ECE 4525: Project Report 2 +Bonus

Sergei Akhmatdinov
Donovan Colo

December 8, 2021

Contents

1 Introduction

In this project report, the design and evaluation of a peripheral I/O (PIO) chip will be discussed. The PIO chip represents a subset of an existing Intel i82C55A chip, with only a single peripheral port, a modified control and status register and only modes 0 and 1 supported. Using the datasheet of the original chip and the given specifications, asynchronous state machines were designed to replicate the chip functionality.

The chip functionality was then tested using the Xilinx Vivado simula- tor, by emulating a behaviour of an external peripheral device and a CPU controlling the peripheral ports and inputs and data ports and inputs respec- tively.

The base project covered mode 0 and mode 1 output only. Input functionality was added as part of the project bonus. For convenience, since the bonus project is a complete system and uses identical code to the base with added functionality, this report will cover the final version of the project, with bonus parts added.

It should be noted that due to time constraints, the VHDL code demonstrated in this report may differ from the VHDL code used in the final demonstration. This is because parts of the code may be modified to make the final design conform better to any initially misunderstood specifications and/or some changes may be made to improve existing functionality shortly after the submission of this report.

2 Design

2.1 Finite State Machine Design

Due to the lengthy process involved in asynchronous state machine design, the paper-and-pencil design process was moved to the Appendix.

The process that was demonstrated in class and used previously in the laboratory assessments was applied to design a rising and a falling edge detector, along with handshake interfaces used by mode 1 for the input and output functionality.

Mode 0 output was implemented as a listener for a rising edge of the ac-

tive low WR signal. Upon receiving a rising edge from WR, the contents of the data bus are either written to the peripheral device or to the control register, depending on the state of the A0 input.

Due to transparency requirements for the Mode 0 input, the interface re- acts to the RD input being low to allow for the data bus to change while the pins on the peripheral inputs change.

Due to the short length of the pulse provided by the edge detectors, their state output was delayed using double inverters to lengthen the pulse.

Mode 1 handshake interface for the peripheral output works as follows: Initially, outputs OBF (Output Buffer Full) and INTR (Interrupt) are high. Input WR is then used to write data to the peripheral bus, using the same logic as Mode 0. A falling edge of WR clears the INTR output and a rising edge sets the OBF output. The system then waits for the peripheral device to acknoledge the output by pulsing the ACK input. Signals INTR and OBF are reset to their original states after a falling and a rising edge of ACK re- spectively. The whole process can then be repeated.

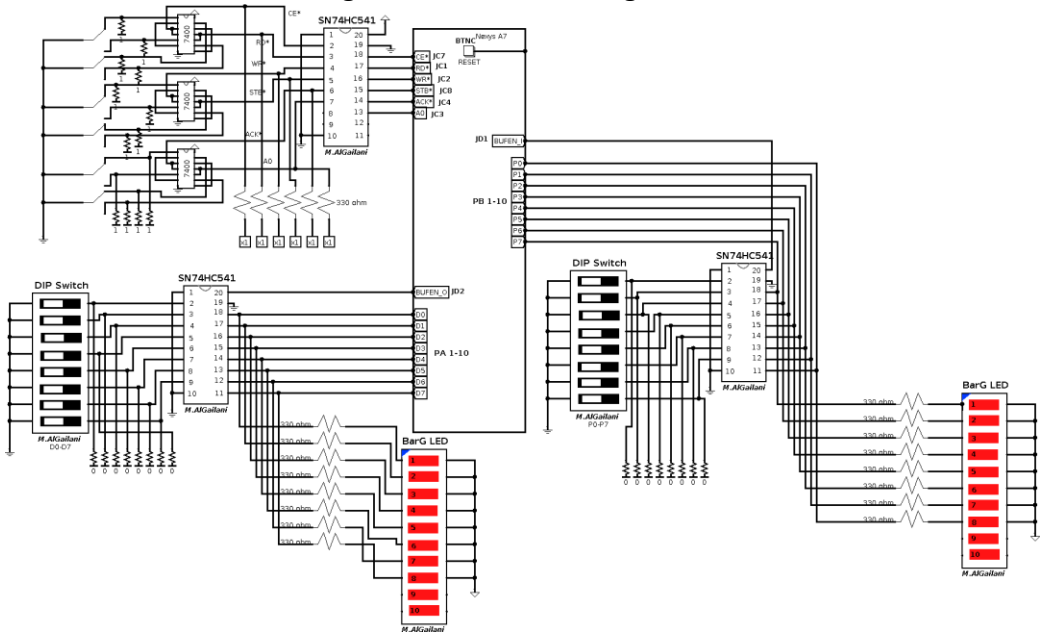
Mode 1 handshake interface for the peripheral input works as follows: Initially, outputs IBF (Input Buffer Full) and INTR (Interrupt) are low. The system waits to start input until a falling edge of the STB (Strobe) signal. The falling and rising edge of STB set signals IBF and INTR respectively. The system then waits to write the data to the data bus upon receiving a pulse of the RD signal, reacting to the falling edge of RD only to copy the data. The pulse of RD resets signals INTR on the falling edge and IBF on the rising edge. The whole process can then be repeated.

INTR output for modes 1 can be disabled by setting the control register bit 1 (INTE) to 0, in which case INTR will always be low.

2.2 Hardware Design

A simplified schematic diagram of the module is provided below:

Figure 1: Schematic Diagram



A 3-state setup was provided for the data and peripheral buses, with each input section protected by a non-inverting buffer. The input buffers are enabled only when input to the data or the peripheral buses is required.

3 quadruple NAND gate chips were used to provide bounce-free inputs CE, RD, WR, ACK, and A0. Although A0 was not required to be bounce free by the specifications, we decided that it was worth using an extra pair of NAND gates for the input. The RESET signal is represented by BTNC on the Nexys A7 board.

2.3 Material Links

Source materials used can be found in the appendix as follows:

- Project Summary Report
- VHDL Source Code
- Rising Edge Detector
- Falling Edge Detector

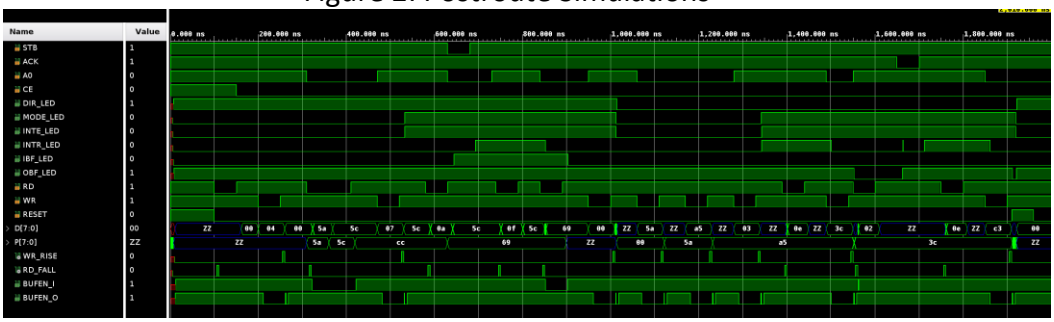
- Mode 1 Output State Machine Module
- Mode 1 Input State Machine Module
- Physical Constraints File
- Testing Stimuli

3 Design Evaluation

3.1 Simulations

Postroute simulations using thetesting stimuliwere first launched to test the design.

Figure 2: Postroute Simulations



Test steps were performed following the Demo Steps documents from the ECE 4525 website.

Since the design starts in mode 0 input (as per specs), it made sense to start testing the design from the bonus steps.

First, RD and CE relationship is checked. We make sure that the outputs remain in high impedance when CE = '1' or when RD and WR are both '0'. Then, we load the control register to mode 0 input. Nothing should change at this moment. Then, we attempt to input our data by writing 0x5a to the peripheral bus. At this moment, CE is 0, A0 is 0 and RD is pulsed low. Upon detecting the falling edge of RD, the bits in the data bus should mirror those in the peripheral bus. The bits in the peripheral bus are then changed

to test transparency. When the bits on the peripheral bus are switched to 0x5c, the bits on the data bus are changed accordingly. But after the rising edge, D register is latched and chages in P do not affect the data bus any longer. This is demonstrated by setting the peripheral bus to 0xcc.

Afterwards, the control register is loaded to mode 1 input, with interrupt enabled. This is done by switching the least significant bits on the data bus to 111 or setting the data bus to 0x07, setting A0 to 1 and pulsing the WR input. We also set the peripheral bus to 0x69 to anticipate the input operation.

To check the status register, we set A0 to 1 and pulse RD. Once in mode 1, this would show 010. The most significant bit (3) is irrelevant when we are in input mode. We then send a strobe pulse (STB) that sets IBF and INTR high. Now, when we read the status register, we should read 111. We then pulse RD to write the peripheral to the data bus. This sets IBF and INTR to 0.

After testing the input, we write the control register to perform output. This is done by writing 000 to the control register.

In output mode, the data bus register contents are copied to the peripheral upon detecting a rising edge of WR when A0 = 0.

We test this functionality in mode 0 by writing 0x5a and 0xa5. The data is latched after every write.

Mode 1 output is then tested by writing 011 to the control register. 0x3c is written to the peripheral. Status register is checked at critical steps, such as before generating the write (WR) and acknowledge pulses.

The system is then reset to check reset to mode 0 input.

4 Conclusions

The peripheral I/O chip performed as expected by the given specifications. Postroute simulations verified correct functionality, which was then further verified with hardware testing and demonstrations.

Asynchronous Finite State Machines for edge detection and handshaking interfaces were created using a pen-and-paper approach.

5 Team Contributions

Sergei

- FSM design
- VHDL Code
- Nexys A7 board hardware, P and D tri-state hardware and wiring
- Bonus Design
- This Report

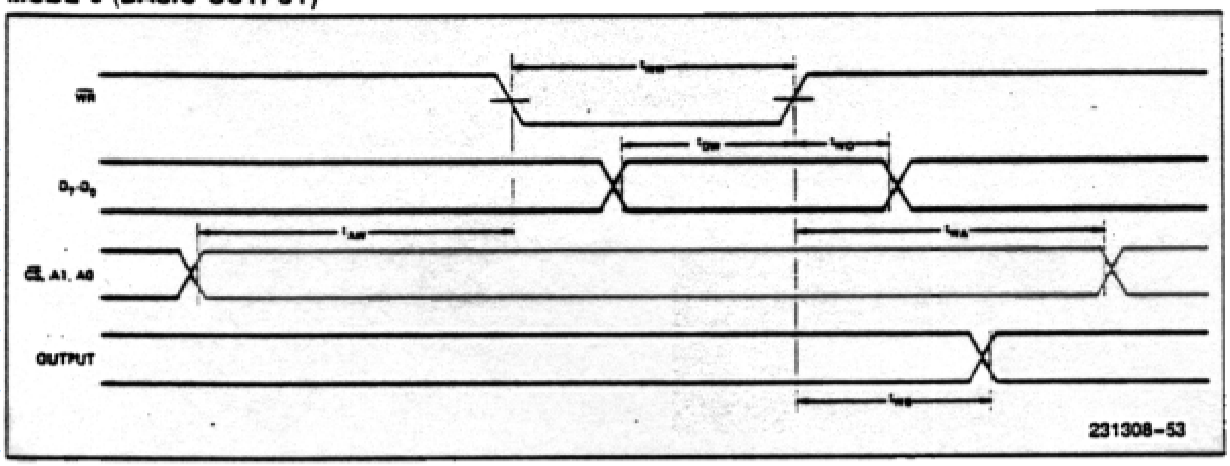
Donovan

- Project Schematic
- VHDL Code
- Bounce-free switch hardware and wiring
- Testing assistance
- Bonus Design

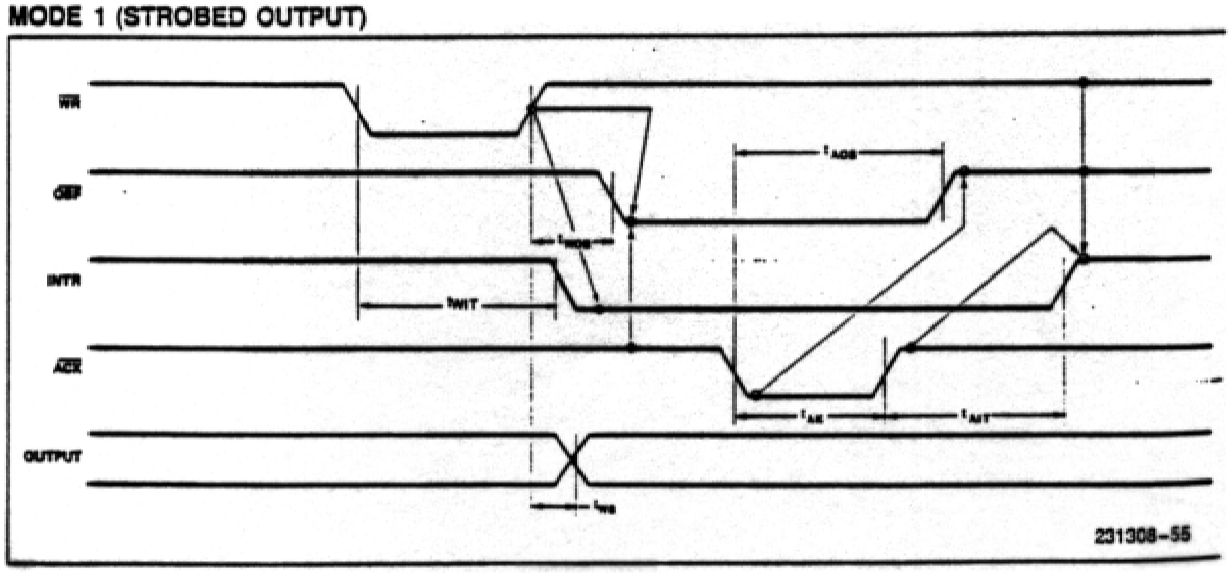
6 Appendix

6.1 FSM Design

MODE 0 (BASIC OUTPUT)



MODE 1 (STROBED OUTPUT)



Rising edge detector FSM

	$\overline{WR}$	$\overline{WR}$
	0	1
0 a	b 0	a 0
1 b	b 0	a 1

	0	1
0	0	0
1	0	1

$Z = WR \text{ AND } Y_1$

$Y_1 =$ not WR

Obviously free of essential hazards

	0	WR
0	1	0
Y_1	1	0

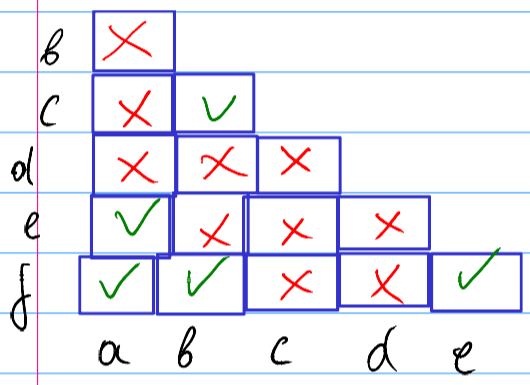
Needs some delay to increase pulse width, introduce double inverters

Mode 1 Out handshake

Outputs: INTR, OBF Primitive state table

st \ WR, ACK	00	01	11	10
a	xx	b 01	(a) 11	e 11
b	f 01	(b) 01	c 00	xx
c	xx	b 01	(c) 00	d 01
d	f 01	xx	a 11	(d) 01
e	f 01	xx	a 11	(e) 11
f	(f) 01	b 01	xx	e 11

Implication table



(a b c d e f)

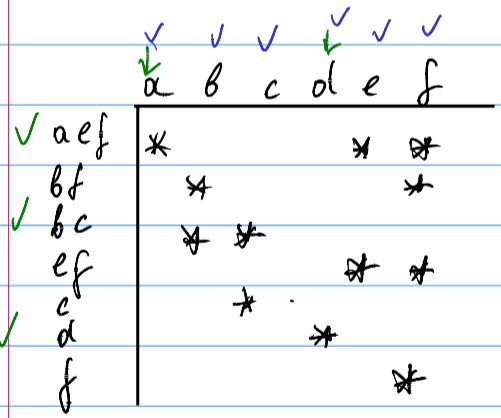
a: (aef) (b c d ef)

b: (bcf) (aef) (cd ef)

c. (c)(bf) (bc) (aef) (d ef)

d. (d)(c)(bf)(bc)(aef)(ef)

e: (ef)(d)(c)(bf)(bc)(aef)(f)



Selected states:

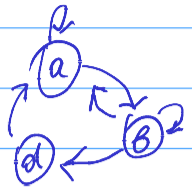
(aef) (bc) (d)

(a) (b) (d)

Reduced state table:

st \ WR, ACK	00	01	11	10
a	(a) 01	b 01	(a) 11	(a) 11
b	a 01	(b) 01	(b) 00	d 01
d	a 01	xx	a 11	(d) 01

Critical-race free assignment



y_2 \ y_1	a	b
d	z	

requires transition state z

State coding:

	y_1 y_2
a	00
d	01
z	11
b	10

Assigned state table:

s \ w, ack	00	01	11	10
a	(00)/01	10/01	(00)/11	(00)/11
d	01	00/01	11/x	00/11	(01)/01
z	11	01/x	10/x	10/x	01/01
b	10	00/01	(10)/01	(10)/00	11/01

Check essential hazards:

Left side:

00 → 10 → 00 → 10

no hazards

Right side:

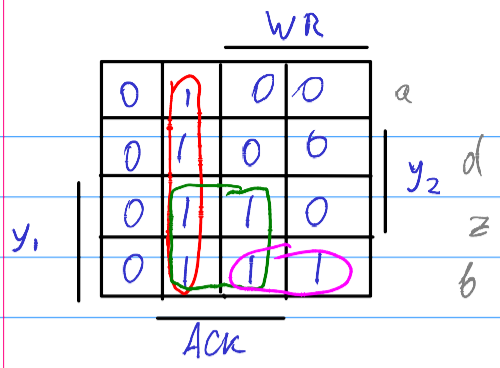
00 → 00 no hazard

01 → 00 → 00 → 00 delay y_2

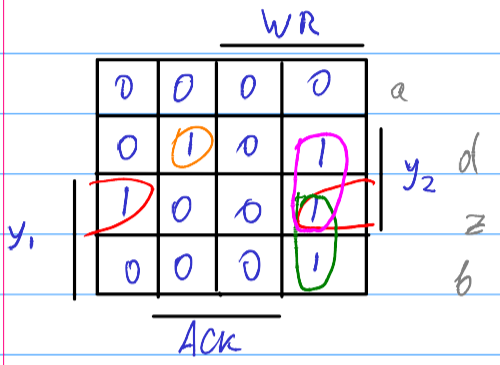
10 → 01 → 00 → 00 delay y_1

Both state vars delayed

K-maps + State & Output eqns (next page)

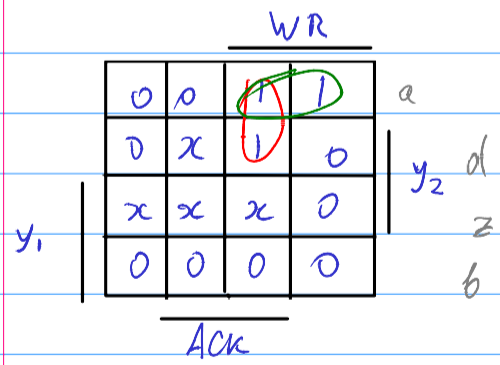


$$y_1 = (y_1 \cdot ACK) + (\overline{WR} \cdot ACK) + (y_1 \cdot \overline{y_2} \cdot WR)$$

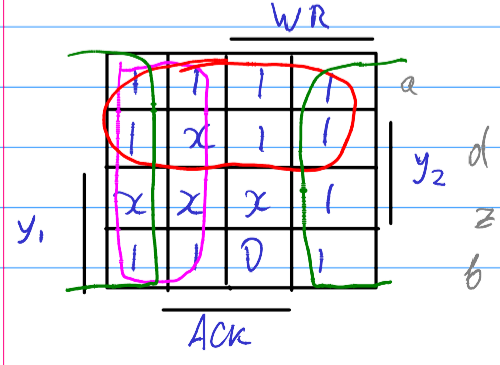


$$y_2 = (y_1 y_2 \cdot \overline{ACK}) + (y_1 \cdot WR \cdot \overline{ACK}) + (y_2 \cdot WR \cdot \overline{ACK}) +$$

$$+ (\overline{y_1} \cdot y_2 \cdot \overline{WR} \cdot ACK)$$



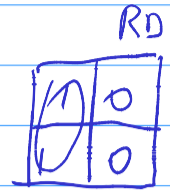
$$INTR = (\overline{y_1} y_2 \cdot WR) + (\overline{y_1} \cdot WR \cdot ACK)$$



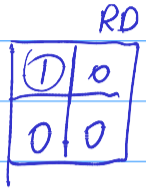
$$OBF = \overline{y_1} + \overline{WR} + \overline{ACK}$$

Falling edge detector for Mode 0 input

		$\overline{RD}$	$\overline{RD}$
		0	1
0	a	b1	(a)0
1	b	(b)0	a0



$Y1 = \text{not } RD$



$Z = \text{NOT } RD \text{ AND not } Y1$

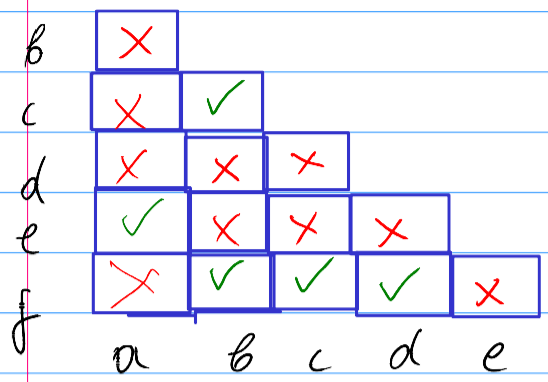
Introduce same delays as Mode 0 output

Mode 1 input — Primitive state table

out : INTR, IBF

$s \backslash rd,wb$	00	01	11	10
a	xx	e00	(a)00	b01
b	f01	xx	c11	(b)01
c	xx	d01	(c)11	b01
d	f01	(d)01	a00	xx
e	f01	(e)00	a00	xx
f	(f)01	d01	xx	b01

Implication table



(a b c d e f)

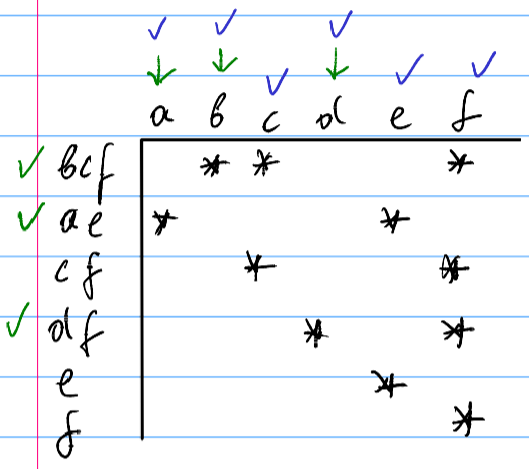
a: (a e) (b c d e f)

b: (b c f) (a e) (c d e f)

c: (c f) (b c f) (a e) (d e f)

d: (d f) (c f) (b c f) (a e) (e f)

e: (e) (d f) (c f) (b c f) (a e) (f)



Selected states:

~~(a e) (b c f) (d f)~~

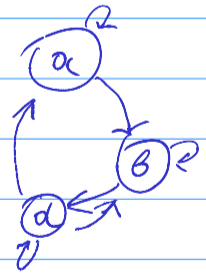
(a e) (b c f) (d)

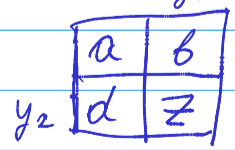
d b d

Reduced state table:

s \ Nxt	00	01	11	10
a	b 01	(a) 00	(a) 00	b 01
b	(b) 01	d 01	(b) 11	(b) 01
d	b 01	(d) 01	a 00	x x

Critical-race free assignment:

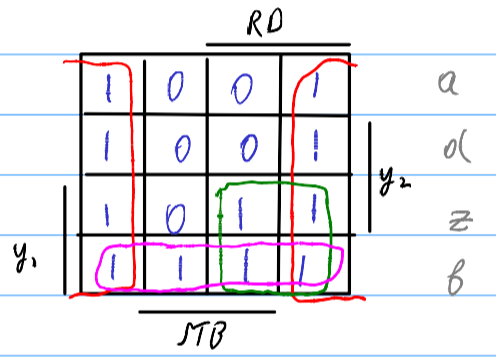




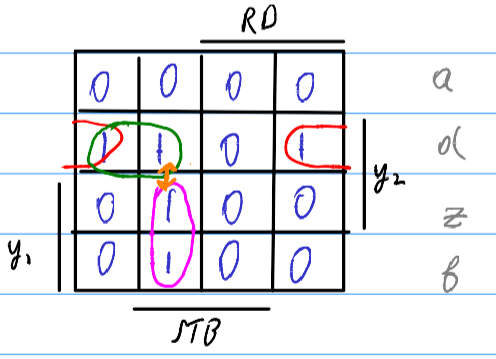
	y_1 y_2
a	00
d	01
z	11
b	10

Assigned state table

s \ RD/STB	00	01	11	10
a	00	10/01	(00)/00	(00)/00	10/01
d	01	11/01	(01)/01	00/00	11/x
z	11	10/01	01/01	10/11	10/x
b	10	(10)/01	11/01	(10)/11	(10)/01

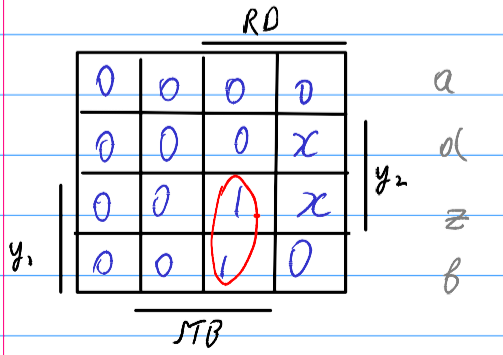


$$y_1 = (1\,\overline{STB}) + (y_1 \cdot RD) + (y_1 \cdot \overline{y_2})$$

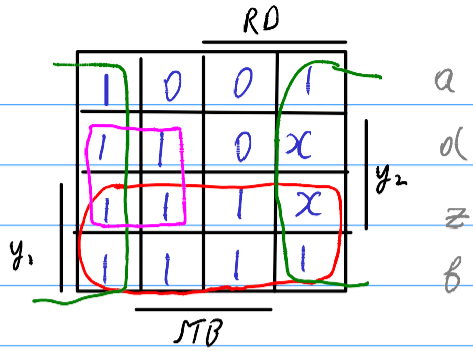


$$y_2 = (\overline{y_1} \cdot y_2 \cdot \overline{STB}) + (\overline{y_1} \cdot y_2 \cdot \overline{RD}) + (y_1 \cdot \overline{RD} \cdot STB) + (y_2 \cdot \overline{RD} \cdot STB)$$

Orange group fixes a static hazard



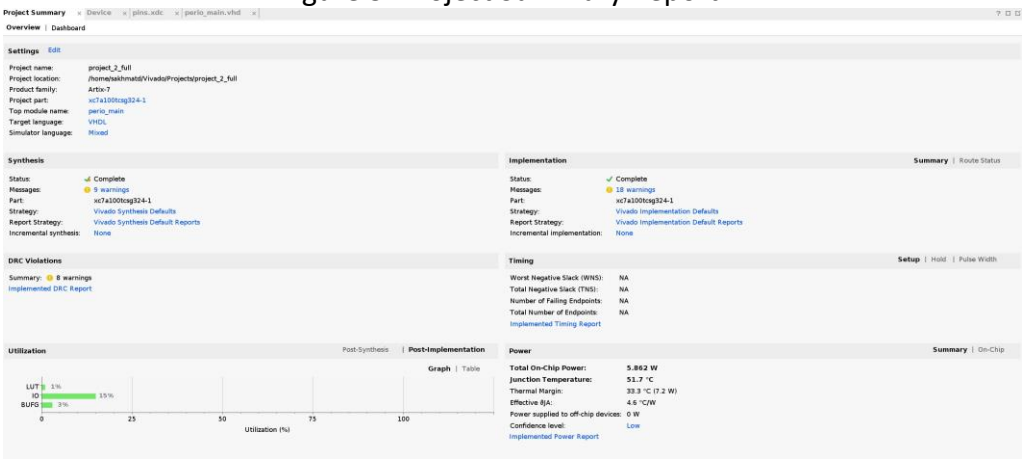
$$INTR = (y_1 \cdot RD \cdot STB)$$



$$IBF = (y_1) + (\overline{STB}) + (y_2 \cdot \overline{RD})$$

6.2 Project

Figure 3: Project Summary Report



```
libraryIEEE; useIEEE.STD_LOGIC_1164.ALL;

entityperio_mainis
        Port(
                CE:instd_logic;
                RD, WR:instd_logic;
                A0:instd_logic;
                RESET:instd_logic; ACK,
                STB:instd_logic;
                --
                D:inoutstd_logic_vector(7downto0); P:inoutstd_logic_vector(7downto0);
                --
                BUFEN_I, BUFEN_O:outstd_logic;
                --Debugsignals
                MODE_LED, INTE_LED, INTR_LED, OBF_LED, IBF_LED, DIR_LED:
    outstd_logic
        );
endperio_main;
architectureBehavioralofperio_mainis
        attributeDONT_TOUCH: string;
```

```
signalD_reg : std_logic_vector(7downto0) := (others=> '0'); signalP_reg :
std_logic_vector(7downto0) := (others=> '0'); signalMODE, INTE, INTR:
std_logic :='0';
signalDIR: std_logic := '1'; signalOBF,IBF:
std_logic:='1';

--Preventrisingedgedetectorsfrombeingoptimizedaway
signalWR_RISE: std_logic :='0';
signalRD_FALL: std_logic := '0'; attributeDONT_TOUCHofWR_RISE:signalis"true";
attributeDONT_TOUCHofRD_FALL:signalis"true";

signalINTR_I, INTR_II, OBF_I, IBF_I: std_logic;
--signalY1o,Y2o:std_logic;
--attributeDONT_TOUCHofY1o,Y2o:signalis"true";

signalmode1i_enable, mode1o_enable: std_logic; componentedge_detector
Port
(
        X:instd_logic;
        RISE:outstd_logic
);
endcomponent;
componentfedge_detector Port
(
        X:instd_logic;
        FALL:outstd_logic
);
endcomponent;
componentperio_mode1w
Port
(
        RESET, ENABLE:instd_logic; WR,
        ACK:instd_logic;
        INTR, OBF, Y1o, Y2o:outstd_logic
);
endcomponent;
componentperio_mode1r
Port
```

```
        (
                ENABLE:instd_logic;
                RESET:instd_logic;
                RD, STB, A0:instd_logic;
                INTR, IBF, Y1o, Y2o:inoutstd_logic
        );
        endcomponent;
begin

BUFEN_I <= '0'when((RD = '0'orRD_FALL = '1')andCE = '0'andA0 = '0'
    andDIR='1')else '1';
BUFEN_O<='0'when((WR='0'orWR_RISE='1')andCE='0')else '1';

mode1i_enable <= DIRandMODE;
mode1o_enable <=notDIRandMODE;

mode1o: perio_mode1w
portmap(
        ENABLE => mode1o_enable,
        RESET => RESET,
        WR => WR,
        ACK => ACK,
        INTR => INTR_I, OBF
        => OBF_I,
        Y1o =>open,
        Y2o =>open

);

mode1i: perio_mode1r
portmap(
        ENABLE => mode1i_enable,
        RESET => RESET,
        RD=> RD,
        STB => STB,
        A0 => A0,
        INTR => INTR_II, IBF
        => IBF_I,
        Y1o =>open,
        Y2o =>open

);

wr_edge: edge_detector portmap(
```

```
            X => WR,
            RISE => WR_RISE
);
rd_edge: fedge_detector portmap(
            X => RD,
            FALL => RD_FALL
);
INTR   <=   INTR_IandINTEwhen(mode1o_enable   =   '1'andRESET   =   '0')else
            INTR_IIandINTEwhen(mode1i_enable='1'andRESET='0')else '0';
INTR_LED <= INTR;
OBF<=OBF_Iwhen(mode1o_enable='1'andRESET='0')else '1';
OBF_LED <= OBF;
IBF<=IBF_Iwhen(mode1i_enable='1'andRESET='0')else '0';
IBF_LED <= IBF;

D <=        (others=> 'Z')when(CE ='1')else
            (0 => IBF, 1 => INTE, 2 => INTR, 3 => OBF,others=> '0')when
      (MODE='1'andRD='0'andA0='1')else
            P_regwhen(DIR ='1')else
            (others=> 'Z');

D_reg <= (others=> '0')when(RESET = '1')else
              Dwhen(WR_RISE = '1'andA0 = '0'andCE = '0'andDIR = '0')
      else
              D_reg;

P_reg <= (others=> '0')when(RESET = '1')else
            Pwhen(RD_FALL='1'andA0='0'andCE='0'andDIR='1'and MODE = '1')else
            Pwhen(RD = '0'andA0 = '0'andCE = '0'andDIR = '1'andMODE
            ='0')else
              P_reg;
P <=        (others=> 'Z')when(CE ='1')else
            D_regwhen(DIR = '0')else (others=>
            'Z');
```

```
--Controlregister

MODE <= '0'when(RESET = '1')else
        D(0)when(WR_RISE='1'andA0='1'andCE='0')else MODE;
MODE_LED <= MODE;

INTE <= '0'when(RESET = '1')else
        D(1)when(WR_RISE='1'andA0='1'andCE='0')else INTE;
INTE_LED <= INTE;

DIR <= '1'when(RESET = '1')else
        D(2)when(WR_RISE='1'andA0='1'andCE='0')else DIR;
DIR_LED <= DIR;

endBehavioral;
```

Listing 1: Source Code

```
libraryIEEE; useIEEE.STD_LOGIC_1164.ALL;

entityedge_detectoris Port
        (
                X:instd_logic;
                RISE:outstd_logic
        );
endedge_detector;

architectureBehavioralofedge_detectoris
        --Thispreventstheentiremodulefrom
        --beingoptimizedaway,fromXilinxManual
        --
    https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug901-vivado-synthesis.pd
        attributeDONT_TOUCH: string;
        signalY1, delay1, delay2, delay3, delay4, delay5, delay6, delay7, delay8 : std_logic;
        attributeDONT_TOUCHofY1, delay1, delay2, delay3, delay4, delay5,
    delay6, delay7, delay8:signalis"true";
begin
        --Adddelaystolengthenthedurationof
```

```
		--thepulse
		Y1 <=not(not(notX)); delay1
		<=not(notY1); delay2
		<=not(notdelay1); delay3
		<=not(notdelay2); delay4
		<=not(notdelay3); delay5
		<=not(notdelay4); delay6
		<=not(notdelay5); delay7
		<=not(notdelay6); delay8
		<=not(notdelay7); RISE <=
		Xanddelay8;

endBehavioral;
```

Listing 2: Rising Edge Detector

```
libraryIEEE; useIEEE.STD_LOGIC_1164.ALL;

entityfedge_detectoris Port
		(
			X:instd_logic;
			FALL:outstd_logic
		);
endfedge_detector;

architectureBehavioraloffedge_detectoris
		--Thispreventstheentiremodulefrom
		--beingoptimizedaway,fromXilinxManual
		--
	https://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug901-vivado-synthesis.pd
		attributeDONT_TOUCH: string;
		signalY1, delay1, delay2, delay3, delay4, delay5, delay6, delay7, delay8 : std_logic;
		attributeDONT_TOUCHofY1, delay1, delay2, delay3, delay4, delay5,
	delay6, delay7, delay8:signalis"true";
begin
		--Adddelaystolengthenthedurationof
		--thepulse
		Y1 <=not(not(notX)); delay1
		<=not(notY1); delay2
		<=not(notdelay1); delay3
		<=not(notdelay2);
```

```
        delay4      <=not(notdelay3);
        delay5      <=not(notdelay4);
        delay6      <=not(notdelay5);
        delay7      <=not(notdelay6);
        delay8      <=not(notdelay7);
        FALL<=notXandnotdelay8;

endBehavioral;
```

Listing 3: Falling Edge Detector

```
libraryIEEE; useIEEE.STD_LOGIC_1164.ALL;

entityperio_mode1ris Port
        (
                ENABLE:instd_logic;
                RESET:instd_logic;
                RD, STB, A0:instd_logic;

        );      INTR, IBF, Y1o, Y2o:inoutstd_logic

endperio_mode1r;

architectureBehavioralofperio_mode1ris signaly1,y2:
        std_logic:= '0';

begin
        --needtolatchstatesandoutputsifA0='1'toallow
        --forstatusregreading y1 <=
        ((notSTB)or
                (y1andRD)or (y1andnoty2))andnotRESETandENABLEwhen(A0
                = '0')
    else

                y1;

        y2   <=  ((noty1andy2andnotSTB)or
                (noty1andy2andnotRD)or
                (y1andnotRDandSTB)or
                (y2andnotRDandSTB))andnotRESETandENABLEwhen(A0
    = '0')else

                y2;

        INTR<=y1andRDandSTBwhen(A0='0')else INTR;
```

```
        IBF<=y1ornotSTBor(y2andnotRD)when(A0='0')else IBF;
endBehavioral;
```

Listing 4: Mode 1 Input

```
libraryIEEE; useIEEE.STD_LOGIC_1164.ALL;
entityperio_mode1wis
        Port
        (
                ENABLE:instd_logic;
                RESET:instd_logic;   WR,
                ACK:instd_logic;
                INTR, OBF, Y1o, Y2o:outstd_logic
        );
endperio_mode1w;

architectureBehavioralofperio_mode1wis
        --attributeDONT_TOUCH:string;
        signaly1,y2:std_logic:= '0';
        --attributeDONT_TOUCHofy1,y2:signalis"true";
begin

        --y1<=((ACKandY1)or
        --(notWRandACK)or
        --(WRandY1andnotY2))andnotRESETandENABLE;

        y1 <= ((y1andACK)or
                (notWRandACK)or
                (y1andnoty2andWR))andnotRESETandENABLE;

        --y2<=((WRandnotACKandY2)or
        --(notWRandY1andY2)or
        --(WRandnotACKandY1)or
        --(notWRandACKandnotY1andY2))andnotRESETand ENABLE;
        y2    <=    ((y1andy2andnotACK)or
                (y1andWRandnotACK)or
                (y2andWRandnotACK)or
```

```
                    (notY1andy2andnotWRandACK))andnotRESETand
    ENABLE;
        --INTR<=(WRandACKandnotY1)or(WRandnotY1andnotY2); INTR
        <= (notY1andnoty2andWR)or(notY1andWRandACK);

        OBF<=notY1ornotWRornotACK;

        Y1o <= Y1;
        Y2o <= Y2;
endBehavioral;
```

Listing 5: Mode 1 Output

```
set_property ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets D*] set_property
ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets IBF*] set_property
ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets OBF*] set_property
ALLOW_COMBINATORIAL_LOOPS TRUE [get_nets INTR_LED*]

# A0 gets registered as a clock

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets A0*]

set_property -dict { PACKAGE_PIN C17 IOSTANDARD LVCMOS33 } [get_ports { D[0] }];
    #IO_L20N_T3_A19_15 Sch=ja[1]
set_property -dict { PACKAGE_PIN D18 IOSTANDARD LVCMOS33 } [get_ports { D[1] }];
    #IO_L21N_T3_DQS_A18_15 Sch=ja[2]
set_property -dict { PACKAGE_PIN E18 IOSTANDARD LVCMOS33 } [get_ports { D[2] }];
    #IO_L21P_T3_DQS_15 Sch=ja[3]
set_property -dict { PACKAGE_PIN G17 IOSTANDARD LVCMOS33 } [get_ports { D[3] }];
    #IO_L18N_T2_A23_15 Sch=ja[4]
set_property -dict { PACKAGE_PIN D17 IOSTANDARD LVCMOS33 } [get_ports { D[4] }];
    #IO_L16N_T2_A27_15 Sch=ja[7]
set_property -dict { PACKAGE_PIN E17 IOSTANDARD LVCMOS33 } [get_ports { D[5] }];
    #IO_L16P_T2_A28_15 Sch=ja[8]
set_property -dict { PACKAGE_PIN F18 IOSTANDARD LVCMOS33 } [get_ports { D[6] }];
    #IO_L22N_T3_A16_15 Sch=ja[9]
set_property -dict { PACKAGE_PIN G18 IOSTANDARD LVCMOS33 } [get_ports { D[7] }];
    #IO_L22P_T3_A17_15 Sch=ja[10]

set_property -dict { PACKAGE_PIN D14 IOSTANDARD LVCMOS33 } [get_ports { P[0] }];
    #IO_L1P_T0_AD0P_15 Sch=jb[1]

set_property -dict { PACKAGE_PIN F16 IOSTANDARD LVCMOS33 } [get_ports {
```

```
P[1] }]; #IO_L14N_T2_SRCC_15 Sch=jb[2]
set_property -dict { PACKAGE_PIN G16 IOSTANDARD LVCMOS33 } [get_ports{P[2] }];
#IO_L13N_T2_MRCC_15 Sch=jb[3]
set_property -dict { PACKAGE_PIN H14 IOSTANDARD LVCMOS33 } [get_ports { P[3] }];
#IO_L15P_T2_DQS_15 Sch=jb[4]
set_property -dict { PACKAGE_PIN E16 IOSTANDARD LVCMOS33 } [get_ports { P[4] }];
#IO_L11N_T1_SRCC_15 Sch=jb[7]
set_property -dict { PACKAGE_PIN F13 IOSTANDARD LVCMOS33 } [get_ports { P[5] }];
#IO_L5P_T0_AD9P_15 Sch=jb[8]
set_property -dict { PACKAGE_PIN G13 IOSTANDARD LVCMOS33 } [get_ports { P[6] }];
#IO_0_15 Sch=jb[9]
set_property -dict { PACKAGE_PIN H16 IOSTANDARD LVCMOS33 } [get_ports { P[7] }];
#IO_L13P_T2_MRCC_15 Sch=jb[10]


set_property -dict { PACKAGE_PIN K1 IOSTANDARD LVCMOS33 } [get_ports { RD
}]; #IO_L23N_T3_35 Sch=jc[1]
set_property -dict { PACKAGE_PIN F6 IOSTANDARD LVCMOS33 } [get_ports { WR
}]; #IO_L19N_T3_VREF_35 Sch=jc[2]
set_property -dict { PACKAGE_PIN J2 IOSTANDARD LVCMOS33 } [get_ports { A0
}]; #IO_L22N_T3_35 Sch=jc[3]
set_property -dict { PACKAGE_PIN G6 IOSTANDARD LVCMOS33 } [get_ports { ACK
}]; #IO_L19P_T3_35 Sch=jc[4]
set_property -dict { PACKAGE_PIN E7 IOSTANDARD LVCMOS33 } [get_ports { CE
}]; #IO_L6P_T0_35 Sch=jc[7]
set_property -dict { PACKAGE_PIN J3 IOSTANDARD LVCMOS33 } [get_ports { STB
}]; #IO_L22P_T3_35 Sch=jc[8]

set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports {
RESET }]; #IO_L9P_T1_DQS_14 Sch=btnc
#set_property -dict { PACKAGE_PIN J4 IOSTANDARD LVCMOS33 } [get_ports{STB }];
#IO_L21P_T3_DQS_35 Sch=jc[9]
#set_property -dict { PACKAGE_PIN E6 IOSTANDARD LVCMOS33 } [get_ports { JC[10] }];
#IO_L5P_T0_AD13P_35 Sch=jc[10]
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports{MODE_LED }];
#IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports{INTE_LED }];
#IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports{INTR_LED }];
#IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports{OBF_LED }];
#IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports{IBF_LED }];
#IO_L7P_T1_D09_14 Sch=led[4]
```

```
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { DIR_LED }];
      #IO_L18N_T2_A11_D27_14 Sch=led[5]

set_property -dict { PACKAGE_PIN H4 IOSTANDARD LVCMOS33 } [get_ports { BUFEN_I }];
      #IO_L21N_T3_DQS_35 Sch=jd[1]
set_property -dict { PACKAGE_PIN H1 IOSTANDARD LVCMOS33 } [get_ports { BUFEN_O }];
      #IO_L17P_T2_35 Sch=jd[2]

#set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports{LED[6] }];
      #IO_L17P_T2_A14_D30_14 Sch=led[6]
#set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports{LED[7] }];
      #IO_L18P_T2_A12_D28_14 Sch=led[7]
#set_property -dict { PACKAGE_PIN V16 IOSTANDARD LVCMOS33 } [get_ports{LED[8] }];
      #IO_L16N_T2_A15_D31_14 Sch=led[8]
#set_property -dict { PACKAGE_PIN T15 IOSTANDARD LVCMOS33 } [get_ports { LED[9] }];
      #IO_L14N_T2_SRCC_14 Sch=led[9]
#set_property -dict { PACKAGE_PIN U14 IOSTANDARD LVCMOS33 } [get_ports { LED[10] }];
      #IO_L22P_T3_A05_D21_14 Sch=led[10]
#set_property -dict { PACKAGE_PIN T16 IOSTANDARD LVCMOS33 } [get_ports { LED[11] }];
      #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
#set_property -dict { PACKAGE_PIN V15 IOSTANDARD LVCMOS33 } [get_ports { LED[12] }];
      #IO_L16P_T2_CSI_B_14 Sch=led[12]
#set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports{LED[13] }];
      #IO_L22N_T3_A04_D20_14 Sch=led[13]
#set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports{LED[14] }];
      #IO_L20N_T3_A07_D23_14 Sch=led[14]
#set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33 } [get_ports { LED[15] }];
      #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
```

Listing 6: Physical Constraints

```
restart

add_force STB 1
add_force ACK 1
add_force CE 1
add_force A0 1
add_force RD 1
add_force WR 1
add_force RESET 1 run
100ns
```

```
add_force RD 0
add_force RESET 0

run 50ns

add_force RD 1
add_force CE 0
run 50ns
add_force -radix hex D 0x04 add_force
WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D
run 50ns
add_force -radix hex P 0x5a
add_force A0 0
add_force RD 0

run 50 ns
add_force -radix hex P 0x5c run
50ns
add_force RD 1
run 10ns
add_force -radix hex P 0xcc
run 50ns
add_force -radix hex D 0x07
add_force A0 1
add_force WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D
run 50ns
```

```
add_force RD 0
run 50ns
add_force RD 1

add_force -radix hex P 0x69
add_force A0 0
add_force STB 0
run 50ns
add_force STB 1
run 50ns

add_force A0 1
run 10ns
add_force RD 0
run 50ns
add_force RD 1
run 50ns

add_force A0 0

add_force RD 0
run 50ns

add_force RD 1
run 10ns
remove_force P
run 50ns

puts"Output"

add_force -radix hex D 0x00 add_force
A0 1
add_force WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D
run 50ns


add_force A0 0
add_force -radix hex D 0x5a
add_force WR 0
run 50ns
add_force WR 1
run 10ns
```

```
remove_force D
run 50ns

add_force -radix hex D 0xa5 add_force
WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D
run 50ns


add_force A0 1
add_force -radix hex D 0x03
add_force WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D
run 50ns

add_force RD 0
run 50ns
add_force RD 1
run 50ns


add_force A0 0
add_force -radix hex D 0x3c
add_force WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D


add_force A0 1
add_force RD 0
run 50ns
add_force RD 1
run 50ns

add_force ACK 0
run 50ns
add_force ACK 1
run 50ns

add_force RD 0
```

```
run 50ns
add_force RD 1
run 50ns


add_force A0 0
add_force -radix hex D 0xc3
add_force WR 0
run 50ns
add_force WR 1
run 10ns
remove_force D

add_force RESET 1
run 50ns add_force
RESET 0 run 50ns
```

Listing 7: Testing File