Kyle Keislar

ECE524/L FPGA/ASIC Design and Optimization Using VHDL Lab



Lab 3

FPAG Features Clock Management, DSP Blocks, DDR, and SRL

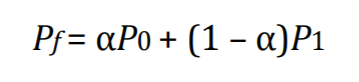
10/09/2020

## 

## Introduction & Problem Statements

In this experiment a basic interleaver was designed. The interleaver combines data from two sources (P0, P1) by measure of a blending factor( alpha and its compliment). The output of the interleaver is then the sum of the products of its two sources and blending factor. The behavior of the interleaver is described in equaition 3.1 below

*Eq 3.1: Interleaver Behavior*



To explore FPGA utilizations and resource mapping, multiple implementations of the fundamental interleaver can be derived. In figures 3.0A-C, increasing complex iterations of the design are realized at the high level.

Fig 3.0.A: Signal Stream Sender High-Level Block Diagram

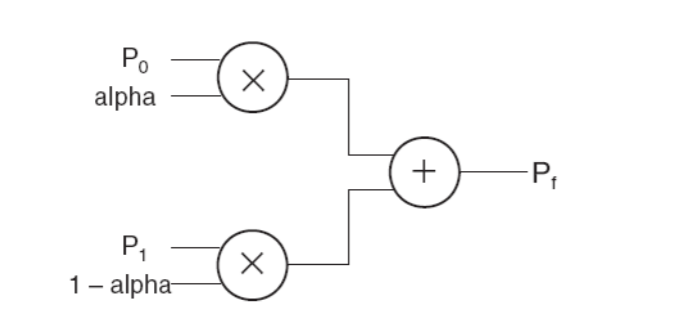


Fig 3.0.B: Signal Stream Interleaver High-Level Block Diagram

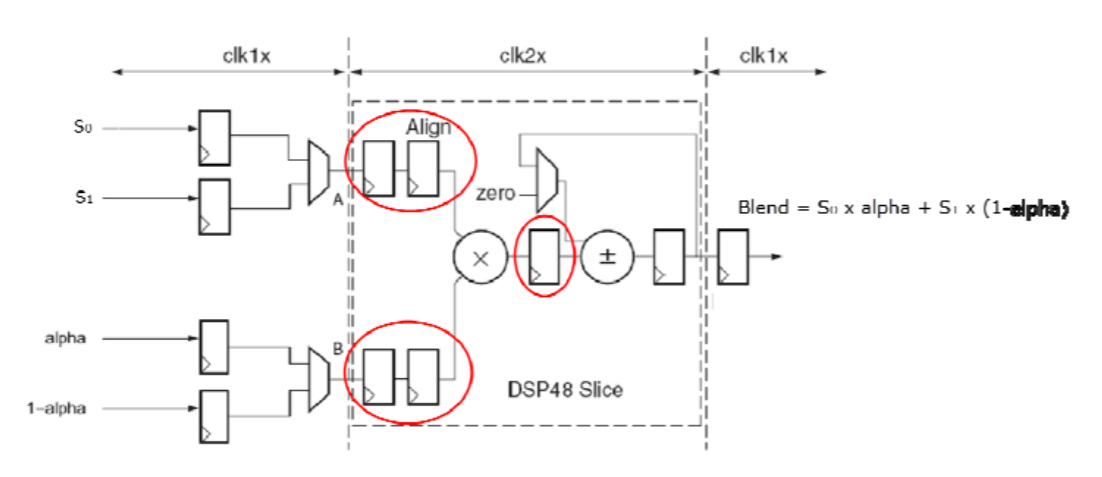
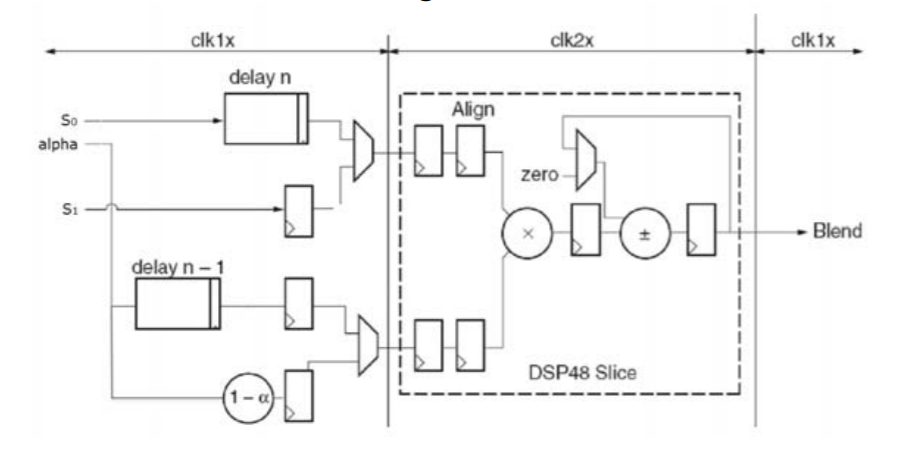


Fig 3.0.B: Signal Stream Interleaver High-Level Block Diagram



The code for the synthesized design of each of these files can be found in the Appendix below.

An implemented design of this filter was not realized using the Zedboard for this experiment.

## Procedure:

***Part I (Direct Implementation)***

Task 1: Write VHDL code to implement the circuit shown in Figure 3.1. In order to verify the design, you can apply two sine waves on P0 and P1 with different amplitude. Change alpha value from 0 to 1. The blend output should make transition from P0 to P1 waveform. You should test your design for several cases in order to prove the correct functionality. For instance, case of alpha = 1 or alpha = 0 to show the output shows either one of the waveforms. A more interesting simulation case should be included to prove the functionality for a linear transition of alpha from 0 to 1. Assume P 0 and P1 are 8 bit values and 0≼ alpha ≼1. Prove that your design is working by providing the simulation.

The VHDL test benches and design for this task can be seen below in the Appendix as items A.1 and A.2.

***Part II (Optimized Implementation)***

Task 2: For the optimized implementation, there are two clock domains in this design. Instantiate theclock manager module with 50 MHz clock input and deskewed output and use the clk2x output of the clock manager to generate 100 MHz clock for your DSP block. Write complete VHDL code to infer all blocks (other than DCM). Prove that your design is working by providing the simulation. You can use the testbench you developed in the first part of the lab.

The VHDL test benches and design for this task can be seen below in the Appendix as items A.1 and A.3.

***Part III (Equalizing delays)***

Task 3: : In the third part of the lab, apply 64 clock cycle delay to P0stream using SRL componentsaccording to Figure 3.3. You can use IP Integrator to generate the SRL components. The testbench developed in the first part of the lab should be slightly modified so you can apply it to the third part of the lab. Provide the simulation waveform to prove your design is functioning.

The VHDL test benches and design for this task can be seen below in the Appendix as items A.4 and A.5. The synthesizable shift register used in this portion of the experiment was utilized from the Xilnix Synthesis Guide and can be seen in the Appendix as item A.6.

Task 4: : Replace the SRL components with Block RAMs (BRAMs) that create the same amount of delayand prove your design works by providing simulation.

## Results (Data):

## *Waveforms graphed from generated & simulated data:*

Fig 3.1: Direct Implementation: Alpha= 1

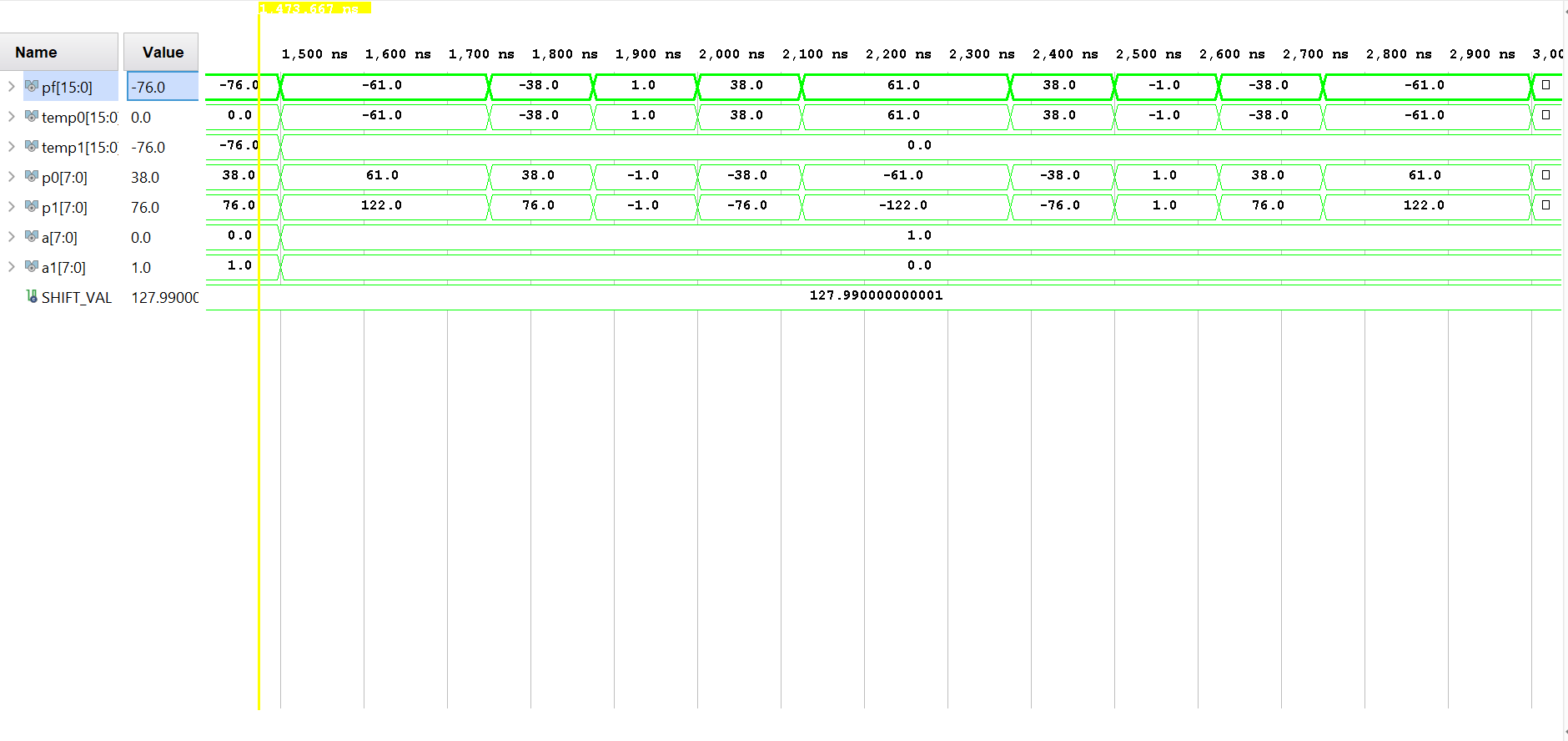


Fig 3.2: Direct Implementation: Alpha changing from 1 to 0

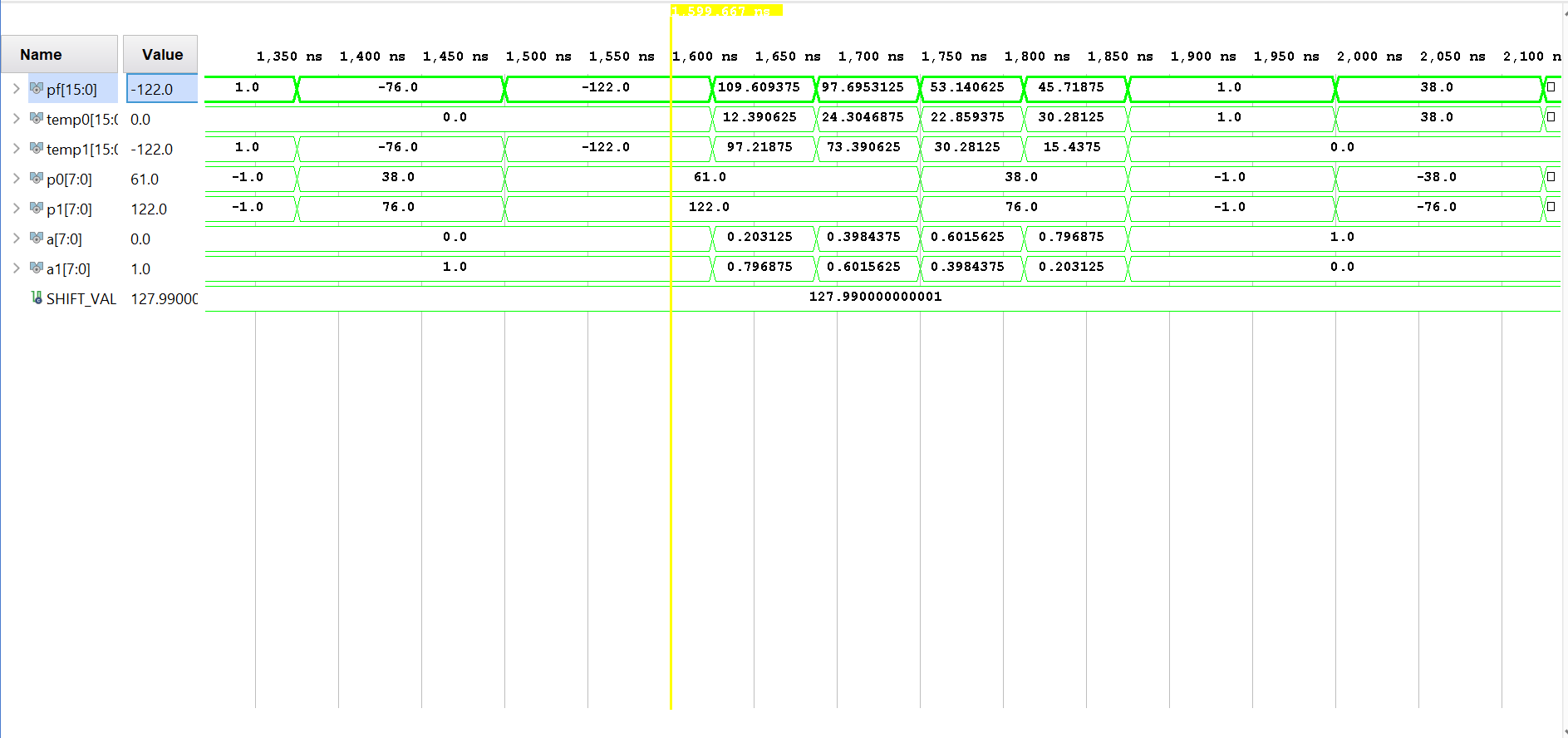


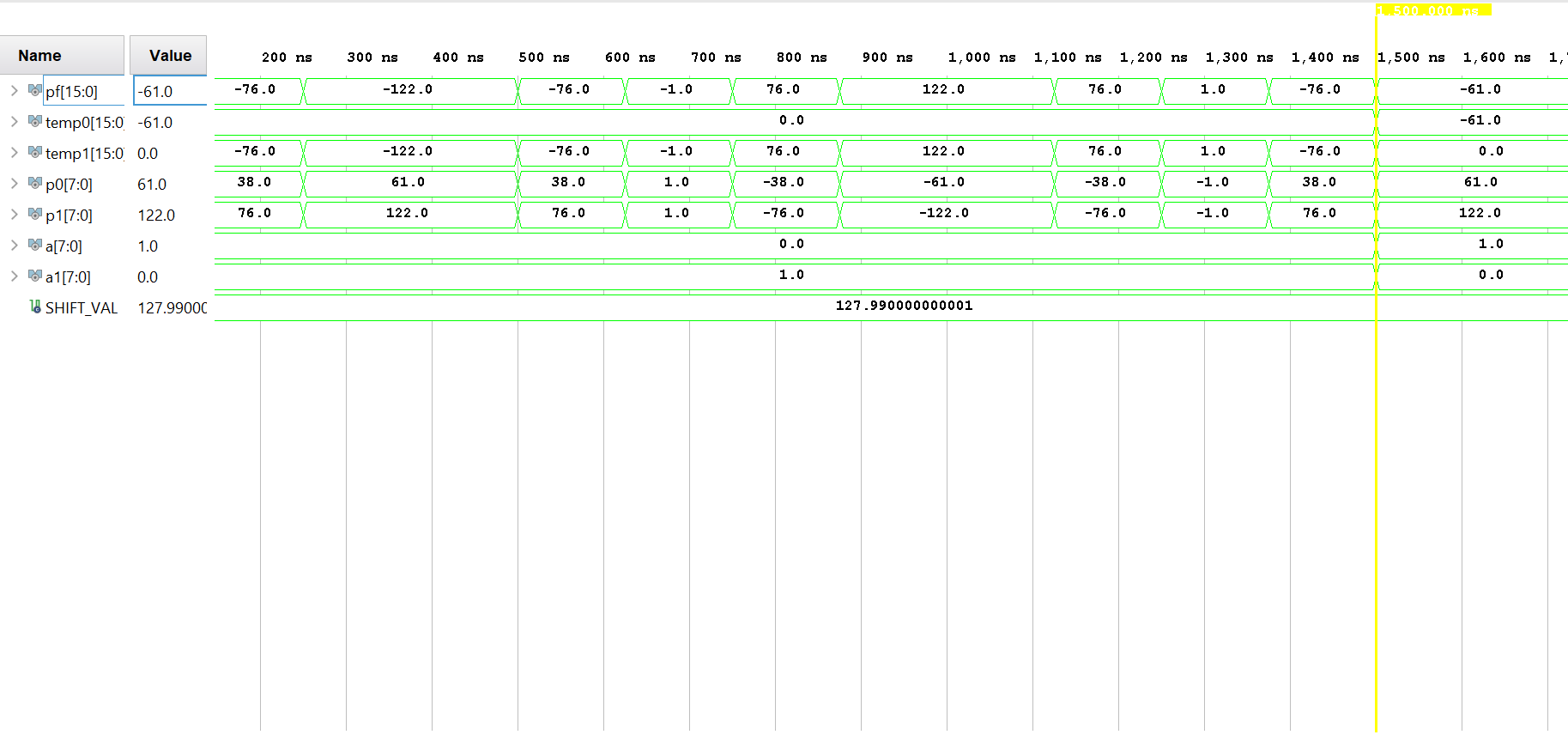
Fig 3.3: Direct Implementation: Alpha= 0

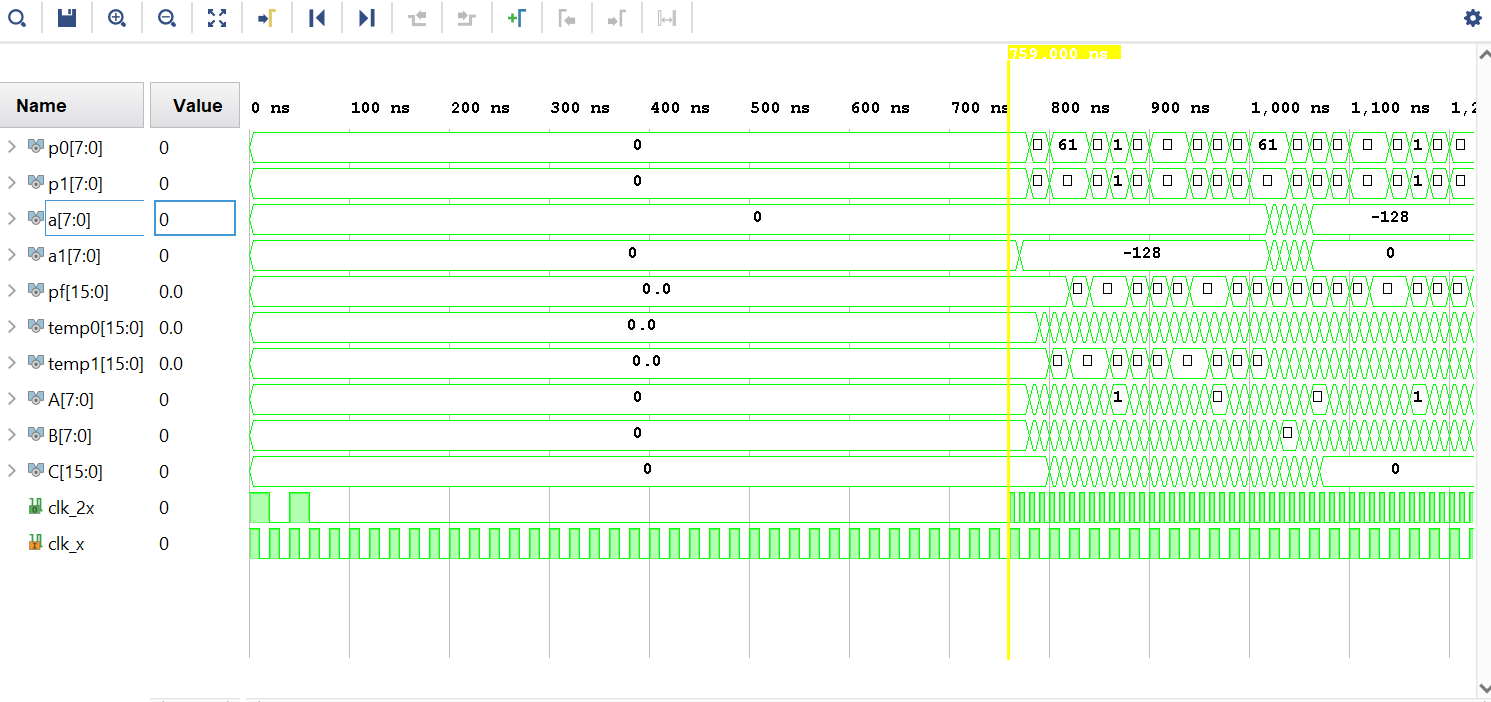
Fig 3.4: Optimized Implementation: Alpha= 1; Clock Sync Delay 

Fig 3.5: Optimized Implementation: Alpha= 1; Clocks Synced

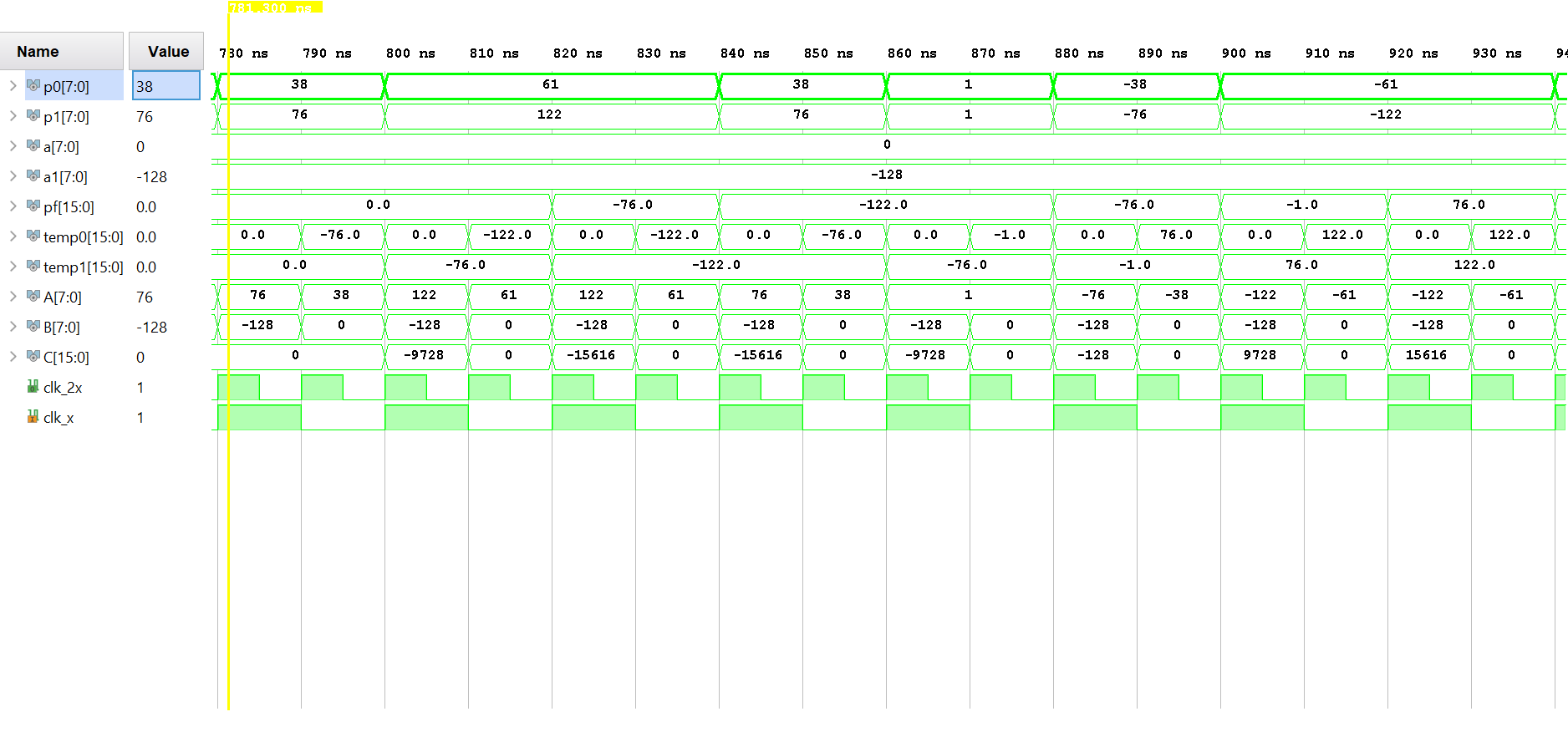


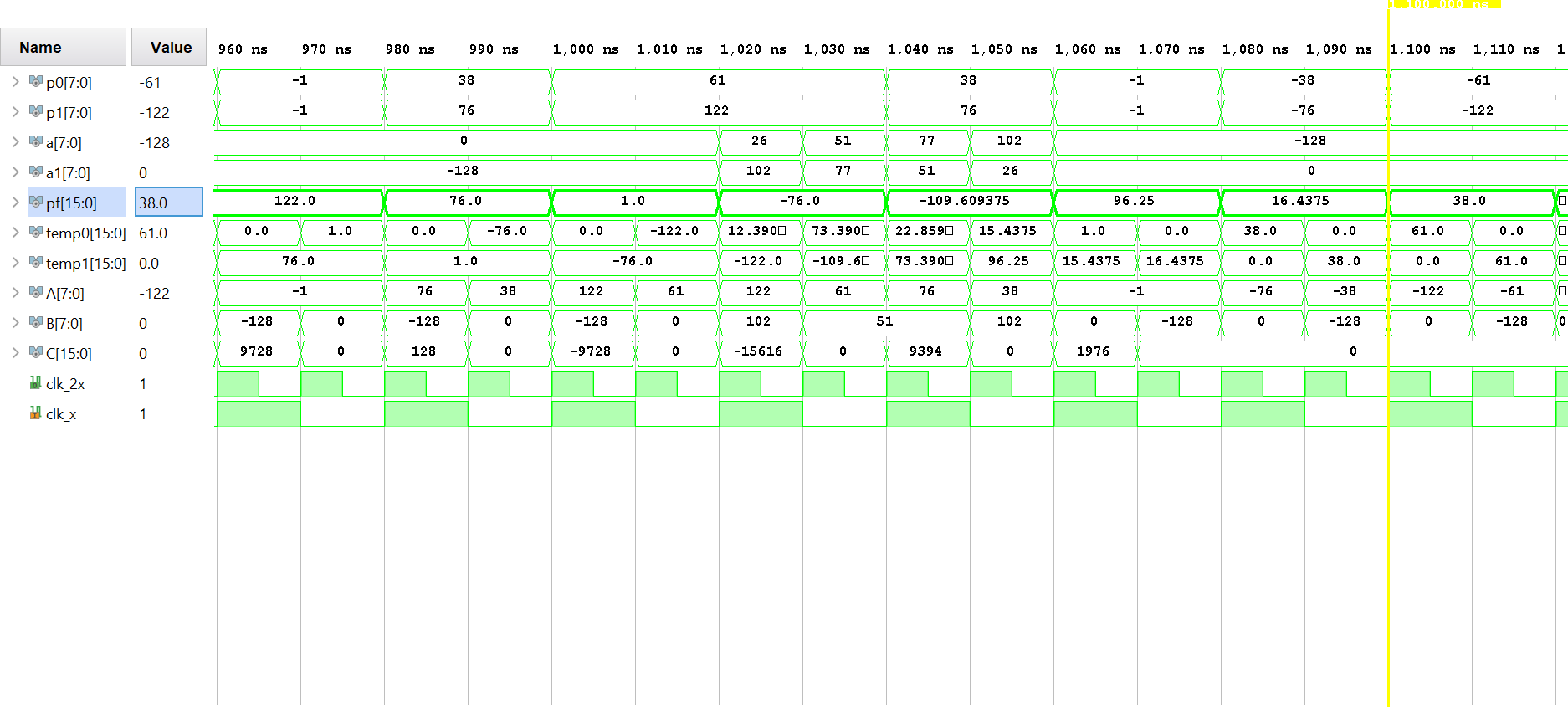
Fig 3.6: Optimized Implementation: Alpha changing from 1 to 0

Fig 3.7: Optimized Implementation: Alpha= 0

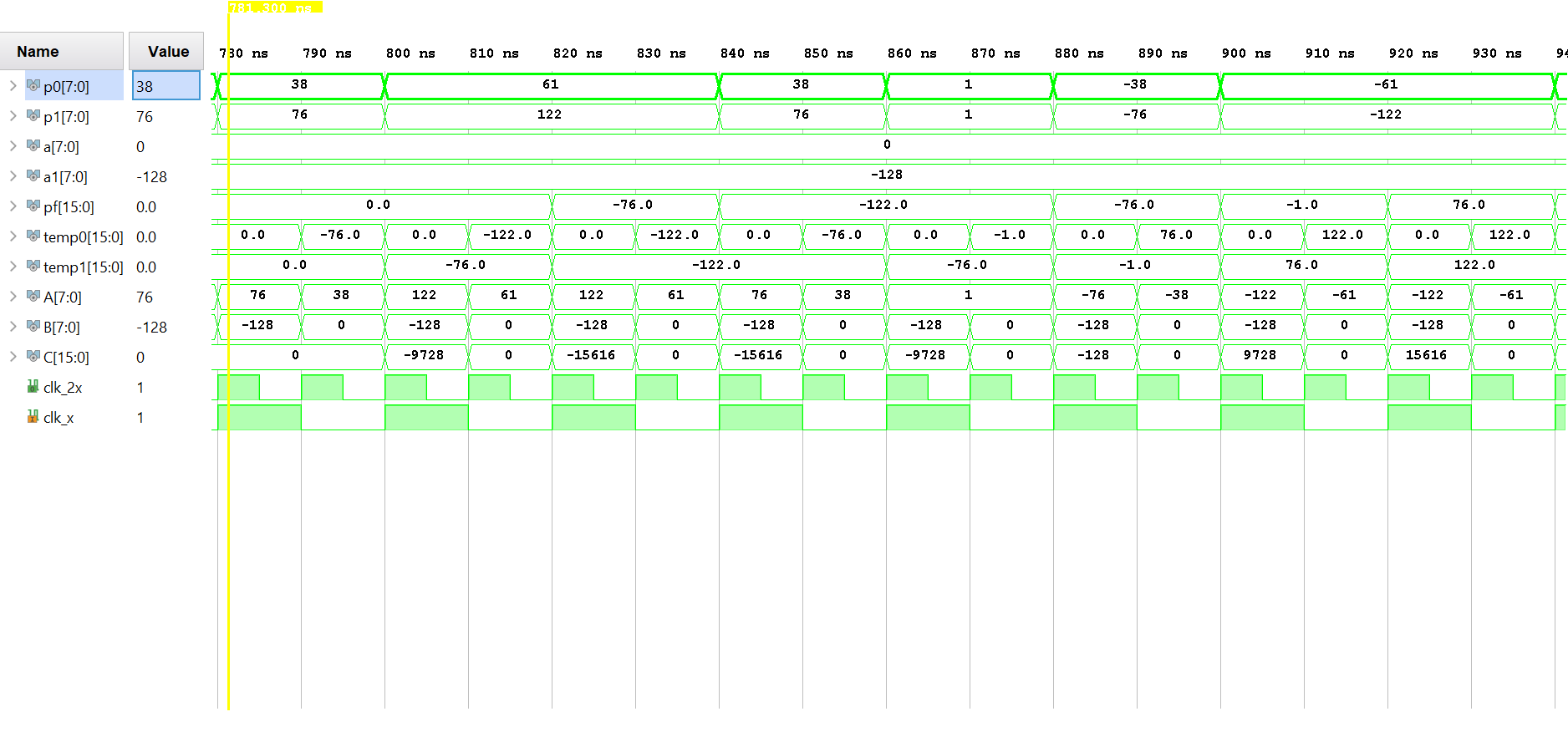


Fig 3.8: Equalized Interleaver Delay: P\_0 input read

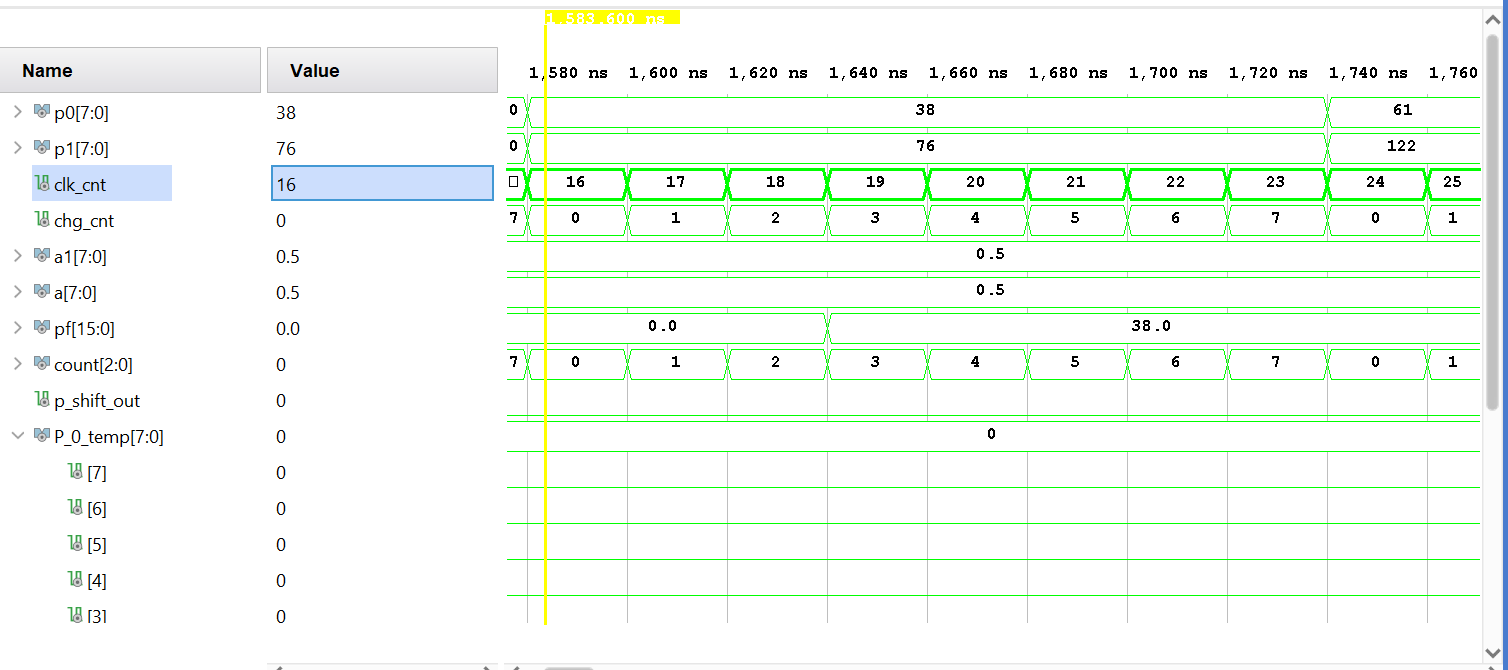
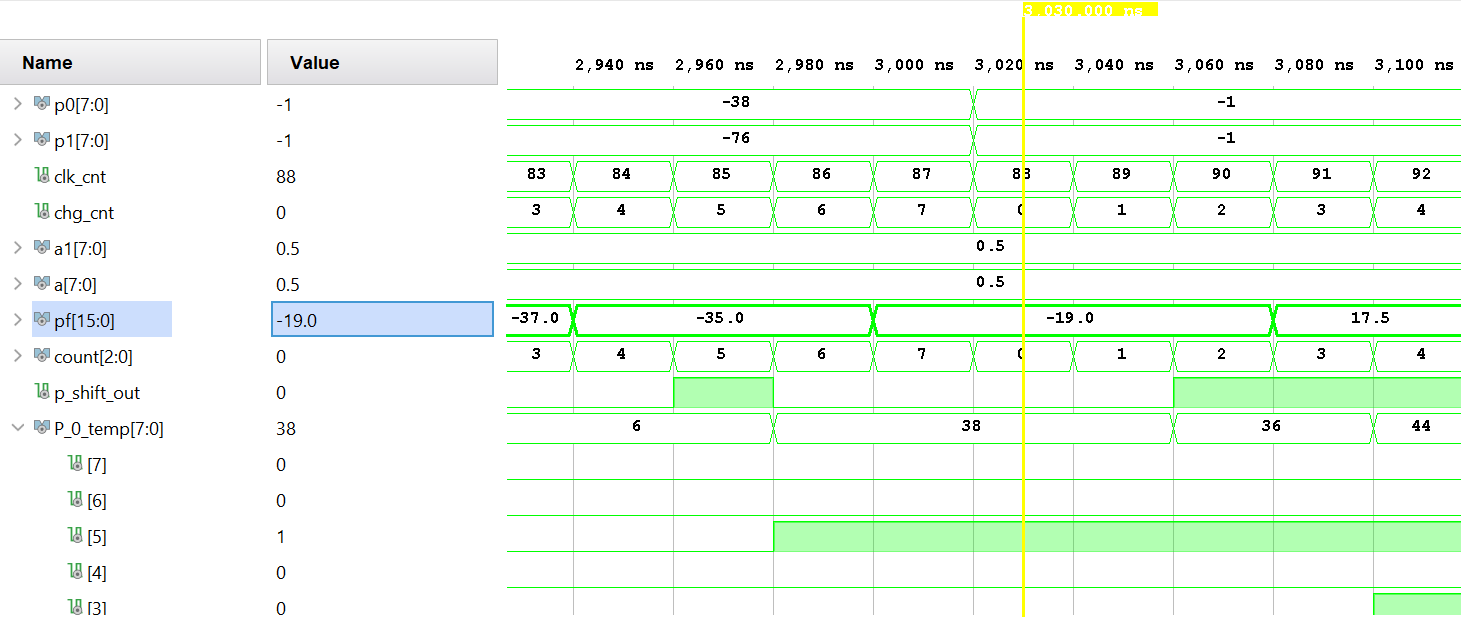


Fig 3.8: Equalized Interleaver Delay: delayed P\_f output calculated from P\_0



## Analysis:

The simulated design for this experiment successfully demonstrates the functionality of the interleaver in experimental tasks 1 and 2. Task 3 was only partially functional, but it can be concluded that the functionally of the Interleaver was preserved in this implementation.

Figure 3.1 shows the initial output of the direct implementation of the interleaver. This output represents the sum product of P\_0, P\_1, alpha and 1-alpha when alpha is 1. The output waveform, P\_f, is comprised entirely of data from P\_0 and is shown to mimic this input. As the simulation progresses, Figure 3.2 shows how the output values can be altered when alpha is changed from 1 to 0 at a linear rate. After five clock pulses, it can be seen that the output has shifted from being composed of the single magnitude sinusoid input P\_0 to the double input sinusoid P\_1. Figure 3.3 shows the output after alpha has changed being passed values from input P\_1. This outcome confirms the working condition of the direct implementation as an blending value of 0 nullifies either of the inputs while a blending value of 1 passes the inputs.

Figure 3.5, 3.6 and 3.7 show a similar conclusion in the optimized implementation of the interleaver from task 1. This implementation utilized the DSP blocks of the FPGA in its synthesized design, resulting in a more efficient use of resources. This is proved by the addition of a second clock (clk\_2) which was constructed using the Vivado clocking wizard to pulse at twice the rate of the original input clock (50 MHZ). In order to synchronize these locks and remove the maximum error possible, the clocking wizard was allotted a total of 38 clock cycles to provide enough input to synchronize. This delay can be seen in Figure 3.4 above and was accounted for in the Task 2 testbench.

Finally in Figure 3.8 and 3.9 a Equalized dealy implementation of the interleaver can be shown to have partial functionality. The original input for one of the blending factors (alpha) and the input values (P\_0) was delayed by 64 clock cycles to provide an instantaneous output. Figure 3.8 shows the first two read input values of P\_0 after the clocks were synchronized (at clk\_cnt=64). Figure 3.9 shows the delayed value of the second input of P\_0 64 clock cycles later. The output values at this time (~3020 ns) equals -19.0. Given the value of alpha, by Eq 3.1 the theoretical values of P\_0 and P\_1 at this time can be derived. These are:

= -19.0

Values delayed value of P\_0 and current value of P\_1 at 3020ns can be observed form the waveform in Figure 3.9 as signals P\_0\_temp and P\_1 respectively. P\_0 is shown to be 38, and P\_1 is shown to be -76. When these numbers are plugged into the Equation above theoretical value can be calculated

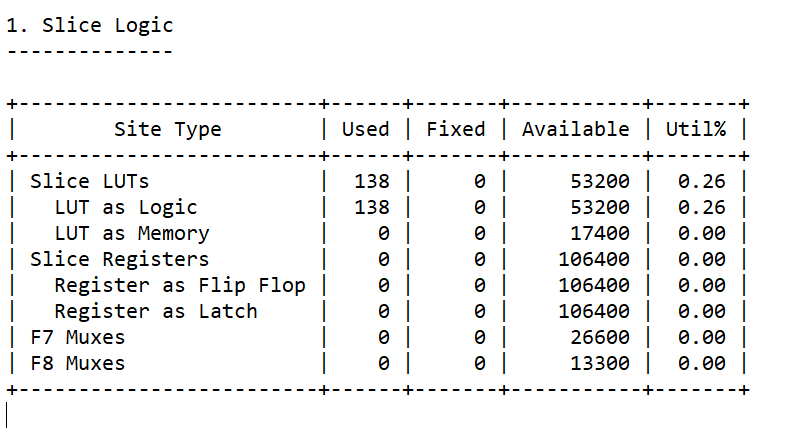
It should be noted that the testbench used for this task did not provide the adequate delays when reading in inputs from file. Further iterations of this equalized delay implementiaon are needed to provide a waveform more representative of the design’s functionality.

APPENDIX

Questions

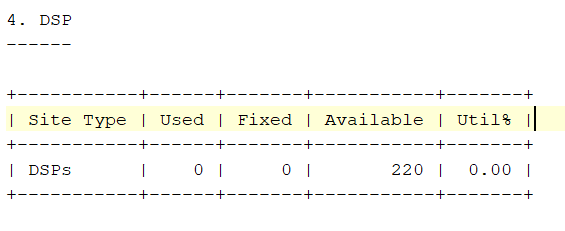
* 1) (Part I: Direct Implementation) Implement the design. What is the resource utilization of the design?

**Utilization report**



* 2) Have multipliers and adder been mapped to the DSP block? If not, can you force the tool tomap these components to the DSP blocks? How do you prove your answers?

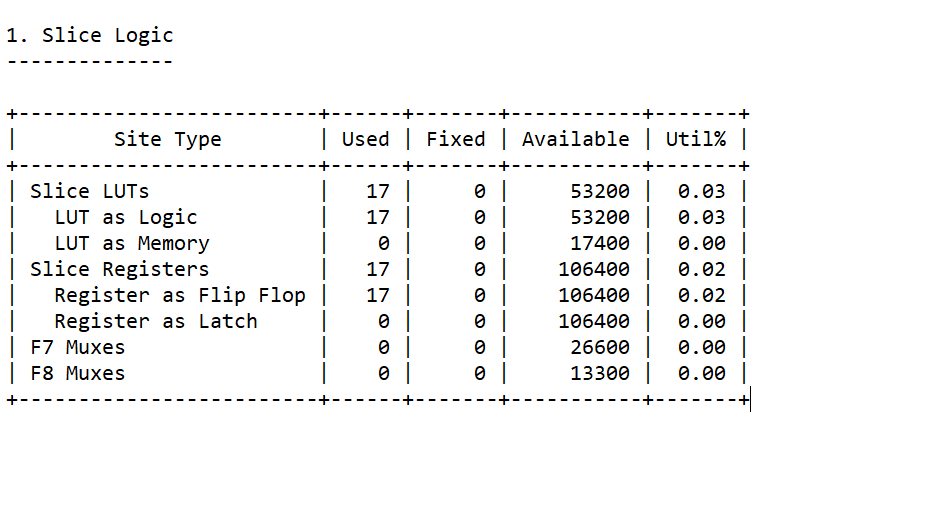
No the multiplexers are not mapped to any DSP blocks. This can be seen in the implemented design utilization report



In order to map the multiplier and adder to the DSP block this specific board component can be inferred by the synthesis tool using a predefined synthesis attribute use\_dsp in the .vhd design file for the interleaver (*lab3\_partI.vhd)*. To satisfy the computational requirements of the interleaver, 3 DSP blocks would need be used. One to calculate P0\*alpha, another to calculate P1\*(1-alpha) and a third to add these two products to generate Pf.

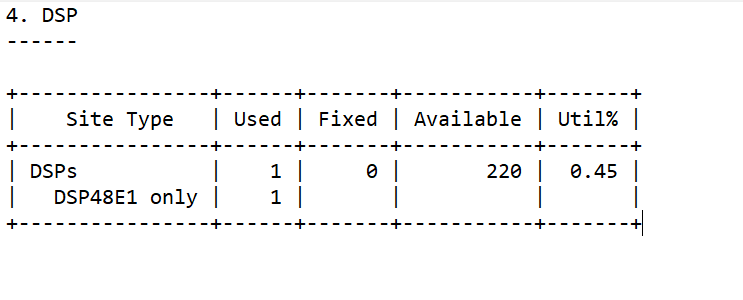
* 3) (Part II: Optimized Implementation) Implement the design. What is the resource utilization of the design?

**Utilization report**



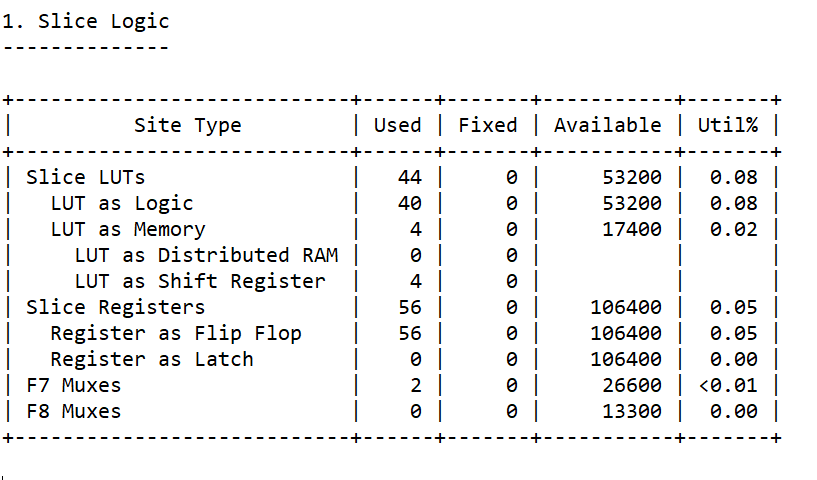
* 4) Have multipliers and adder been mapped to the DSP block? If not, can you force the tool tomap these components to the DSP blocks? How do you prove your answers?

Yes, The implemented utilization report lists one DSP48 block being used on the ZedBoard. This confers with the arithmetic operations performed in a clock cycle (1 multiplication and 1 addition).



* 5) (Part III: Equalizing Delays; SRL) Implement the design. What is the resource utilization of the design?

**Utilization report**



* 6) Can you implement the same delay using Block RAM (BRAM)? How? Explain your answer.

Yes. A BRAM implementation could be done by writing a data stream of bit values of P\_0 and alpha to any memory address. A functional delay can then be realized by reading these values form a memory address 64 indicies higher than the original written address. A dual port ram could be utilized to read and write alpha, P\_0 to different address ranges.

* 7) (Part III: Equalizing Delays; BRAM) Implement the design. What is the resource utilization of the design?

**Utilization report**

1. *Tb\_Lab3\_PartI.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/08/2020 03:14:49 PM

-- Design Name:

-- Module Name: tb\_lab3\_PartI - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**use** ieee**.**math\_real**.all;**

**use** std**.**textio**.all;**

**use** ieee**.**std\_logic\_textio**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** tb\_lab3\_PartI **is**

-- Port ( );

**end** tb\_lab3\_PartI**;**

**architecture** Behavioral **of** tb\_lab3\_PartI **is**

--Constant Definitions

**CONSTANT** CP**:** TIME **:=** 125ns**;**--clock period

**CONSTANT** ALPHA\_VAL**:** real **:=** 0.00**;**--blend value

**CONSTANT** ONE\_ALPHA\_VAL**:** real **:=** 1.00**-**ALPHA\_VAL**;** --blend value inverse

**CONSTANT** A\_LEN**:** integer **:=** 8**;** --blend value bit length

**CONSTANT** P\_LEN**:** integer **:=** 8**;** --pixel input value bit length

**CONSTANT** SHIFT\_VAL**:** real **:=** 2**\*\***real**(**A\_LEN**-**1**)-**0.01**;** --real value representing the # of bits to shift left (arithmetic shift with real inputs)

--not synthesizable, excludes sign bit shift

--Signal Definitions

**signal** p0**,**p1**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** a**,**a\_chg**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);** --amount ot change alpha by

**signal** a1**,**a1\_chg**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);** --amount to change one allpha by

**signal** pf**:** std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**SIGNAL** PULSE**:** TIME**:=**CP**\***0.5**;**--pulse width

**signal** m\_clk**:** std\_logic **:=** '0'**;** --master clock, filter reset

**signal** chg\_cnt**:** integer **:=** 0**;** --used to read in files, gradually change alpha in Part I test bench

**Component** lab3\_partI

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** **component;**

**begin**

uut**:** lab3\_partI--Signal Stream Blender

**Port** **Map(** P\_0 **=>** p0**,**

P\_1 **=>** p1**,**

alpha **=>**a**,**

one\_alpha**=>**a1**,**

P\_f **=>**pf**);**

--Clock Processs

m\_clock**:process** --free running clock

**begin**

m\_clk **<=** '1'**;**

**wait** **for** PULSE **;**

m\_clk **<=** '0'**;**

**wait** **for** PULSE**;**

**end** **process;**

--Test Vectors

**process** **(**m\_clk**)**

**variable** a\_calc**,** a1\_calc**:** real **:=**0.00**;** --used to calculate gradual shift of alpha values

**begin**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

chg\_cnt **<=** chg\_cnt **+**1**;**

**end** **if;**

**if(**chg\_cnt**<=**12**)** **then** --after 12 clock pulses (value chosen to show previous perodicity of sin wave P0)

a\_calc **:=** 0.00**;**

a1\_calc**:=**SHIFT\_VAL**;**

a\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a\_calc**),**A\_LEN**));** --alpha is 0

a1\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a1\_calc**),**A\_LEN**));** --one\_alpha is 1

**elsif(**a\_chg **<**"10000000"**)** **then**

a\_calc **:=** a\_calc **+(**SHIFT\_VAL**/**5.00**);**

a1\_calc**:=**a1\_calc **-(**SHIFT\_VAL**/**5.00**);**

a\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a\_calc**),**A\_LEN**));** --alpha is changing towards 1

a1\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a1\_calc**),**A\_LEN**));** --one\_alpha is changing towards 0

**elsif(**a\_chg **>=**"10000000"**)** **then**

a\_calc **:=** SHIFT\_VAL**;**

a1\_calc**:=**0.00**;**

a\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a\_calc**),**A\_LEN**));** --alpha is 1

a1\_chg**<=**std\_logic\_vector**(to\_unsigned(**natural**(**a1\_calc**),**A\_LEN**));** --one\_alpha is 0

**end** **if;**

a**<=** a\_chg**;**

a1**<=** a1\_chg**;**

**end** **process;**

P0\_read\_in**:process(**m\_clk**)**--reads in data for P0

**file** TVs **:** text **open** read\_mode **is** "P0\_in.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_in **:** line**;**

**variable** read\_from\_line **:** integer**;**

**variable** r\_temp **:** std\_logic\_vector**(**7 **downto** 0**);**

**begin**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**);** --read read line data from file into bit array

**read(**read\_in**,**r\_temp**);** --read read line data from file into bit array a

P0 **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**,**P\_LEN**));**

**end** **if;**

**end** **process;**

P1\_read\_in**:process(**m\_clk**)**

**file** TVs **:** text **open** read\_mode **is** "P1\_in.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_in**:** line**;**

**variable** read\_from\_line **:** integer**;**

**variable** r\_temp **:** std\_logic\_vector**(**7 **downto** 0**);**

**begin**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**);** --read read line data from file into bit array

**read(**read\_in**,**r\_temp**);** --read read line data from file into bit array a

P1 **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**,**P\_LEN**));**

**end** **if;**

**end** **process;**

Pf\_write**:process**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** Output**:**text **open** write\_mode **is** "Pf\_out.txt"**;**

**variable** write\_in**:** line**;** --reads in a line from the text file

**variable** write\_line\_to\_file**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**pf**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

**end** **process;**

**end** Behavioral**;**

1. *Lab3\_PartI.vhd*

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**std\_logic\_signed**.all;**

**entity** lab3\_partI **is**

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** lab3\_partI**;**

**architecture** Behavioral **of** lab3\_partI **is**

**signal** temp0**,**temp1**:**std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**begin**

temp0 **<=** P\_0**\***alpha**;**

temp1 **<=** P\_1**\***one\_alpha**;**

P\_f **<=** P\_0**\***alpha **+** P\_1**\***one\_alpha**;**

**end** Behavioral**;**

1. *Lab3\_PartII.vhd*

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**std\_logic\_signed**.all;**

**entity** DSP\_inter **is**

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

clk**:** **in** STD\_LOGIC**;**

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** DSP\_inter**;**

**architecture** Behavioral **of** DSP\_inter **is**

**signal** A**,**B**,**a\_temp**,**one\_a\_temp**,**P\_0\_temp**,**P\_1\_temp**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** C**,**temp0**,**temp1**:**std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** clk\_2**,**add\_m**:** std\_logic **:=** '0'**;** --clk x 2, internal adder mode

**component** clk\_x\_2x **is**

**port** **(**

clk\_2x **:** **out** STD\_LOGIC**;**

clk\_x **:** **in** STD\_LOGIC

**);**

**end** **component;**

**attribute** use\_dsp **:** string**;**

**attribute** use\_dsp **of** A**,**B**,**C **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp0 **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp1 **:** **signal** **is** "yes"**;**

**begin**

clk\_io**:** clk\_x\_2x

**Port** **Map(** clk\_2x **=>** clk\_2**,**

clk\_x **=>** clk**);**

reg\_io**:process(**clk**)**--register inputs

**begin**

**if(**clk'**event** **AND** clk**=**'1'**)** **then**

a\_temp **<=** alpha**;**

one\_a\_temp**<=**one\_alpha**;**

P\_0\_temp**<=**P\_0**;**

P\_1\_temp**<=**P\_1**;**

P\_f**<=**temp1**;**

**end** **if;**

**end** **process;**

add\_mode**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

add\_m**<=** **NOT** add\_m**;**

**end** **if;**

**end** **process;**

dsp\_proc**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

temp0**<=**A**\***B**;**

temp1**<=**temp0**+**C**;**

**end** **if;**

**end** **process;**

----muxes

A **<=** P\_0 **when** add\_m**=**'0' **else**

P\_1 **when** add\_m**=**'1' **;**

B **<=** alpha **when** add\_m**=**'0' **else**

one\_alpha **when** add\_m**=**'1' **;**

C **<=** **(Others** **=>**'0'**)** **when** add\_m**=**'0' **else**

temp1 **when** add\_m**=**'1' **;**

**end** Behavioral**;**

1. Lab3\_PartIII.vhd

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**std\_logic\_signed**.all;**

**entity** DSP\_inter **is**

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

clk**:** **in** STD\_LOGIC**;**

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** DSP\_inter**;**

**architecture** Behavioral **of** DSP\_inter **is**

**signal** A**,**B**,**a\_temp**,**one\_a\_temp**,**P\_0\_temp**,**P\_1\_temp**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** C**,**temp0**,**temp1**:**std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** clk\_2**,**add\_m**:** std\_logic **:=** '0'**;** --clk x 2, internal adder mode

**component** clk\_x\_2x **is**

**port** **(**

clk\_2x **:** **out** STD\_LOGIC**;**

clk\_x **:** **in** STD\_LOGIC

**);**

**end** **component;**

**attribute** use\_dsp **:** string**;**

**attribute** use\_dsp **of** A**,**B**,**C **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp0 **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp1 **:** **signal** **is** "yes"**;**

**begin**

clk\_io**:** clk\_x\_2x

**Port** **Map(** clk\_2x **=>** clk\_2**,**

clk\_x **=>** clk**);**

reg\_io**:process(**clk**)**--register inputs

**begin**

**if(**clk'**event** **AND** clk**=**'1'**)** **then**

a\_temp **<=** alpha**;**

one\_a\_temp**<=**one\_alpha**;**

P\_0\_temp**<=**P\_0**;**

P\_1\_temp**<=**P\_1**;**

P\_f**<=**temp1**;**

**end** **if;**

**end** **process;**

add\_mode**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

add\_m**<=** **NOT** add\_m**;**

**end** **if;**

**end** **process;**

dsp\_proc**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

temp0**<=**A**\***B**;**

temp1**<=**temp0**+**C**;**

**end** **if;**

**end** **process;**

----muxes

A **<=** P\_0 **when** add\_m**=**'0' **else**

P\_1 **when** add\_m**=**'1' **;**

B **<=** alpha **when** add\_m**=**'0' **else**

one\_alpha **when** add\_m**=**'1' **;**

C **<=** **(Others** **=>**'0'**)** **when** add\_m**=**'0' **else**

temp1 **when** add\_m**=**'1' **;**

**end** Behavioral**;**

**A.4**  *Lab3\_partIII.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/08/2020 02:45:13 PM

-- Design Name:

-- Module Name: lab3\_partI - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**std\_logic\_signed**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** DSP\_inter **is**

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

clk**:** **in** STD\_LOGIC**;**

rst**:** **in** STD\_LOGIC**;**

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** DSP\_inter**;**

**architecture** Behavioral **of** DSP\_inter **is**

**signal** A**,**B**,**a\_temp**,**one\_a\_temp**,**P\_0\_temp**,**P\_1\_temp**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** C**,**temp0**,**temp1**:**std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** clk\_2**,**add\_m**:** std\_logic **:=** '0'**;** --clk x 2, internal adder mode

--64-bit shift reg components

**signal** a\_shift\_in**,**a\_shift\_out**,**p\_shift\_in**,**p\_shift\_out**:** STD\_LOGIC **:=**'0'**;**

**signal** clk\_en**:** STD\_LOGIC **:=**'1'**;**

**signal** count**:** std\_logic\_vector**(**2 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**CONSTANT** depth**:** integer **:=** 64**;**

**component** clk\_x\_2x **is** --clocking wizard

**port** **(**

clk\_2x **:** **out** STD\_LOGIC**;**

clk\_x **:** **in** STD\_LOGIC

**);**

**end** **component;**

**component** shift\_registers\_1 **is** --Xilnix Synthesis Guide: Inferred 64 Bit SR

**generic(**

DEPTH **:** integer **:=** 32

**);**

**port(**

clk **:** **in** std\_logic**;**

clken **:** **in** std\_logic**;**

SI **:** **in** std\_logic**;**

SO **:** **out** std\_logic

**);**

**end** **component;**

**attribute** use\_dsp **:** string**;**

**attribute** use\_dsp **of** A**,**B**,**C **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp0 **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp1 **:** **signal** **is** "yes"**;**

**begin**

clk\_io**:** clk\_x\_2x

**Port** **Map(** clk\_2x **=>** clk\_2**,**

clk\_x **=>** clk**);**

SR\_64\_P**:**shift\_registers\_1

**Generic** **Map(**

DEPTH **=>** depth**)**

**Port** **Map(** clk **=>** clk**,**

clken**=>** clk\_en**,**

SI **=>**p\_shift\_in**,**

SO**=>**p\_shift\_out**);**

SR\_64\_alpha**:**shift\_registers\_1

**Generic** **Map(**

DEPTH **=>** depth**)**

**Port** **Map(** clk **=>** clk**,**

clken**=>** clk\_en**,**

SI **=>**a\_shift\_in**,**

SO**=>**a\_shift\_out**);**

alpha\_shift\_io**:** **process(**clk**,**rst**)**

**begin**

**if(**rst**=**'0'**)** **then**

**if(**clk'**event** **AND** clk**=**'1'**)** **then**

**case** count **is**

**when** "111" **=>**

a\_shift\_in **<=** alpha**(**0**);**

p\_shift\_in **<=** P\_0**(**0**);**

a\_temp**(**7**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**7**)<=** p\_shift\_out**;**

**when** "000" **=>**

a\_shift\_in **<=** alpha**(**1**);**

p\_shift\_in **<=** P\_0**(**1**);**

a\_temp**(**0**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**0**)<=** p\_shift\_out**;**

**when** "001" **=>**

a\_shift\_in **<=** alpha**(**2**);**

p\_shift\_in **<=** P\_0**(**2**);**

a\_temp**(**1**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**1**)<=** p\_shift\_out**;**

**when** "010" **=>**

a\_shift\_in **<=** alpha**(**3**);**

p\_shift\_in **<=** P\_0**(**3**);**

a\_temp**(**2**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**2**)<=** p\_shift\_out**;**

**when** "011" **=>**

a\_shift\_in **<=** alpha**(**4**);**

p\_shift\_in **<=** P\_0**(**4**);**

a\_temp**(**3**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**3**)<=** p\_shift\_out**;**

**when** "100" **=>**

a\_shift\_in **<=** alpha**(**5**);**

p\_shift\_in **<=** P\_0**(**5**);**

a\_temp**(**4**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**4**)<=** p\_shift\_out**;**

**when** "101" **=>**

a\_shift\_in **<=** alpha**(**6**);**

p\_shift\_in **<=** P\_0**(**6**);**

a\_temp**(**5**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**5**)<=** p\_shift\_out**;**

**when** "110" **=>**

a\_shift\_in **<=** alpha**(**7**);**

p\_shift\_in **<=** P\_0**(**7**);**

a\_temp**(**6**)** **<=** a\_shift\_out**;**

P\_0\_temp**(**6**)<=** p\_shift\_out**;**

**when** **others** **=>**

a\_shift\_in**<=**'0'**;**

p\_shift\_in**<=**'0'**;**

**end** **case;**

count **<=** count **+**"001"**;**

**end** **if;**

**end** **if;**

**end** **process;**

reg\_io**:process(**clk**)**--register inputs

**begin**

**if(**clk'**event** **AND** clk**=**'1'**)** **then**

one\_a\_temp**<=**one\_alpha**;**

P\_1\_temp**<=**P\_1**;**

P\_f**<=**temp1**;**

**end** **if;**

**end** **process;**

add\_mode**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

add\_m**<=** **NOT** add\_m**;**

**end** **if;**

**end** **process;**

dsp\_proc**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

temp0**<=**A**\***B**;**

temp1**<=**temp0**+**C**;**

**end** **if;**

**end** **process;**

----muxes

A **<=** P\_0\_temp **when** add\_m**=**'0' **else**

P\_1\_temp **when** add\_m**=**'1' **;**

B **<=** a\_temp **when** add\_m**=**'0' **else**

one\_a\_temp **when** add\_m**=**'1' **;**

C **<=** **(Others** **=>**'0'**)** **when** add\_m**=**'0' **else**

temp1 **when** add\_m**=**'1' **;**

**end** Behavioral**;**

**A.5**  tb\_*Lab3\_partIII.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/08/2020 03:14:49 PM

-- Design Name:

-- Module Name: tb\_lab3\_PartI - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**use** ieee**.**math\_real**.all;**

**use** std**.**textio**.all;**

**use** ieee**.**std\_logic\_textio**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** tb\_DSP\_inter **is**

-- Port ( );

**end** tb\_DSP\_inter**;**

**architecture** Behavioral **of** tb\_DSP\_inter **is**

--Constant Definitions

**CONSTANT** CP**:** TIME **:=** 20ns**;**--50 MHz clock period

**CONSTANT** ALPHA\_VAL**:** real **:=** 0.00**;**--blend value

**CONSTANT** ONE\_ALPHA\_VAL**:** real **:=** 1.00**-**ALPHA\_VAL**;** --blend value inverse

**CONSTANT** A\_LEN**:** integer **:=** 8**;** --blend value bit length

**CONSTANT** P\_LEN**:** integer **:=** 8**;** --pixel input value bit length

**CONSTANT** SHIFT\_VAL**:** real **:=** 2**\*\***real**(**A\_LEN**-**1**)-**0.01**;** --real value representing the # of bits to shift left (arithmetic shift with real inputs)

--not synthesizable, excludes sign bit shift

--Signal Definitions

**signal** p0**,**p1**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** a**,**a\_chg**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);** --amount ot change alpha by

**signal** a1**,**a1\_chg**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);** --amount to change one allpha by

**signal** pf**:** std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**SIGNAL** PULSE**:** TIME**:=**CP**\***0.5**;**--pulse width

**signal** m\_clk**:** std\_logic **:=** '0'**;** --master clock

**signal** reset**:** std\_logic **:=** '1'**;** -- reset

**signal** clk\_cnt**,**chg\_cnt**:** integer **:=** 0**;** --used to read in files, gradually change alpha in Part I test bench

--Counts clk changes; used in Part III test bench

**signal** w**:** STD\_LOGIC**:=** '0'**;**

**Component** DSP\_inter

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

clk**:** **in** STD\_LOGIC**;**

rst**:** **in** STD\_LOGIC**;**

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** **component;**

**begin**

uut**:** DSP\_inter--Signal Stream Blender

**Port** **Map(** P\_0 **=>** p0**,**

P\_1 **=>** p1**,**

alpha **=>**a**,**

one\_alpha**=>**a1**,**

clk **=>** m\_clk**,**

rst **=>** reset**,**

P\_f **=>**pf**);**

--Clock Processs

m\_clock**:process** --free running clock

**begin**

m\_clk **<=** '1'**;**

**wait** **for** PULSE **;**

m\_clk **<=** '0'**;**

**wait** **for** PULSE**;**

**end** **process;**

r**:process** --reset for clock sync (~760ns clock delay @ 50 MHZ/ 100 MHz sync)

**begin**

**wait** **for** CP**\***64**;**

reset**<=** '0'**;**

**wait;**

**end** **process;**

clk\_count**:** **process(**m\_clk**,**reset**)**

**begin**

**if(**reset**=**'0'**)** **then**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

clk\_cnt**<=** clk\_cnt**+**1**;**

chg\_cnt**<=**chg\_cnt**+**1**;**

**if(**chg\_cnt**=**7**)** **then**

chg\_cnt**<=**0**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

--Test Vectors

**process**

**begin**

a1**<=**"01000000"**;**

a**<=**"01000000"**;**

**wait;**

**end** **process;**

P0\_read\_in**:process(**reset**,**chg\_cnt**,**m\_clk**)**--reads in data for P0

**file** TVs **:** text **open** read\_mode **is** "P0\_in.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_in **:** line**;**

**variable** read\_from\_line **:** integer**;**

**variable** r\_temp **:** std\_logic\_vector**(**7 **downto** 0**);**

**begin**

**if(**reset**=**'0'**)** **then**

**if(**chg\_cnt**=**7**)** **then**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**);** --read read line data from file into bit array

**read(**read\_in**,**r\_temp**);** --read read line data from file into bit array a

P0 **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**,**P\_LEN**));**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

P1\_read\_in**:process(**reset**,**chg\_cnt**,**m\_clk**)**

**file** TVs **:** text **open** read\_mode **is** "P1\_in.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_in**:** line**;**

**variable** read\_from\_line **:** integer**;**

**variable** r\_temp **:** std\_logic\_vector**(**7 **downto** 0**);**

**begin**

**if(**reset**=**'0'**)** **then**

**if(**chg\_cnt**=**7**)** **then**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**);** --read read line data from file into bit array

**read(**read\_in**,**r\_temp**);** --read read line data from file into bit array a

P1 **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**,**P\_LEN**));**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Pf\_write**:process** **(**reset**,**m\_clk**)**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** Output**:**text **open** write\_mode **is** "Pf\_out.txt"**;**

**variable** write\_in**:** line**;** --reads in a line from the text file

**variable** write\_line\_to\_file**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'0'**)** **then**

**if(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**pf**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**end** **if;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

**A.6**  *Shift\_Registers\_1.vhd*

-- 32-bit Shift Register

-- Rising edge clock

-- Active high clock enable

-- foor loop-based template

-- File: shift\_registers\_1.vhd

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** shift\_registers\_1 **is**

**generic(**

DEPTH **:** integer **:=** 32

**);**

**port(**

clk **:** **in** std\_logic**;**

clken **:** **in** std\_logic**;**

SI **:** **in** std\_logic**;**

SO **:** **out** std\_logic

**);**

**end** shift\_registers\_1**;**

**architecture** archi **of** shift\_registers\_1 **is**

**signal** shreg **:** std\_logic\_vector**(**DEPTH **-** 1 **downto** 0**);**

**begin**

**process(**clk**)**

**begin**

**if** **rising\_edge(**clk**)** **then**

**if** clken **=** '1' **then**

**for** i **in** 0 **to** DEPTH **-** 2 **loop**

shreg**(**i **+** 1**)** **<=** shreg**(**i**);**

**end** **loop;**

shreg**(**0**)** **<=** SI**;**

**end** **if;**

**end** **if;**

**end** **process;**

SO **<=** shreg**(**DEPTH **-** 1**);**

**end** archi**;**

**A.7 BRAM IMPLEMENTAION.VHD**

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 10/08/2020 02:45:13 PM

-- Design Name:

-- Module Name: lab3\_partI - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**std\_logic\_signed**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** DSP\_inter **is**

**Port(**P\_0**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --first pixel data stream input

P\_1**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --second pixel data stream input

alpha**:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** --blending factor

one\_alpha**:** **in** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);**-- inverse blending factor

clk**:** **in** STD\_LOGIC**;**

rst**:** **in** STD\_LOGIC**;**

P\_f**:** **out** STD\_LOGIC\_VECTOR**(**15 **downto** 0**));**--Output equal to :: P\_0\*alpha + P\_1\*one\_alpha

**end** DSP\_inter**;**

**architecture** Behavioral **of** DSP\_inter **is**

**signal** A**,**B**,**a\_temp**,**one\_a\_temp**,**P\_0\_temp**,**P\_1\_temp**:** std\_logic\_vector**(**7 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** C**,**temp0**,**temp1**:**std\_logic\_vector**(**15 **downto** 0**)** **:=** **(Others** **=>**'0'**);**

**signal** clk\_2**,**add\_m**:** std\_logic **:=** '0'**;** --clk x 2, internal adder mode

--64-bit shift reg components

**signal** a\_shift\_in**,**a\_shift\_out**,**p\_shift\_in**,**p\_shift\_out**:** STD\_LOGIC **:=**'0'**;**

**signal** en**:** STD\_LOGIC **:=**'1'**;**

**signal** count**:** std\_logic\_vector**(**6 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**CONSTANT** depth**:** integer **:=** 64**;**

**component** clk\_x\_2x **is** --clocking wizard

**port** **(**

clk\_2x **:** **out** STD\_LOGIC**;**

clk\_x **:** **in** STD\_LOGIC

**);**

**end** **component;**

**component** blk\_mem\_gen\_0 **is**

**port** **(**

doutb **:** **out** STD\_LOGIC\_VECTOR **(** 7 **downto** 0 **);**

clkb **:** **in** STD\_LOGIC**;**

clka **:** **in** STD\_LOGIC**;**

enb **:** **in** STD\_LOGIC**;**

ena **:** **in** STD\_LOGIC**;**

addrb **:** **in** STD\_LOGIC\_VECTOR **(** 5 **downto** 0 **);**

addra **:** **in** STD\_LOGIC\_VECTOR **(** 5 **downto** 0 **);**

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

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

wea **:** **in** STD\_LOGIC\_VECTOR **(** 0 **to** 0 **)**

**);**

**end** **component;**

**attribute** use\_dsp **:** string**;**

**attribute** use\_dsp **of** A**,**B**,**C **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp0 **:** **signal** **is** "yes"**;**

**attribute** use\_dsp **of** temp1 **:** **signal** **is** "yes"**;**

**begin**

clk\_io**:** clk\_x\_2x

**Port** **Map(** clk\_2x **=>** clk\_2**,**

clk\_x **=>** clk**);**

bram**:**blk\_mem\_gen\_0

**Port** **Map(**

douta **=>**a\_temp**;**

doutb **=>**P\_0\_temp**;**

clkb **=>** clk**;**

clka **=>** clk

enb **=>** en**;**

ena**=>**en**;**

addrb **=>** count**;**

addra **=>** count**;**

dina **:** alpha

dinb**:** P\_0**;**

wea **:** 2**);**

reg\_io**:process(**clk**)**--register inputs

**begin**

**if(**clk'**event** **AND** clk**=**'1'**)** **then**

one\_a\_temp**<=**one\_alpha**;**

P\_1\_temp**<=**P\_1**;**

P\_f**<=**temp1**;**

**end** **if;**

**end** **process;**

add\_mode**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

add\_m**<=** **NOT** add\_m**;**

**end** **if;**

**end** **process;**

dsp\_proc**:** **process(**clk\_2**)**

**begin**

**if(**clk\_2'**event** **AND** clk\_2**=**'1'**)** **then**

temp0**<=**A**\***B**;**

temp1**<=**temp0**+**C**;**

**end** **if;**

**end** **process;**

----muxes

A **<=** P\_0\_temp **when** add\_m**=**'0' **else**

P\_1\_temp **when** add\_m**=**'1' **;**

B **<=** a\_temp **when** add\_m**=**'0' **else**

one\_a\_temp **when** add\_m**=**'1' **;**

C **<=** **(Others** **=>**'0'**)** **when** add\_m**=**'0' **else**

temp1 **when** add\_m**=**'1' **;**

**end** Behavioral**;**