Universal Asynchronous Receiver Transmitter (UART) design

By

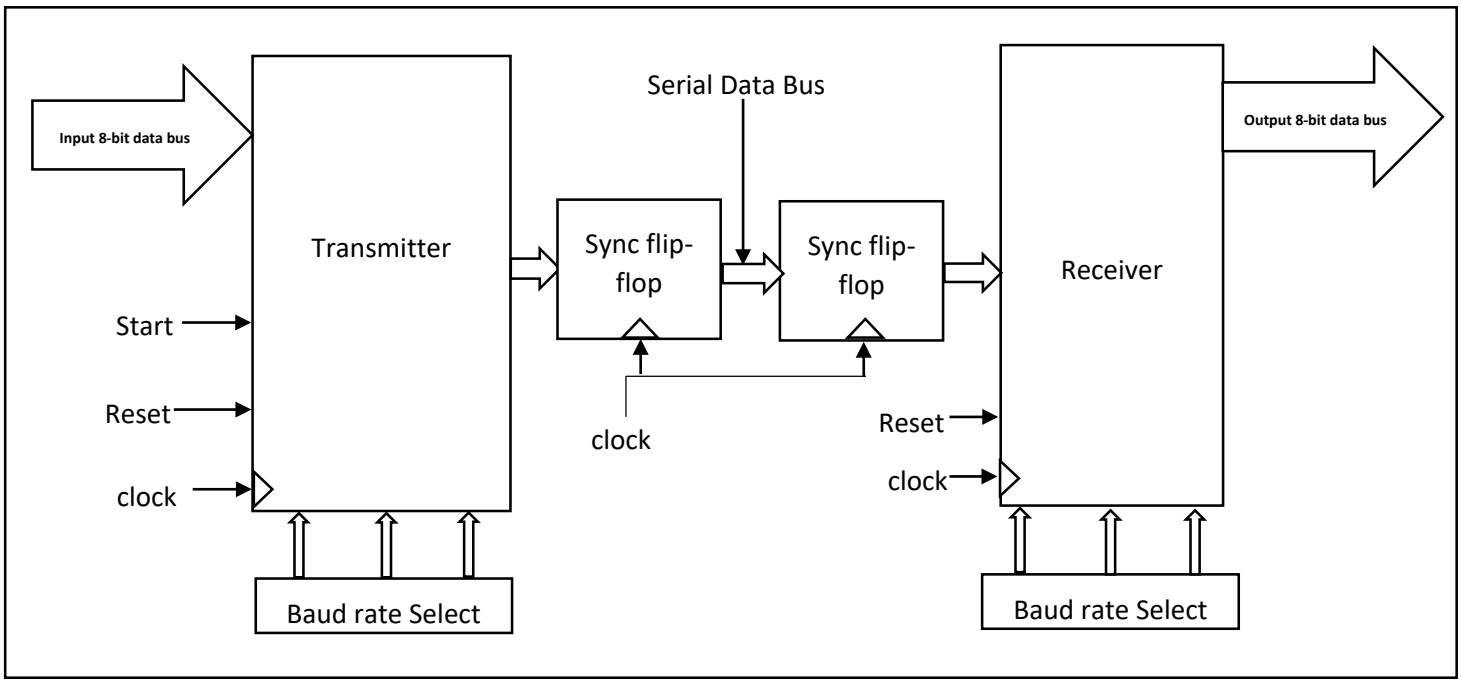
Krishna Subramanian

Report Bugs & Issues : < https://github.com/mongrelgem/USART-RTL-Physical-Design/issues >

Universal Asynchronous Receiver Transmitter (UART):

Universal Asynchronous Receiver Transmitter is used for asynchronous communication between two or more modules which are running at different clock frequencies. UART uses a fixed baud rate for synchronization of a synchronous data transfer. The baud rate on the receiver and transmitter must be same for data synchronization.

Block Diagram:



In this project we are using two modules, a transmitter and a receiver. Both modules have a baud rate selection option. User can select between 8 different baud rates which are 9600, 14400, 19200, 38400, 57600, 115200, 128000, and 256000.

The transmitter module receives 8-bit data parallelly from the system using 8-bit data bus and receives 2 control signals (start and reset) and a clock. Tx module starts sending out the data serially, at selected baud rate, when it detects high on start signal. Sync flip flop is used to mitigate the metastability issues. The receiver receives the data serially on serial data bus and it is tuned to the same baud rate as the transmitter.

A UART protocol is followed for data communication which is as follows,

UART Frame Format:

Bit no.	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	START	**8-bit data**								**END**
	START	**D0**	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**END**

The UART transmits and receives the LSB first. The UART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA and BAUDCON). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

- The start bit signals the receiver that a new character is coming.
- The next five to eight bits, depending on the code set employed, represent the character. The next one or two bits are always logic high, i.e., '1' condition and called the stop bit(s). They signal the receiver that the character is completed.
- Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

Detail working:

Transmitter

1. The transmitter module receives the 8-bit parallel data and waits for start signal to go high.
2. The user can select baud rate from 8 different options available.

OPERATION	CODE
9600	000
14400	001
19200	010
38400	011
57600	100
115200	101
128000	110
256000	111

3. The 3-bit baud rate select works as select lines for a 8:1 multiplexer and baud rate to be used is selected.
4. Once the inputs are latched and baud rate is selected, start signal is ready to go high.
5. Once the start signal goes high a start bit is sent on the serial bus followed by 8-bit user data and a stop bit.

Receiver

1. The receiver receives this serial data which was sent at a fixed baud rate on its input port. The receiver MUST be configured to the same baud rate at which transmitter is sending the data to properly receive data.
2. Once the receiver detects the start bit at the input for half baud cycle, it starts latching the incoming data.
3. As soon as receiver receives all 8-bits it checks for stop bits and then goes to IDLE state to receive next incoming data.
4. Simultaneously the current data is made available on the output port which can be used parallelly.

After deciding flow of design as specified above we designed this system in behavioral VerilogHDL.

Physical design of the standard cells:

To synthesize the behavioral code a library needs to be designed. To design this library first step was to layout the cells. Following are the cells which we included in the library. Aoi22, Oai21, Oai211, Inverter, Nand2, Nor2, Xor2, 2:1 multiplexer, and a falling edge triggered active high reset D flip-flop. (See appendix)

Every layout in the library is optimized for area on chip. After laying out all the 9 cells in the physical library, we created schematics for these cells and verified their functionality using HSPICE once DRC/LVS was cleared.

Next step was to generate the library in the liberty (.lib) format and also ASCII format description of physical cells in LEF format.

Generating library in liberty format and .db format:

The tools used to generate this library were SiliconsmartACE and Library compiler. Siliconsmart takes all spice netlist files for each cell as an input and generates the .lib library file. This process was automated using a Perl script where we had to make changes in the instance file. The instance files get automatically generated when spice netlists are imported. By default, all input and output pins were getting defined with inout direction. So, we wrote a script to change this direction of pins according to their behavior. This step generated the .lib format library.

The synthesis tool (Design Vision) cannot read this liberty formatted file. Therefore, using library compiler we converted this .lib file to .db file so that design vision can read this library file and the behavioral design can be synthesized.

Synthesis and verification of behavioral design using generated library:

The behavioral VerilogHDL code is synthesized using the freshly created library to generate the mapped netlist. This generated mapped netlist is nothing but a structural description of the system. Before proceeding further this mapped netlist needs to be verified. We simulated the structural code and behavioral code and it was found that both verilogs generate exact same output for same testbench. The input and output waveforms for both is shown below.

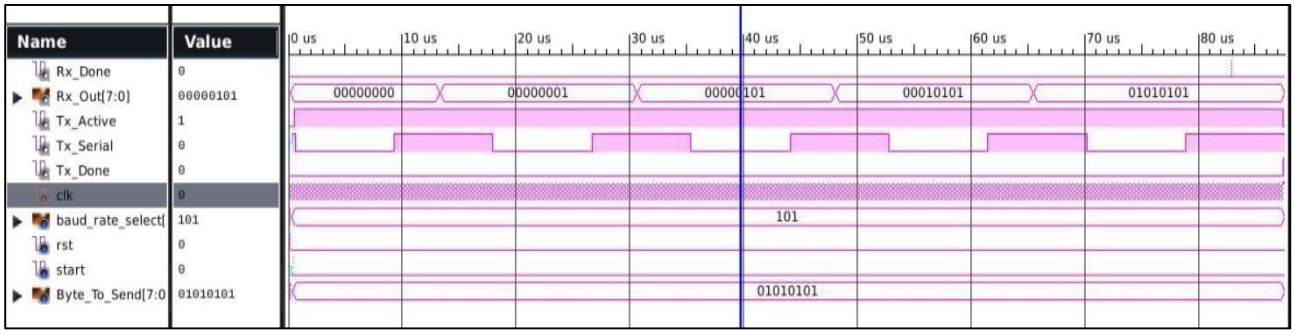


Figure 1 Output waveforms for behavioral code

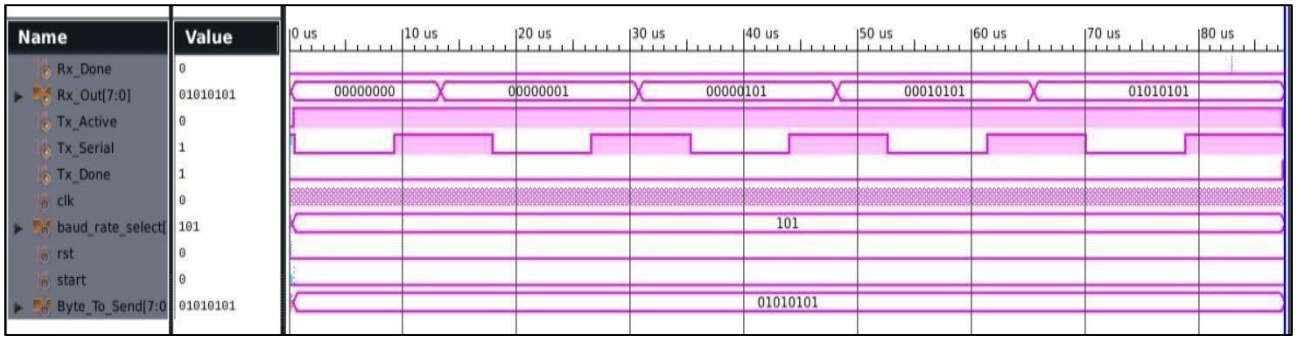


Figure 2 Output waveforms of mapped netlist

Floor planning, Power planning, and PNR (Place and Route):

Now that the design verification is done we proceeded for floor planning. As mentioned before, from the physical library which we had designed we generated an LEF file which is used by Auto place and Route tool (Encounter). With the mapped netlist generated after synthesis and the LEF file the floor plan was made using Encounter. Also, power planning and auto routing was done. Once this was done, we were able to generate the DEF file which describes this floorplan in the format that virtuoso can understand. But because of some bugs in the tool we had to write a script to fix the potential DRC errors for vias.

Physical layout and schematic of the final design:

Now that everything is ready with routed floor plan in DEF format, we imported the final layout in the cadence. The synthesized mapped netlist is used to generate the schematic of the final design.

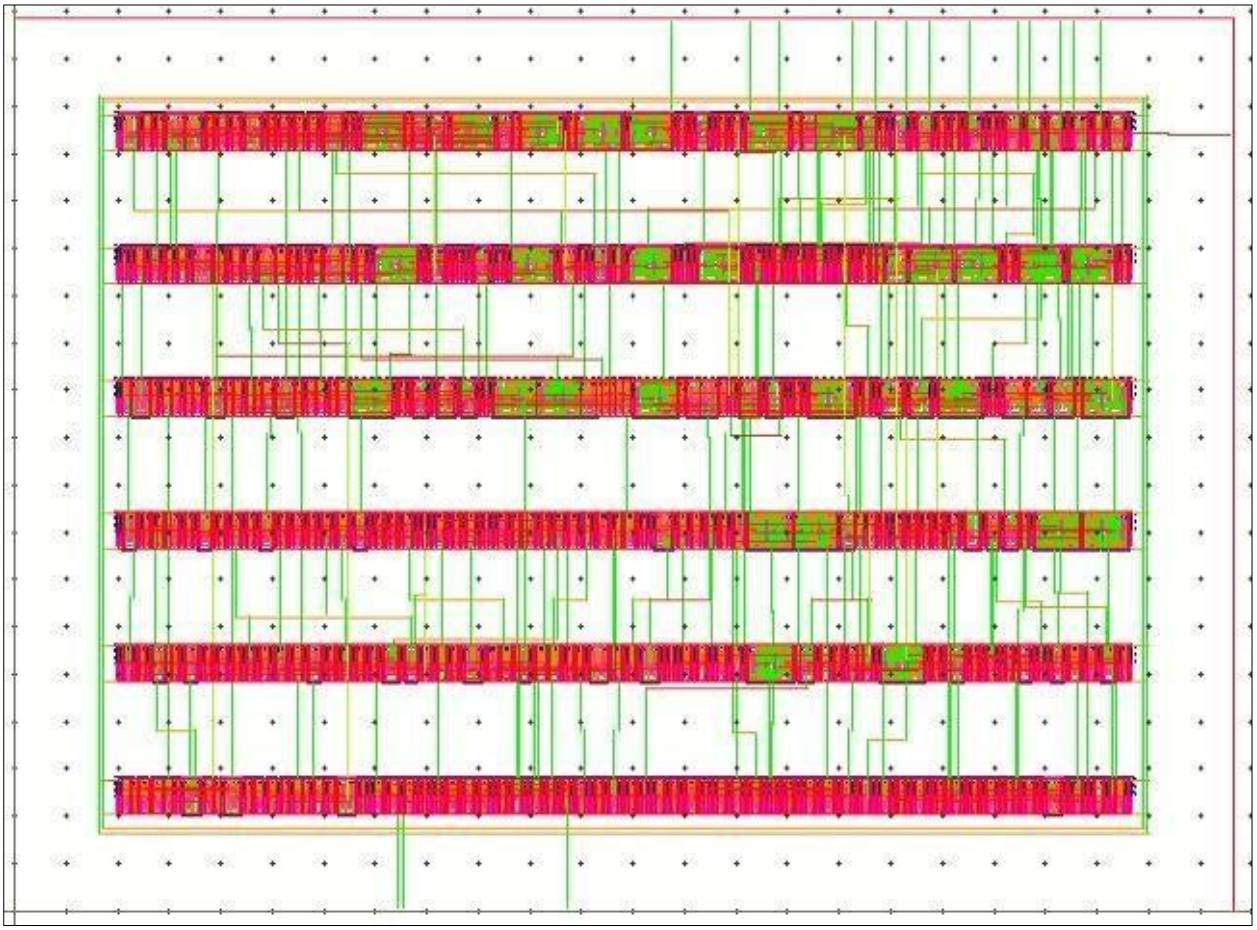


Figure 3 Layout of Transmitter unit

Figure 4 Schematic of Transmitter unit

Figure 5 DRC Report

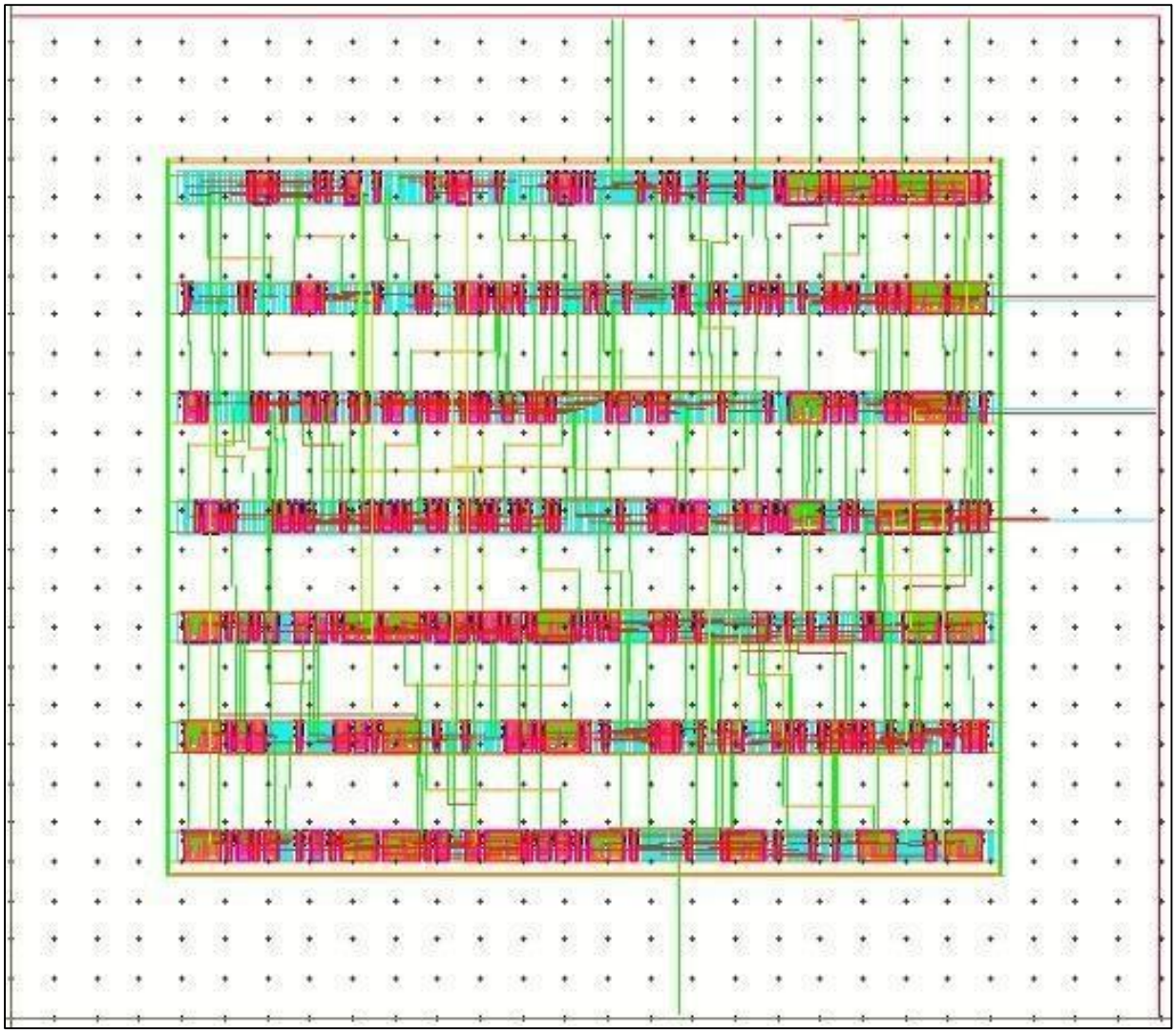


Figure 6 Layout of receiver block

Figure 7 DRC report

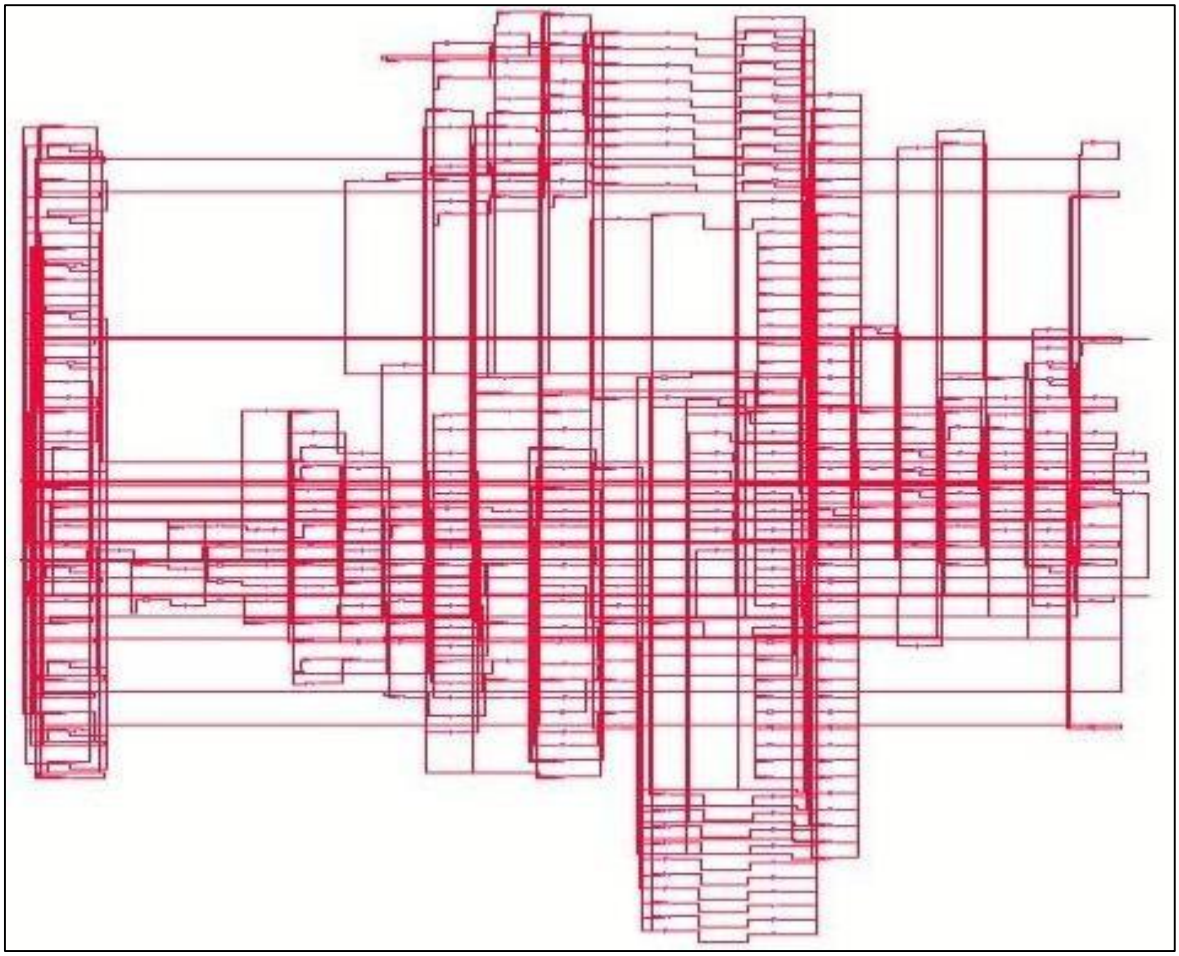


Figure 8 Schematic of receiver block

Static timing analysis:

Primetime report:

```
{engnx05:~/cad/primetime} pt_shell -f primetime.script

PrimeTime (R)

Version M-2016.12-SP3-1 for linux64 - Jul 18, 2017

Copyright (c) 1988 - 2017 Synopsys, Inc.
This software and the associated documentation are proprietary to Synopsys, Inc.
This software may only be used in accordance with the terms and conditions of a
written license agreement with Synopsys, Inc. All other use, reproduction, or
distribution of this software is strictly prohibited.

############################################################## #
Define search path verilog file and library and variables
##############################################################
set search_path &amp;quot;* ~/cad/primetime&amp;quot;
```

```
   * ~/cad/primetime source
   variables1 rst rst
   ############################################################## #
   link library
   ##############################################################
   set link_library $library_file library65.db
   set target_library $library_file library65.db
   #set link_library [list $library_file ]
   #set target_library [list $library_file ]

   ############################################################## #
   link design
   ############################################################## remove_design
   -all
   Error: Nothing matched for designs: there are none loaded (SEL-005)
   0
   read_verilog $verilog_file
   1
   ############################################################## #
   Define IO parameters
   ##############################################################
   set_driving_cell -lib_cell $driving_cell -input_transition_rise $input_transition -
input_transition_fall
   $input_transition [all_inputs]
   Loading verilog file
   Loading db file
   Linking design uart_tx...
   1
   set_load $load [all_outputs]
   1
   ##############################################################
   ##############################################################
   #define the clock - for comb circuit we may not need to use any clock
   ############################################################## create_clock
   -name clk -period $clock_period [get_ports $clock_pin_name]
   1
   set_clock_transition -rise -max $input_transition [get_clocks clk]
   1
   set_clock_transition -fall -max $input_transition [get_clocks clk]
   1

   ############################################################## #
   set condition
   ##############################################################
   set timing_slew_propagation_mode worst_slew worst_slew
   set timing_report_unconstrained_paths true true
   set power_enable_analysis true true
   set_disable_timing [get_ports $reset_pin_name]
   1
   ############################################################## #
   analyze delay and power
   ############################################################## check_timing
   Warning: Some timing arcs have been disabled for breaking timing loops or because of
constant propagation. Use the &#39;report_disable_timing&#39; command to get the list
of these
disabled timing arcs. (PTE-
   003) Information: Checking
   &#39;no_input_delay&#39;.
   Information: Checking
   &#39;no_driving_cell&#39;.
```

```
Information: Checking &#39;unconstrained_endpoints&#39;.
Warning: There are 3 endpoints which are not constrained for maximum delay.

Information: Checking &#39;unexpandable_clocks&#39;.
Information: Checking &#39;latch_fanout&#39;.
Information: Checking &#39;no_clock&#39;.
Information: Checking &#39;partial_input_delay&#39;.
Information: Checking &#39;generic&#39;.
Information: Checking &#39;loops&#39;.

Information: Checking &#39;generated_clocks&#39;.
Information: Checking &#39;pulse_clock_non_pulse_clock_merge&#39;.
Information: Checking &#39;pll_configuration&#39;.
0
update_timing
1
report_timing -transition_time -delay min_max -capacitance -input_pins
****************************************
Report : timing
-path_type full
-delay_type min_max
-input_pins
-max_paths 1
-transition_time
-capacitance
-sort_by slack
Design : uart_tx
Version: M-2016.12-SP3-1
Date : Sat Aug 4 16:42:52 2018
****************************************

Startpoint: Data_Byte_reg[0]
(falling edge-triggered flip-flop clocked by clk&#39;) Endpoint:
Data_Byte_reg[0]
(falling edge-triggered flip-flop clocked by clk&#39;)
Path Group: clk Path
Type: min

Point Cap Trans Incr Path
-------------------------------------------------------------------------------
clock clk&#39; (fall edge) 0.00 0.00 0.00 clock network delay (ideal) 0.00
0.00 Data_Byte_reg[0]/CLK (dff) 0.00 0.00 0.00 f
Data_Byte_reg[0]/QOUT (dff) 0.00 0.05 0.11 0.11 f
U88/C (aoi22) 0.05 0.00 0.11 f
U88/OUT (aoi22) 0.00 0.04 0.06 0.17 r
U87/IN (inv) 0.04 0.00 0.17 r
U87/OUT (inv) 0.00 0.02 0.04 0.21 f
Data_Byte_reg[0]/DIN (dff) 0.02 0.00 0.21 f data
arrival time 0.21

clock clk&#39; (fall edge) 60.00 0.00 0.00
clock network delay (ideal) 0.00 0.00
clock reconvergence pessimism 0.00 0.00
Data_Byte_reg[0]/CLK (dff) 0.00 f library
hold time 7.03 7.03 data required time
7.03
-------------------------------------------------------------------------------
data required time 7.03 data arrival time -0.21
-------------------------------------------------------------------------------
slack (VIOLATED) -6.82
```

```
    Startpoint: Tx_Enable_reg
    (falling edge-triggered flip-flop clocked by clk&#39;)

    Endpoint: Tx_Enable_reg
    (falling edge-triggered flip-flop clocked by clk&#39;)
    Path Group: clk Path
    Type: max

    Point Cap Trans Incr Path
    ---------------------------------------------------------------------------
    clock clk&#39; (fall edge) 60.00 0.00 0.00 clock network delay (ideal) 0.00
    0.00 Tx_Enable_reg/CLK (dff) 60.00 0.00 0.00 f
    Tx_Enable_reg/QOUT (dff) 10.00 0.00 71.44 71.44 r
    U65/A (nand2) 0.00 0.00 71.44 r
    U65/OUT (nand2) 0.00 0.04 0.03 71.48 f
    U64/D (oai211) 0.04 0.00 71.48 f
    U64/OUT (oai211) 0.00 0.76 0.05 71.53 r
    Tx_Enable_reg/DIN (dff) 0.76 0.00 71.53 r data
    arrival time 71.53

    clock clk&#39; (fall edge) 0.00 200.00 200.00
    clock network delay (ideal) 0.00 200.00 clock
    reconvergence pessimism 0.00 200.00
    Tx_Enable_reg/CLK (dff) 200.00 f library
    setup time -5.99 194.01 data required time
    194.01
    ---------------------------------------------------------------------------
    data required time 194.01 data arrival time -71.53
    ---------------------------------------------------------------------------
    slack (MET) 122.49

    1
    update_power
    Information: Checked out license &#39;PrimeTime-PX&#39; (PT-019)
    Warning: Neither event file or switching activity data present for power
    estimation.
The command will propagate switching activity values for power
    calculation. (PWR-246) Information: Running switching activity
    propagation with 4 threads!
    Information: Running averaged power analysis... (PWR-601)
    Information: Running power calculation with 4 threads. (PWR-602)
    1
    report_power
    ****************************************
    Report : Averaged Power
    Design : uart_tx
    Version: M-2016.12-SP3-1
    Date : Sat Aug 4 16:42:53 2018
    ****************************************

    Attributes ----------
    i - Including register clock pin internal power u
    - User defined power group

    Internal Switching Leakage Total
    Power Group Power Power Power Power ( %) Attrs
    ---------------------------------------------------------------------------

    clock_network 4.699e-06 1.071e-06 1.552e-10 5.770e-06 (60.42%) i
```

```
register -1.264e-07
3.459e-06 4.122e-08 3.374e-06 (35.32%)
combinational 1.888e-07 2.019e-07 1.621e-08 4.069e-07 ( 4.26%)
sequential 0.0000 0.0000 0.0000 0.0000 ( 0.00%) memory 0.0000
0.0000 0.0000 0.0000 ( 0.00%) io_pad 0.0000 0.0000 0.0000
0.0000 ( 0.00%) black_box 0.0000 0.0000 0.0000 0.0000 ( 0.00%)

Net Switching Power = 4.731e-06 (49.54%)
Cell Internal Power = 4.762e-06 (49.86%) Cell
Leakage Power = 5.758e-08 ( 0.60%)
---------
Total Power = 9.551e-06 (100.00%)

1
1
Information: Defining new variable &#39;driving_cell&#39;. (CMD-041)
Information: Defining new variable &#39;library_file&#39;. (CMD-041)
Information: Defining new variable &#39;verilog_file&#39;. (CMD-041)
Information: Defining new variable &#39;input_transition&#39;. (CMD-041)
Information: Defining new variable &#39;timing_slew_propagation_mode&#39;. (CMD-
041)
Information: Defining new variable &#39;clock_period&#39;. (CMD-041)
Information: Defining new variable &#39;load&#39;. (CMD-041)
Information: Defining new variable &#39;reset_pin_name&#39;. (CMD-041)
Information: Defining new variable &#39;clock_pin_name&#39;. (CMD-041)
1
```

Appendix: Individual cell's layout, schematic and pitches

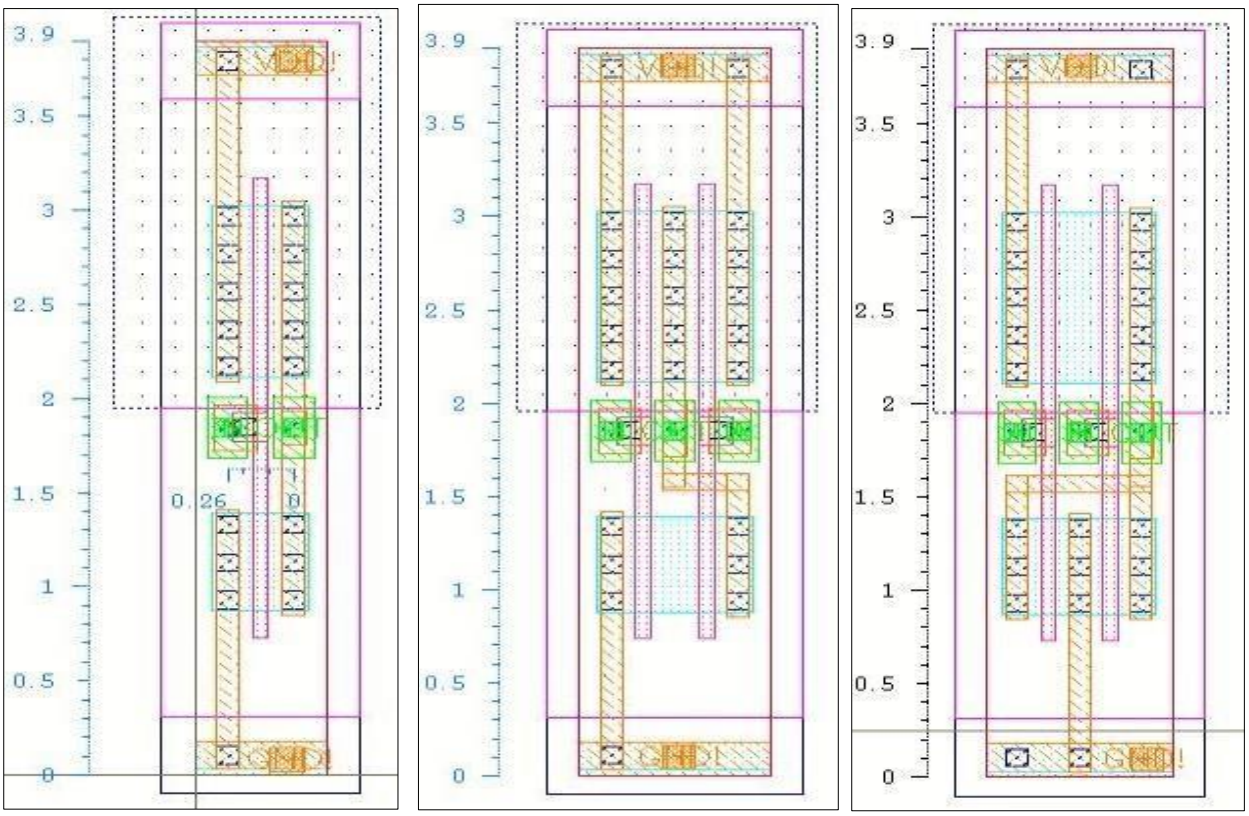


Figure 9 Layout of Inverter, nand2, and nor2

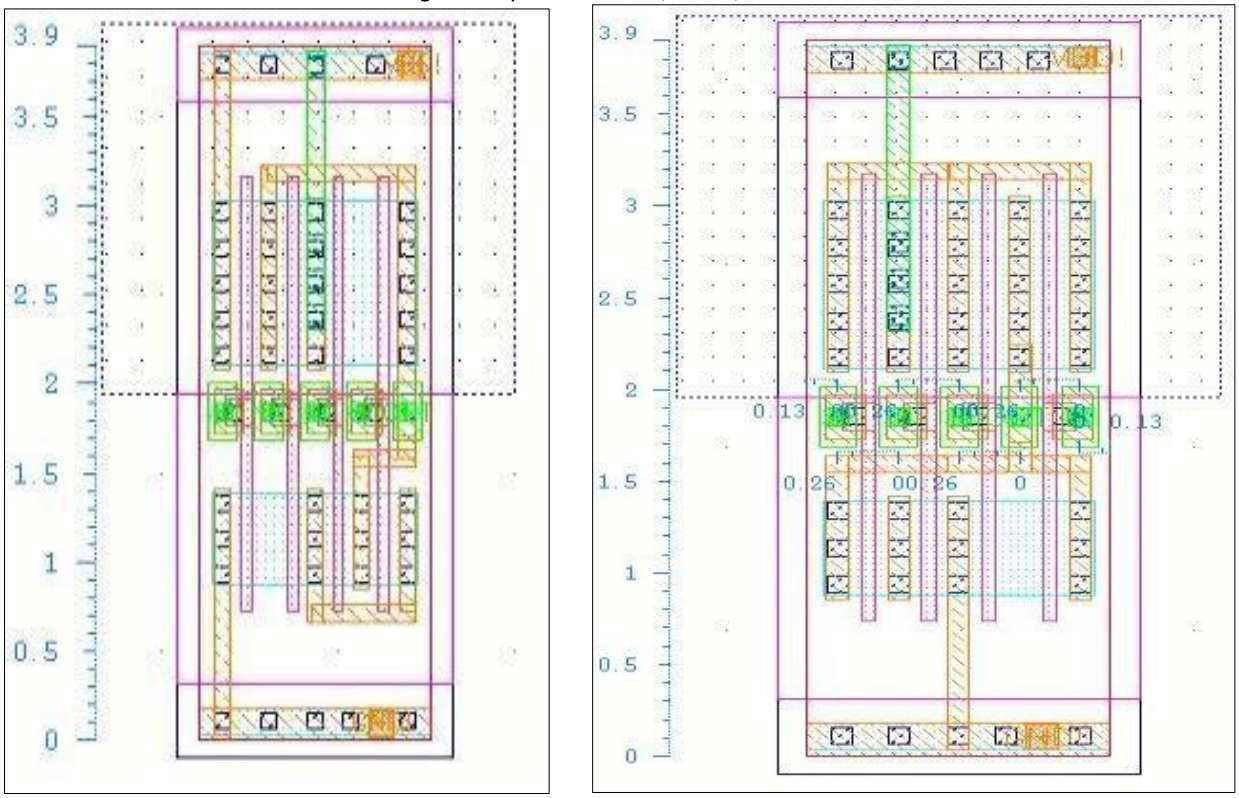


Figure 10 Layout of oai211 and aoi22

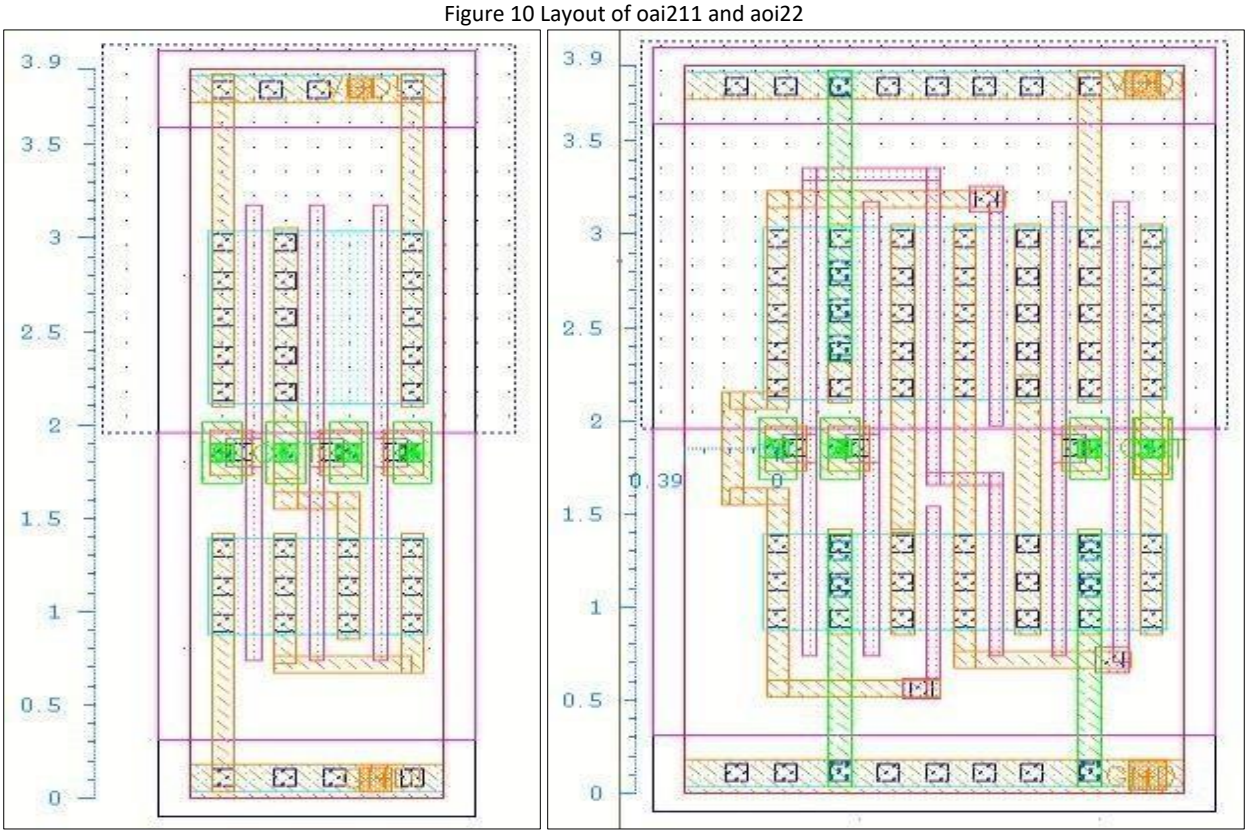


Figure 11 Layout of oai21 and mux 2:1

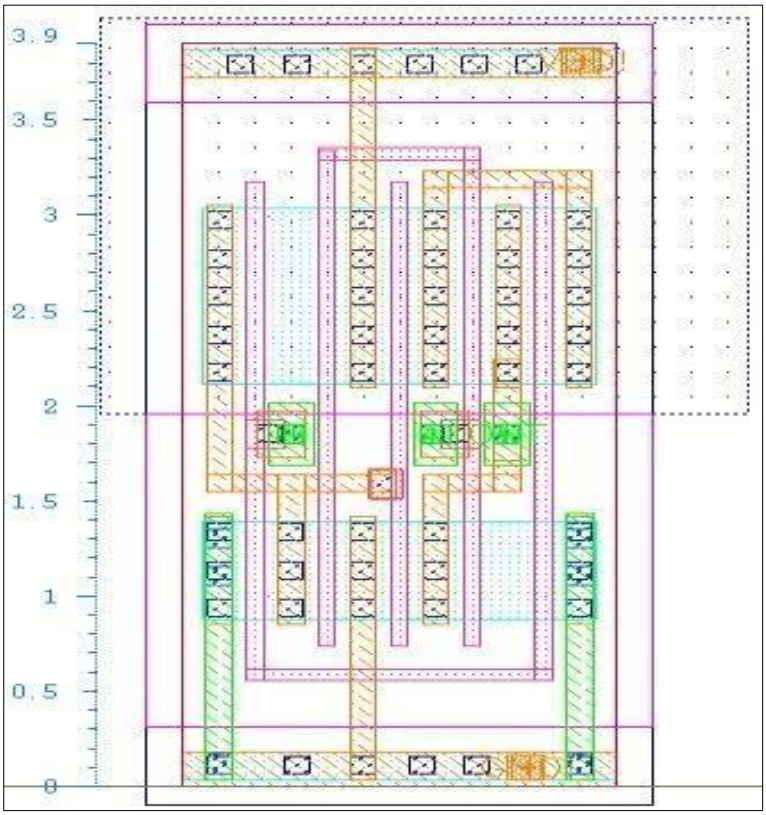


Figure 12 Layout of Xor

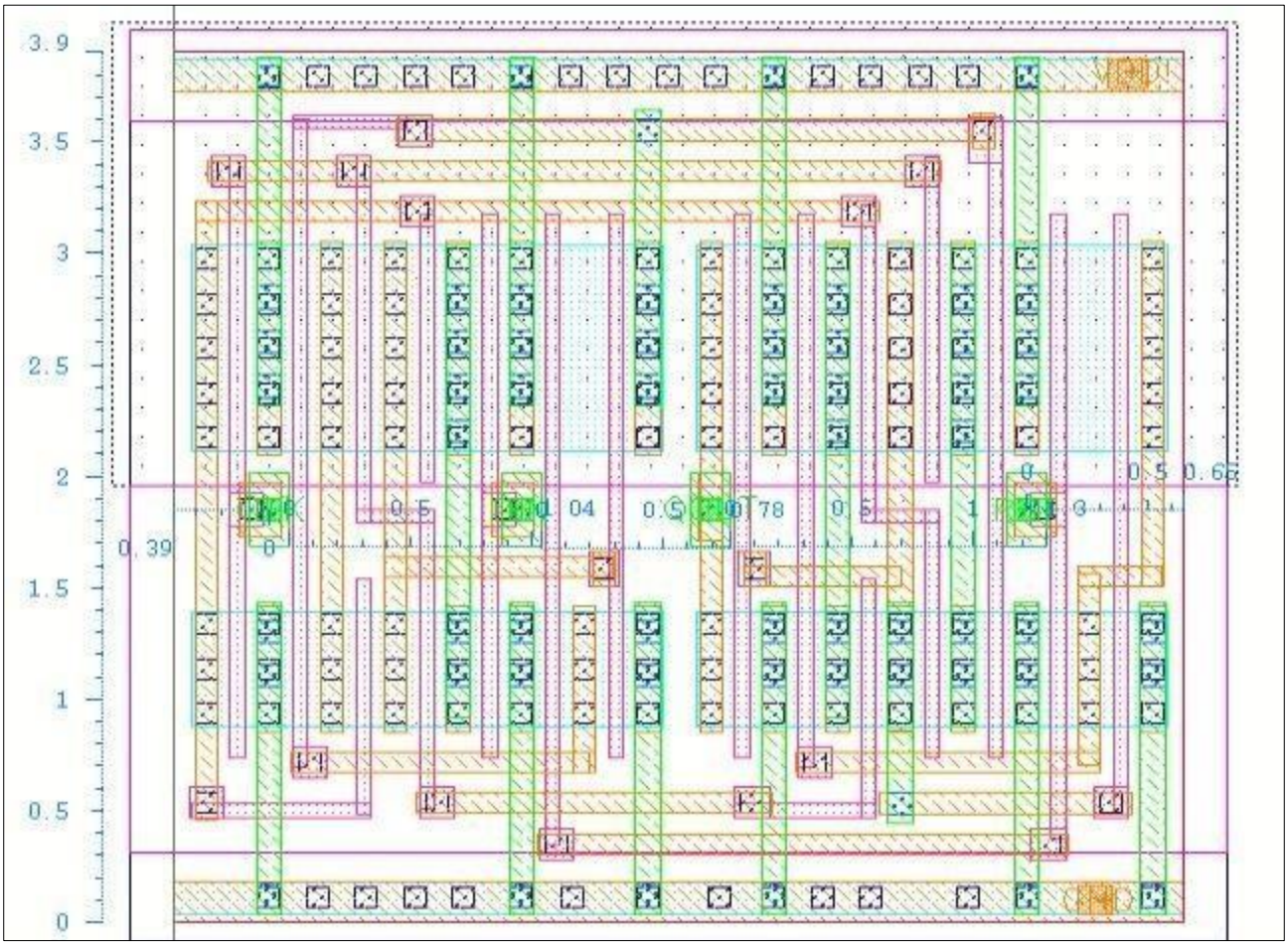


Figure 13 Layout of D flip-flop

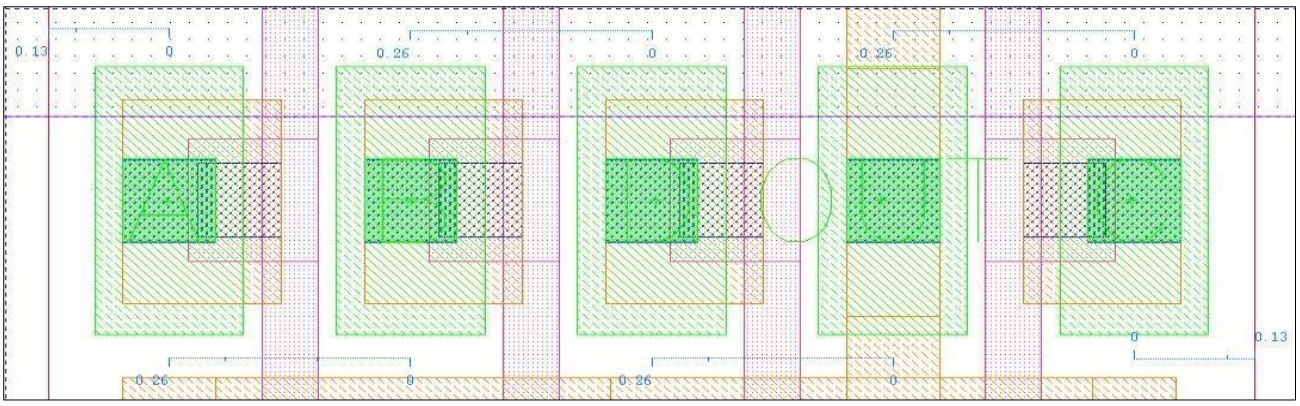


Figure 14 Pitch distance

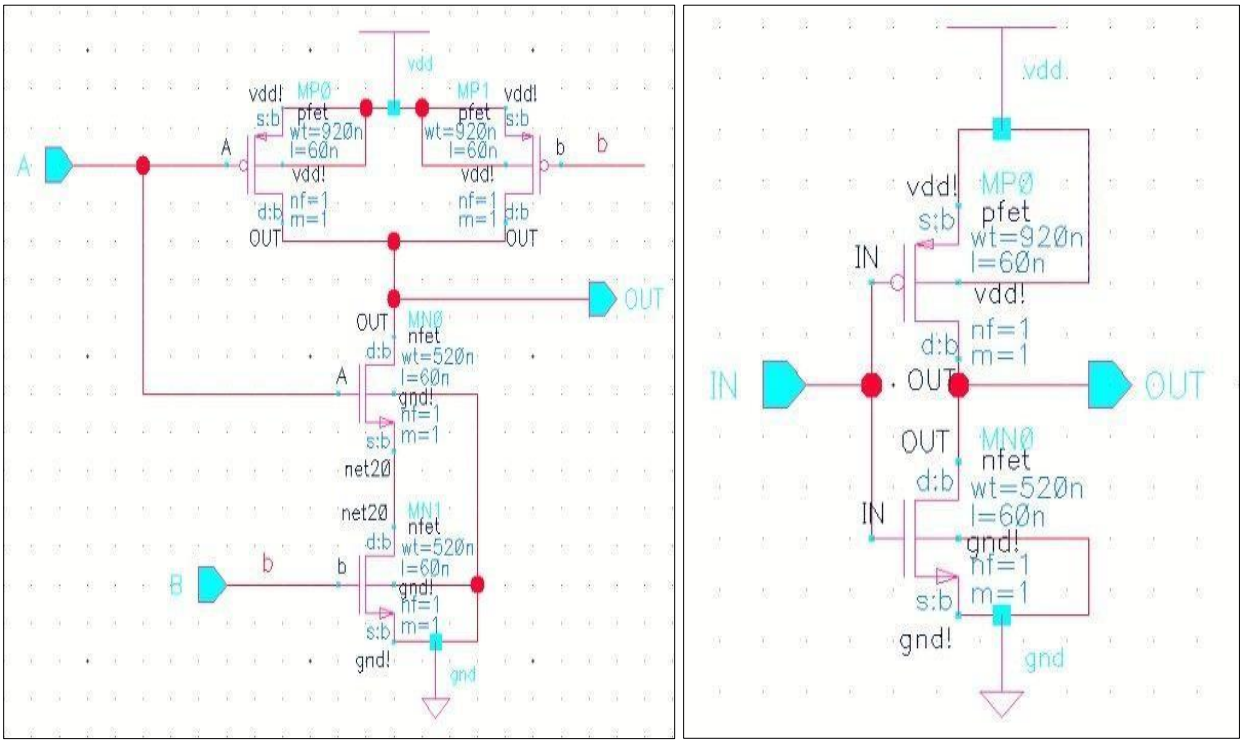


Figure 15 Schematic of Nand2 and Inverter

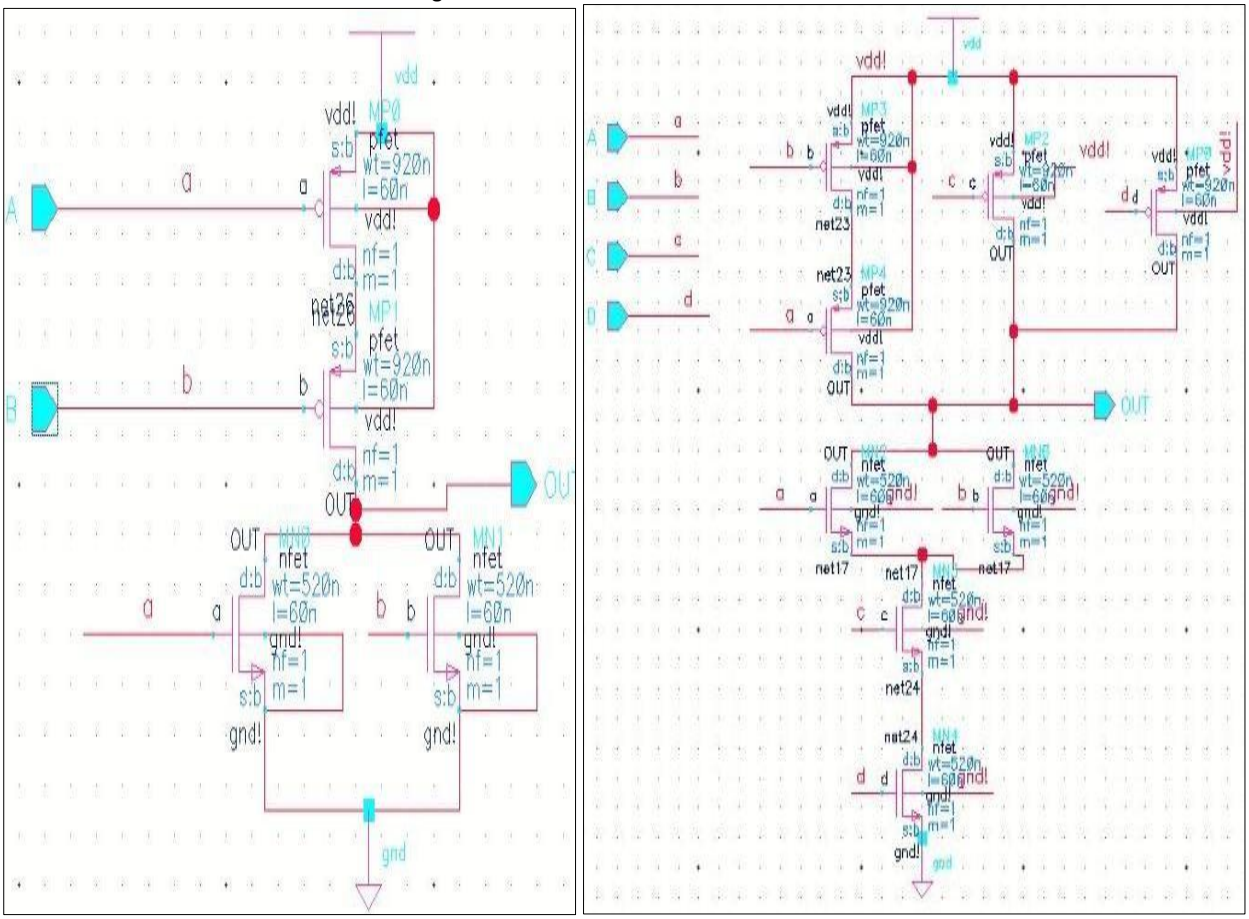


Figure 16 Schematic of Nor2 and Oai211

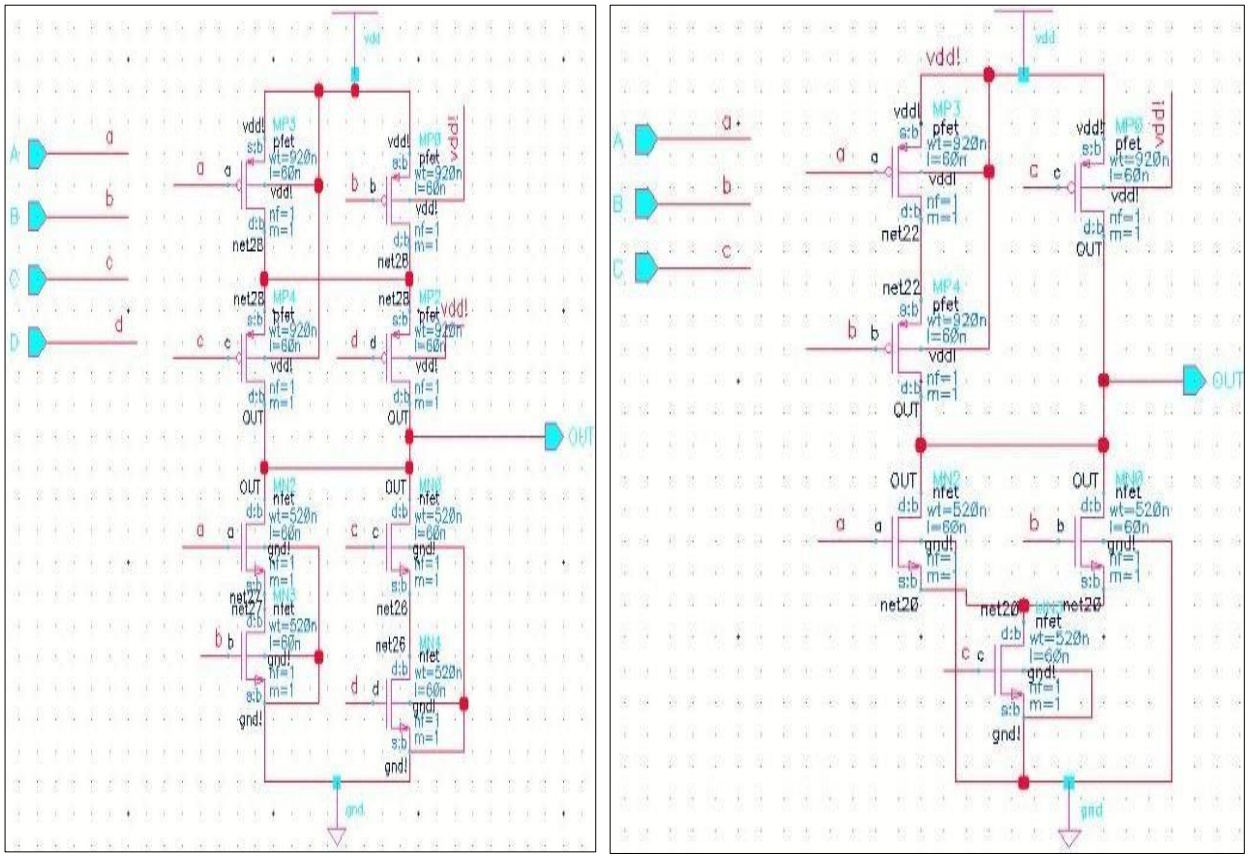


Figure 17 Schematic of aoi22 and oai21

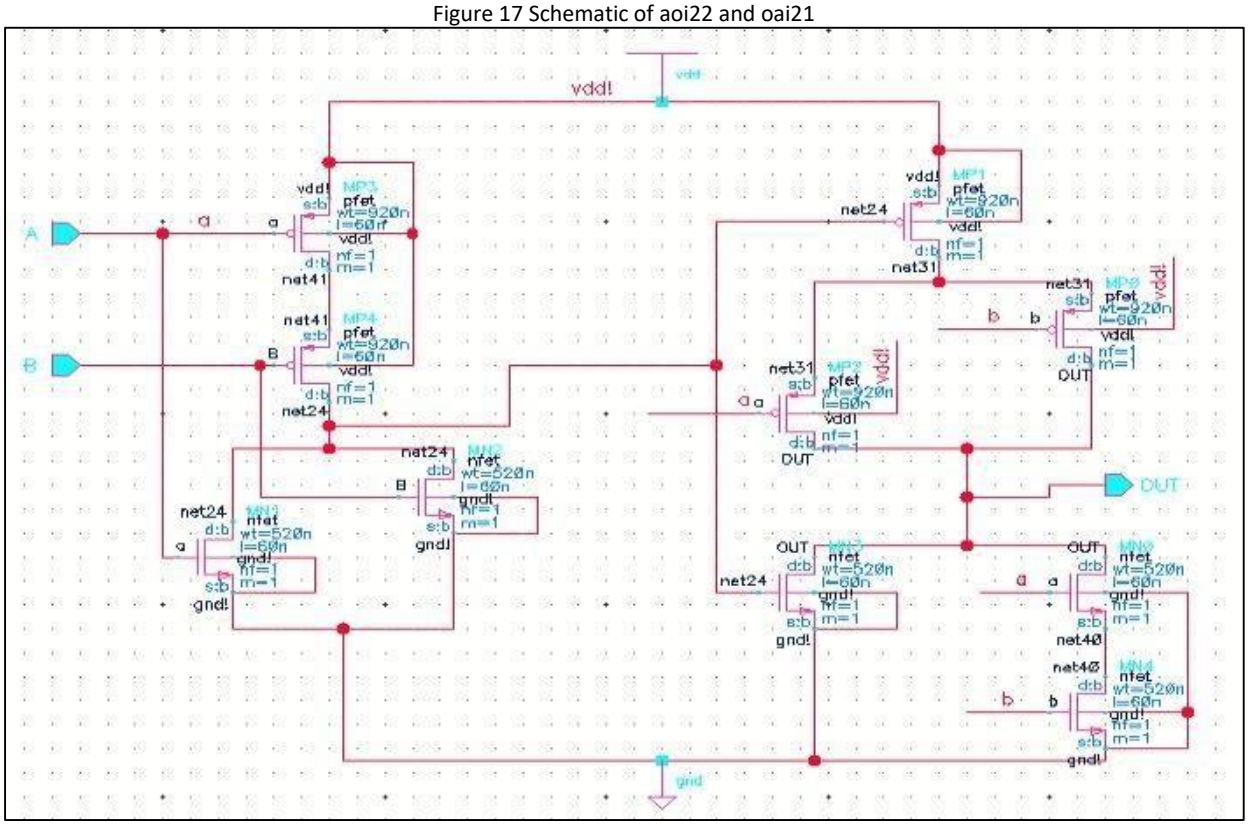


Figure 18 Schematic of xor2

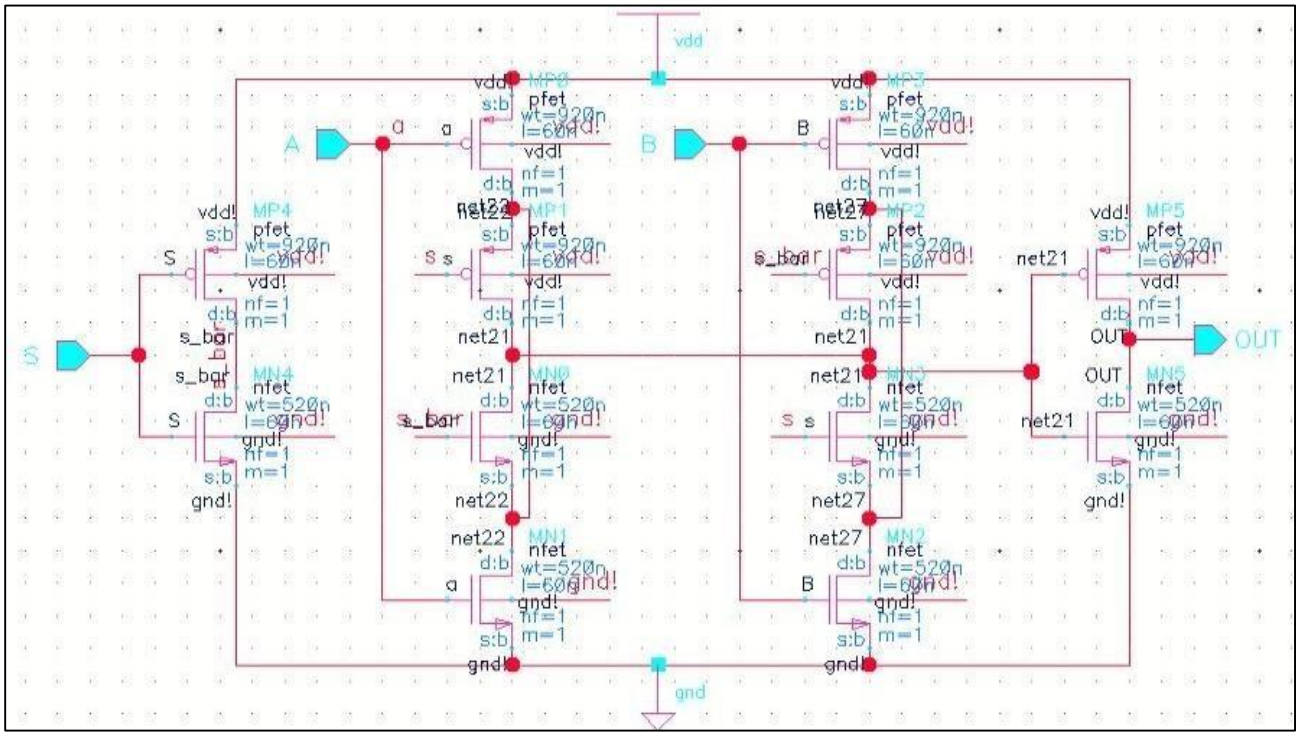


Figure 19 Schematic of mux 2:1

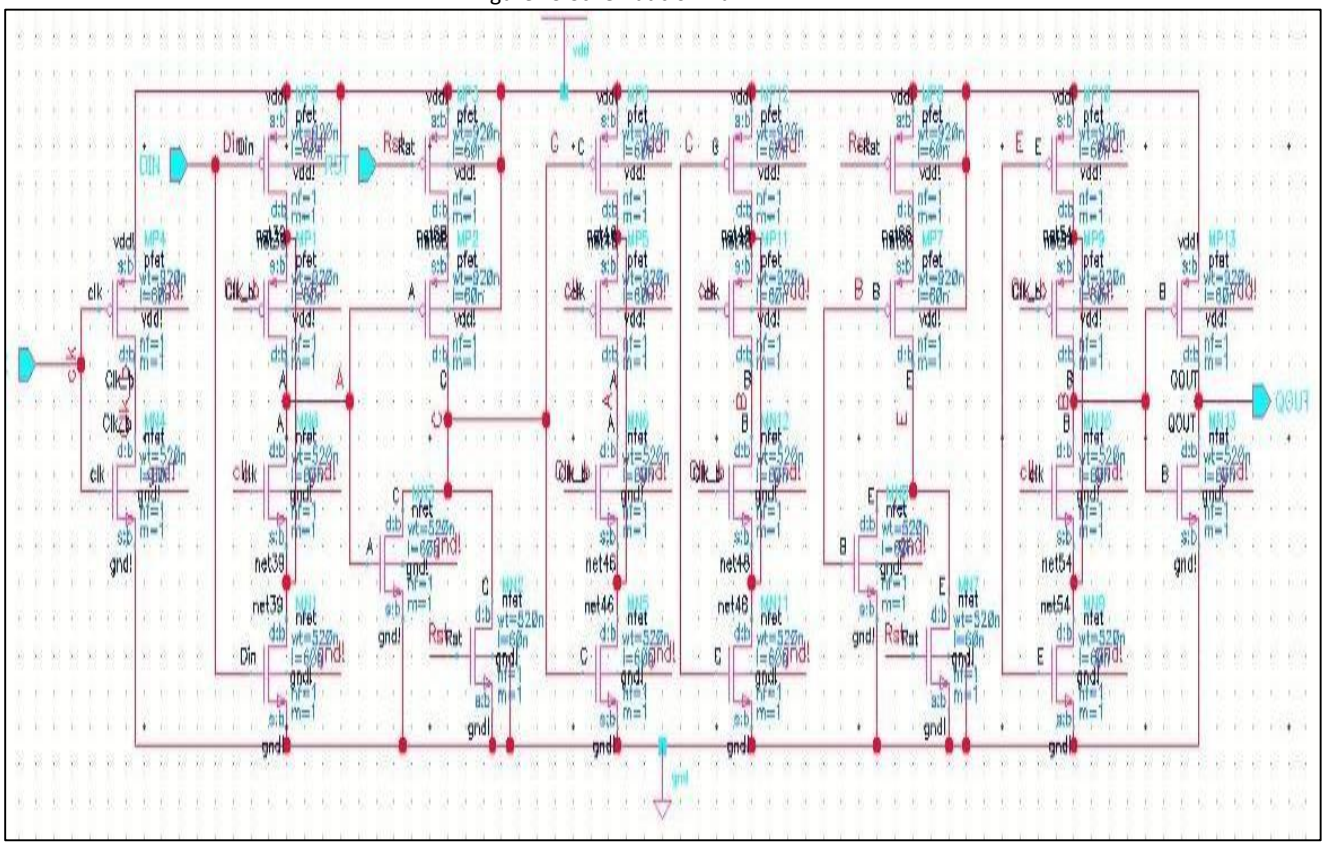


Figure 20 Schematic of D flip-flop