CSU22022 Computer Architecture I

Prof. Michael Manzke

Processor Assignment: Full Project

Instructions for the Blackboard Submission of the Assignment

20th November 2022

Version 1.0

The "Processor Assignment: Full Project" document (CSU22022 Processor Project 2022-2023 V1.0.pdf) specifies the VHDL entities that you need to implement.

It also provides names for all the entities (e.g., RF_Register32Bit_XXXXXXXX) with XXXXXXXX as your student ID and its instantiation names (e.g., RegisterXX, XX=00 to 31; TempRegXX, XX=01 to 15) with XX as the running number of the instantiation.

The file that implements the entity must have this name (e.g., RF_Register32Bit_XXXXXXXX.vhd) and the entity itself must have the same name (e.g., entity RF_Register32Bit_XXXXXXXX is).

Any instantiation of this entity must use the specified instantiation name (e.g., Register02: RF_Register32Bit_XXXXXXXX port map(…);).

All files associated with this entity must also follow this naming convention: (e.g., RF_Register32Bit_XXXXXXXX.vhd
RF_Register32Bit_XXXXXXXX_TB.vhd
RF_Register32Bit_XXXXXXXX_SchematicXX.jpg
RF_Register32Bit_XXXXXXXX_TDXX.jpg
RF_Register32Bit_XXXXXXXX_Doc.pdf)

Entities that do not comply with this naming convention will receive a zero mark.

Furthermore, the "Processor Assignment: Full Project Checklist" document (CSU22022 Processor Project 2022-2023 Checklist V1.0.pdf) specifies the order in which you must implement, test, and document the entities, e.g.

01	RF_Mux16_32Bit_XXXXXXXX.vhd	
	RF_Mux16_32Bit_XXXXXXXX_TB.vhd	
	RF_Mux16_32Bit_XXXXXXXX_SchematicXX.jpg	
	RF_Mux16_32Bit_XXXXXXXX_TDXX.jpg	
	RF_Mux16_32Bit_XXXXXXXX_Doc.pdf (only if needed)	
	Is working	

02	RF_Mux32_32Bit_XXXXXXXX.vhd	
	RF_Mux32_32Bit_XXXXXXXX_TB.vhd	
	RF_Mux32_32Bit_XXXXXXXX_SchematicXX.jpg	
	RF_Mux32_32Bit_XXXXXXXX_TDXX.jpg	
	RF_Mux32_32Bit_XXXXXXXX_Doc.pdf (only if needed)	
	Is working	

Only fully implemented, tested, and documented entities should be ticked on the checklist and submitted to Blackboard. Before you make ticks for an entity, you must have all these documents: Entity_XXXXXXXX.vhd, Entity_XXXXXXXX_TB.vhd, Entity_XXXXXXXX Entity_SchematicXX.jpg, Entity_XXXXXXXX_TDXX.jpg, and Entity_XXXXXXXX_Doc.jpg (only if needed). Also, you must confirm that the entity is working.

Partially implemented, tested, and documented entities (e.g., the timing diagrams are not annotated) will receive a zero mark.

Partially implemented, tested, and documented entities should not be ticked on the checklist and no files for this entity should be submitted to Blackboard.

Therefore, if you ticked on the checklist and submitted the entities 01 to 07 and it transpires that you didn't annotate the timing diagrams for entity 05, you will receive a mark for entities 01 to 04 but a zero mark for the entities 05 to 07.

Please find following an example for a fully implemented, tested, and documented entities:

XX	RippleCarryAdder3Bit_XXXXXXXX.vhd	☑
	RippleCarryAdder3Bit_XXXXXXXX_TB.vhd	☑
	RippleCarryAdder3Bit_XXXXXXXX_SchematicXX.jpg	☑
	RippleCarryAdder3Bit_XXXXXXXX_TDXX.jpg	☑
	RippleCarryAdder3Bit_XXXXXXXX_Doc.pdf (only if needed)	☑
	Is working	☑

RippleCarryAdder3Bit_XXXXXXXX.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity RippleCarryAdder3Bit_XXXXXXXX is

   Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
          B : in STD_LOGIC_VECTOR (2 downto 0);
          C_IN : in STD_LOGIC;
          SUM : out STD_LOGIC_VECTOR (2 downto 0);
          C_OUT : out STD_LOGIC;
          V : out STD_LOGIC);

end RippleCarryAdder3Bit_XXXXXXXX;

architecture Behavioral of RippleCarryAdder3Bit_XXXXXXXX is

COMPONENT FullAdder_XXXXXXXX
   PORT(
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        C_IN : in STD_LOGIC;
        SUM : out STD_LOGIC;
        C_OUT : out STD_LOGIC
       );
   END COMPONENT;

Signal C0_to_C1, C1_to_C2, C2_to_COut : STD_LOGIC;

begin

  -- Instantiate Full Adder Bit 0
  BIT0: FullAdder_XXXXXXXX PORT MAP (
         A => A(0),
         B => B(0),
         C_IN => C_IN,
         SUM => SUM(0),
         C_OUT => C0_to_C1
        );

  -- Instantiate Full Adder Bit 1
  BIT1: FullAdder_XXXXXXXX PORT MAP (
```

```
        A => A(1),
        B => B(1),
        C_IN => C0_to_C1,
        SUM => SUM(1),
        C_OUT => C1_to_C2
       );

  -- Instantiate Full Adder Bit 2
  BIT2: FullAdder_XXXXXXXX PORT MAP (
        A => A(2),
        B => B(2),
        C_IN => C1_to_C2,
        SUM => SUM(2),
        C_OUT => C2_to_COut
       );

   -- Carry and Overflow
   C_OUT <= C2_to_COut
   V <= C2_to_COut XOR C1_to_C2 after 3ns;

end Behavioral;
```

You may work with other students, but you cannot share code. Sharing code is plagiarism.

RippleCarryAdder3Bit_XXXXXXXX_TB.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity RippleCarryAdder3Bit_XXXXXXXX_TB is
--  Port ( ); We don't need ports
end RippleCarryAdder3Bit_XXXXXXXX_TB;

architecture Sim of RippleCarryAdder3Bit_XXXXXXXX_TB is

-- Component Declaration for the Unit Under Test (UUT)

component RippleCarryAdder3Bit_XXXXXXXX

Port ( A : in STD_LOGIC_VECTOR (2 downto 0);
      B : in STD_LOGIC_VECTOR (2 downto 0);
      C_IN : in STD_LOGIC;
      SUM : out STD_LOGIC_VECTOR (2 downto 0);
      C_OUT : out STD_LOGIC;
      V : out STD_LOGIC);
end component;

--Inputs

   signal A_TB : STD_LOGIC_VECTOR (2 downto 0) := (others => '0');
   signal B_TB : STD_LOGIC_VECTOR (2 downto 0) := (others => '0');
   signal C_IN_TB : STD_LOGIC := '0';

--Outputs

   signal SUM_TB : STD_LOGIC_VECTOR (2 downto 0)  := (others => '0');
```

```
   signal C_OUT_TB : STD_LOGIC := '0';
   signal V_TB : STD_LOGIC := '0';

begin

-- Instantiate the Unit Under Test (UUT)
   uut: RippleCarryAdder3Bit_XXXXXXXX PORT MAP (
        A => A_TB,
        B => B_TB,
        C_IN => C_IN_TB,
        SUM => SUM_TB,
        C_OUT => C_OUT_TB,
        V => V_TB
        );

 stim_proc: process
   begin

   -- Test Vector A

      A_TB <= "101";
      B_TB <= "110";
      C_IN_TB <= '0';

      wait for 60 ns;

   -- Test Vector B

      A_TB <= "101";
      B_TB <= "111";
      C_IN_TB <= '0';

      wait for 60 ns;

   -- Test Vector C

      A_TB <= "001";
      B_TB <= "011";
      C_IN_TB <= '0';

      wait for 60 ns;

   -- Test Vector D

      A_TB <= "001";
      B_TB <= "010";
      C_IN_TB <= '0';

      wait for 60 ns;

   -- Test Vector E

      A_TB <= "010";
      B_TB <= "110";
      C_IN_TB <= '0';

      wait for 60 ns;
```

```
  -- Test Vector F, worst case propagation delay

    A_TB <= "110";
    B_TB <= "001";
    C_IN_TB <= '1';

    wait for 60 ns;

  end process;
end Sim;
```

You may not use loops in your VHDL design or simulation code.

RippleCarryAdder3Bit _XXXXXXXX_ Doc.pdf

A

Negative number + negative number with overflow flag (V-flag)

V <= C3 XOR C2

					Unsigned value	Two's complement
	C3	**C2**	**C1**	**C0**		
	1	0	0	0		
A		**A2**	**A1**	**A0**		
		1	0	1	5	-3
B		**B2**	**B1**	**B0**		
		1	1	0	6	-2
+						
SUM		**S2**	**S1**	**S0**		
	1	0	1	1	11	-5

B

Negative number + negative number without overflow flag (V-flag)

V <= C3 XOR C2

					Unsigned value	Two's complement
	C3	**C2**	**C1**	**C0**		
	1	1	1	0		
A		**A2**	**A1**	**A0**		
		1	0	1	5	-3
B		**B2**	**B1**	**B0**		
		1	1	1	7	-1
+						
SUM		**S2**	**S1**	**S0**		
	1	1	0	0	12	-4

C

Positive number + positive number with overflow flag (V-flag)

V <= C3 XOR C2
1

					Unsigned value	Two's complement
	C3	C2	C1	C0		
	0	1	1	0		
A		A2	A1	A0		
		0	0	1	1	1
B		B2	B1	B0		
		0	1	1	3	3
+						
SUM		S2	S1	S0		
	0	1	0	0	4	-4

D

Positive number + positive number without overflow flag (V-flag)

V <= C3 XOR C2
0

					Unsigned value	Two's complement
	C3	C2	C1	C0		
	0	0	0	0		
A		A2	A1	A0		
		0	0	1	1	1
B		B2	B1	B0		
		0	1	0	2	2
+						
SUM		S2	S1	S0		
	0	0	1	1	3	3

E

Positive number + nagative number without overflow flag (V-flag)

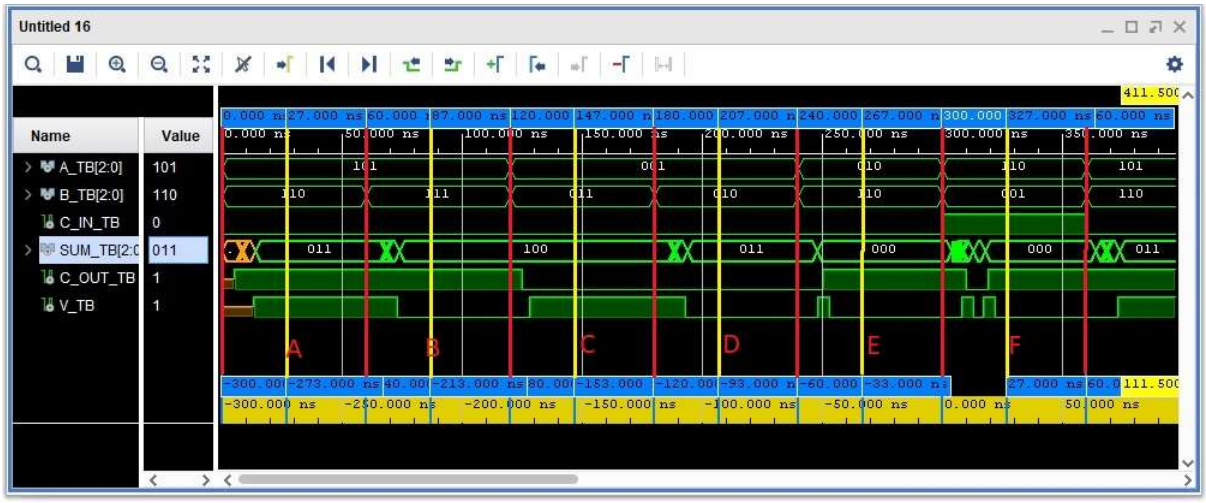
V <= C3 XOR C2
0

					Unsigned value	Two's complement
	C3	C2	C1	C0		
	1	1	0	0		
A		A2	A1	A0		
		0	1	0	2	2
B		B2	B1	B0		
		1	1	0	6	-2
+						
SUM		S2	S1	S0		
	1	0	0	0	8	0

F	Nagative number + positive number without overflow flag (V-flag) Worst case propagation delay, C0 is set							
	V <= C3 XOR C2							
	0							
		C3	C2	C1	C0	Unsigned value		
		1	1	1	1		Two's complement	
	A		A2	A1	A0			
			1	1	0	6	-2	
	B		B2	B1	B0			
			0	0	1	1	1	
	+							
	SUM		S2	S1	S0			
		1	0	0	0	8	0	C0 = 1 Important

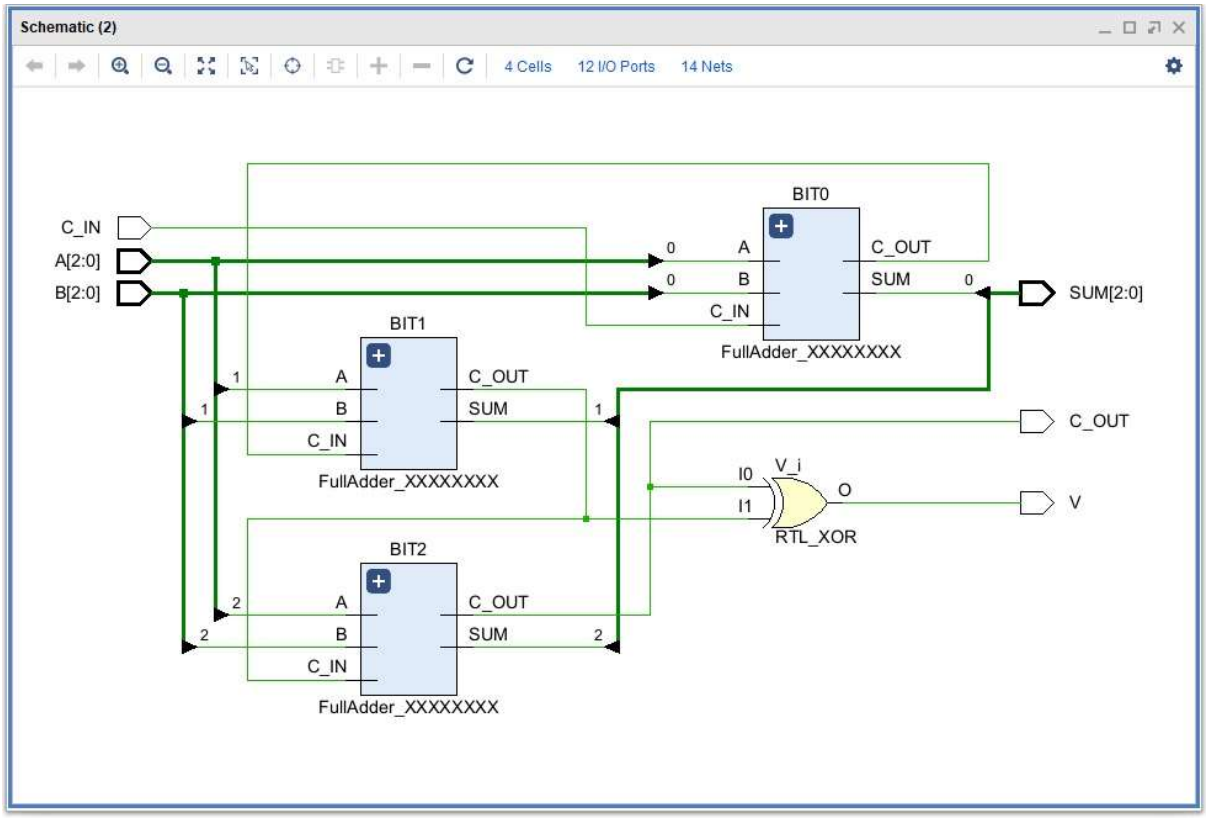
Depending on the complexity of you testbench you may need a separate document (e.g., RippleCarryAdder3Bit _XXXXXXXX _Doc.pdf) to specify the evaluation of your VHDL design. Otherwise, you can do this on the waveform screenshot, but the waveform screenshot must be in the JPG-file format.

RippleCarryAdder3Bit _XXXXXXXX_TD01.jpg



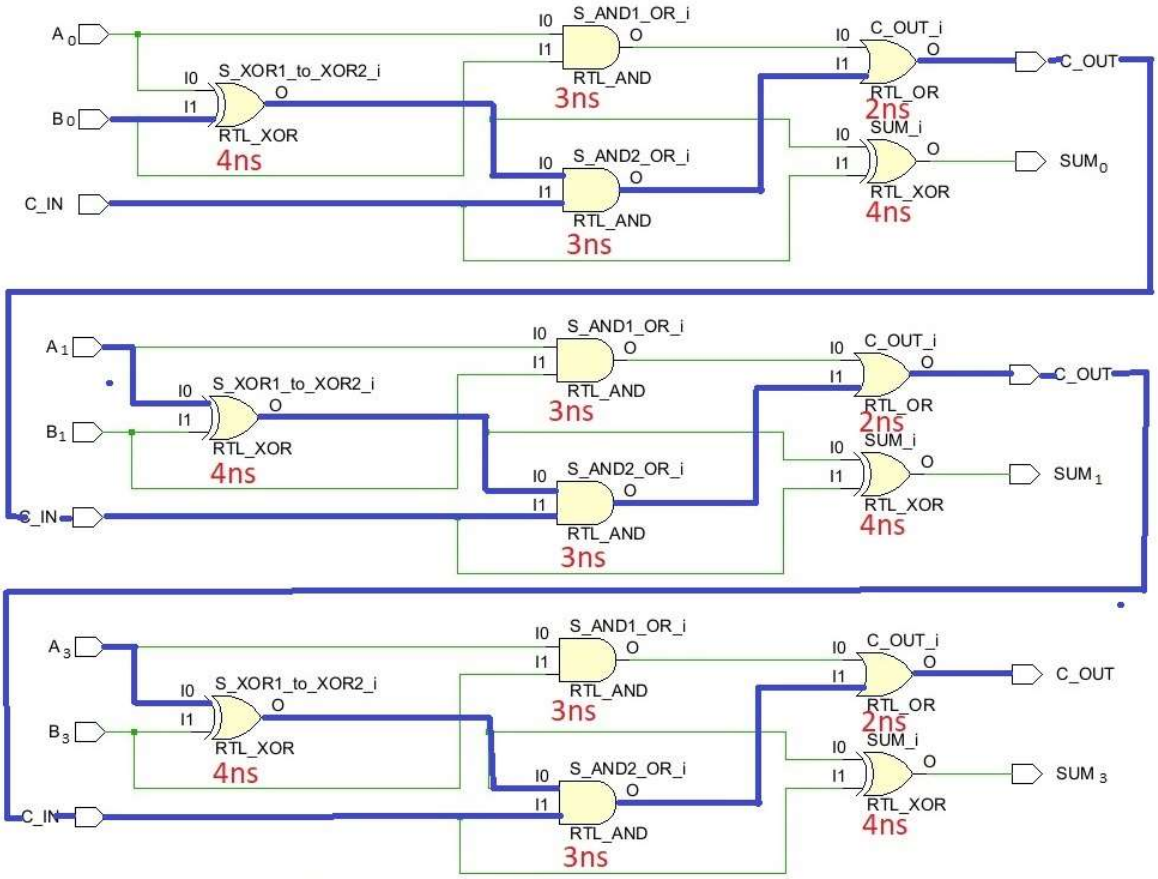
You must take a screenshot of the waveform and save it in the JPG-file format. Important, you must annotate the waveform. In this case with reference to the RippleCarryAdder3Bit _XXXXXXXX _Doc.pdf document.

RippleCarryAdder3Bit_XXXXXXXX__Schematic01.jpg



You must take a screenshot of the schematic and save it in the JPG-file format.

RippleCarryAdder3Bit_XXXXXXXX__Schematic02.jpg



C_OUT||SUM(1000) <=A(110)+B(001)+C_IN(1)

Total gate delay 19ns = (4ns + 3ns + 2ns) + 2(3ns +2ns)

You must take a screenshot of the schematic and save it in the JPG-file format.

Furthermore, all entities must be tested according to the Simulation Procedure (CSU22022 Processor Project 2022-2023 Simulation Procedure V1.0).

All files must be submitted to Blackboard as individual files, no zip-files.

You must print this checklist, tick the boxes, sign it, and submit it with your milestone assignment to Blackboard.

All files must be submitted in the checklist order:

RF_Mux16_32Bit_XXXXXXXX.vhd
RF_Mux16_32Bit_XXXXXXXX _TB.vhd
RF_Mux16_32Bit_XXXXXXXX _SchematicXX.jpg
RF_Mux16_32Bit_XXXXXXXX _TDXX.jpg
RF_Mux16_32Bit_XXXXXXXX _Doc.jpg
RF_Mux32_32Bit_XXXXXXXX.vhd
RF_Mux32_32Bit_XXXXXXXX _TB.vhd
RF_Mux32_32Bit_XXXXXXXX _SchematicXX.jpg
RF_Mux32_32Bit_XXXXXXXX _TDXX.jpg
RF_Mux32_32Bit_XXXXXXXX _Doc.jpg

-
-
-

CSU22022 Processor Project 2022-2023 Checklist V1.0.pdf