University of Southern California

Viterbi School of Engineering

EE477L

MOS VLSI Circuit Design

Phase 2 Project Report:

Arbiter Design

Li-Wei (Richard) Liu	**5297015518**
Hung-Ting (Andrew) Tsai	**4995041703**
Yo-Chwen Wang	**8331269140**

Professor: Dr. Shahin Nazarian

Due Date: 11/7/2023

Table of Contents

Report Guideline for phase 2

- Clearly explain the states of your FSM and specify the design (Moore or Mealy) used in your design.
- Test the entire functionality ofyour design using different values such that every state is visited once (make sure to include waveforms).
- Report the minimum time period of your design.

Part 1: Arbiter Design

1-bit DFF gate Design:

Arbiter Specifications from project description

1. If only one of the logic blocks (either multiplier or divider) asks for the grant, the arbiter should grant that block access to the output and let the requesting block pass its results to the output.
2. If none of the two blocks asks for access, the arbiter should specify no grant by setting an output signal STALL to 1. When the STALL signal is 1, we do not care about the output.
3. If both blocks request access, the arbiter should check the last grant and switch the priority. If the grant was given to the multiplier previously, the divider should now be granted access, and the last grant is updated to the divider. Similarly, if the grant was given to the divider previously, the multiplier should now be granted access, and the last grant is updated to the multiplier.
4. The divider has priority on reset, so if none of the blocks was previously given access and now both ask for it, give access to the divider. The divider has priority on reset, which means that when you activate the reset signal to 1 for an active-high reset or 0 for an active-low reset at the time "t" if both multiplier and divider ask for the grant at the time "t+1", you should choose divider.

Arbiter Design Assumptions

1. If just one of the logic block requests, the arbiter should grant that block for output, even if it has been granted previously.
2. The arbiter will produce a stall signal when none of the devices ask for requests; however, based on the designed schematic, it will still produce an output. The output will not be considered whenever the stall signal is “1”.
3. If both devices ask for requests, the arbiter will check the last grant given and prevent granting repeatedly to the same device.
4. If the system is stalling for this clock, and two devices both ask for requests after, the divider should have higher priority than that of the multiplier.

FSM Design of the Arbiter:

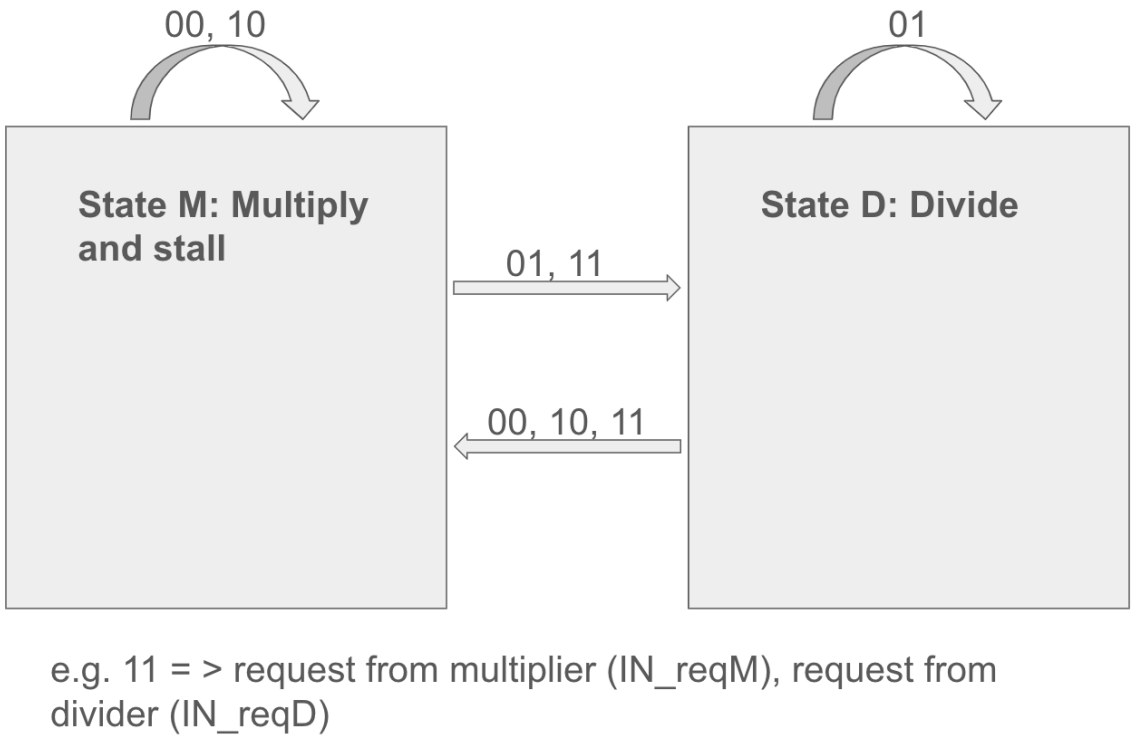


Figure 1: State diagram of a Mealy FSM of the proposed arbiter

Figure 1 above describes a Mealy FSM implementation of a proposed arbiter design. It consists of two states, state M and state D for multiplication and division respectively. The 2-bit inputs are IN_reqM and IN_reqD, two request signals from two sub-devices of the system. This FSM has been implemented with a single DFF in our schematic. The default state in our design will be on state M with a stalling signal, denoting that the system just started.

The stalling logic in the above design will be produced conditionally in state M. In this way, we can satisfy both assumptions 3 and 4 in the previous sections, the system will always move to state D whenever the divider requests after stalling, preventing giving multiplier repeated grants and giving the divider higher priority. Truth tables, final boolean calculations, and circuit implementation will be demonstrated in the next section.

reqM	reqD	S_DFF	G1 (S)	G2 (STALL)
0	**0**	**0**	**0**	**1**
0	**0**	**1**	**0**	**1**
0	**1**	**0**	**1**	**0**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**0**
1	**0**	**1**	**0**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**0**	**0**

Table 1: Truth table of the proposed design

G1 (S)				
G1 \ MD	00	01	11	10
0	0	1	1	0
1	1	1	0	0

Table 2: Karnaugh Map of G1 (S)

G2 (Stall)				
G1 \ MD	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Table 3: Karnaugh Map of G2 (Stall)

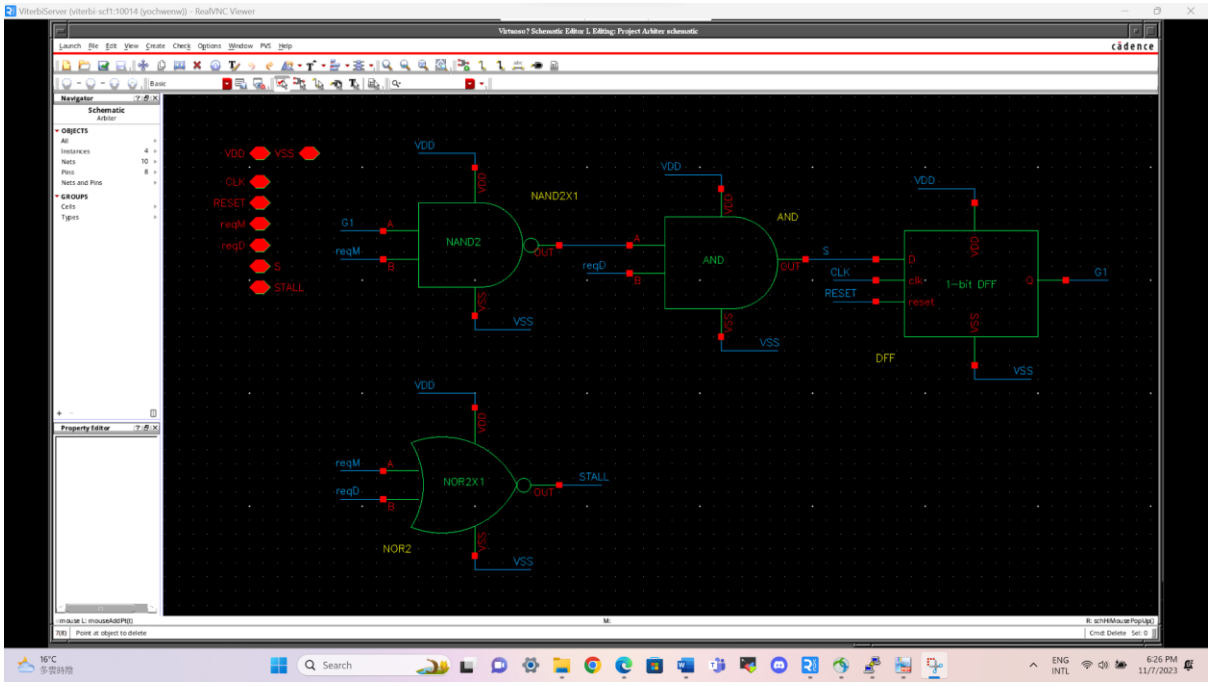
From the Karnaugh Map above, we can derive the Boolean equation of both granting signals: G1 and G2.

- $G1 = reqM' \cdot reqD + S_{DFF}' \cdot reqD + S_{DFF} \cdot reqM' = S_{DFF}' \cdot reqD + S_{DFF} \cdot reqM'$
- $G2 = reqM \ NOR \ reqD$

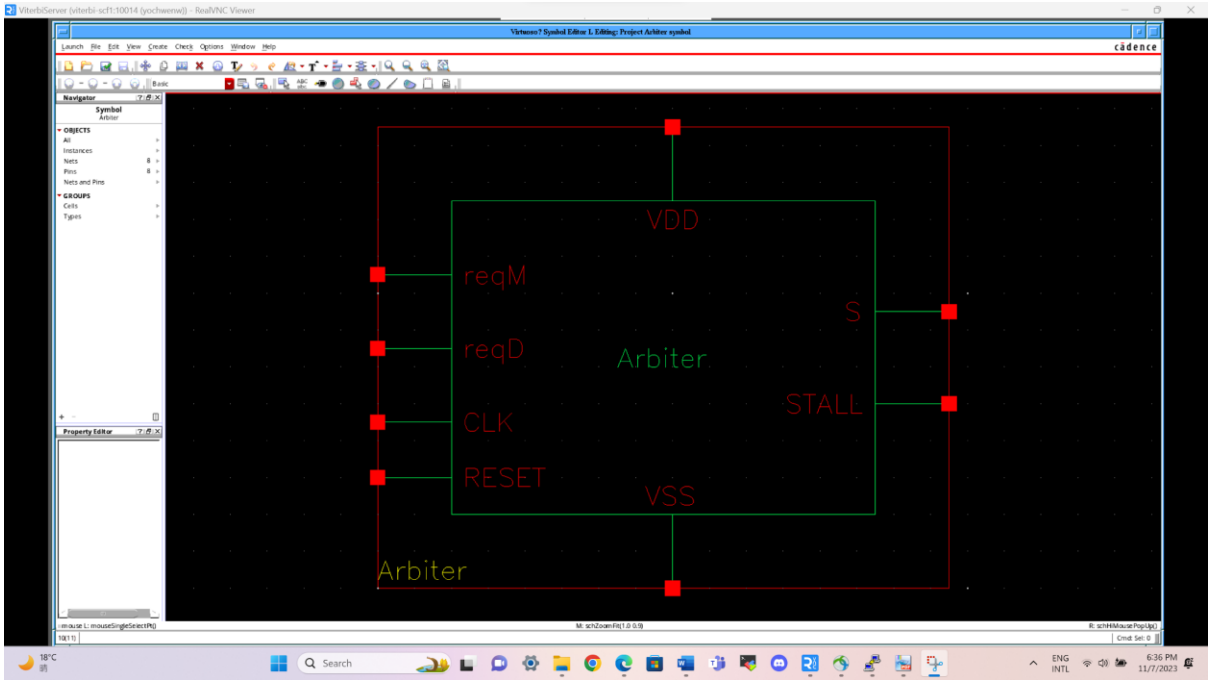
Further logic circuits will be shown in the schematics below.

Arbiter Schematic, Symbol, and Functional Verification

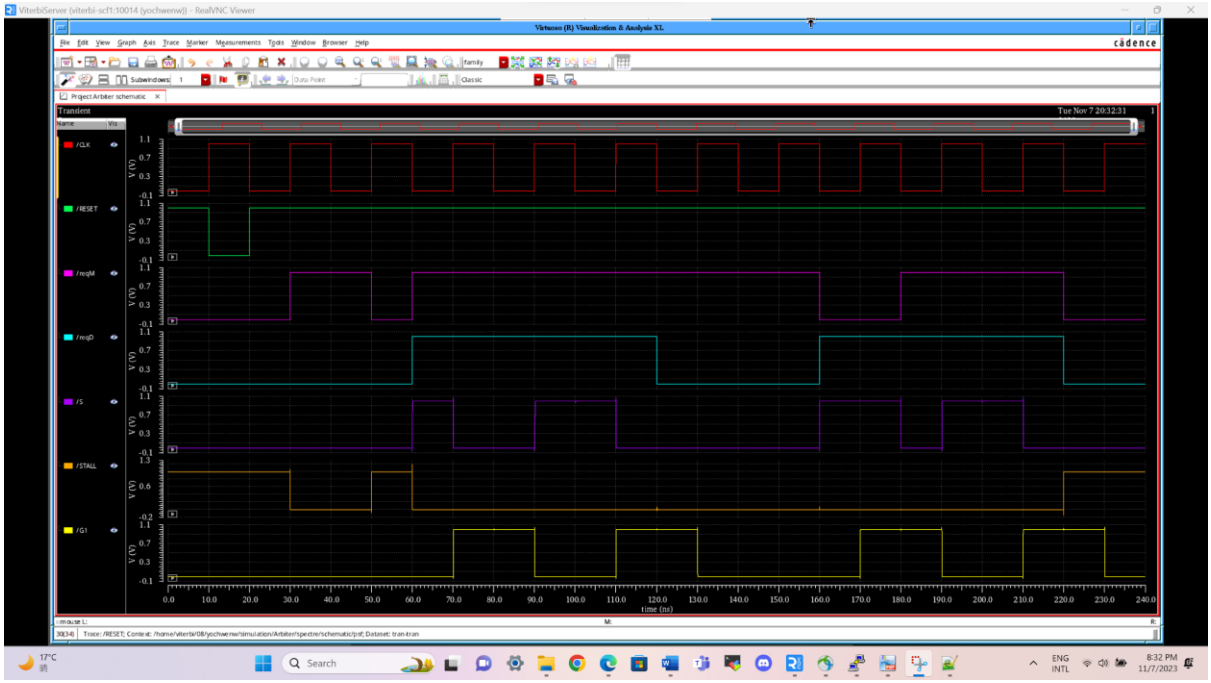
Schematic:



Symbol:



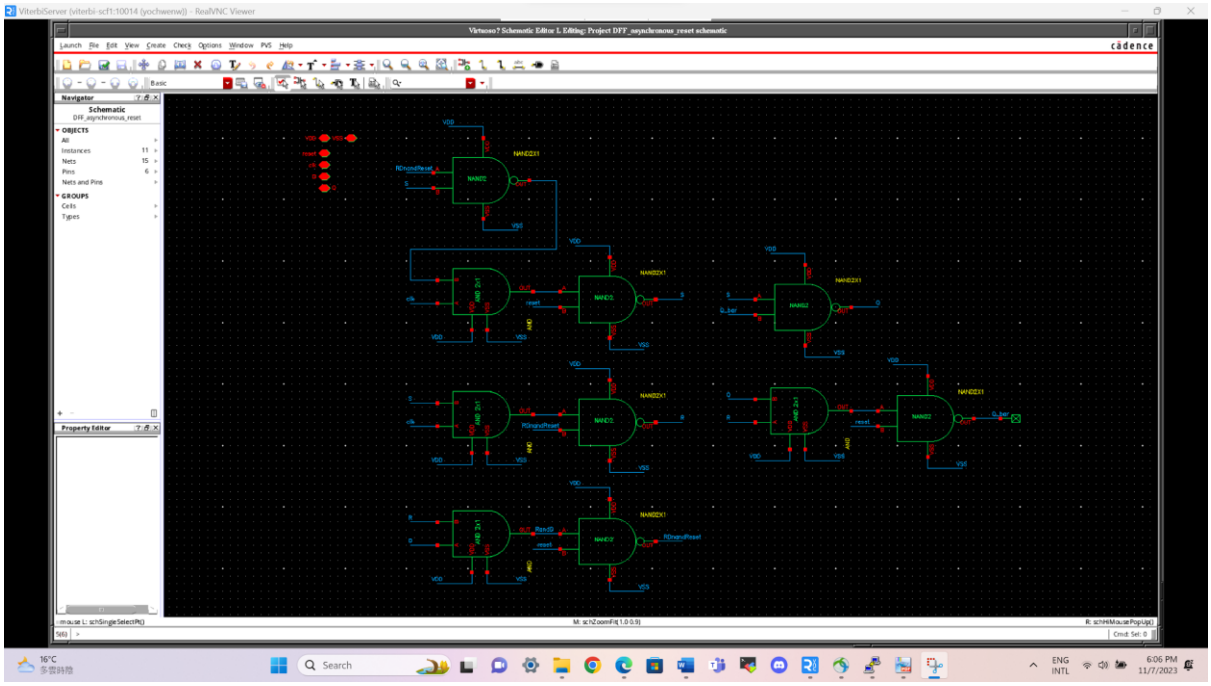
Functional Verification:



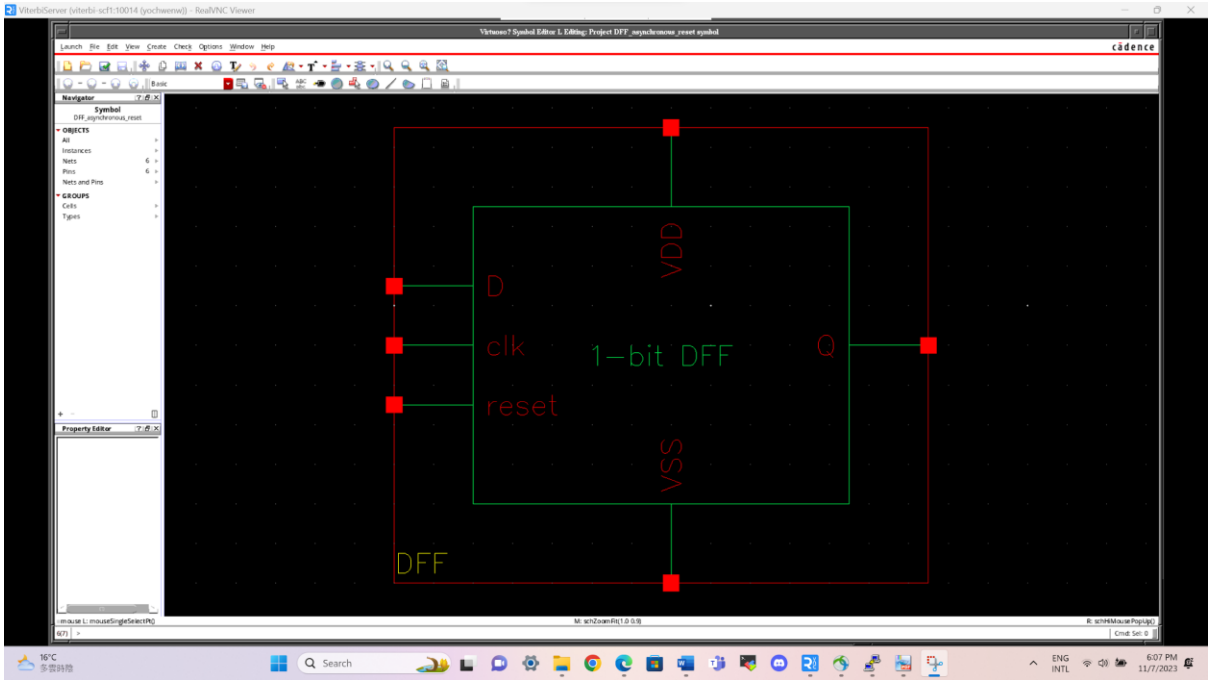
Part 2: Minimum Time Period

1-bit DFF gate Design:

Schematic:



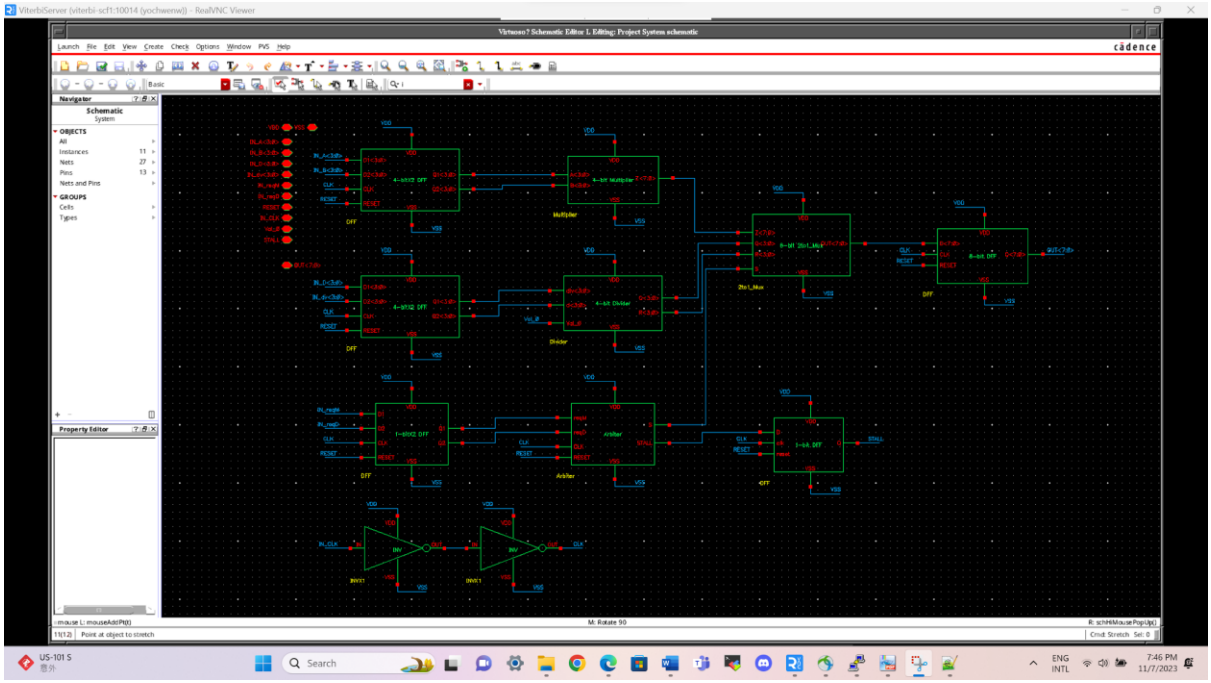
Symbol:



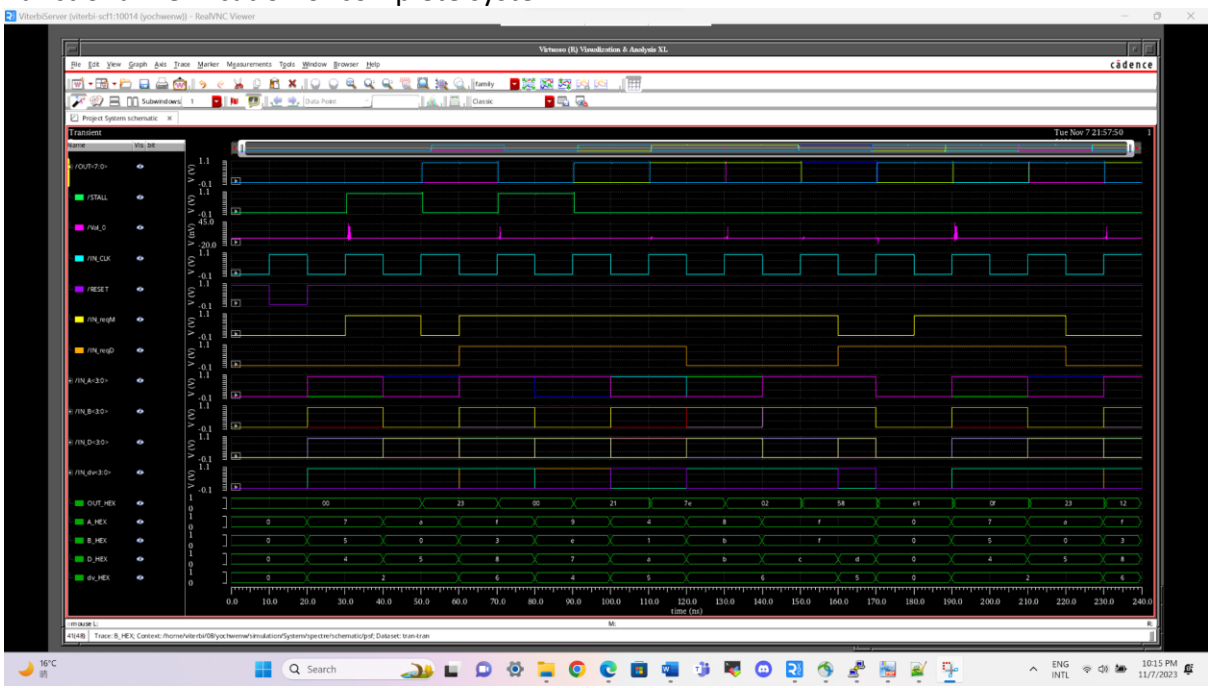
Functional Verification for DFF:

Complete system with multiplier, divider, and arbiter

Schematic:



Functional Verification of complete system:



Minimum time period delay measurements:

You should report the minimum time period of your design. The minimum time period is when the system's clock outputs the correct results with no setup or hold violations.

Note: Do not connect the x1 and x16 inverter before the input DFFs and output DFFs.

The minimum time period set for each DFFs is 1ns. The measured minimum time period of the complete system is 3.3625 ns. In summary, the minimum time period delay can be calculated by subtracting 2 ns (2 DFFs) from the result, which makes it 1.3625 ns.

Appendix

// Vector file for testing

```
radix 4 4 4 4
io i i i i
vname IN_A<[3:0]> IN_B<[3:0]> IN_D<[3:0]> IN_dv<[3:0]>
tunit ns
slope 0.01
vih 1.0
vil 0.0

0  0 0 0 0
1  0 0 0 0
2  7 5 4 2
3  7 5 4 2
4  A 0 5 2
5  A 0 5 2
6  F 3 8 6
7  F 3 8 6
8  9 E 7 4
9  9 E 7 4
10 4 1 A 5
11 4 1 A 5
12 8 B B 6
13 8 B B 6
14 F F C 6
15 F F C 6
16 F F D 5
17 0 0 0 1
18 0 0 0 1
19 7 5 4 2
20 7 5 4 2
21 A 0 5 2
22 A 0 5 2
23 F 3 8 6
```