LOGIC DESIGN - WEEK 3 - REPORT

Pham Duc Anh Khoa - 2053140
Nguyen Thi Ngoc Nhi - 2052632
Ngo Chan Phong - 2053321
Le Hoang Minh - 2052595
Tran Hoang Minh Quan - 2053380
Phan Tien Vinh - 2052323
Nguyen Tien Trung - 2052353

11 - 05 - 2021

Work division

Pham Duc Anh Khoa - Doing Question 3
Nguyen Thi Ngoc Nhi - Doing Question 1
Ngo Chan Phong - Doing Question 1
Le Hoang Minh - Question 2
Phan Tien Vinh - Question 4
Nguyen Tien Trung - Doing report

1 Answering the questions

1.1 The above declaration describes flip flop with synchronous or asynchronous reset ? Explain ?

The above declaration describe **synchronous** reset.
Explain: the **reset** condition only activate when the **clock** is positive edge (**always @(posedge clk)**)

1.2 What is the active level (LOW/HIGH) of reset and set signal ?

- **Reset** : High. When we associate HIGH with TRUE/1 it is called positive logic. HIGH is TRUE or 1 because in digital hardware the higher voltage is associated with the logic 1 state
- **Set** : Low. When we associate LOW with FALSE/0 it is called negative logic. LOW is FALSE or 0 because the 0 logic state in hardware is represented by the lower voltage, usually close to 0 volts.

1.3 Between set and reset signals, which is higher priority

The reset signal has highest priority, because this declaration describes synchronous reset. And when asserted (=1) forces the state of the flip-flop to 0 on the rising edge of the clock. The set signal has priority over the d input. And when asserted forces the flip-flop to 1 on the rising-edge of the clock.

1.4 Could we change the operator $\Leftarrow$ by =? Explain ?

We can change the operator $\Leftarrow$ by =. It is Recommended to use non-blocking assignment "$\Leftarrow$" for sequential logic and blocking assignment "=" for combinational logic.
We use non-blocking statement "$\Leftarrow$" to avoid race – condition. Using blocking statement "=" can cause a race-around condition. But the declaration above doesn't contain any race- condition so we can replace $\Leftarrow$ by =, which wont be cause any big difference.

1.5 The always block in dff sync module is triggered by level (level-sensitive) or edge (edge-sensitive) ?

The always block in **dff_sync** module is triggered by **edge sensitive** (**posedge clk**)

1.6 Could we change the declaration of rq from reg to wire ? Explain ?

We **cannot** change the declaration from reg to wire.
Wire: are used for connecting different elements. They need to be driven by either continuous assign statement or from a port of a module.
Reg: are also used for connecting but they can retain their value till next value is assigned to them, which means if it is holding some value, we should declare it as "reg".

1.7 Could we move the place of assign lines to above always block. Does q and q bar receive the appropriate value of rq after moving ? Explain your answer ?

- We **cannot** move the assign line above **always** block because the flip flop should only be activated when clock is positive edge.
- The **q** and **q bar** will not receive the appropriate value of rq after moving, because **rq's** value is assigned in the **always** block.

1.8 Is there any different between using the operator " ! " and "∼" in this case.

- " ! " stands for logical negation. " ! " means "reverse value".
- "∼" stands for bit-wise negation. "∼"' means the negation of the value.

2 Implement the J-K flipflop that satisfies the following requirements:

- Set and Reset are low-level activate.
- Set has higher priority than Reset.
- Set is asynchronous. While Reset is synchronous.
- The flip-flop works at negative edge of clock.

2.1 Simulation of the design

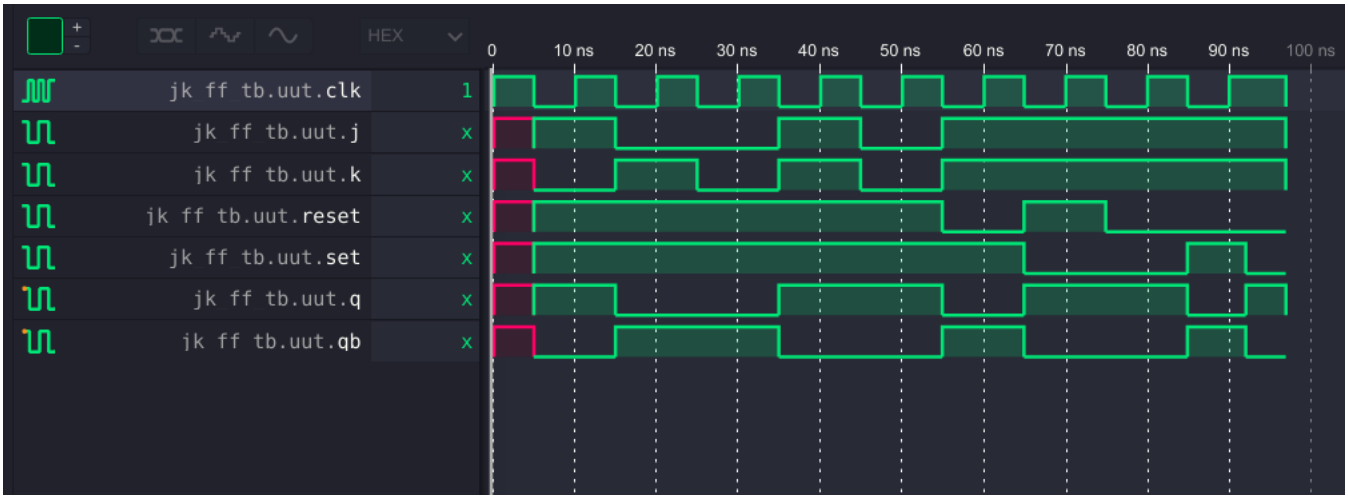

Figure 1: J-K flip flop simulation.

2.2 Simulation break down

- From the time 0ns → 55ns: The waveform show that the design of J-K flip flop work as **negative edge**.
- From the time 55ns → 65ns: The waveform show that **reset** works when it is **active - low**.
- From the time 65ns → 75ns: The waveform show that **set** works when it is **active - low**.
- From the time 75ns → 85ns: The waveform show that **set** has higher priority than **reset**.
- From the time 85ns → 100ns: The waveform show that **set** is **asynchronous**.

3 Design and implement 4-bit synchronous counter that satisfy the following requirement:

- Full counter (from 0 to 15).
- Using **only structural model**. (You can reuse the dff sync module our your implemented JK-flip flop).
- Prototype: module counter(clk, count, RCO), where:
  → **clk**: the input clock.
  → **count**: the count output.
  → **RCO**: is 1 when the value of the counter is 15, 0 otherwise.

3.1 Simulation of the design

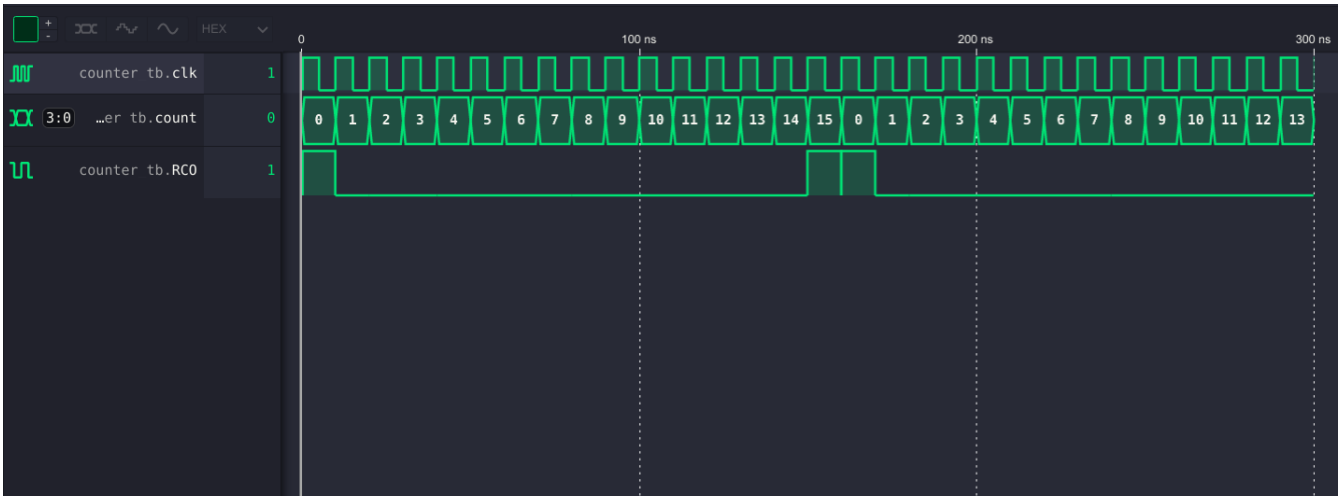

Figure 2: 4 bit synchronous counter

3.2 Simulation break down

- From the time 0ns → 155ns: The waveform show that the design of 4 bit synchronous counter works properly.
- From the time 155ns → 165ns: The waveform show that **RCO** works properly when count is 15 or 0.

4 Redesign the question 3 by using either RTL or Behavioral model:

4.1 Simulation of the design

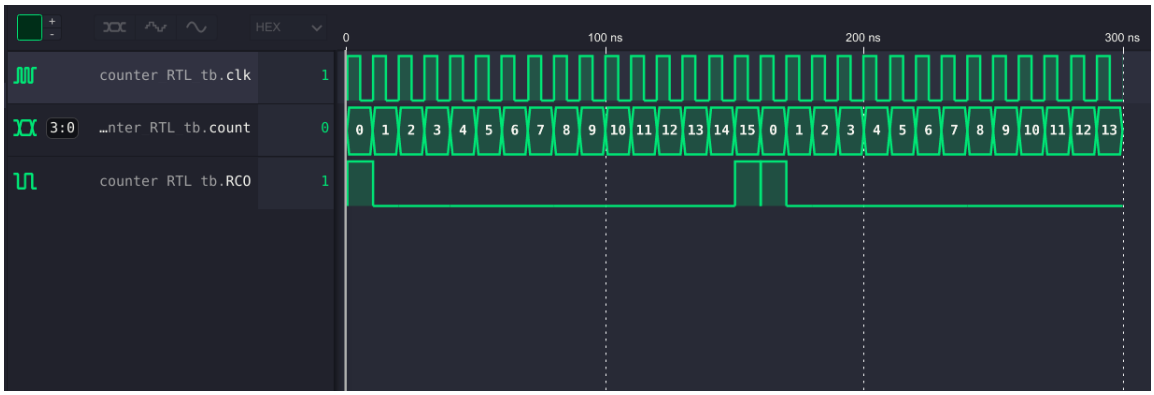

Figure2: 4 bit synchronous counter with RTL structure.

4.2 Simulation break down

- The waveform show that the design works just like the design on Question 3.