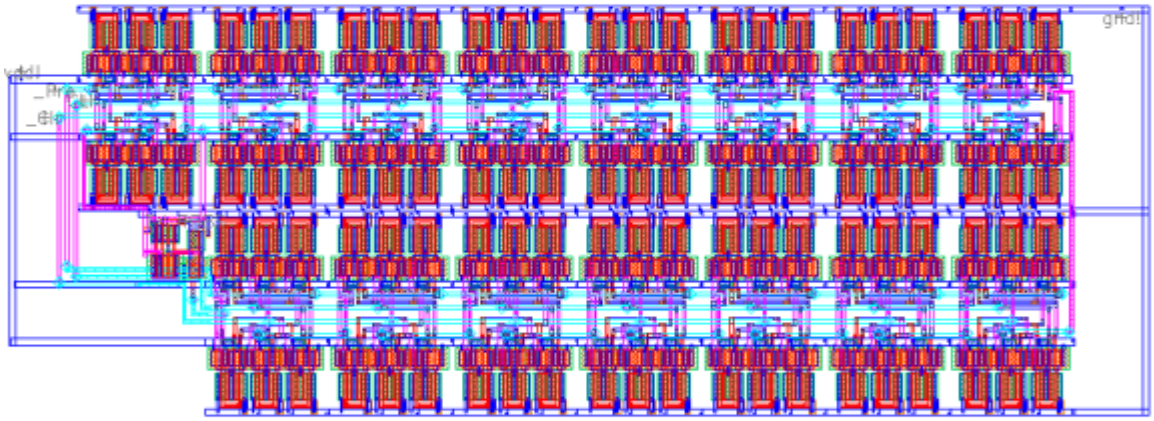


PRBG_new_D

PRBG_new_D schematic

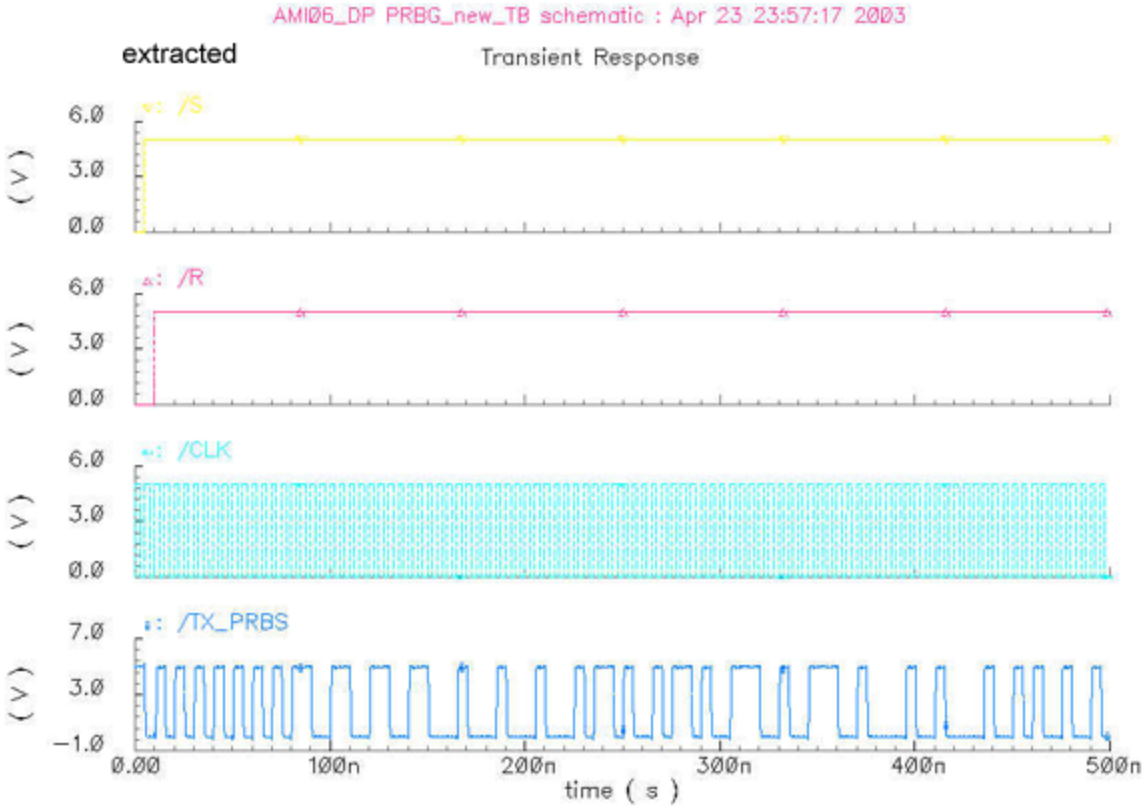


PRBG_new_D layout

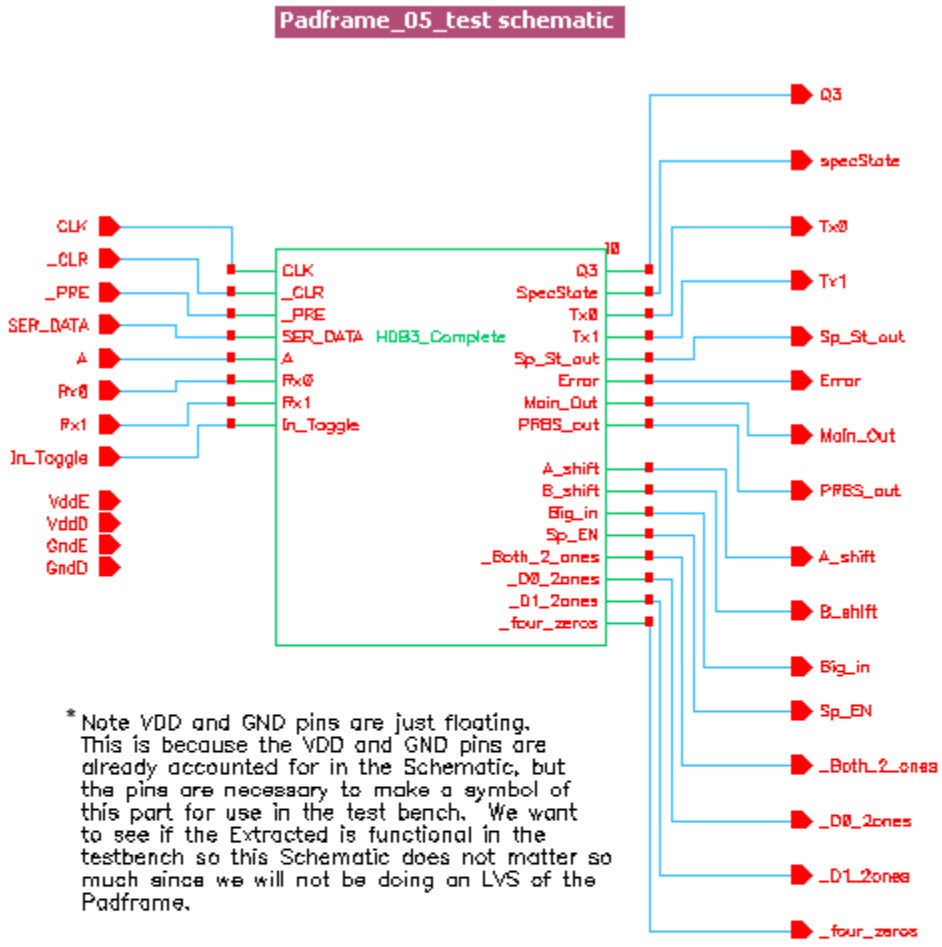


```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 16:21:18 2003
    completed ....Tue Nov  4 16:21:40 2003
    CPU TIME = 00:00:07  TOTAL TIME = 00:00:22
*********   Summary of rule violation for cell "PRBG_new_D layout"   *********
   Total errors found: 0
```

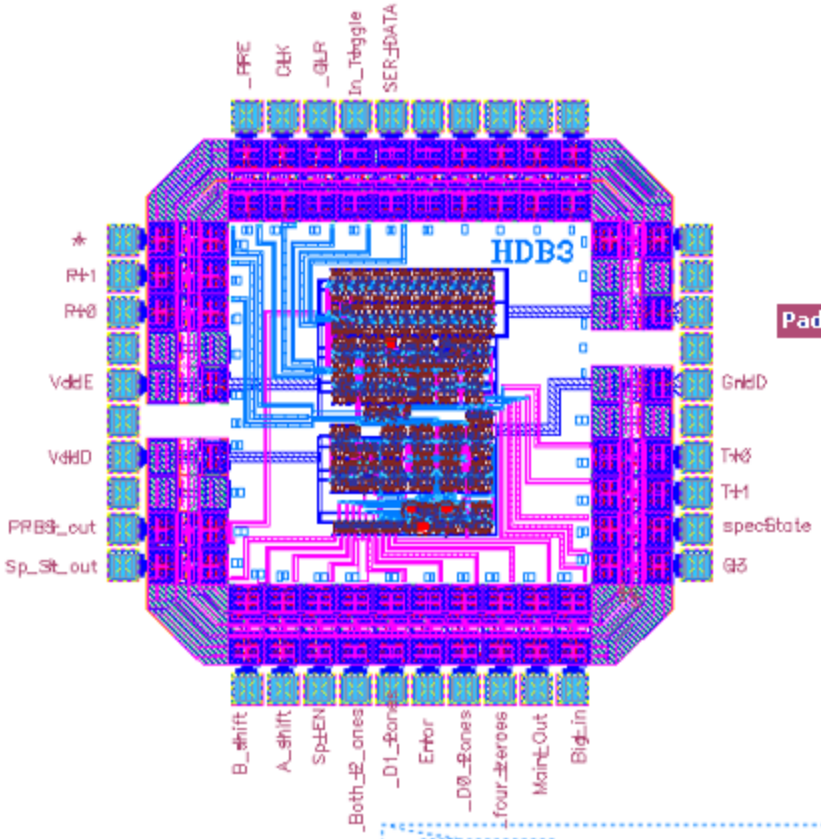
```
@(#)$CDS: LVS version 4.4.6 06/01/2001 20:24 (cds11612) $

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         462             nets
         6               terminals
         546             pmos
         546             nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         282             nets
         6               terminals
         276             pmos
         276             nmos


    Terminal correspondence points
           1     CLK
           2     TX_PRBS
           3     _Clr
           4     _Pre
           5     gnd!
           6     vdd!

The net-lists match.
```

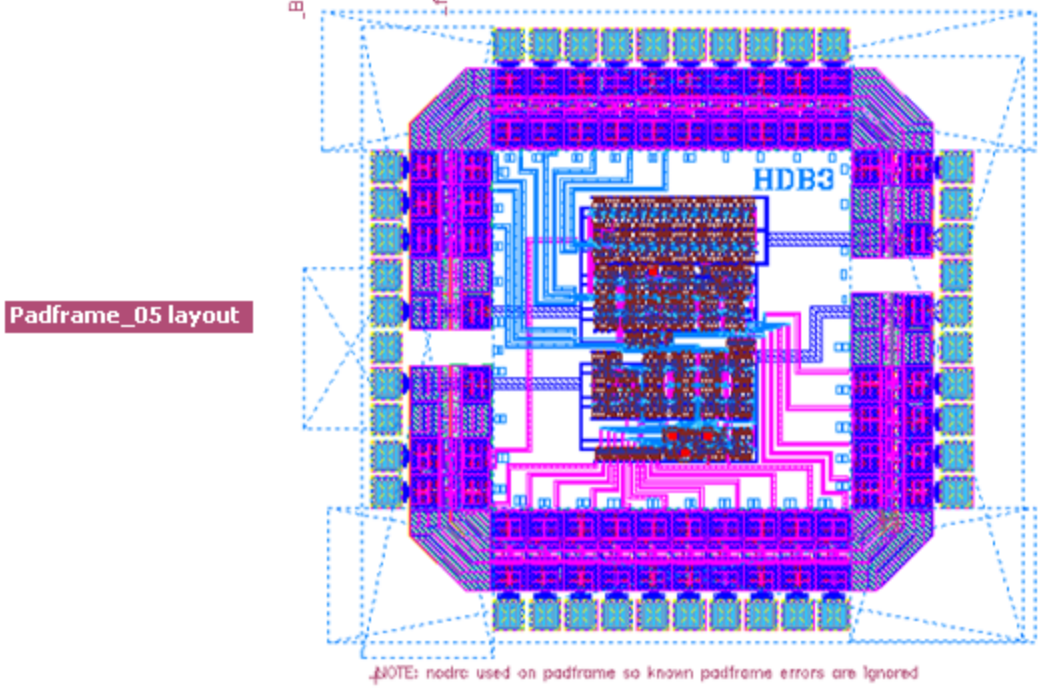


Padframe_05_test



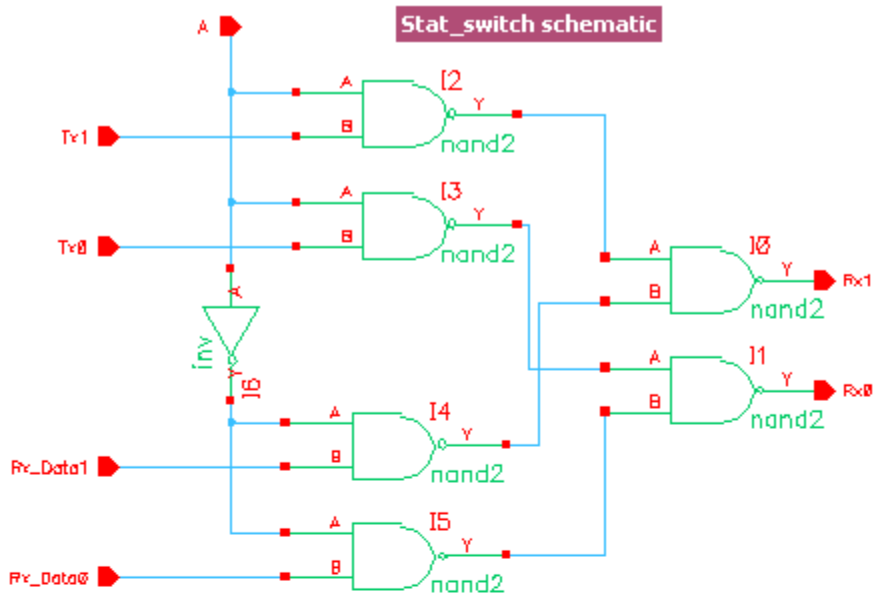

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 18:02:34 2003
    completed ....Tue Nov  4 18:04:51 2003
    CPU TIME = 00:00:46  TOTAL TIME = 00:02:17
*********   Summary of rule violation for cell "Padframe_05 layout"   ********
   Total errors found: 0
```



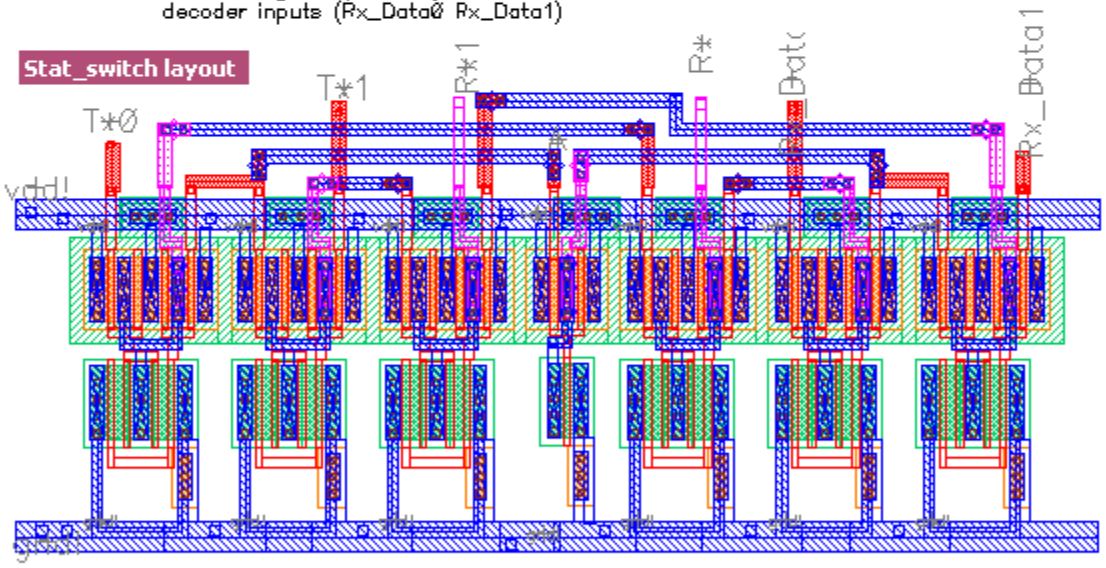


NOTE: nodrc used on padframe so known padframe errors are ignored

Stat_Switch

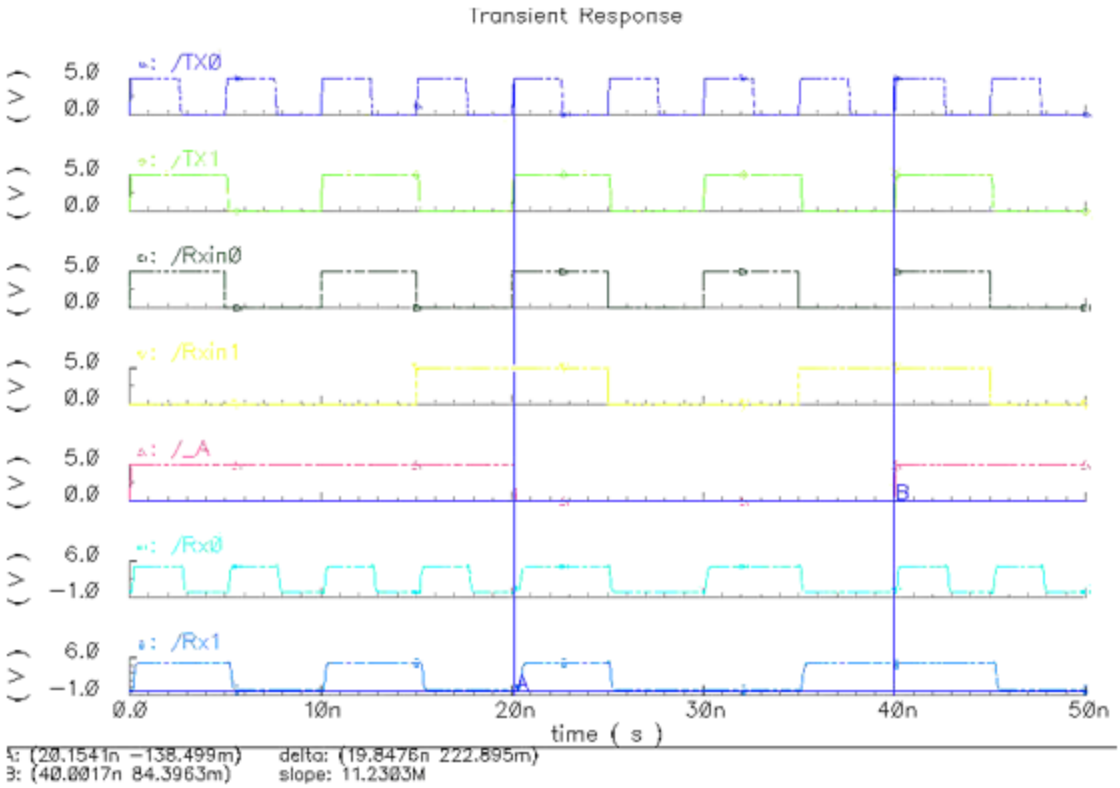


Stat_switch allows switching between encoder signals (Tx0 Tx1) and manual decoder inputs (Rx_Data0 Rx_Data1)

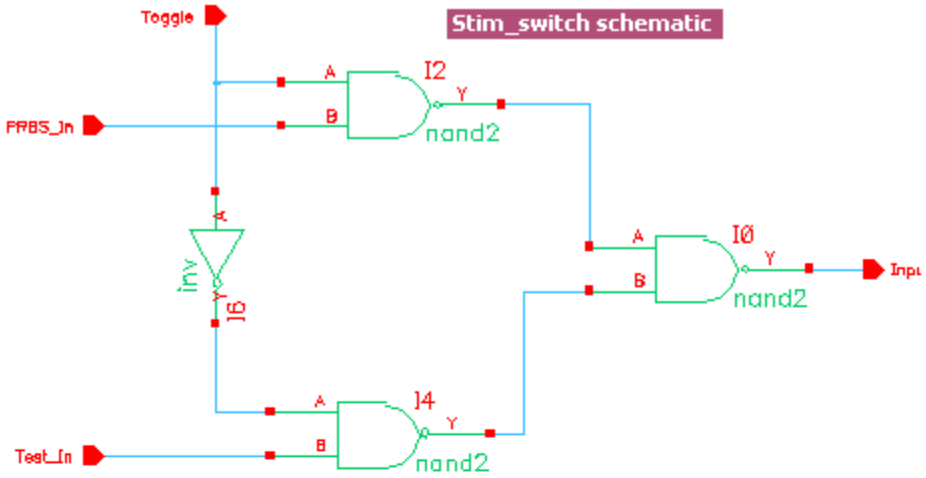



```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 18:54:21 2003
    completed ....Tue Nov  4 18:54:25 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:04
*********   Summary of rule violation for cell "Stat_switch layout"   ****
   Total errors found: 0
```

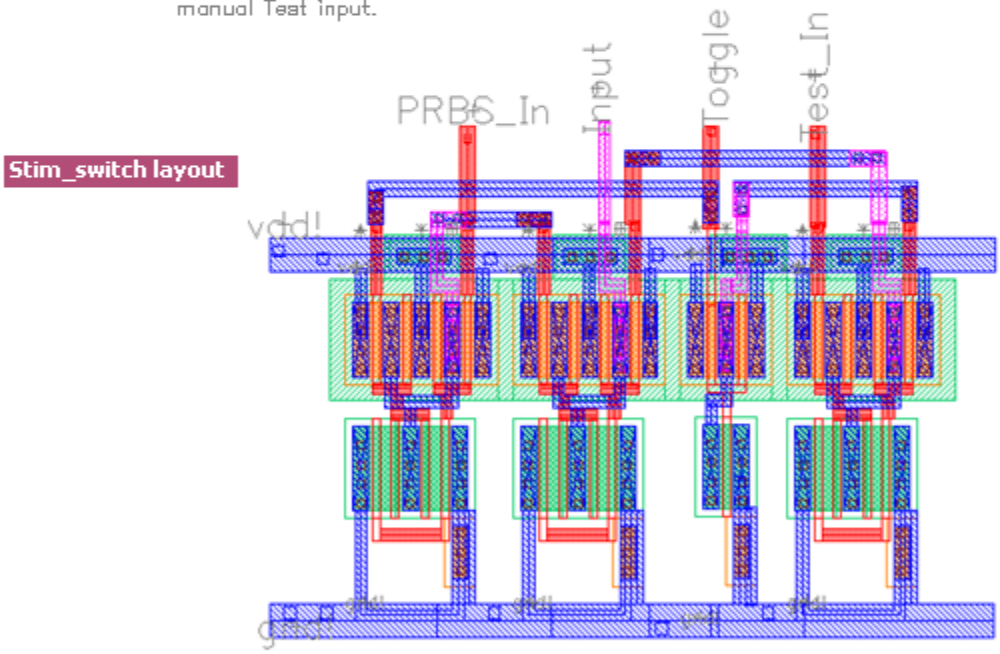
```
Like matching is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         26              nets
         9               terminals
         26              pmos
         25              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         20              nets
         9               terminals
         13              pmos
         13              nmos


    Terminal correspondence points
           1     A
           2     Rx0
           3     Rx1
           4     Rx_Data0
           5     Rx_Data1
           6     Tx0
           7     Tx1
           8     gnd!
           9     vdd!

The net-lists match.
```



Stim_Switch

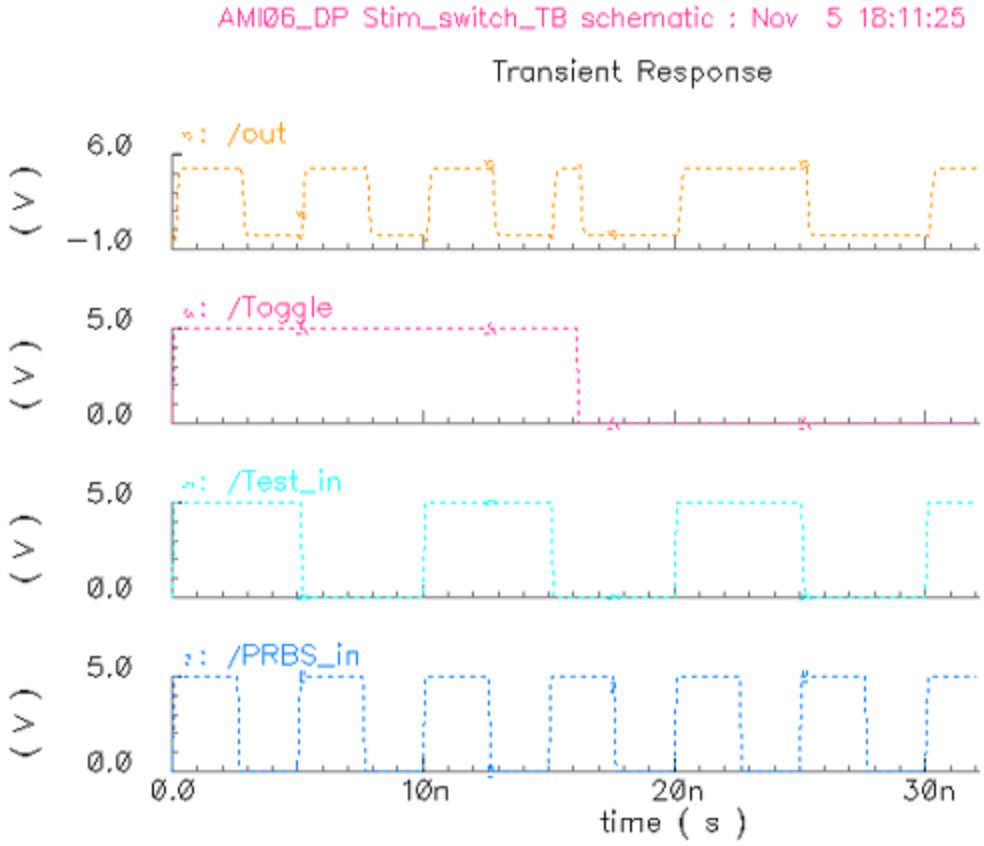


Stim_switch allows switching between the self generated PRBS input and the manual Test input.



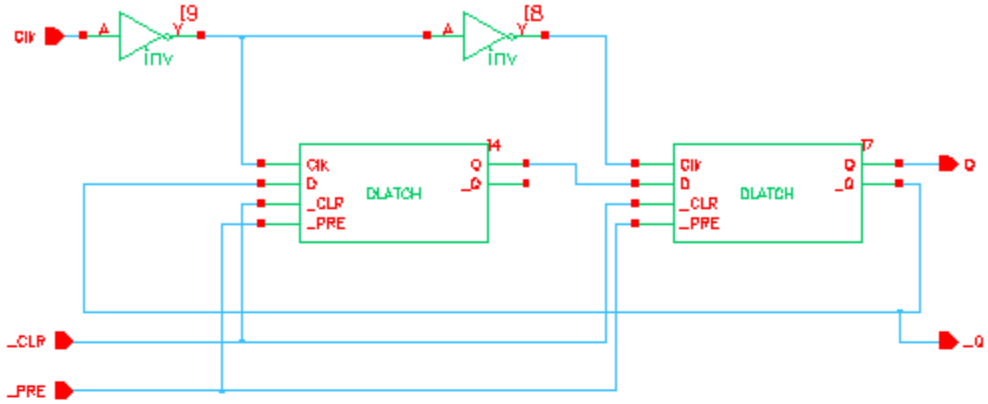

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 19:06:21 2003
    completed ....Tue Nov  4 19:06:22 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "Stim_switch layout"   *********
   Total errors found: 0
```

```
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
        15              nets
        6               terminals
        14              pmos
        13              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
        12              nets
        6               terminals
        7               pmos
        7               nmos


    Terminal correspondence points
          1     Input
          2     PRBS_In
          3     Test_In
          4     Toggle
          5     gnd!
          6     vdd!

The net-lists match.
```

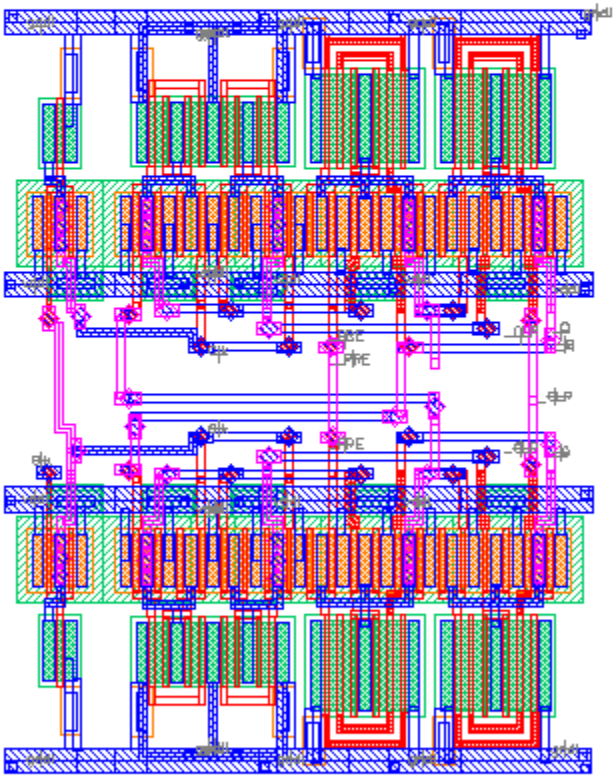


TFF

TFF schematic

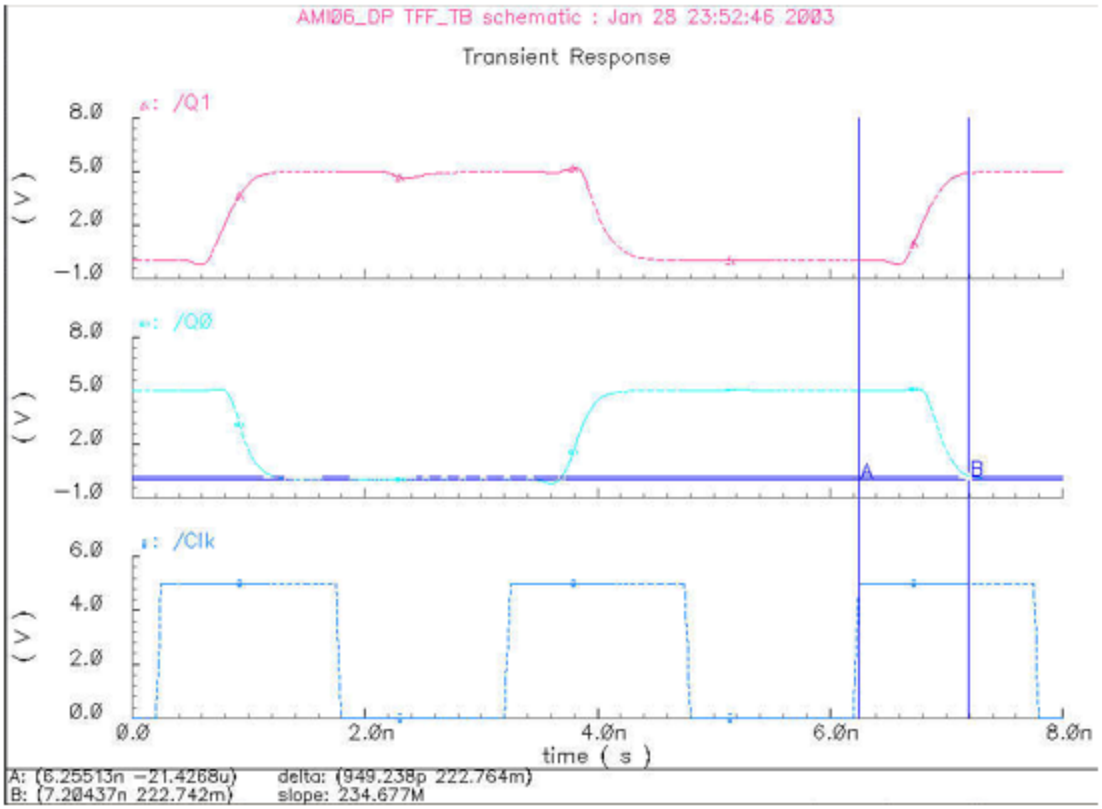


TFF layout

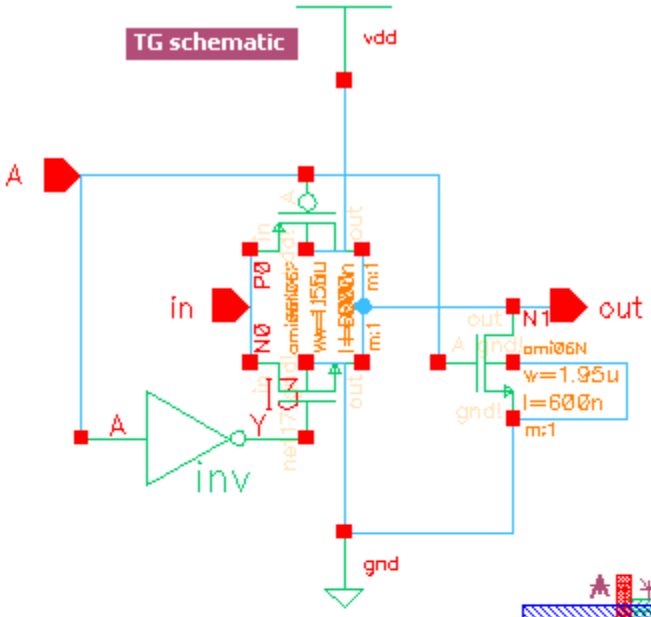


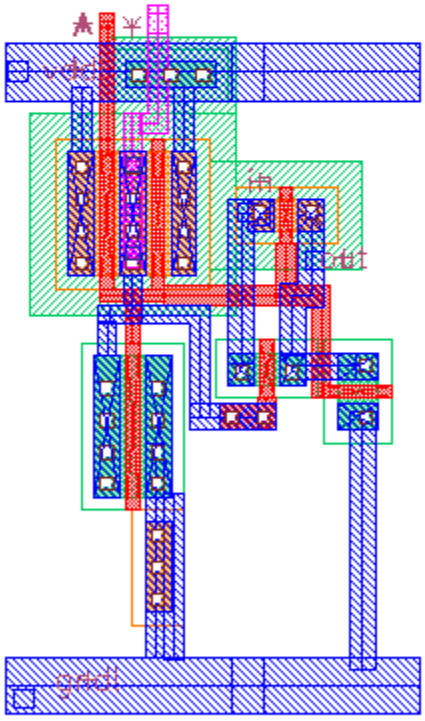
```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 19:22:44 2003
    completed ....Tue Nov  4 19:22:45 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:01
*********   Summary of rule violation for cell "TFF layout"   ******
   Total errors found: 0
```

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         39              nets
         7               terminals
         44              pmos
         42              nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         27              nets
         7               terminals
         22              pmos
         22              nmos


    Terminal correspondence points
          1     Clk
          2     Q
          3     _CLR
          4     _PRE
          5     _Q
          6     gnd!
          7     vdd!

The net-lists match.
```



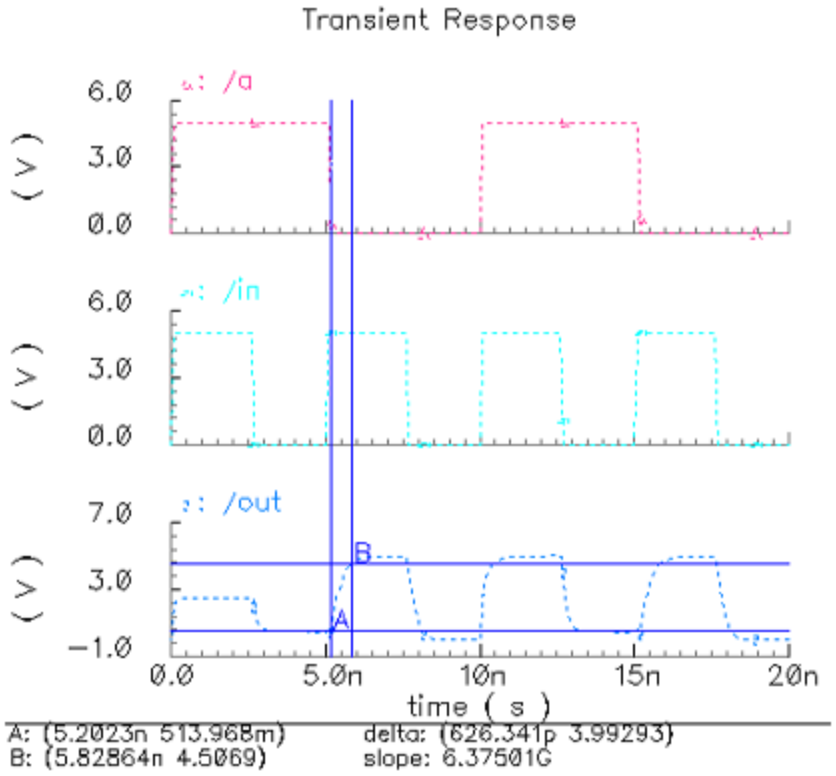
TG



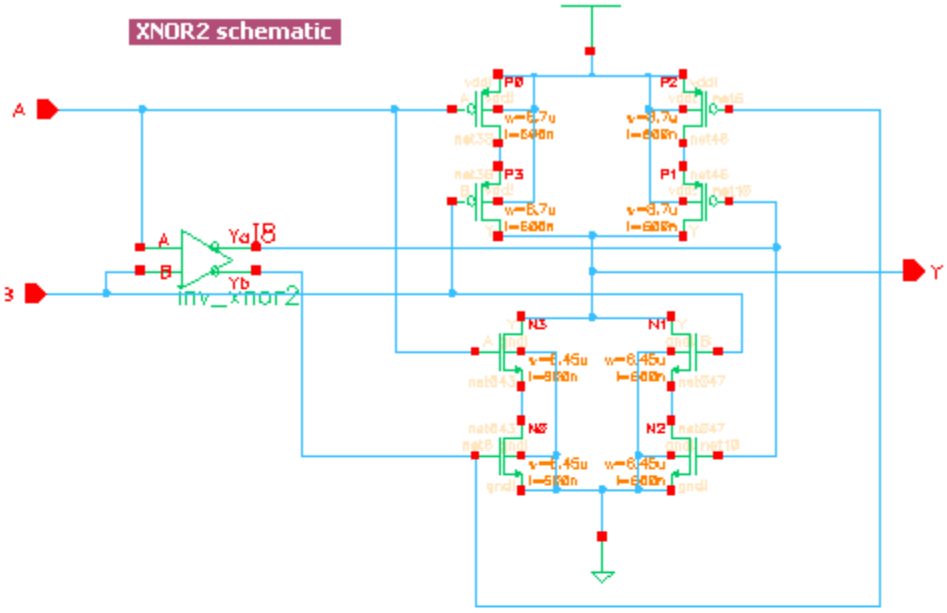


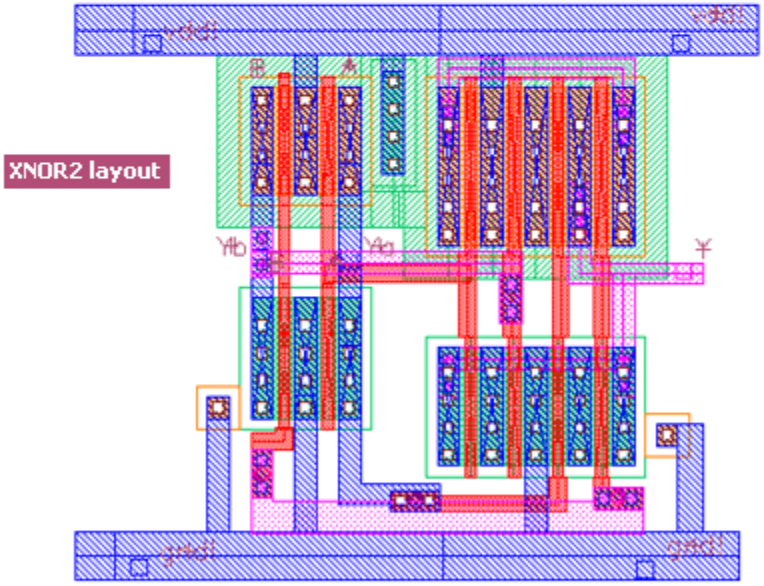

```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 19:36:27 2003
    completed ....Tue Nov  4 19:36:27 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violation for cell "TG layout"   ******
   Total errors found: 0
```

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

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         6               nets
         3               terminals
         3               pmos
         3               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         6               nets
         5               terminals
         2               pmos
         3               nmos


    Terminal correspondence points
          1     A
          2     in
          3     out

The net-lists match.
```



XNOR2






```
Running drclayout analysis
Flat mode
Full checking.
DRC started.......Tue Nov  4 20:01:55 2003
    completed ....Tue Nov  4 20:01:55 2003
    CPU TIME = 00:00:00  TOTAL TIME = 00:00:00
*********   Summary of rule violation for cell "XNOR2 layout"   ********
   Total errors found: 0
```

```
Compiling Diva LVS rules...

    Net-list summary for /home/eecad37/cell/LVS/layout/netlist
       count
         11              nets
         5               terminals
         6               pmos
         6               nmos

    Net-list summary for /home/eecad37/cell/LVS/schematic/netlist
       count
         11              nets
         5               terminals
         6               pmos
         6               nmos


    Terminal correspondence points
          1     A
          2     B
          3     Y
          4     gnd!
          5     vdd!

The net-lists match.
```