Homework Assignment 4

(Due Mar. 4th at the beginning of the class)

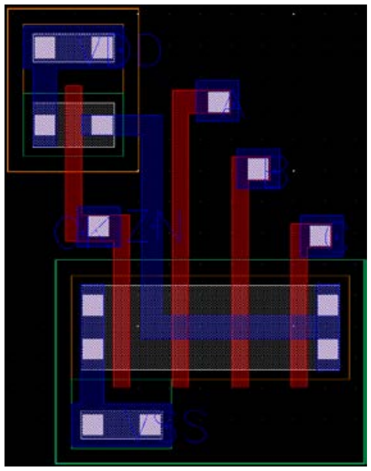
0. Preparation for homework 4
   - Download the following file into your working directory.
     - o wget http://www.eecs.wsu.edu/~ee434/Homework/hw04.tar.gz
   - Unzip it.
     - o tar xvzf hw04.tar.gz
   - Source synopsys.sh
     - o source synopsys.sh

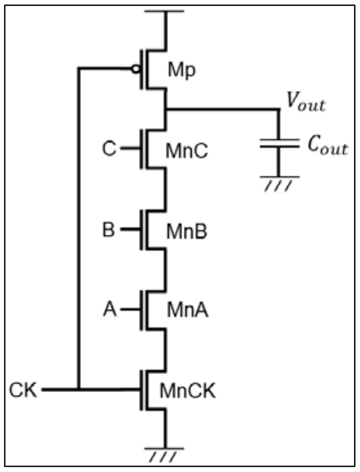
1. [Pseudo-nMOS, **20 points**]
   - Design a pseudo-nMOS inverter for the following design spec:
     - o Simulate the inverter with the following load cap and input signal
       - ▪ Load cap: 10fF
       - ▪ The fall & rise transition time of the input signal: 10ps
     - o Spec
       - ▪ Rise time < 100ps
       - ▪ Fall time < 100ps
       - ▪ $V_{OL} < 100mV$
       - ▪ $V_{OH} > 900mV$
       - ▪ $NM_L > 150mV$
       - ▪ $NM_H > 300mV$
   - [**Submit**]
     - o The size of the NMOS and PMOS transistors
     - o Rise time, fall time
     - o V_{IH}, V_{IL}, V_{OH}, V_{OL}, NM_L, NM_H
     - o Average power for falling and rising transitions

(1) Let's first satisfy the fall and rise time constraints.
   - Wn=45nm, Wp=70nm: t_f=288ps, t_r=167ps.
   - Wn=135nm, Wp=140nm: t_f=103ps, t_r=91.1ps.
   - Wn=210nm, Wp=140nm: t_f=53.1ps, t_r=96.5ps (t_r is too close to 100ps, so let's upsize Wp).
   - Wn=210nm, Wp=210nm: t_f=64.4ps, t_r=59.4ps.

(2) DC characteristics
   - Wn=210nm Wp=210nm: V_{OL}>100mV (We should upsize the NMOS TR).
   - Wn=840nm Wp=210nm: V_{OL}=120mV
   - $\boxed{\text{Wn=1260nm Wp=210nm}}$: V_{OL}=97mV, V_{IH}=575mV, V_{IL}=300mV, V_{OH}=940mV => NM_L=203mV, NM_H=365mV, p_{fall}=218uW, p_{rise}=66.5uW

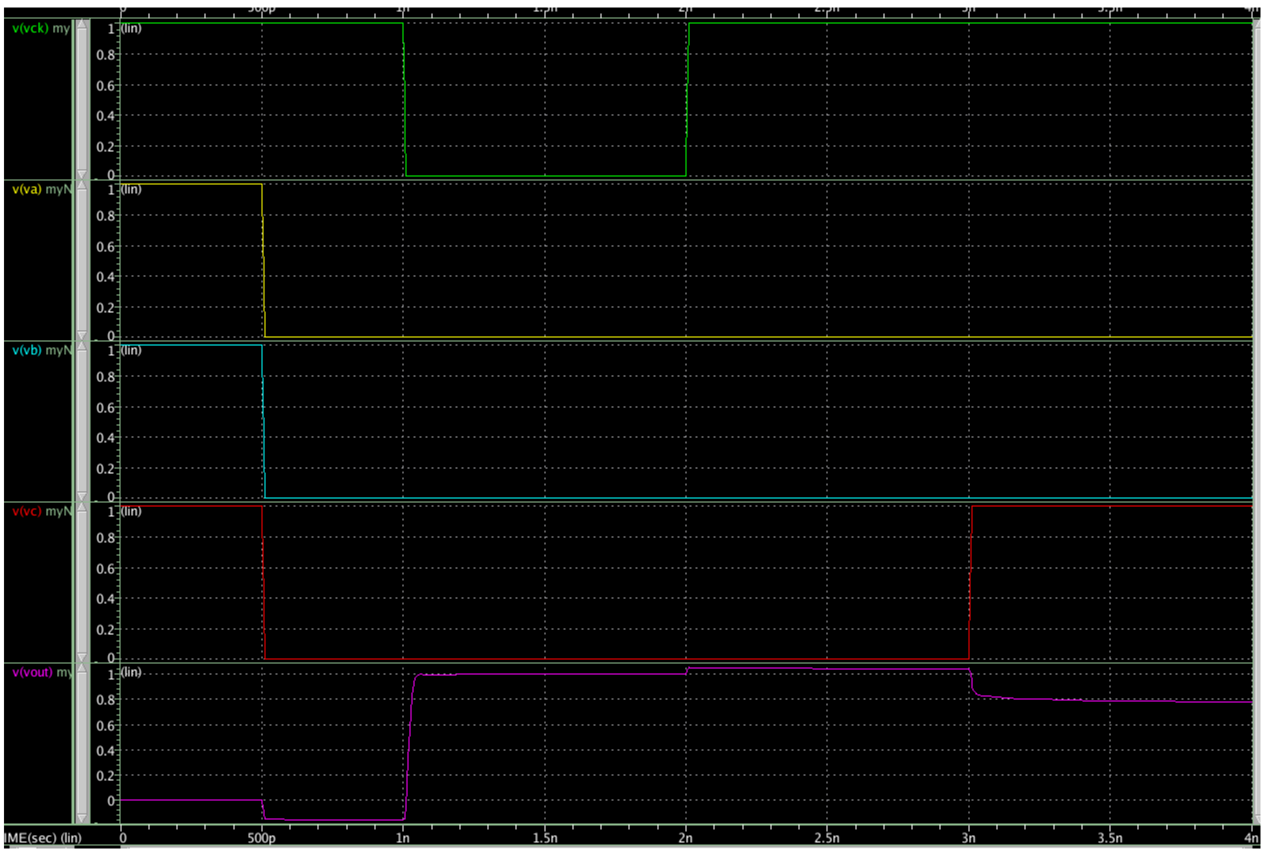
2. [Dynamic CMOS, **20 points**]
   - We design a three-input NAND gate using the dynamic CMOS design style.
   - Open myNAND3_pex.cdl and see the netlist of the NAND gate. I drew a layout for the NAND gate and extracted parasitic RC. myNAND3_pex.cdl includes all the parasitic RC.
   - Open myNAND3_simul.sp and see the netlist. It is used to simulate the NAND gate.
   - The followings show the layout and schematic of the NAND gate.





   - Add four signal waveforms (CK, A, B, C) to simulate charge sharing.
     - Load cap: 10fF
     - CK: $V_{DD} \rightarrow 0 \rightarrow V_{DD}$
     - When CK is V_{DD} (before it goes down to 0), set A, B, and C to V_{DD} so that it can fully discharge the output capacitor and all the parasitic capacitors.
     - Then, set A, B, and C to 0 before CK goes to 0.
     - Then, CK goes to 0 and the gate will charge the output capacitor.
     - Then, CK goes to V_{DD}.
     - Then, set C to V_{DD} so that charge sharing can happen between the output capacitor and the parasitic capacitor between MnC and MnB.
     - Perform the same simulation, but set both B and C to V_{DD} so that charge sharing can happen among the output capacitor, the parasitic capacitors between MnC and MnB and between MnB and MnA.
   - [**Submit**]
     - Vout when only C is set to V_{DD} for C_{out} =10fF, 9fF, …, 1fF.

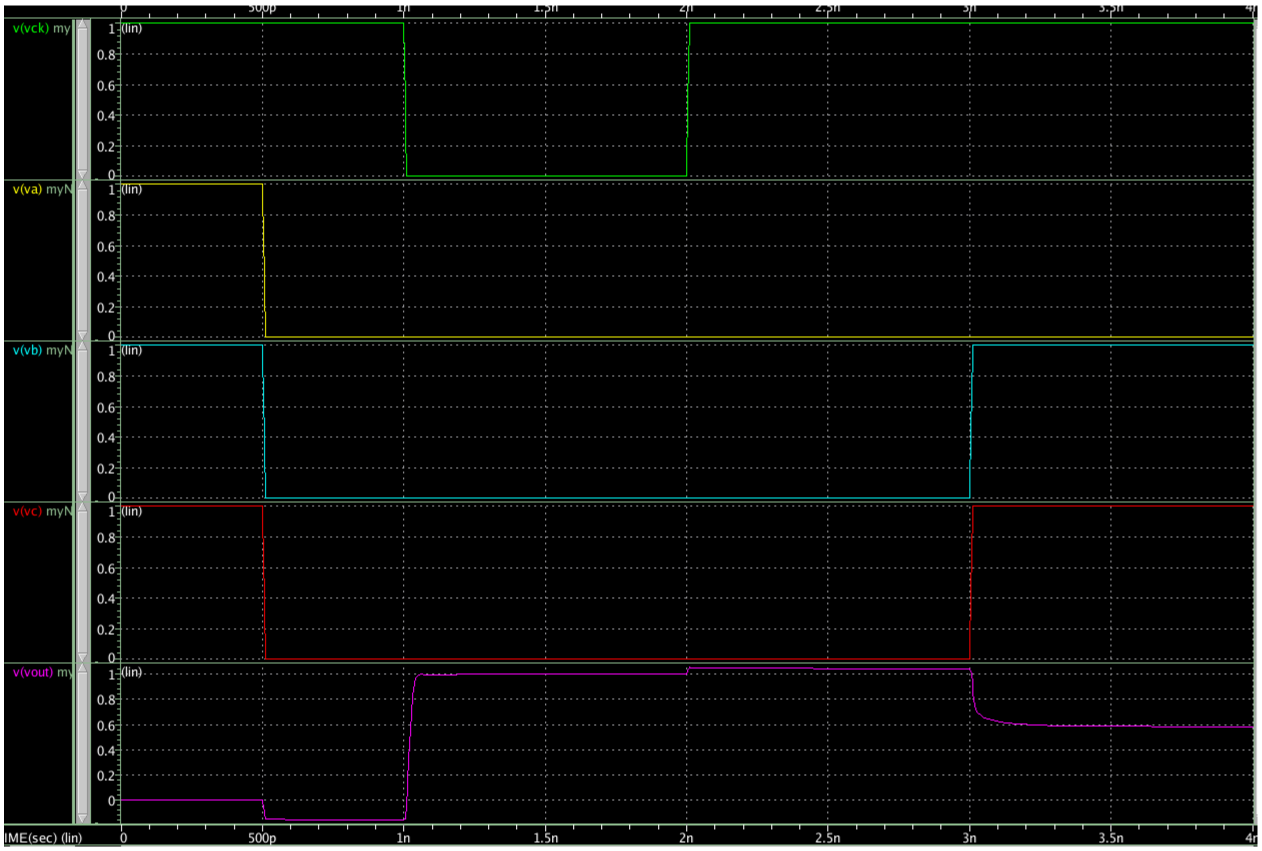
Waveform



C_{out} = 1fF ~ 10fF: V_{out} = 0.8V, 0.874V, 0.909V, 0.929V, 0.941V, 0.950V, …, 0.969V

- [**Submit**]
  - Vout when both C and B are set to V_{DD} for C_{out} =10fF, 9fF, …, 1fF.

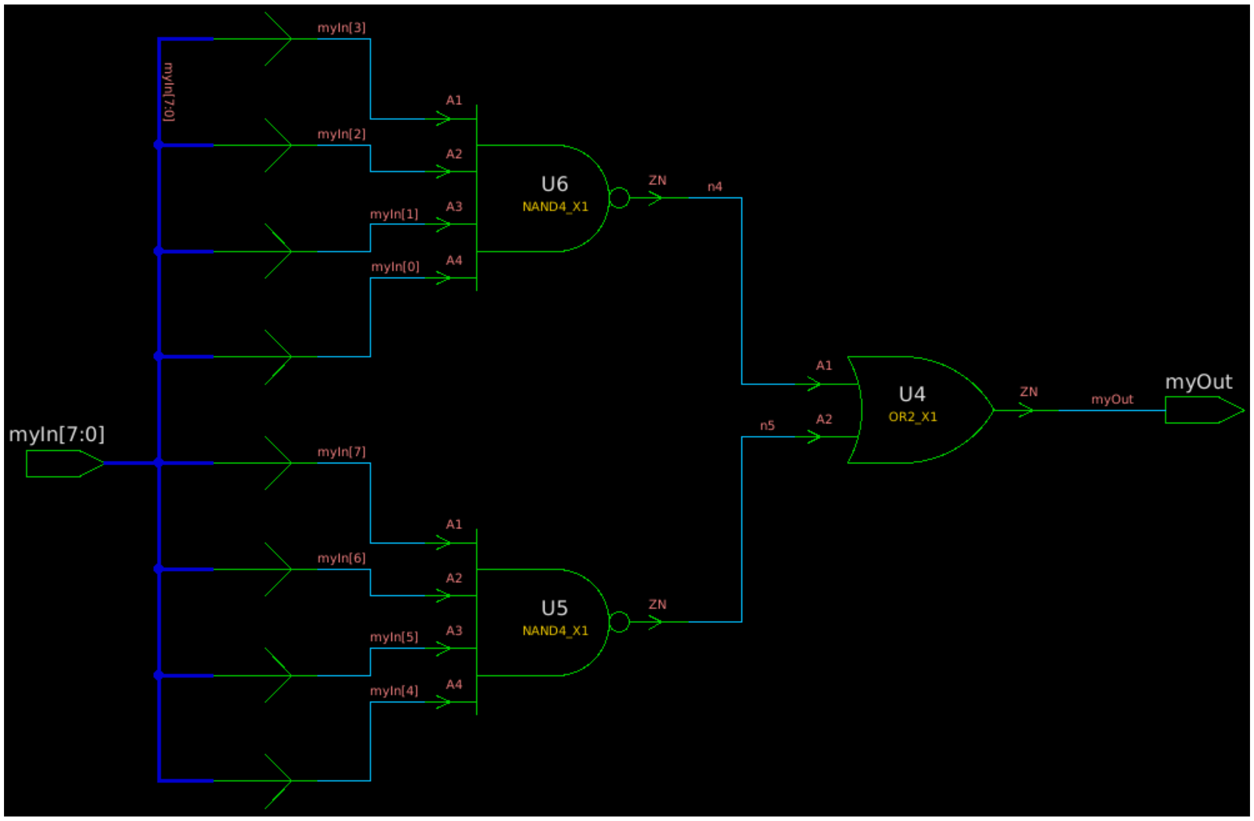
Waveform



C_{out} = 1fF ~ 10fF: V_{out} = 0.633V, 0.734V, …, 0.933V

3. [Synthesis, **20 points**]
   - In this problem, we will synthesize a netlist for a few gates.
   - Make sure you have the following files in your working directory.
     - NangateOpenCellLibrary_typical_ecsm.db
     - nand8.v
   - Source synopsys.sh.
   - Run Design Compiler (DC).
     - design_vision –no_gui
   - In DC, run the following commands.
     - set link_library {NangateOpenCellLibrary_typical_ecsm.db}
     - set target_library {NangateOpenCellLibrary_typical_ecsm.db}
     - read_file -format verilog {nand8.v}
     - compile -exact_map
     - write -format verilog -output nand8_mapped.v
     - exit
   - The two “set” statements set up target libraries.
   - “read_file” reads HDL source codes.
   - “compile” compiles (synthesizes) the source codes.

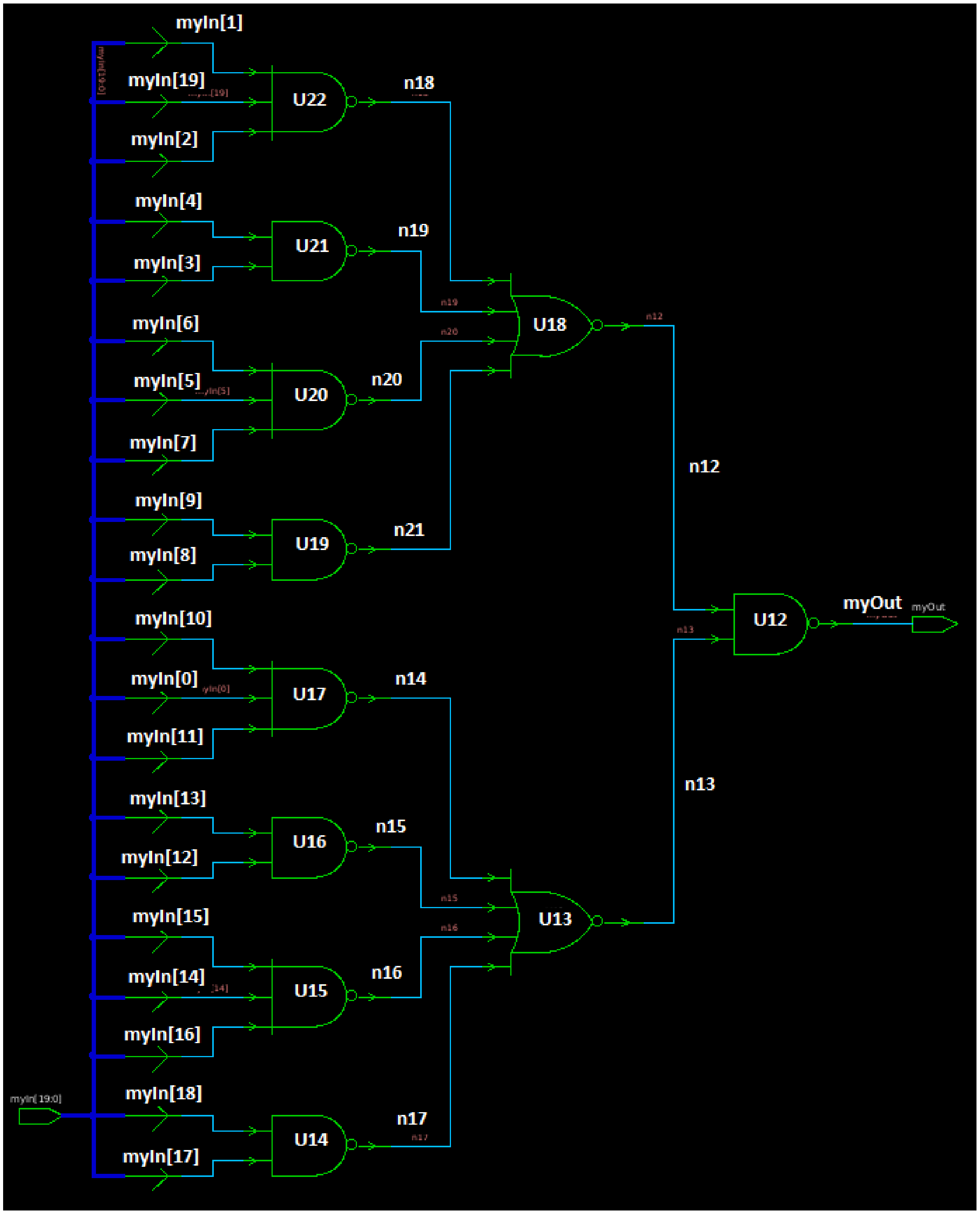
- “write” writes the synthesized code into the file specified after “-output”.
- Open “nand8.v” and see the function of the module.
- Open “nand8_mapped.v” and see the function of the module. Are they equal?
- [**Submit**] Draw a schematic for the netlist of “nand8_mapped.v”.



- Implement a 20-input nand gate by modifying “nand8.v” and synthesize it.
- [**Submit**] Draw a schematic for the netlist of the synthesized 20-input nand gate.

Code:

```
  module myNAND20 (myIn, myOut);

    input [19:0] myIn;

    output [19:0] myOut;

    assign myOut = !(myIn[0] && myIn[1] && myIn[2] && myIn[3] && myIn[4] && myIn[5]
&& myIn[6] && myIn[7] && myIn[8] && myIn[9] && myIn[10] && myIn[11] && myIn[12]
&& myIn[13] && myIn[14] && myIn[15] && myIn[16] && myIn[17] && myIn[18] &&
myIn[19]);

  endmodule
```

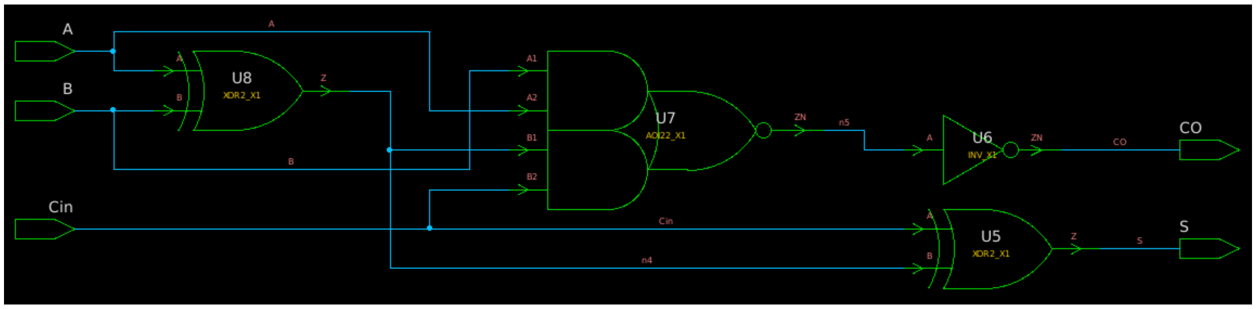


- Implement a full adder and synthesize it.
    - o Primary inputs: A, B, CI
    - o Primary outputs: S, CO

  - Use ^, &&, and || for XOR, logical AND, and logical OR operations in Verilog.
  - Use parentheses to prioritize the operations.
  - Use two assignments, one for S and the other for CO.
- [**Submit**] Draw a schematic for the netlist of the synthesized full adder.

Code

```
module myFA1 (A, B, Cin, S, CO);

 input A, B, Cin;

 output S, CO;

 assign S = A ^ B ^ Cin;

 assign CO = (A && B) || (B && Cin) || (Cin && A);

endmodule
```


A
B
Cin
A
B
U8
XOR2_X1
Z
A
B
A1
A2
B1
B2
U7
AOI22_X1
ZN
n5
A
U6
INV_X1
ZN
CO
CO
Cin
n4
A
B
U5
XOR2_X1
Z
S
S