Avery Peiffer

ECE 1195

Practicum

My VHDL code is on the following pages. Below are two screenshots from my simulation and explanations about my design.

Graphical user interface

Description automatically generated

In this example, my design appears to be off by one clock cycle. I think this is due to the zero signal from the counter not being wired correctly, or something to that effect. However, it looks to be generally correct other than this one bug, as the ChecksumError signal goes high because the checksum is compared to 0x99 at the very end.

Diagram

Description automatically generated

Here is an example of my design working on a data stream. The first two packets (99 and 66) are read in, after which the length of 01 is read in. The next packet, 03, is the data, prompting the DataValid signal to go high. The next packet, 04, is the checksum. Since the running total of 03 and the checksum do not match, the ChecksumError signal goes high at the following clock cycle.

**Fsm.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity fsm is

Port (

clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR(7 downto 0);

Checksum : in STD\_LOGIC\_VECTOR (7 downto 0);

LengthIsZero : in STD\_LOGIC;

Counter\_load : out STD\_LOGIC;

Length\_init : out STD\_LOGIC\_VECTOR (7 downto 0);

ChecksumError : out STD\_LOGIC;

DataValid : out STD\_LOGIC

);

end fsm;

architecture Behavioral of fsm is

type state is (S0, S1, S2, S3, S4);

signal pr\_state, nx\_state : state;

begin

-- FSM register

process(reset, clock)

begin

if (reset = '1') then

pr\_state <= S0;

elsif (clock'event and clock='1') then

pr\_state <= nx\_state;

end if;

end process;

-- Next state function

process(DataIn, pr\_state)

variable length : STD\_LOGIC\_VECTOR(7 downto 0);

begin

case pr\_state is

when S0 =>

if (DataIn = X"99") then

nx\_state <= S1;

else

nx\_state <= S0;

end if;

when S1 =>

if (DataIn = X"66") then

nx\_state <= S2;

else

nx\_state <= S0;

end if;

when S2 =>

Length\_init <= DataIn;

nx\_state <= S3;

-- Read in payload data

when S3 =>

if (LengthIsZero = '1') then

nx\_state <= S4;

else

nx\_state <= S3;

end if;

-- Do checksum

when S4 =>

nx\_state <= S0;

end case;

end process;

-- Output function

process(pr\_state)

begin

case pr\_state is

when S0 =>

ChecksumError <= '0';

DataValid <= '0';

Counter\_load <= '0';

when S1 =>

when S2 =>

Counter\_load <= '1';

when S3 =>

DataValid <= '1'; -- Will control counter and register EN signals

Counter\_load <= '0';

when S4 =>

if (CheckSum = DataIn) then

ChecksumError <= '0';

else

ChecksumError <= '1';

end if;

DataValid <= '0';

end case;

end process;

end Behavioral;

**Datapath.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity datapath is

Port(

clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR(7 downto 0);

ChecksumError : out STD\_LOGIC;

DataValid : out STD\_LOGIC

);

end datapath;

architecture arc of datapath is

component add8

Port(

CI : IN std\_logic;

A : IN std\_logic\_vector (7 DOWNTO 0);

B : IN std\_logic\_vector (7 DOWNTO 0);

CO : out std\_logic;

S : out std\_logic\_vector (7 DOWNTO 0));

end component;

component counter

Port (D : IN std\_logic\_vector (7 DOWNTO 0);

clk : IN std\_logic;

en : IN std\_logic;

load : IN std\_logic;

rst : IN std\_logic;

updown : IN std\_logic;

Q : OUT std\_logic\_vector (7 DOWNTO 0);

Zero : OUT std\_logic);

end component;

component reg8

Port (CLK : IN std\_logic;

D : IN std\_logic\_vector (7 DOWNTO 0);

EN : IN std\_logic;

RST : IN std\_logic;

Q : OUT std\_logic\_vector (7 DOWNTO 0));

end component;

component fsm

Port (clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR(7 downto 0);

Checksum : in STD\_LOGIC\_VECTOR (7 downto 0);

LengthIsZero : in STD\_LOGIC;

Counter\_load : out STD\_LOGIC;

Length\_init : out STD\_LOGIC\_VECTOR (7 downto 0);

ChecksumError : out STD\_LOGIC;

DataValid : out STD\_LOGIC);

end component;

signal temp\_C : STD\_LOGIC;

signal reg\_Q : STD\_LOGIC\_VECTOR(7 downto 0);

signal reg\_D : STD\_LOGIC\_VECTOR(7 downto 0);

signal Length\_init : STD\_LOGIC\_VECTOR (7 downto 0);

signal Length\_count : STD\_LOGIC\_VECTOR (7 downto 0);

signal DataValid\_en : STD\_LOGIC;

signal Zero\_out : STD\_LOGIC;

signal Counter\_load\_out :STD\_LOGIC;

signal Counter\_en : STD\_LOGIC;

begin

adder: add8 PORT MAP(

CI => temp\_C,

A => DataIn,

B => reg\_Q,

CO => temp\_C,

S => reg\_D

);

count: counter PORT MAP(

D => DataIn,

CLK => clock,

EN => Counter\_en,

load => Counter\_load\_out,

rst => reset,

updown => '0',

Q => Length\_count,

Zero => Zero\_out

);

reg: reg8 PORT MAP(

CLK => clock,

D => reg\_D,

EN => DataValid\_en,

RST => reset,

Q => reg\_Q

);

control: fsm PORT MAP(

clock => clock,

reset => reset,

DataIn => DataIn,

Checksum => reg\_Q,

LengthIsZero => Zero\_out,

Length\_init => Length\_init,

ChecksumError => ChecksumError,

DataValid => DataValid\_en,

Counter\_load => Counter\_load\_out

);

DataValid <= DataValid\_en;

Counter\_en <= DataValid\_en or Counter\_load\_out;

end arc;

**CheckSummer.vhd**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity CheckSummer is

Port ( clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR (7 downto 0);

ChecksumError : out STD\_LOGIC;

DataValid : out STD\_LOGIC);

end CheckSummer;

architecture arc of CheckSummer is

component datapath

PORT(

clock : in STD\_LOGIC;

reset : in STD\_LOGIC;

DataIn : in STD\_LOGIC\_VECTOR(7 downto 0);

ChecksumError : out STD\_LOGIC;

DataValid : out STD\_LOGIC

);

end component;

begin

dp : datapath PORT MAP(

clock => clock,

reset => reset,

DataIn => DataIn,

ChecksumError => ChecksumError,

DataValid => DataValid

);

end arc;