Lab 2 Advanced: FPGA Implementation of 3-SAT Solving

Submission Due Dates:

Demo:	2025/09/23 17:20
Source Code:	2025/09/23 18:30
Report:	2025/09/28 23:59

Objectives

- Getting familiar with Verilog combinational/sequential behavior modeling.
- Practicing switch and LED control on the FPGA board.

Introduction to the SAT problem

- ∨ (OR), ∧ (AND), ¬ (NOT) are the Boolean operators.
- **Literal**: A literal is either a Boolean variable (e.g., A) or its negation (e.g., ¬A). For example, in the expression (A ∨ ¬B), both A and ¬B are literals.
- **Clause**: A clause is a disjunction (logical OR, denoted ∨) of one or more literals. For example, (A ∨ ¬B ∨ C) is a clause with three literals.
- **SAT Problem**: Given a Boolean formula (a conjunction of clauses) and determine whether there exists an assignment of true/false values to its variables that makes the entire formula true. If such an assignment exists, the formula is said to be **satisfiable**.
- **3-SAT problem** is that there are exactly 3 literals in each clause.

Examples for 3-SAT problem:

1. Satisfiable:

   $$\underbrace{(A \lor B \lor \neg C)}_{\text{clause1}} \land \underbrace{(\neg A \lor B \lor D)}_{\text{clause2}} \land \underbrace{(\neg C \lor \neg D \lor E)}_{\text{clause3}}$$

   In this example, one satisfying assignment: A=1, B=0, C=1, D=1, E=1.
2. Unsatisfiable:

   $$\underbrace{(A \lor A \lor A)}_{\text{clause1}} \land \underbrace{(\neg A \lor \neg A \lor \neg A)}_{\text{clause2}}$$

   In this example, no assignment for A can satisfy both clauses.

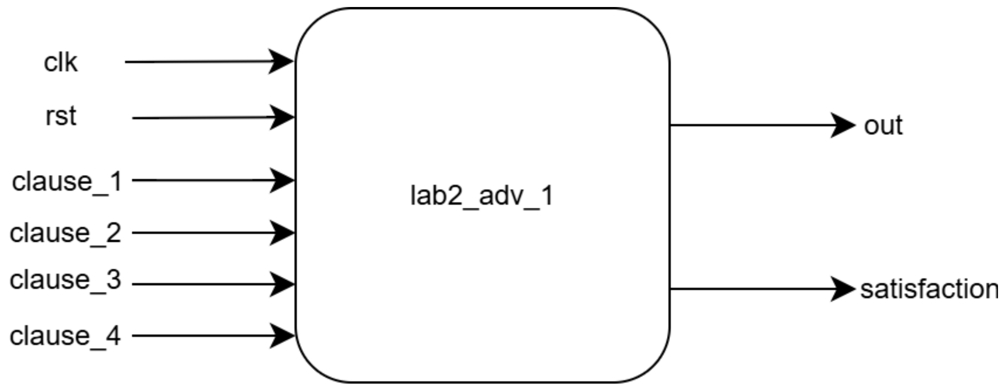
For more details, see Wikipedia: Boolean Satisfiability Problem:
https://en.wikipedia.org/wiki/Boolean_satisfiability_problem

Action Items

1. lab2_adv_1.v (35%)

Design a circuit that solves a 3-SAT problem with 4 clauses using brute force.

- Try all 2^3 possibilities (from 3'b000 to 3'b111) in one cycle to determine if the formula is satisfiable
- Output the first satisfying Boolean assignment, if it exists.



a. IO List and Specification

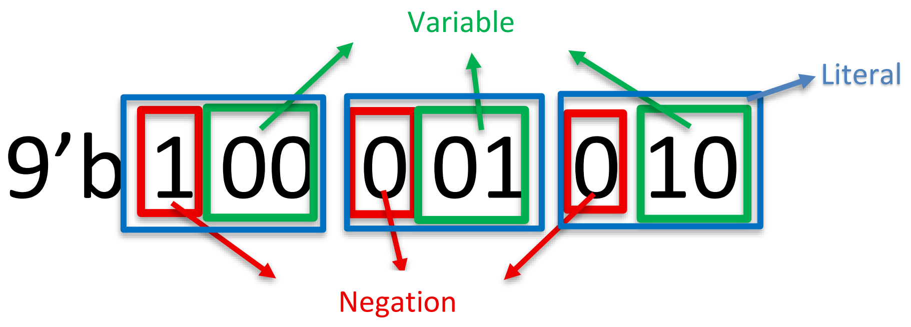
Input Signals	Bit Width
clk	1
rst	1
clause_1	9
clause_2	9
clause_3	9
clause_4	9

Output Signals	Bit Width
out	3
satisfaction	1

- clk:
  - A 100 MHz clock.
  - All outputs should change at positive edges.
- rst:
  - Synchronous active-high reset; when rst == 1'b1, reset all outputs to 0
- clause_1, clause_2, clause_3, clause_4:
  - There are 3 variables (A, B, and C), encoded by 2'b00, 2'b01, 2'b10, respectively. We use one additional bit to indicate whether the variable is negated. Therefore, each literal is represented using 3 bits: the leftmost bit indicates negation, and the two rightmost bits represent the variable ID (00→A, 01→B, 10→C).

For example, 3’b010 corresponds to C, 3’b101 corresponds to ¬B.

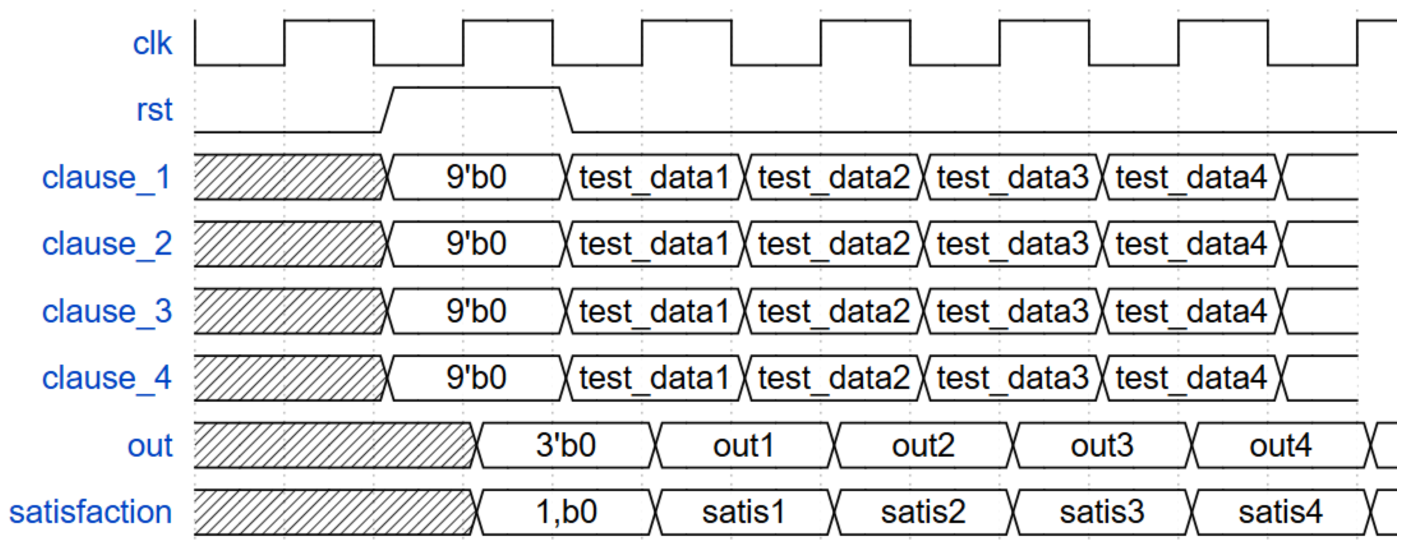
- Each clause contains three literals. Therefore, we use 9 bits to represent a clause. For example, the clause 9’b100_001_010 corresponds to ¬A ∨ B ∨ C.



- Note that clauses may update at the negative clock edge.

- out:
  - You need to test the Boolean assignment from 3’b000 to 3’b111, when the design finds the **first** Boolean assignment for the variables that satisfies the entire Boolean formula (clause_1∧ clause_2∧ clause_3∧ clause_4), the **out** signal indicates the 3-bit assignment.
  - The variable A, denoted by 2'b00, corresponds to out[0]. The variable B, denoted by 2'b01, corresponds to out[1]. The variable C denotes by 2'b10 corresponds to out[2]. **(e.g., 3'b100 for C=1, B=0, A=0; 0 for false and 1 for True)**.
  - Otherwise, **out** should be 3'b111.
  - The **out** signal updates at the **positive clock edge**.
- satisfaction:
  - When the FPGA finds a satisfying assignment, satisfaction is set to 1'b1.
  - Otherwise, it remains 1'b0.
  - The **satisfaction** signal updates at the **positive clock edge**.

b. Waveform example for the input/ouput



- After rst goes low, the test begins, and input data (clauses) are updated at the negative clock edge.
- Use the brute-force method to evaluate the formula by testing all possible Boolean assignments within one cycle.
  - Evaluate the Boolean assignment from 3’b000, 3’b001, 3’b010, to 3’b111.
- After the negative edge (when inputs are updated), update the outputs (out and satisfaction) at the next positive clock edge.
- Therefore, the circuit produces a new result every cycle, and each output value remains valid for exactly one cycle.

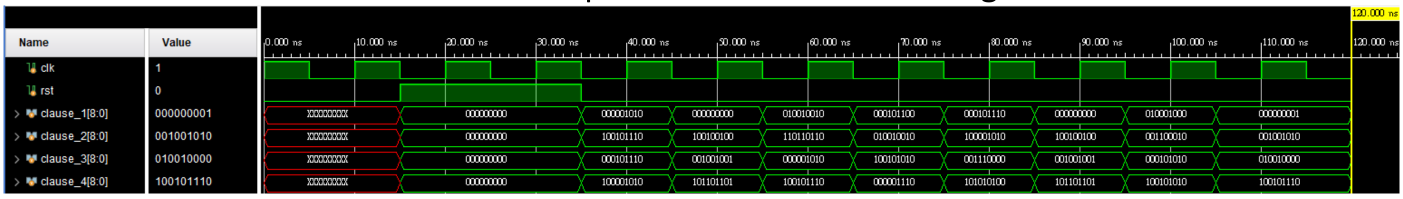
c. You must use the following template for your design

```
`timescale 1ns/1ps
module lab2_adv_1 (
    input clk,
    input rst,
    input [8:0] clause_1,
    input [8:0] clause_2,
    input [8:0] clause_3,
    input [8:0] clause_4,
    output reg satisfaction,
    output [2:0] out
);

    // Output signals can be reg or wire
    // add your design here

endmodule
```

2. lab2_adv_1_t.v (15%)

Write your own testbench (lab2_adv_1_t.v) to verify the design (lab2_adv_1.v). In your report, please explain how you built your testbench.

The testbench must at least cover the patterns as in the following waveform:


Name
Value
clk 1
rst 0
clause_1[8:0] 000000001
clause_2[8:0] 001001010
clause_3[8:0] 010010000
clause_4[8:0] 100101110
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.000 ns
90.000 ns
100.000 ns
110.000 ns
120.000 ns
clause_1: XXXXXXXXX 000000000 000001010 000000000 010010010 000101100 000101110 000000000 010001000 000000001
clause_2: XXXXXXXXX 000000000 100101110 100100100 110110110 010010010 100001010 100100100 001100010 001001010
clause_3: XXXXXXXXX 000000000 000101110 001001001 000001010 100101010 001110000 001001001 000101010 010010000
clause_4: XXXXXXXXX 000000000 100001010 101101101 100101110 000001110 101010100 101101101 100101010 100101110

Brief summary of the testbench (refer to the waveform above):

- The duration of the simulation is 120ns.
- The reset signal (rst) must be activated at 15ns and last for 20 ns.
- Test data: 8 sets of clause inputs (provided in the template).
- The evaluation starts at 35ns (when rst goes down).

- The inputs (‘clause_1’, ‘clause_2’, ‘clause_3’, and ‘clause_4’) change at negative clock edges.
- The outputs (‘out’ and ‘satisfaction’) change their values at positive clock edges.

3. lab2_adv_2.v (50%)

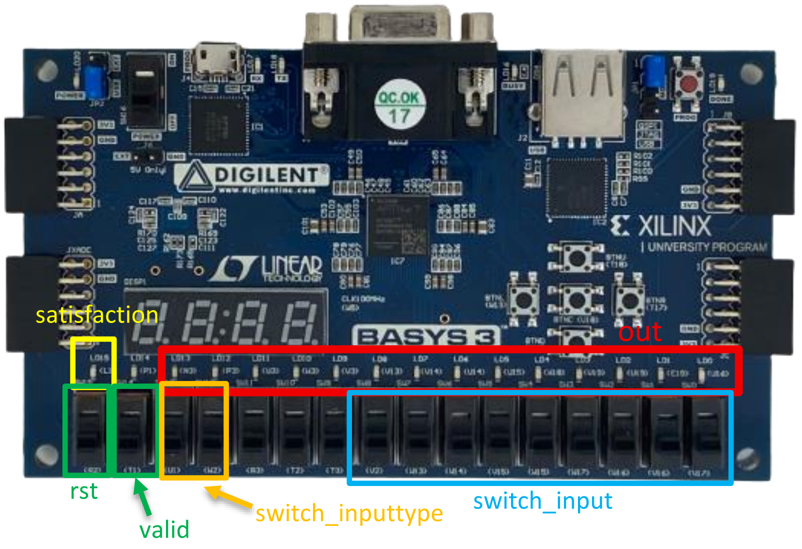
Implement a 3-SAT verification system on the FPGA.

- **Inputs**: Three clauses and a Boolean assignment entered via the switches.
- **Outputs**: Satisfaction result displayed by the LEDs.

a. IO List and Specification

Input Connection	
rst	Connected to SW15
valid	Connected to SW14
switch_inputtype	Connected to SW13-SW12
switch_input	Connected to SW8-SW0

Output Connection	
out	Connected to LD13-LD0
satisfaction	Connected to LD15



- rst
  - Asynchronous active-high reset; when rst == 1’b1, reset outputs to 0.
- valid
  - If valid == 1'b1, load switch inputs into the design.
  - If valid == 1’b0, no matter how the switches are changed, the values of the clauses and Boolean assignment remain unchanged.
- switch_inputtype
  - Determines which clause is updated by the value of switch_input:
    a. 2’b00 and valid==1: load clause_1 (9-bit value from SW8-SW0).

   b. 2’b01 and valid==1: load clause_2 (9-bit value from SW8-SW0).
   c. 2’b10 and valid==1: load clause_3 (9-bit value from SW8-SW0).
   d. 2’b11 and valid==1: load the Boolean assignment (3-bit value from SW2-SW0; SW8-SW3 unused).
  - Once the value of switch_inputtype changed, the previous input should be stored if valid==1’b1.
- switch_input
  - Clause input (9 bits: SW8-SW0):
    Where SW8 is the most significant bit and SW0 is the least significant bit. For example: 9’b000_101_010 (SW8, SW7…, SW1, SW0)
  - Boolean assignment (3 bits: SW2-SW0):
    a. Variable A → SW0
    b. Variable B → SW1
    c. Variable C → SW2
    For example, if SW0 is on (SW0=1), A=1.
  - Each clause has an initial value of 9’b000_000_000; The Boolean assignment has an initial value of 3’b000.
- satisfaction
  - LD15 is set to 1'b1 if all three clauses are satisfied.
  - Otherwise, LD15 is set to 1'b0.
- out: (per-clause satisfaction)
  - If Clause 1 is satisfied, LD13-LD10 = 4’b1111.
  - If Clause 2 is satisfied, LD8-LD5 = 4’b1111.
  - If Clause 3 is satisfied, LD3-LD0 = 4’b1111.
  - LD9 and LD4 remain 1’b0.

b. Example Operations

Step	Action	Register Values	out	satisfaction
1	SW15=1	Clause1: 000_000_000 Clause2: 000_000_000 Clause3: 000_000_000 Boolean Assignment (BA): 000	00000000000000	0
2	SW15=0	Clause1: 000_000_000 Clause2: 000_000_000 Clause3: 000_000_000 Boolean Assignment (BA): 000	00000000000000	0

3	SW14=1
	SW2=1
	(Clause 1)
		Clause1: 000_000_100
		Clause2: 000_000_000
		Clause3: 000_000_000
		BA: 000
			11110000000000	0

4	SW14=0
	SW2=0
	SW12=1
	SW5=1
	SW14=1
	(Clause 2)
		Clause1: 000_000_100
		Clause2: 000_100_000
		Clause3: 000_000_000
		BA: 000
			11110111100000	0

5	SW14=0	Clause1: 000_000_100	11110111101111	1

	SW5=0
	SW12=0
	SW13=1
	SW8=1
	SW14=1
	(Clause 3)
		Clause2: 000_100_000
		Clause3: 100_000_000
		BA: 000		

6	SW14=0
	SW8=0
	SW12=1
	SW0=1
	SW14=1
	(Boolean Assignment)
		Clause1: 000_000_100
		Clause2: 000_100_000
		Clause3: 100_000_000
		BA: 001
			11110111101111	1

7	SW14=0
	SW0=0
	SW13=0
	SW2=1
	SW5=1
	SW8=1
	SW14=1
	(Clause 2)
		Clause1: 000_000_100
		Clause2: 100_100_000
		Clause3: 100_000_000
		BA: 001
			11110000001111	0

- **Steps 1 and 2:**
  a. Reset the value of clauses, Boolean assignment, and output signals to 0.
- **Step 3:**
  a. Set SW14 (valid bit) to 1. Now the value of SW8-SW0 corresponds to Clause 1 (switch_inputtype=2'b00).
  b. Then set SW2 to change the value of Clause 1.
  c. Since the Boolean assignment (3'b000) satisfies Clause 1, LD13-LD10 is 4'b1111.
- **Step 4:**
  a. Set SW14 (valid) to 0. Then set SW2=0.
  b. Set SW12=1. Now switch_inputtype=2'b01 (Clause 2).
  c. Set SW5=1 and SW14=1 to change the value of Clause 2.
  d. Since the Boolean assignment (3'b000) satisfies Clause 2, LD8-LD5 is 4'b1111.
- **Step 5:**
  a. Set SW14=0 and then SW5=0.
  b. Set SW12=0 and SW13=1 (switch_inputtype=2'b10: Clause 3).
  c. Set SW8=1 and then SW14=1 to change the value of Clause 3.
  d. Since the Boolean assignment (3'b000) satisfies Clause 3, LD3-LD0 is 4'b1111. Moreover, because all the 3 clauses are satisfied, the satisfaction bit becomes 1.
- **Step 6:**
  a. Set SW14=0 and then SW8=0.
  b. Set SW12=1 (switch_inputtype=2'b11: Boolean assignment).
  c. Set SW0=1 and then SW14=1 to change the Boolean assignment to 3'b001. Although we have changed the Boolean assignment, the assignment still satisfies all 3 clauses. Therefore, the satisfaction is still 1, and the out signal is still 14'b11110111101111.
- **Step 7:**

a. Set SW14=0 and then SW0=0.
b. Set SW13=0 (switch_inputtype=2'b01: Clause 2).
c. Set SW2=SW5=SW8=1 and then SW14=1 to change the value of Clause 2.
d. Since the Boolean assignment (3'b001) doesn't satisfy Clause 2, LD8-LD5 is 4'b0000.

c. You must use the following template for your design

```
module lab2_adv_2 (
   input clk,
   input rst,
   input [8:0] switch_input,
   input [1:0] switch_inputtype,
   input valid,
   output reg [13:0] out,
   output reg satisfaction
);
// Output signals can be reg or wire
// add your design here

endmodule
```

4. Questions & Discussion

Please answer the following questions in your report.

1. What are the differences between a combinational circuit and a sequential circuit? Please explain how each of them works in detail.
2. For an **unsatisfiable 3-SAT problem**, how would you extend your Verilog design to solve the **MAX-SAT** problem (find a Boolean assignment that satisfies the maximum number of clauses)?
3. In lab2_adv_2, what **unexpected behavior** might occur if the valid signal is removed? Explain why?

5. Report Guidelines

Your report should include but not limit to the following items.

A. Lab Implementation (35%)
   1. Block diagram of the design with explanation.
   2. Description of each essential block.
   3. State diagram(s) for FSM(s) with explanation, if applicable.
   4. Waveform screenshots and testing process, with explanation.

B. Questions & Discussions (50%)
   Provide your answer to the Questions & Discussions in the lab assignment.

C. Problem Encountered (10%)
   Describe the problems you encountered, solutions you developed, and the discussion. Explaining them with code segments or diagrams is recommended.

D. Suggestions (5%)
   Optional: Feedback on the lab, course suggestions, or even a light-hearted joke.

Attention

- ✓ DO NOT copy-and-paste code segments from the PDF materials to compose your design. It may also paste invisible non-ASCII characters, leading to hard-to-debug syntax errors.
- ✓ You should hand in three source files, including **lab2_adv_1.v, lab2_adv_1_t.v, lab2_adv_2.v**. Upload each source file individually. DO NOT hand in any compressed ZIP files, which will be considered an incorrect format.
- ✓ You should also hand in your report as **lab2_adv_report_StudentID.pdf** (i.e., lab2_adv_report_111456789.pdf).
- ✓ You should be able to answer questions from TA during the demo.
- ✓ You need to prepare the bitstream files before the lab demo to make the demo process smooth.