CMPE 240 (DIGITAL SYSTEMS)

Lab Handbook

v. 0.2.1

Binnur GÖRER
Yavuz KÖROĞLU
İbrahim ÖZCAN
Alper ŞEN

April 7, 2018

DEPARTMENT OF COMPUTER ENGINEERING
BOĞAZIÇI UNIVERSITY

Contents

Chapter 1

Course Syllabus

Instructor: Alper Sen
Lectures: TThTh 578, BM A2
Lab Sections: FF 12, FF 34 BM 11 (HWLab)
Problem Session: T6 BM A2
Email: alper.sen@boun.edu.tr (technical/assignment questions must be directed to the class discussion group or TAs)
Office Hours: After class, BM 25
Teaching Assistants:
Binnur Görer Email: binnur.gorer@boun.edu.tr Office Hours: TBA, BM 37
Yavuz Köroğlu Email: yavuz.koroglu@boun.edu.tr Office Hours: TBA, BM 21
İbrahim Özcan Email: ibrahim.ozcan@boun.edu.tr Office Hours: TBA, BM 37
Required Textbook: Frank Vahid, Digital Design, Wiley, 2011
Frank Vahid, Verilog for Digital Design, Wiley, 2011
Supplemental Textbooks (Optional):
John F. Wakerly, Digital Design, Fourth Edition, Prentice Hall, 2006
Richard S. Sandige, Digital Design Essentials, Prentice Hall, 2002.
M. Morris Mano, Digital Design, Third Edition, Prentice Hall, 2002.
R.H. Katz, Contemporary Logic Design, Benjamin Cummings, 1994.
Course Website: moodle.boun.edu.tr
Tentative Grading:

- Labs: 25%
- Attendance+Quiz: 5%
- Midterm 1: 25%
- Midterm 2: 25%
- Final: 20%
- Total: 100%

Right to take the final exam:

- Each midterm grade > 15
- Lab grade average > 60

Course Organization: The course is organized as a series of lectures by the instructor, problem sessions and laboratory sessions by the TAs, reading, and exams. There will be two midterms and one final exam.
Class Discussion Forums: TBA You are to send all technical/HW/exam questions to the appropriate discussion forums. Only private emails to me.
Mailing List: Use Moodle

Chapter 2

Rules

2.1 Academic Honesty

Cheating will not be tolerated. You must not show your work to any student in the class. You must not search for solutions on the internet, copy, study, analyze or even look at any source code produced by other CMPE 240 students from this year, or from years past. We will check for cheating, and any incident will be reported to the department. Procedure for cheating behavior is given as follows:

2.1.1 Definitions

Cheating includes, but is not limited to, the following actions:

- Copying using unauthorized materials, copying from someone else's paper, allowing others to copy from his/her own paper, or giving unauthorized assistance to others during a test or examination.
- Working with others in completing a take-home examination or assignment unless the instructor has allowed independent action or allowed any specific aid could be accepted by students.
- Submitting of material in whole or part for academic evaluation that has been prepared by another student(s), individual(s) or community agency.
- Submitting, without prior permission of the instructor, any work by a student which has at any time been submitted in identical or similar form by that student in fulfillment of any other academic requirement at any institution.
- Marking or submitting an examination or evaluation material in a manner designed to deceive the grading system.
- Obtaining in a fraudulent manner any material relating to a student's academic work (theft of examination, collusion with a university employee, etc.).

- Attempting to influence or change an academic evaluation, grade, or record by unfair means.
- Permitting another student to substitute for one's self in any academic evaluation.
- Wilfully damaging the academic work or efforts of another student.
- Trying to deceive the instructor and/or assistant by any means such as denying the submission of a homework/project.

2.1.2 Actions Against

The following actions will be taken against all sides involved in cheating:

- The student will receive a zero or negative grade for the related test or evaluation. In addition, the student may receive the grade "F", or one letter less grade than the normal grade depending on the severity of cheating as judged by the evaluator, for the related course. For example, an organized cheating action in a final exam may result in "F" grade.
- The responsible instructor will file for the disciplinary action. Students must already be aware of related clauses of YÖK Disiplin Yönetmeliği.
- The student will not be issued any letter of reference by any instructor of the department.
- The student will not be accepted to any further graduate program such as the MS and PhD programs in the department.
- The student's name, the course, the year and the semester and the specific cheating behavior will be logged in a departmental database for further reference to the student. Regarding students of other departments, the respective department will also be informed of the specific cheating behavior.

2.1.3 Course Evaluation

Standard

2.1.4 Add/Drop Policy

Standard

2.2 Lab Regulations

1. **If you are late more than 15 minutes, you will not be allowed to participate.**
2. You should turn off your cellular phones before entering the laboratory and keep it in your bag. If we detect the usage of the phone, we will not warn you but you will get 0 points (NO EXCUSES).

3. Talking with another student except your partner will be counted as cheating.
4. Drinking and eating are not allowed (except water). You should not leave any trash on your desk.
5. Preliminary works will be graded and you have to submit the preliminary works to the online submission system before the deadline (NO EXCEPTIONS such as sending via mail after the deadline).
6. Do your experiment step by step as indicated in the lab manual or as requested by your assistants. You should prepare your preliminary work using the template files provided. Preliminary works prepared by using other formats are not accepted.
7. Untidy and ambiguous works will lose grades or might not be graded at all.
8. You are not allowed to bring any documents related to the course. Your preliminary works will be available in your lab computers.
9. You can not use soft resources like your computers as well as portable devices (USB etc.).
10. If any of the above rules are violated, we will not warn you but you will lose points. Hence, please respect the rules.
11. Both cheating and allowing one's own work to be cheated will be penalized severely. This applies for both written and experimental work, including projects and exams. Procedure for the cheating behavior of the department will be applied (see Section 2.1.2).

Chapter 3

Submissions

3.1 Submissions

1. You should submit your preliminary work for the experiment to:
   https://moodle.boun.edu.tr/

2. The report of the experiment should be arranged using the corresponding template provided to you. It will not be accepted otherwise. Please **only** include .pdf version of your report in the submissions.

3. In order to submit the project, you should **zip** the **verilog files** and the **report** with the name in the form of
   <SessionID>_<GroupID>_<StudentID1>_<StudentID2>_PRE<exp_num>.zip
   (Example: 1_12_2009101101_2009202202_PRE1.zip
   Group 12 in session 1 (FF12) for the 1^{st} experiment)
   (Example: 2_7_2009101101_2009202202_PRE3.zip
   Group 7 in session 2 (FF34) for the 3^{rd} experiment)

   The zip file should not contain any sub-folders.

4. Your submission should contain all verilog codes (.v files) and your report as a PDF file. Please do not put extra files (such as .vvp and .vcd files), since the maximum file limit is 1MB.

5. Submissions that do not obey any of the rules above will be graded zero.

6. **The deadline for submissions is Wednesday, 23.59.**

7. If a cheating is detected between two groups for the preliminary work, both groups will be banned from attending the following lab sessions.

Chapter 4

Important Notes for Report Format

You have to use the template provided to you while preparing the report. Untidy and ambiguous works will lose grades or might not be graded at all. Please don't try to rewrite the template file. Don't add any additional tables.

4.1 Circuit Drawing

For circuit drawing, you can use any of the following tools;

`http://www.digikey.com/schemeit` (online version is free)

`https://www.circuitlab.com/` (online demo version is free)

`http://www.circuit-diagram.org/` (free download)

`http://www.cburch.com/logisim/` (free download)

1. Be sure of that you show the inputs and outputs clearly.
2. Be sure of that all the wires are connected as they should be and your circuit is complete.
3. You do not have to draw the inside design of components which are used more than once. For example, if you are using adder for different functionalities, draw circuit for adder for once and name it. Then you can reference to it when you need adder in your design.
4. First draw small components, then combine them and get more complicated circuits.

4.2 Truth Table Ordering

While filling the truth table, you have to follow the input sequence given in the report template. Don't change the order of inputs. The truth table should also follow the numerical increase for inputs. An example truth table is shown in Table 4.2, where X,Y and Z are inputs and F is output.

X	**Y**	**Z**	**F**
0	0	0	X
0	0	1	X
0	1	0	X
0	1	1	X
1	0	0	X
1	0	1	X
1	1	0	X
1	1	1	X

4.3 General Rules for Notation

Please help us and be careful about the followings at all experiments and midterms:

1. Don't use !, $\sim$ and overline for negation (ex: $!X$, $\sim X$ or $\overline{X}$). Please use X'.
2. For the boolean expression minimization staff, don't ever use two rules in one step.
3. Don't minimize your equations by dividing them into several parts. Please use the given inference rules on the whole expression.
4. Don't use rules like "same process again". Write the exact name of the rule at each step even if you used it above steps.
5. Please be careful about the hierarchy and gate/behavioral levelness of your code. Never code using both.

Chapter 5

Tools used in the Experiments

5.1 Icarus Verilog

Icarus Verilog is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can generate an intermediate form called vvp assembly. This intermediate form is executed by the "vvp" command.
Link : `http://bleyer.org/icarus/`.
Installation Guide : `http://iverilog.wikia.com/wiki/Installation_Guide`

5.2 GTKWave

GTKWave is a fully featured GTK+ based wave viewer for Unix, Win32, and Mac OSX which reads Verilog VCD/EVCD files.
Link : `http://gtkwave.sourceforge.net`

We recommend that you get familiar with all the tools above. Any text editor is sufficient to write Verilog code, but you should prefer one that supports correct syntax highlighting such as Notepad++ or Gedit. Please execute the following instructions to compile your source codes, generate the timing diagram and view the timing diagram, respectively. Let's say you created a testbench file like testbench.v and a source verilog code like source.v

```
iverilog -o output.vvp testbench.v source.v
vvp output.vvp
gtkwave TimingDiagram.vcd
```

Chapter 6

Experiments

Table of Experiments

- Hardware Implementation of a Combinational Circuit
- Analysis and Implementation of a Problem Definition
- Implementation of a Boolean Expression
- Implementation of a Regular Expression Detector
- Analysis of a Sequential Circuit
- Arithmetic Logic Unit

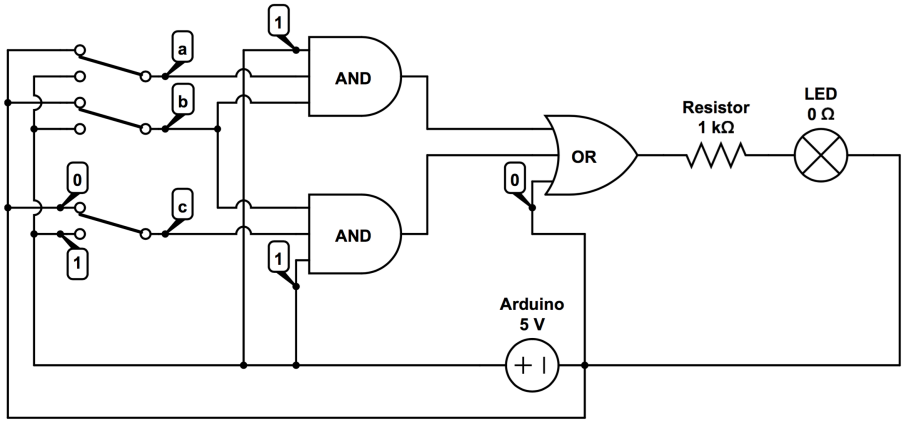
6.1 Experiment 1 (Hardware Implementation of a Combinational Circuit)

6.1.1 Aim

In this experiment, your knowledge about the working principles of the combinatorial logic gates which is explained in section 2.4 of the course book will be tested.

6.1.2 Problem

You will implement the following circuit on a breadboard.

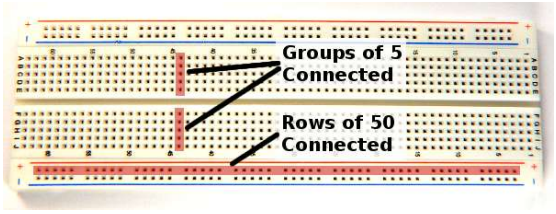


6.1.3 Preliminary Work

Before the experiment, you should study the working principles of all circuit components described below.

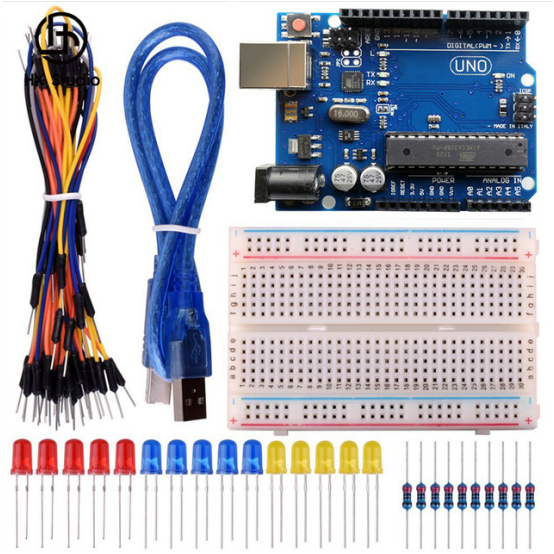
BreadBoard

To understand breadboards, we recommend you to watch the video tutorial at
`https://youtu.be/fq6U5Y14oM4`



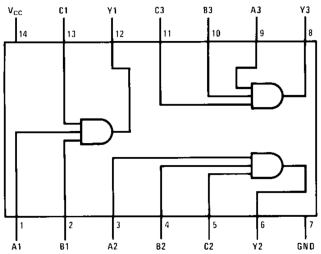
A breadboard is a board on which you can put circuit components and connections. There are two rows at the top and two rows at the bottom. These rows are power rails and each row is internally connected, i.e. they will always have the same voltage. Rows

with plus sign (+) expect a positive voltage source. Rows with minus sign (-) expect a negative voltage source. In the middle, there are 63 columns and 10 rows. Each of these columns is internally connected but divided by a center divider. We put chips s.t. one side of the chip is connected to one side of the center divider and the other side of the chip is connected to the other side of the center divider.

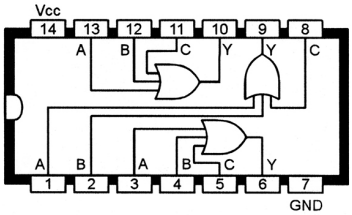


Along with the breadboard, we provide one Arduino as a $5V$ power source, and one USB cable to power the Arduino. We also provide sufficient amount of connector cables/wires, LED lamps, switches, and $1k\Omega$ resistors. Finally, we provide one AND gate and one OR gate. We show the diagrams of these gates below.

3-input AND Gate (74LS11N)



3-input OR Gate (74HCT4075E)



6.1.4 Lab Work

1. Request one Arduino, one USB connector, one 74LS11N chip, one 74HCT4075E chip, one resistor, one led lamp, three switches, and 30 connector cables/wires.
2. Implement and then demonstrate to the assistants the circuit on the breadboard.
3. Answer the questions on the lab sheet.

6.2 Experiment 2 (Analysis and Implementation of a Problem)

6.2.1 Aim

In this experiment, you will design and implement a circuit by analysing the given problem. Please refer to Chapter 4 of the book for more details.

6.2.2 Problem

A robot which plays soccer has two different behavior modes(b): defensive or offensive. In the defensive mode, the robot positions itself in its own halfcourt and supports other team members. In the offensive mode, the robot acts in a way to gain ball control and score a goal. We know that the behavior mode of the robot(b) (1:defensive, 0:offensive) is determined by three factors:

i_0: Distance of the robot to the ball (1: far, 0: near).

i_1: If the golie of the opponent team is penalized (1: not penalized, 0: penalized).

i_2: If the total number of offensive robots in the team is larger than 2 (1: (total number>2) , 0: (total number<=2)).

You will create and implement the circuit design of the behavior mode b in terms of its inputs i_0, i_1, and i_2. The situation of the game described by the above three factors determines the behavior mode of the robot as follows;

- If the total number of offensive robots is larger than 2, and the goalie of the opponent team is penalized, and the ball is far, then the behavior is defensive.
- Else if the ball is far, and the goalie of the opponent team is penalized, then the behavior is offensive.
- Else if the ball is far, then the behavior is offensive.
- Else if the total number of offensive robots is larger than 2, or the goalie of the opponent team is penalized, then the behavior is defensive.
- Otherwise, the behavior is defensive.

You have only 2 AND, 1 OR, and 3 NOT gates at your disposal. The AND and OR gates are binary, i.e. they take exactly two inputs. You are expected to design the circuit which uses the fewest number of gates as possible. Designs with the fewest number of gates will get the highest grades.

6.2.3 Preliminary Work

You should prepare the following materials for the preliminary work:

1. Fill the truth table given in the template according to the design requirements.
2. Write the sum of products (SOP) expression for the truth table.
3. Find the minimized SOP expression by using Boolean theorems and axioms (show your work clearly, minimize the equation you previously found in step 2 using the laws used in class, using exactly one law per line). The minimized expression must also be in SOP form.
4. Write the product of sums (POS) expression for the truth table.
5. Find the minimized POS expression by using Boolean theorems and axioms (show your work clearly, minimize the equation you previously found in step 4 using the laws used in class, using exactly one law per line). The minimized expression must also be in POS form.
6. Draw the circuit for the minimized expression using only binary (which means it has two inputs) AND, OR, and NOT gates.
7. Write the **gate** level verilog code for the circuit chosen at the previous step. You can use the built-in AND, OR, and NOT gates.
8. Write the code for the testbench in order to test all possible input combinations.
9. Verify that the outputs of your implementation is the same as the truth table.

Submit your code and report considering the submission rules described in Section 3.1. One submission per group is required.

6.2.4 Lab Work

1. Repeat all the procedures done in the preliminary work for the problem given in the lab section. Fill the report, write the Verilog code, and demonstrate your implementation to the assistants.

6.3 Experiment 3 (Implementation of a Boolean Expression)

6.3.1 Aim

In this experiment, your knowledge to minimize the number of gates by using decoder and multiplexer gate.

6.3.2 Problem

A circuit takes five inputs denoted as y_1, y_0, x_2, x_1, and x_0. This circuit has only one output z. The output is determined as shown in the following table.

y_1	y_0	z
0	0	PRIME(x_2, x_1, x_0)
0	1	ODD(x_2, x_1, x_0)
1	0	FIBONACCI(x_2, x_1, x_0)
1	1	SQUAREROOT(x_2, x_1, x_0)

We define PRIME, ODD, FIBONACCI, and SQUAREROOT as follows. $x_2x_1x_0$ represents a three bit unsigned number.

$$PRIME?(x_2, x_1, x_0) = \begin{cases} 1 & x_2x_1x_0 \text{ is prime} \\ 0 & \text{otherwise} \end{cases}$$

$$ODD?(x_2, x_1, x_0) = \begin{cases} 1 & x_2x_1x_0 \text{ is odd} \\ 0 & \text{otherwise} \end{cases}$$

$$FIBONACCI?(x_2, x_1, x_0) = \begin{cases} 1 & x_2x_1x_0 \text{ is an element of Fibonacci Sequence} \\ 0 & \text{otherwise} \end{cases}$$

$$SQUAREROOT?(x_2, x_1, x_0) = \begin{cases} 1 & \text{The square root of } x_2x_1x_0 \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

In the implementation, you are only allowed to use one Decoder, one Multiplexer, two binary logic gates (standard gates like AND, OR, XOR, XNOR, NAND, NOR) and NOT gates (as many as needed). Multiplexer can have at most 8 input (i.e. 8 way mux), and decoder can have at most 3 input (i.e. 3 to 8 decoder).

Note: For example, $x_2x_1x_0$ = 010 (2) occurs in Fibonacci Sequence. Therefore for y_1y_0 = 10, z should be 1.

Note: For example, The square root of $x_2x_1x_0$ = 100 (4) is 2 and it is an integer. Therefore for y_1y_0 = 11, z should be 1.

6.3.3 Preliminary Work

Before the experiment, you should prepare following materials:

1. Fill the truth table.

2. Analyze the expression and make your design using the allowed components to implement the expression. Then, draw the circuit of your design. If you will use other components than the allowed ones or more than the allowed numbers, you will lose point.

3. Write a verilog code for the circuit drawn in the previous step. Verilog code should have two components. First, you have to write behavioural level verilog code which implements the functionality of multiplexer and/or decoder. Secondly, you have to write the gate level verilog code for your circuitry which uses the implemented components and additional **built-in** gates (i.e. *AND*, *OR*, *NAND*, *NOR*, *XOR*, or *XNOR*).

4. Write the verilog code for the testbench in order to test all possible input combinations.

Then, follow the steps described in Section 3.1 to submit your preliminary work.

6.3.4 Lab Work

1. Load your submitted code given by the assistants and demonstrate that the circuit implements the truth table of the boolean expression assigned to your lab section by using the testbench waveforms you have generated.

2. Do the same procedure as in preliminary work for the problem definition given in the lab section and demonstrate your implementation.

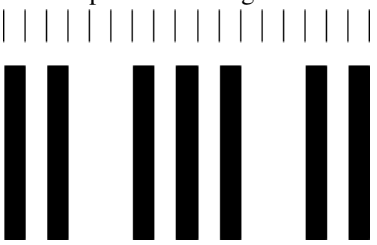
6.4 Experiment 4 (Implementation of a Bar Code Reader)

6.4.1 Aim

In this experiment, your knowledge to design a sequential circuit will be tested. Please refer to book chapter 7 for more details.

6.4.2 Problem

A bar code represents bits as alternating white and black bands. The black bands are of uniform width, while the white bands have width either equal to, or triple of, the width of the white bands. Below is an example of a code-word using the bar code. The tick marks on top show the single widths.



Assume that a bar-code reader translates the bands into symbols, B for black, W for white, one symbol per single width. Thus the symbol sequence for the code-word above would be

B W B W W W B W B W B W W W B W B

A bar pattern represents a binary sequence as follows: a 0 is encoded as BWB, while a 1 is encoded as BWWW. The code can start only with B so the finite state machine will idle until it receives B. After that, when it receives its first W, it knows that the code has started. The transducer will give output 0 or 1 as soon it has determined the next bit from the bar pattern. If the bit is not known yet, it will give output _ . Thus for the input sequence above, the machine will produce;

_ _ 0 _ _ 1 _ _ 0 _ 0 _ _ 1 _ _ 0

B W B W W W B W B W B W W W B W B

The machine will output the symbol end when it subsequently encounters two B's in row, at which point it will return to its initial state.
The machine will output _ if an unexpected sequence occurs. For example, see the following sequence;

_ _ 0 _ _ _ _ _ 1 _ _ 0

B W B W W B W W W B W B

Note: The machine starts with reset input. Active-high and synchronous reset is used.

Hint: You need to find and apply a suitable input/output representation to convert symbol inputs/outputs to binary inputs/outputs.

6.4.3 Preliminary Work

You should apply and report 5-step controller process explained in the class as follows:

1. Capture the FSM: Create finite state machine that describes the desired behavior of the controller.
2. Create the architecture: Create standard architecture by using a state register (of the appropriate width) and combinational logic.
3. Encode the states: Assign a unique binary number to each state. Each binary number representing a state is known as an encoding. Any encoding is acceptable as long as each state has a unique encoding.
4. Create the state table: Create a truth table for the combinational logic such that the logic will generate the correct FSM outputs and next state signals. Ordering the inputs with state bits first makes this truth table describe the state behavior, so the table is a state table.
5. Implement the combinational logic: Implement the combinatorial logic using any method.
6. Write the Verilog code of the regular expression detector. You are free to write in behavioral or gate level code.
7. Write the Verilog code for the testbench waveform in order to test possible input sequences. Use at least five different 32 symbols long or longer sequences for your testbench.
8. Verify the functionality of your implementation.
9. Verify that the detector detects all the example inputs given above.

Submit your code and report considering the submission rules described in Section 3.1. One submission per group is required.

6.4.4 Lab Work

1. Repeat all the procedures done in the preliminary work for the problem given in the lab section. Fill the report, write the Verilog code, and demonstrate your implementation to the assistants.

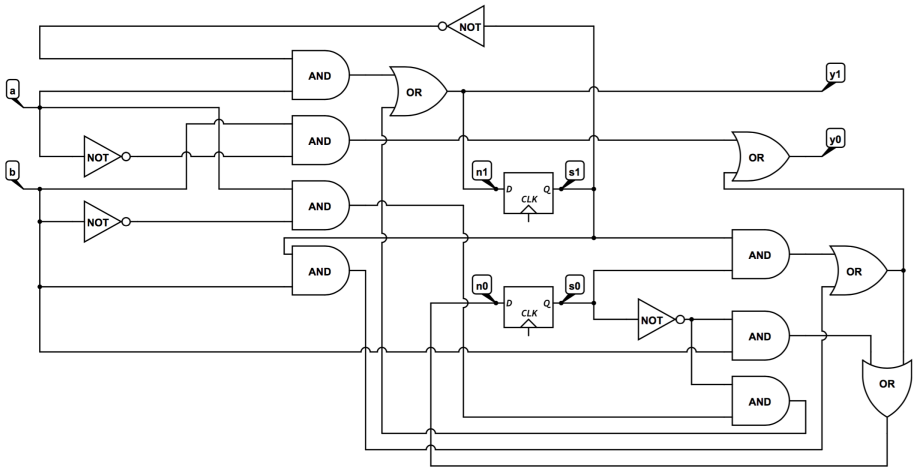
6.5 Experiment 5 (Analysis of a Sequential Circuit)

6.5.1 Aim

In this experiment, your will analyze a given sequential circuit.

6.5.2 Problem

In the circuit below, the initial state is $s_1 = 0, s_0 = 0$. a and b are inputs. y_1 and y_0 are outputs. Reset is synchronous.



6.5.3 Preliminary Work

1. State the inputs and outputs of the state registers.
2. State the inputs and outputs of the combinational block of the sequential circuit.
3. Draw the truth table for the combinational circuit.
4. Draw the finite state machine by using the table obtained in previous step.
5. Is this a Moore or Mealy Machine? (No explanation, only short answer)
6. Write the testbench and the **behavioral level** verilog code for the corresponding finite state machine.

6.5.4 Lab Work

1. Demonstrate your testbench for the preliminary work.
2. A new sequential system will be given. You are required to follow the same steps with the preliminary work for the new sequential circuit.

6.6 Experiment 6 (Arithmetic Logic Unit)

6.6.1 Aim

In this experiment, your knowledge to implement an Arithmetic Logic Unit (ALU) which is explained in section 4.9 of the course book will be tested.

6.6.2 Problem

You will implement verilog code for a 5-bit arithmetic logic unit (ALU). In ALU, there will be 2-bit select input (S), 5-bit data inputs (X, Y), 5-bit output (F), 1-bit output (C_{out}), 1-bit output ($Overflow$), and the functional table will be as follows:

S	Negative	Overflow	Zero	C_{out}	**F**	**Representation**
00	0	0	1	X ! = Y	00000	Unsigned
01	0	0	X[5:3] $\times$ Y[3:1]			Unsigned
10	X + (Y[3:0] / 2)					2's Complement
11	X + Y + 1					2's Complement

Note:

- Let's say Y = 10110; if the representation is unsigned, Y[2:0] will be 6 in decimal; if the representation is 2's complement signed, Y[2:0] will be -2 in decimal.
- Overflow output is only valid for the operations performed in 2's Complement representation (namely third and fourth operation shown in the above table). For other cases, it is always 0.
- Zero output will be 1 when the F value become 0.
- Negative output will be 1 when the F value become negative for 2's complement operations.
- Consider overflow only for addition and subtraction.
- For the S = 10, use Y[3:0] as 4 bit an unsigned number.

6.6.3 Preliminary Work

Before the experiment, you should prepare the following materials:

1. Design circuits with minimum number of components for each function of the ALU.
2. Merge all operations with select inputs and organize outputs.
3. Try to minimize your implementation by using repetitions (see the hint below).
4. Draw final circuit as the final design of the ALU with corresponding. functional tables and calculate the number of chips in your design.

5. Write the behavioral level verilog code of the ALU.
6. Write the verilog code for the testbench waveform in order to test all possible input combinations.
7. Verify the functionality of your implementation.

Then, follow the steps described in Section 3.1 to submit your preliminary work.

Hint: You can use one operation to implement another operation. For example, you can use subtraction for the comparison operation or addition for the multiplication operation.

6.6.4 Lab Work

1. Load your submitted code given by the assistants and demonstrate that the verilog code works with the same principles as the ALU assigned for your lab.
2. Do the same procedure as in preliminary work for the problem definition given in the lab section and demonstrate your implementation