# Lab 4

## TopIO\_Interface

Interfaces with an FPGA. Here's a detailed explanation of its functionality:

**Module Instantiations:**

1. **Top Module:** The Top module is instantiated with its ports mapped to the relevant signals.
2. **Segment Decoders:** The segment decoders are instantiated to display values on the 7-segment displays:
   * DecoderModuleXHex0 and DecoderModuleXHex1 display the X vector.
   * DecoderModuleYHex2 and DecoderModuleYHex3 display the Y vector.
   * DecoderModuleOutHex4 and DecoderModuleOutHex5 display the ALUout vector.
3. **Clock Divider (PLL):** The pll\_wrap module divides the input clock by 2^L, where L is set to 6.

The code:

 Manages inputs from switches and keys.

 Displays processed data on 7-segment displays.

 Controls LEDs based on ALU flags.

 Outputs a PWM signal.

 Includes a clock divider for internal timing.

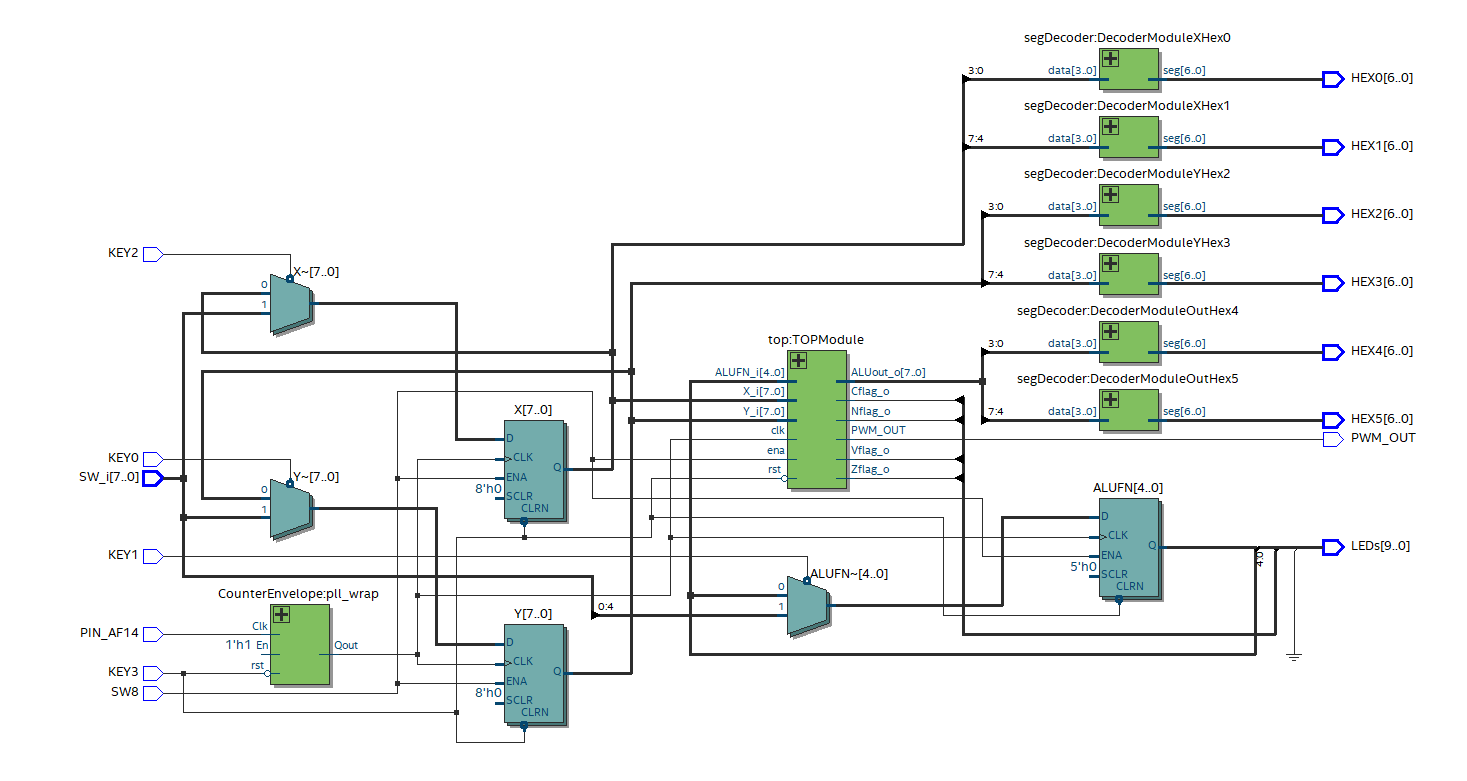


Figure 1 rtl of IO interface

## Top

A module named top, which includes two instantiated components: PWM and ALU.

defines a top-level module (top) that interfaces with both a PWM generator and an ALU. The module performs the following functions:

* **PWM Component:**
  + Processes the input vectors Y\_i and X\_i to generate a PWM signal (PWM\_OUT) based on the enable (ena), reset (rst), and clock (clk) signals.
  + Uses the ALU function code (ALUFN\_i) to control or modify the PWM generation.
* **ALU Component:**
  + Performs arithmetic and logical operations on the input vectors Y\_i and X\_i based on the ALU function code (ALUFN\_i).
  + Outputs the result of the ALU operation (ALUout\_o) and sets the status flags (Nflag\_o, Cflag\_o, Zflag\_o, Vflag\_o) accordingly.

In essence, the top module orchestrates the interaction between the PWM generation and the ALU operations, handling the inputs and outputs efficiently through these two instantiated components.

## 

## PWM

The PWM module generates a Pulse Width Modulation (PWM) signal based on input control signals and vectors. Here is a high-level description of what it does and how it works logically:

**Functionality**

The module generates a PWM signal where the duty cycle is determined by comparing a counter with input thresholds. The behavior is controlled by the ALUFN input.

**Logical Operation**

**Initialization:**

* + When the reset (RST) signal is high, the counter (COUNTER\_PWM) and the PWM output register (PWM\_OUT\_REG) are reset to zero.

**Clock Process:**

* + On the rising edge of the clock (CLK), if the enable signal (ENA) is high and the most significant bits of ALUFN are "00", the PWM generation logic is active.

**Counter Increment:**

* + The counter increments on each clock cycle. If the counter reaches the value of Y\_PWM, it resets to zero, creating a repetitive counting cycle.

**PWM Signal Generation:**

* + The PWM output register (PWM\_OUT\_REG) is set based on the counter value and the values of X\_PWM and Y\_PWM:
    - If ALUFN is "00000", the PWM signal is high when the counter is between X\_PWM and Y\_PWM.
    - If ALUFN is "00001", the PWM signal is low when the counter is between X\_PWM and Y\_PWM, and high otherwise.

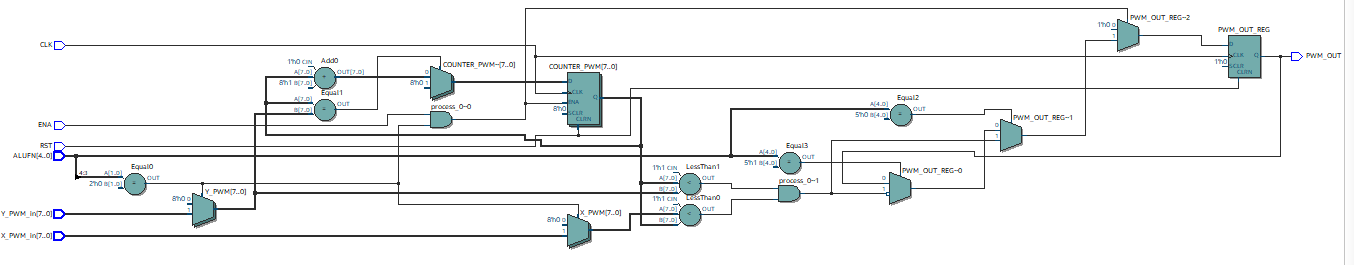


Figure 2 RTL of PWM

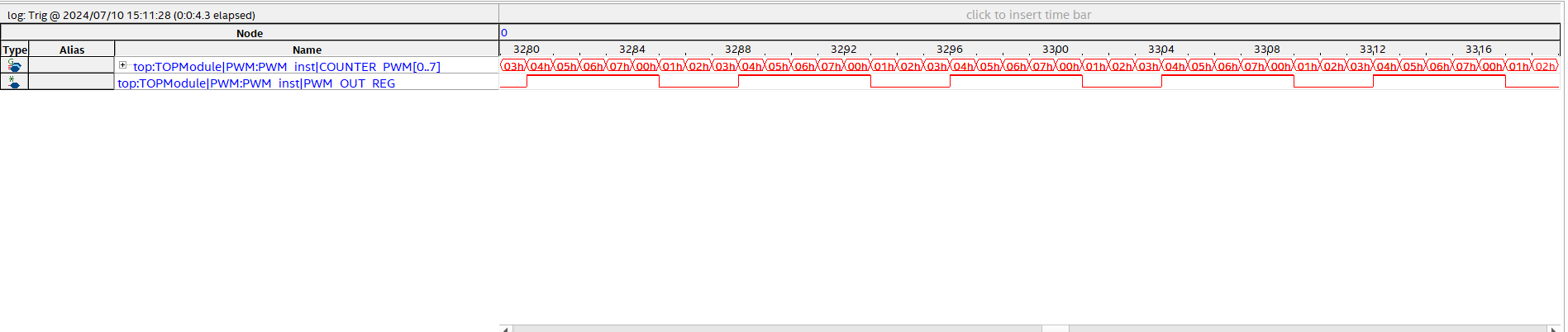


Figure 3 - Wave form in Quartus

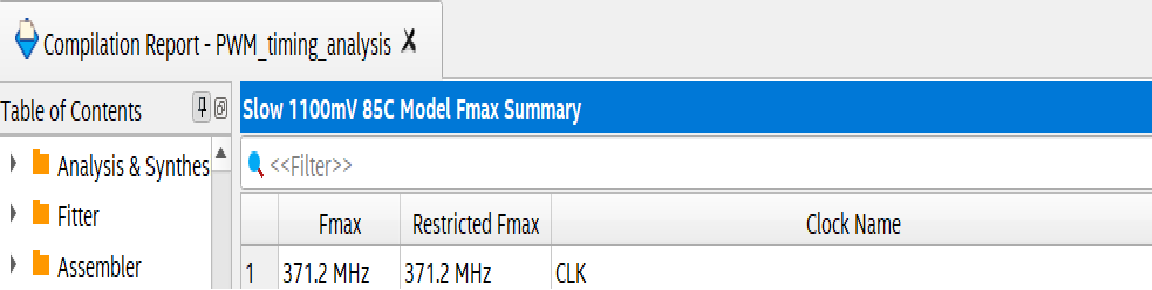


Figure 4 - max freq

## ALU

The ALU module in VHDL performs arithmetic, logical, and shifting operations based on the input control signals. Here is a high-level description of what it does and how it works logically:

**Functionality**

The module computes the result of various operations (addition/subtraction, logical operations, and shifts) based on the ALUFN\_i control signal. It then outputs the result along with several status flags.

**Logical Operation**

1. **Component Instantiation:**
   * **AdderSub Component:** Handles addition and subtraction operations.
   * **Shifter Component:** Handles bitwise shift operations.
   * **Logic Component:** Handles logical operations (AND, OR, etc.).
2. **Input Gating:**
   * Inputs to the specific components (AdderSub, Shifter, Logic) are gated based on the most significant bits of ALUFN\_i:
     + 01 for AdderSub
     + 10 for Shifter
     + 11 for Logic
3. **Overflow Detection:**
   * Overflow (Vflag) is calculated for addition and subtraction based on specific conditions:
     + **Addition Overflow (Vflag\_add):** Occurs when adding two positive numbers results in a negative or adding two negative numbers results in a positive.
     + **Subtraction Overflow (Vflag\_sub):** Occurs when subtracting a positive number from a negative results in a positive or subtracting a negative number from a positive results in a negative.
4. **Output Logic:**
   * The result from the appropriate component is selected based on ALUFN\_i and assigned to ALUOUT.
   * The status flags are calculated based on the result and the ALUFN\_i:
     + **Zero Flag (Zflag\_o):** Set if the result is zero.
     + **Negative Flag (Nflag\_o):** Set if the most significant bit of the result is 1.
     + **Carry Flag (Cflag\_o):** Set based on the carry out from the AdderSub or Shifter components.
5. **Output Assignment:**
   * The final result is assigned to ALUout\_o.

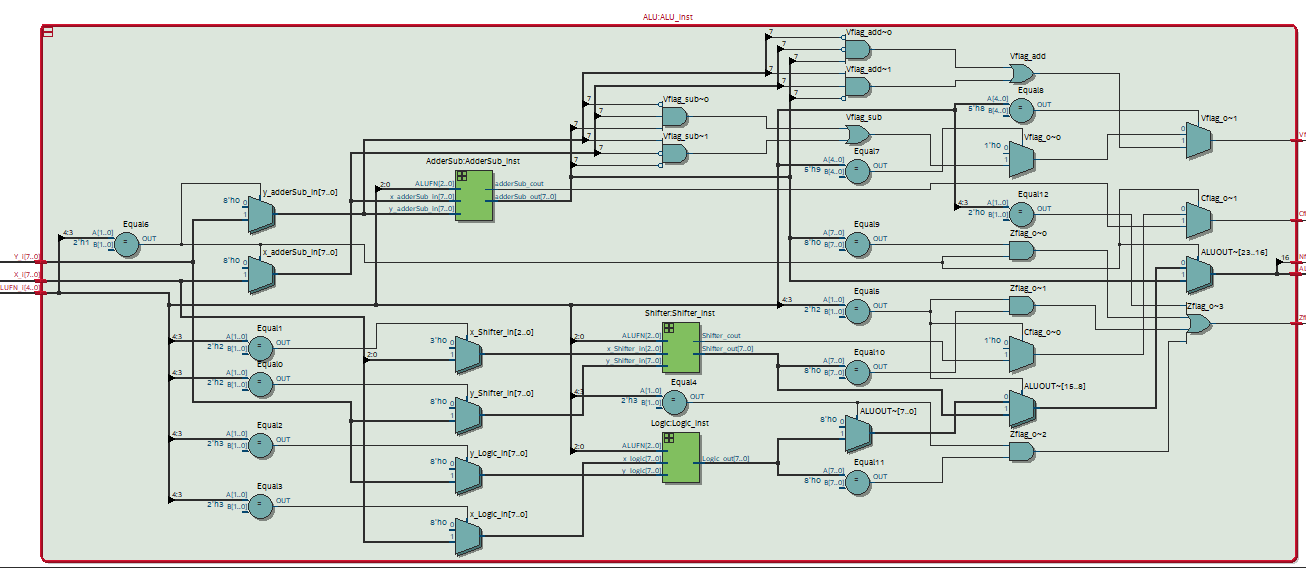


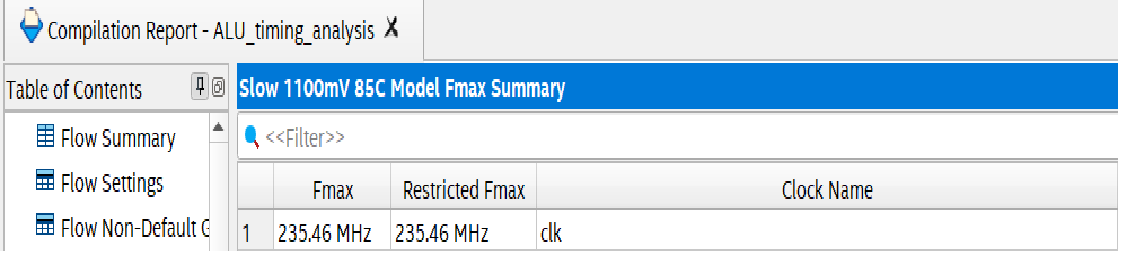
Figure 5 -RTL of ALU

Figure 6 - max freq of alu

## TB

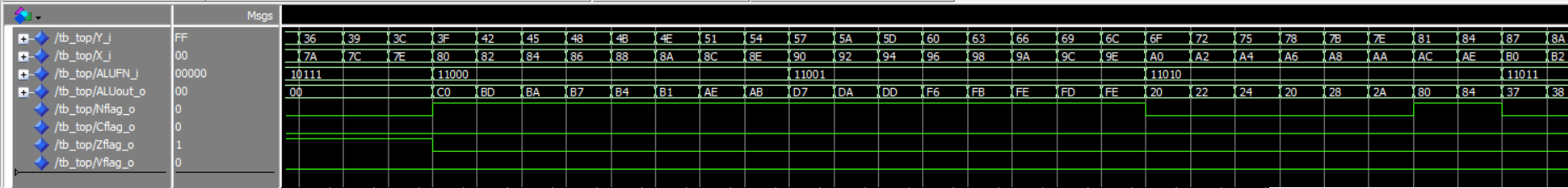
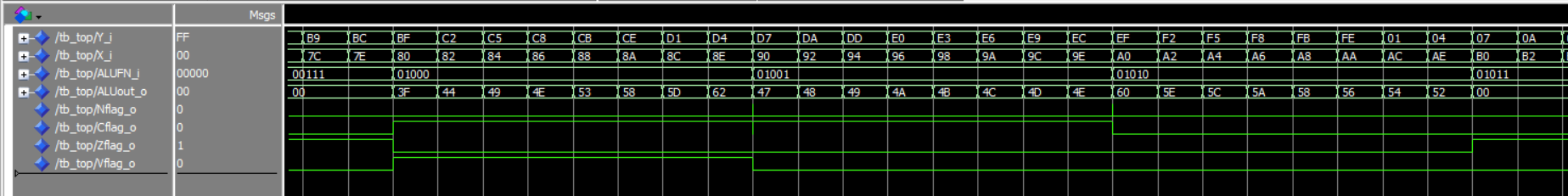
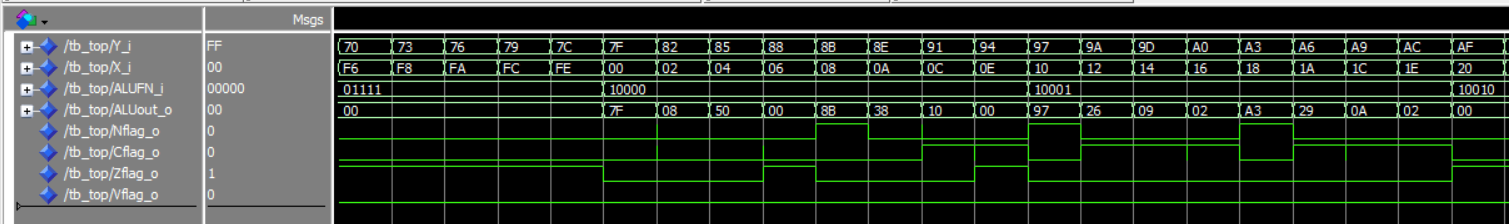


Figure 7 - alu logic



## Critical path

