Computer Architectures

Lab Report 2

Datapath Design

Y3839090 & Y3840426| Computer Architectures | ELE00009I

# **ALL DataPaths**

**ALUParam.vhd (From Lab 1)(Used for All DataPaths)**

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

--

-- Uni : University of York

-- Course : Electronic Engineering

-- Module : Computer Architectures

-- Engineers : Y3839090 & Y3840426

--

-- Create Date : 13:19:20 02/17/2017

-- Design Name : ALU\_param - Behavioral

-- Description : A paramateriable integer ALU.

--

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

**library** IEEE**;**

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

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.ALL;**

**entity** ALU\_param **is**

**Generic** **(**

N **:** natural **:=** 8 -- data size in bits

**);**

**Port** **(**

A **:** **in** STD\_LOGIC\_VECTOR **(**N**-**1 **downto** 0**);**

B **:** **in** STD\_LOGIC\_VECTOR **(**N**-**1 **downto** 0**);**

X **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**N**)-**1 **downto** 0**);** -- shift/rotate amount input

ctrl **:** **in** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);** -- control signals from opcode

O **:** **out** STD\_LOGIC\_VECTOR **(**N**-**1 **downto** 0**);**

flags **:out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**)** -- flags

**);**

**end** ALU\_param**;**

**architecture** Behavioral **of** ALU\_param **is**

-- internal signed signal for A and B inpy=uts

**signal** A\_itrn **:** SIGNED **(**N**-**1 **downto** 0**);**

**signal** B\_itrn **:** SIGNED **(**N**-**1 **downto** 0**);**

-- internal integer for X

**signal** X\_itrn **:** integer**;**

-- internal signed signal for output

**signal** O\_itrn **:** SIGNED **(**N**-**1 **downto** 0**);**

-- max positive and negitive N bit signed numbers

**constant** max\_pos **:** SIGNED **(**N**-**1 **downto** 0**)** **:=** **to\_signed((** 2 **\*\*** **(**N**-**1**)** **)** **-** 1**,** N**);**

**constant** max\_neg **:** SIGNED **(**N**-**1 **downto** 0**)** **:=** **to\_signed(** **-**2 **\*\*** **(**N**-**1**)** **,** N**);**

**begin**

A\_itrn **<=** signed**(**A**);** -- converts A to signed and maps the result to A\_itrn

B\_itrn **<=** signed**(**B**);** -- converts A to signed and maps the result to B\_itrn

-- converts X to integer and maps the result to X\_itrn

X\_itrn **<=** **to\_integer(**unsigned**(**X**));**

-- converts O\_itrn to a plain std\_logic\_vector and maps it to O

O **<=** std\_logic\_vector**(**O\_itrn**);**

-- Main ALU multiplexer for each possible command

O\_itrn **<=**

A\_itrn **when** ctrl **=** "0000" **else** -- Output A

A\_itrn and B\_itrn **when** ctrl **=** "0100" **else** -- Output A & B

A\_itrn or B\_itrn **when** ctrl **=** "0101" **else** -- Output A || B

A\_itrn xor B\_itrn **when** ctrl **=** "0110" **else** -- Output A xor B

not A\_itrn **when** ctrl **=** "0111" **else** -- Output not A

A\_itrn **+** 1 **when** ctrl **=** "1000" **else** -- Output A + 1

A\_itrn **-** 1 **when** ctrl **=** "1001" **else** -- Output A - 1

A\_itrn **+** B\_itrn **when** ctrl **=** "1010" **else** -- Output A + B

A\_itrn **-** B\_itrn **when** ctrl **=** "1011" **else** -- Output A - B

**SHIFT\_LEFT** **(**A\_itrn **,** X\_itrn**)** **when** ctrl **=** "1100" **else** -- Output A sla X

**SHIFT\_RIGHT** **(**A\_itrn **,** X\_itrn**)** **when** ctrl **=** "1101" **else** -- Output A sra X

**ROTATE\_LEFT** **(**A\_itrn **,** X\_itrn**)** **when** ctrl **=** "1110" **else** -- Output A rotl X

**ROTATE\_RIGHT** **(**A\_itrn **,** X\_itrn**)** **when** ctrl **=** "1111" **else** -- Output A rotr X

**(others** **=>**'U'**);**

**(Continued.)**

-- Overflow flag

flags**(**7**)** **<=**

-- Will overflow if you add one to the max positive value

'1' **when** ctrl **=** "1000" and A\_itrn **=** max\_pos **else**

-- Will overflow if you minus one to the max negitive value

'1' **when** ctrl **=** "1001" and A\_itrn **=** max\_neg **else**

-- Will overflow if two neg values added give a pos result

'1' **when** ctrl **=** "1010" and A\_itrn**(**N**-**1**)** **=** '1' and B\_itrn**(**N**-**1**)** **=** '1' and O\_itrn**(**N**-**1**)** **=** '0' **else**

-- Will overflow if two pos values added give a neg result

'1' **when** ctrl **=** "1010" and A\_itrn**(**N**-**1**)** **=** '0' and B\_itrn**(**N**-**1**)** **=** '0' and O\_itrn**(**N**-**1**)** **=** '1' **else**

-- Will overflow if a pos value is subtracted from a neg value gives a pos result

'1' **when** ctrl **=** "1011" and A\_itrn**(**N**-**1**)** **=** '1' and B\_itrn**(**N**-**1**)** **=** '0' and O\_itrn**(**N**-**1**)** **=** '0' **else**

-- Will overflow if a neg value is subtracted from a pos value gives a neg result

'1' **when** ctrl **=** "1011" and A\_itrn**(**N**-**1**)** **=** '0' and B\_itrn**(**N**-**1**)** **=** '1' and O\_itrn**(**N**-**1**)** **=** '1'

-- If none of the above are true then the result hasn't overflown

**else** '0'**;**

-- Other flags

flags**(**6**)** **<=** '1' **when** O\_itrn **>=** 0 **else** '0'**;** -- grater than or equal to zero

flags**(**5**)** **<=** '1' **when** O\_itrn **<=** 0 **else** '0'**;** -- less than or equal to zero

flags**(**4**)** **<=** '1' **when** O\_itrn **>** 0 **else** '0'**;** -- grater than zero

flags**(**3**)** **<=** '1' **when** O\_itrn **<** 0 **else** '0'**;** -- less than zero

flags**(**2**)** **<=** '1' **when** O\_itrn **=** 1 **else** '0'**;** -- one flag

flags**(**1**)** **<=** '1' **when** O\_itrn **/=** 0 **else** '0'**;** -- not zero flag

flags**(**0**)** **<=** '1' **when** O\_itrn **=** 0 **else** '0'**;** -- zero flag

**end** Behavioral**;**

**EasyPrint.vhd (Used in testbenches)**

*A custom package that adds functions that make it easy to print std\_logic\_vectors in report statements. Two of these function are from stack overflow comments links to the origanl source of the functions are provided.*

**library** IEEE**;**

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

**USE** ieee**.**numeric\_std**.ALL;**

**package** easyprint **is**

**function** s\_tostr**(val** **:** std\_logic\_vector**)** **return** string**;**

**function** u\_tostr**(val** **:** std\_logic\_vector**)** **return** string**;**

**function** to\_bstring**(**sl **:** std\_logic**)** **return** string**;**

**function** to\_bstring**(**slv **:** std\_logic\_vector**)** **return** string**;**

**end** easyprint**;**

**package** **body** easyprint **is**

-- ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

-- From http://stackoverflow.com/a/24336034 By Morten Zilmer

-- Allows printing a std\_logic\_vector as a string that represents it's binary form.

-- ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

**function** to\_bstring**(**sl **:** std\_logic**)** **return** string **is**

**variable** sl\_str\_v **:** string**(**1 **to** 3**);** -- std\_logic image with quotes around

**begin**

sl\_str\_v **:=** std\_logic'**image(**sl**);**

**return** "" **&** sl\_str\_v**(**2**);** -- "" & character to get string

**end** **function;**

**function** to\_bstring**(**slv **:** std\_logic\_vector**)** **return** string **is**

**alias** slv\_norm **:** std\_logic\_vector**(**1 **to** slv'**length)** **is** slv**;**

**variable** sl\_str\_v **:** string**(**1 **to** 1**);** -- String of std\_logic

**variable** res\_v **:** string**(**1 **to** slv'**length);**

**begin**

**for** idx **in** slv\_norm'**range** **loop**

sl\_str\_v **:=** to\_bstring**(**slv\_norm**(**idx**));**

res\_v**(**idx**)** **:=** sl\_str\_v**(**1**);**

end loop;

return res\_v;

end function;

-- ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

-- converts an std\_logic\_vector to a string that represents it's signed value

function s\_tostr(val : std\_logic\_vector) return string is

begin

return integer'image( to\_integer(signed(val)) );

end function;

-- converts an std\_logic\_vector to a string that represents it's unsigned value

function u\_tostr(val : std\_logic\_vector) return string is

begin

return integer'image( to\_integer(unsigned(val)) );

end function;

end easyprint;

**(Continued.)**

res\_v**(**idx**)** **:=** sl\_str\_v**(**1**);**

**end** **loop;**

**return** res\_v**;**

**end** **function;**

-- ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ==== ====

-- converts an std\_logic\_vector to a string that represents it's signed value

**function** s\_tostr**(val** **:** std\_logic\_vector**)** **return** string **is**

**begin**

**return** integer'**image(** **to\_integer(**signed**(val))** **);**

**end** **function;**

-- converts an std\_logic\_vector to a string that represents it's unsigned value

**function** u\_tostr**(val** **:** std\_logic\_vector**)** **return** string **is**

**begin**

**return** integer'**image(** **to\_integer(**unsigned**(val))** **);**

**end** **function;**

**end** easyprint**;**

**Reg.vhd (Used for all Datapaths)**

**library** IEEE**;**

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

-- A paramateriable synchronous reset D-type registers with enable

**entity** Reg **is**

**Generic(**

data\_size**:** natural **:=** 8 -- How many bits will the register deal with

**);**

**Port** **(**

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

rst **:** **in** STD\_LOGIC**;** -- synchronus reset

en **:** **in** STD\_LOGIC**;** -- synchronus reset

data\_in **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- input

data\_out **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**)** -- output

**);**

**end** Reg**;**

**architecture** Behavioral **of** Reg **is**

**begin**

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

**begin**

-- Synchronise to the clock

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

**if** **(**rst **=** '1'**)** **then**

-- Reset to zero

data\_out **<=** **(others** **=>** '0'**);**

**elsif** **(**en **=** '1'**)** **then**

-- Pass the input to the output

data\_out **<=** data\_in**;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

**Architecture B**

**DataPathB.vhd**

**library** IEEE**;**

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

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.all;**

**entity** DataPath\_B **is**

**GENERIC(**

data\_size **:** natural **:=** 16**;**

num\_registers **:** natural **:=** 32

**);**

**Port** **(**

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

R\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address A

R\_B **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address B

W\_EN **:** **in** STD\_LOGIC**;** -- Register write enable

W\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Write address

IMM **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Itermediate value

AL **:** **in** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);** -- ALU control

SH **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**data\_size**)-**1 **downto** 0**);** -- Shift amount

M\_A **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

S **:** **in** STD\_LOGIC\_VECTOR **(**4 **downto** 1**);** -- Selector control

flags **:** **out** STD\_LOGIC\_VECTOR**(**7 **downto** 0**);** -- ALU flags

M\_B **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory output (write)

M\_DA **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

M\_in **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**)** -- Memory input (read)

**);**

**end** DataPath\_B**;**

**architecture** Behavioral **of** DataPath\_B **is**

**signal** reg\_in **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- register write data

**signal** A\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Data from register at address R\_A

**signal** B\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Data from register at address R\_B

**signal** B\_mux **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- B input to the ALU

**signal** ALU\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Output of the ALU

**begin**

-- Multiplexer on the ALU's B input

B\_mux **<=** B\_data **when** S**(**1**)** **=** '0' **else** IMM**;**

-- Multiplexer for the memory address

M\_DA **<=** M\_A **when** s**(**3**)** **=** '1' **else** ALU\_out**;**

-- Multiplexer for the register write data

reg\_in **<=** ALU\_out **when** s**(**4**)** **=** '0' **else** M\_in**;**

-- Memory write (output) connection

M\_B **<=** B\_data**;**

-- The ALU

ALU**:** **entity** work**.**ALU\_param

**GENERIC** **MAP(**

N **=>** data\_size

**)**

**PORT** **MAP(**

A **=>** A\_data**,**

B **=>** B\_mux**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

rst **=>** '0'

**);**

**end** Behavioral**;**

**(Continued.)**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

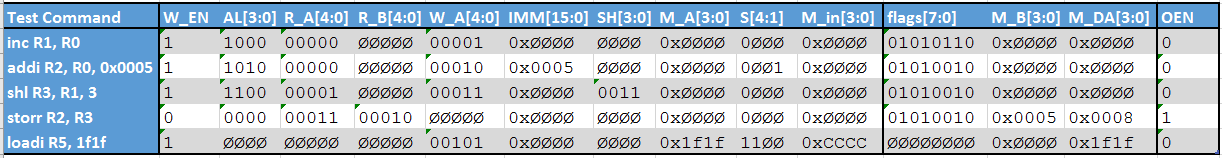
rst **=>** '0'

**);**

**end** Behavioral**;**

**Test Data for DataPath B**

Here is the control signals and expected outputs that we are using to test datapath B.



**DataPathB-TB.vhd**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** work**.**DigEng**.ALL;**

**USE** work**.**easyprint**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**ENTITY** DataPathB\_TB **IS**

**END** DataPathB\_TB**;**

**ARCHITECTURE** behavior **OF** DataPathB\_TB **IS**

-- Constants

**constant** data\_size **:** NATURAL **:=** 16**;**

**constant** num\_registers **:** NATURAL **:=** 32**;**

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

**COMPONENT** DataPath\_B

**GENERIC(**

data\_size **:** natural**;**

num\_registers **:** natural

**);**

**PORT(**

R\_A **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_EN **:** **IN** std\_logic**;**

W\_A **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

clk **:** **IN** std\_logic**;**

IMM **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

AL **:** **IN** std\_logic\_vector**(**3 **downto** 0**);**

SH **:** **IN** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**);**

M\_A **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** **IN** std\_logic\_vector**(**4 **downto** 1**);**

flags **:** **OUT** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** **OUT** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** **OUT** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_in **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**)**

**);**

**END** **COMPONENT;**

--Inputs

**signal** R\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** R\_B **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** W\_EN **:** std\_logic **:=** '0'**;**

**signal** W\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** clk **:** std\_logic **:=** '0'**;**

**signal** IMM **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** AL **:** std\_logic\_vector **(**3 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** SH **:** std\_logic\_vector **(**log2**(**data\_size**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** M\_A **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** S **:** std\_logic\_vector **(**4 **downto** 1**)** **:=** **(others** **=>** '0'**);**

**signal** M\_in **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

--Outputs

**signal** flags **:** std\_logic\_vector **(**7 **downto** 0**);**

**signal** M\_B **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** M\_DA **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** OEN **:** std\_logic **:=** '0'**;**

-- Clock period definitions

**constant** clk\_period **:** time **:=** 10 ns**;**

**constant** wait\_time **:** time **:=** clk\_period**;**

-- Test data for self checking test bench

**type** TEST\_VECTOR **is** **RECORD**

W\_EN **:** std\_logic**;**

AL **:** std\_logic\_vector**(**3 **downto** 0**);**

R\_A **:** STD\_LOGIC\_VECTOR**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_A **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

--W\_EN, AL, R\_A, R\_B, W\_A, IMM, SH, M\_A, S, M\_in, flags, M\_B, M\_DA, OEN

**(** '1'**,** "1000"**,** "00000"**,** "-----"**,** "00001"**,** "----------------"**,** "----"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010110"**,** "----------------"**,** "----------------"**,** '0'**),**

**(** '1'**,** "1010"**,** "00000"**,** "-----"**,** "00010"**,** X"0005"**,** "----"**,** "----------------"**,** "0--1"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0'**),**

**(** '1'**,** "1100"**,** "00001"**,** "-----"**,** "00011"**,** "----------------"**,** "0011"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0'**),**

**(** '0'**,** "0000"**,** "00011"**,** "00010"**,** "-----"**,** "----------------"**,** "----"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** X"0005"**,** X"0008"**,** '1'**),**

**(** '1'**,** "----"**,** "-----"**,** "-----"**,** "00101"**,** "----------------"**,** "----"**,** X"1f1f"**,** "11--"**,** X"CCCC"**,** "--------"**,** "----------------"**,** X"1f1f"**,** '0'**)**

**);**

**(Continued.)**

--Outputs

**signal** flags **:** std\_logic\_vector **(**7 **downto** 0**);**

**signal** M\_B **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** M\_DA **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** OEN **:** std\_logic **:=** '0'**;**

-- Clock period definitions

**constant** clk\_period **:** time **:=** 10 ns**;**

**constant** wait\_time **:** time **:=** clk\_period**;**

-- Test data for self checking test bench

**type** TEST\_VECTOR **is** **RECORD**

W\_EN **:** std\_logic**;**

AL **:** std\_logic\_vector**(**3 **downto** 0**);**

R\_A **:** STD\_LOGIC\_VECTOR**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_A **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

--W\_EN, AL, R\_A, R\_B, W\_A, IMM, SH, M\_A, S, M\_in, flags, M\_B, M\_DA, OEN

**(** '1'**,** "1000"**,** "00000"**,** "-----"**,** "00001"**,** "----------------"**,**

"----"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010110"**,** "----------------"**,** "----------------"**,** '0'**),**

**(**'1'**,** "1010"**,** "00000"**,** "-----"**,** "00010"**,** X"0005"**,** "----"**,** "----------------"**,** "0--1"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0'**),**

**(**'1'**,** "1100"**,** "00001"**,** "-----"**,** "00011"**,** "----------------"**,**

"0011"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0'**),**

**(**'0'**,** "0000"**,**  "00011"**,** "00010"**,** "-----"**,** "----------------"**,**

"----"**,** "----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** X"0005"**,** X"0008"**,** '1'**),**

**(**'1'**,** "----"**,** "-----"**,** "-----"**,** "00101"**,** "----------------"**,**

"----"**,** X"1f1f"**,** "11--"**,** X"CCCC"**,** "--------"**,** "----------------"**,** X"1f1f"**,** '0'**)**

**);**

**(Continued.)**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_B

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

clk **=>** clk**,**

IMM **=>** IMM**,**

AL **=>** AL**,**

SH **=>** SH**,**

M\_A **=>** M\_A**,**

S **=>** S**,**

flags **=>** flags**,**

M\_B **=>** M\_B**,**

M\_DA **=>** M\_DA**,**

M\_in **=>** M\_in

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

S **<=** test\_vectors**(**i**).**S**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **rising\_edge(**clk**);**

-- Check that the actual outputs are the same as were expecting

-- Have to use std\_match when comparing meta values like '-'

**assert** **std\_match(**flags**,** test\_vectors**(**i**).**flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_B**,** M\_B**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_B**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_B**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]"

**severity** error**;**

-- If there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_B**,** test\_vectors**(**i**).**M\_B**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**))**

**report** lf **&**" [ OK ] Test " **&** integer'**image(**i**)&** " was successful!"

**severity** note**;**

**wait** **until** **falling\_edge(**clk**);**

**end** **loop;**

**wait;**

**end** **process;**

**END;**

**(Continued.)**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]"

**severity** error**;**

-- If there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_B**,** test\_vectors**(**i**).**M\_B**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**))**

**report** lf **&**" [ OK ] Test " **&** integer'**image(**i**)&** " was successful!"

**severity** note**;**

**wait** **until** **falling\_edge(**clk**);**

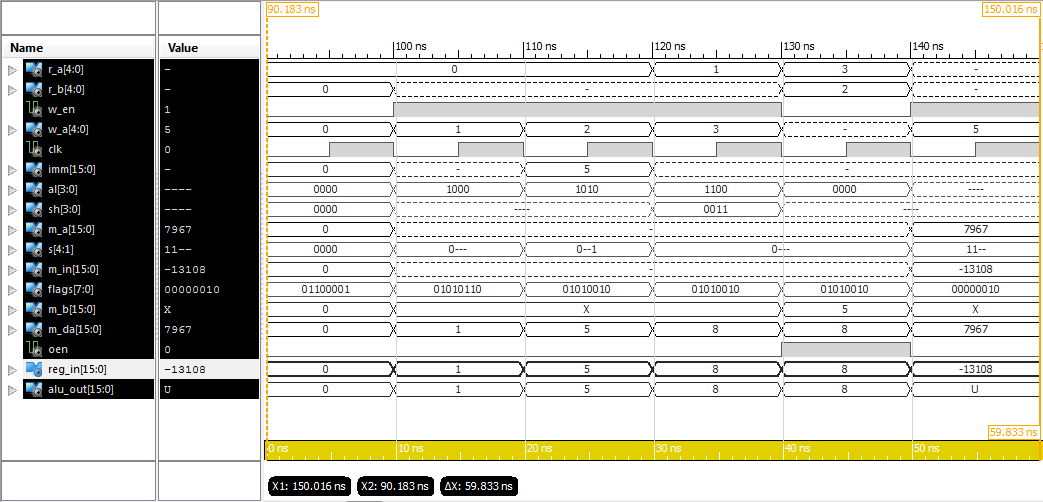
**end** **loop;**

**wait;**

**end** **process;**

**END;**

**Simulations**



This Screenshot shows the entire simulation for DataPath B. The reg\_in signal is the data that will be written to the register bank.

**iSim Console Output**

ISim P.28xd (signature 0xa0883be4)

This is a Full version of ISim.

Time resolution is 1 ps

WARNING: Simulation object /datapathb\_tb/test\_vectors was not traceable in the design for the following reason:

ISim does not yet support tracing of constant and generic multi-dimensional arrays.

Simulator is doing circuit initialization process.

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">=": metavalue detected, returning FALSE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<=": metavalue detected, returning FALSE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">": metavalue detected, returning FALSE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<": metavalue detected, returning FALSE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD."/=": metavalue detected, returning TRUE

at 0 ps, Instance /datapathb\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

Finished circuit initialization process.

at 105 ns(1): Note: [ OK ] Test 0 was successful! (/datapathb\_tb/).

at 115 ns(1): Note: [ OK ] Test 1 was successful! (/datapathb\_tb/).

at 125 ns(1): Note: [ OK ] Test 2 was successful! (/datapathb\_tb/).

at 135 ns(1): Note: [ OK ] Test 3 was successful! (/datapathb\_tb/).

at 145 ns(1): Note: [ OK ] Test 4 was successful! (/datapathb\_tb/).

ISim>

***All but the first meta value warnings have been removed to improve readability.***

**HDL Synthesis**

=========================================================================

\* HDL Synthesis \*

=========================================================================

Synthesizing Unit <DataPath\_B>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\DataPath\_B.vhd".

data\_size = 16

num\_registers = 32

WARNING:Xst:647 - Input <S<2:2>> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.

Summary:

inferred 3 Multiplexer(s).

Unit <DataPath\_B> synthesized.

Synthesizing Unit <ALU\_param>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\ALU\_param.vhd".

N = 16

Found 16-bit adder for signal <A\_itrn[15]\_B\_itrn[15]\_add\_27\_OUT> created at line 69.

Found 16-bit adder for signal <A\_itrn[15]\_GND\_6\_o\_add\_31\_OUT> created at line 1253.

Found 16-bit subtractor for signal <A\_itrn[15]\_B\_itrn[15]\_sub\_26\_OUT<15:0>> created at line 70.

Found 16-bit subtractor for signal <A\_itrn[15]\_GND\_6\_o\_sub\_30\_OUT<15:0>> created at line 1320.

Found 16-bit shifter rotate right for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_right\_17\_OUT> created at line 3021

Found 16-bit shifter rotate left for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_left\_19\_OUT> created at line 3012

Found 16-bit shifter arithmetic right for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_right\_21\_OUT> created at line 2982

Found 16-bit shifter logical left for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_left\_23\_OUT> created at line 2973

Found 16-bit 13-to-1 multiplexer for signal <O\_itrn> created at line 42.

Found 16-bit comparator greater for signal <flags<3>> created at line 99

Found 16-bit comparator greater for signal <flags<4>> created at line 100

Summary:

inferred 1 Adder/Subtractor(s).

inferred 2 Comparator(s).

inferred 16 Multiplexer(s).

inferred 4 Combinational logic shifter(s).

Unit <ALU\_param> synthesized.

Synthesizing Unit <regbank>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\otherRegBank.vhd".

Found 16-bit register for signal <REG02>.

Found 16-bit register for signal <REG03>.

Found 16-bit register for signal <REG04>.

Found 16-bit register for signal <REG05>.

Found 16-bit register for signal <REG06>.

Found 16-bit register for signal <REG07>.

Found 16-bit register for signal <REG08>.

Found 16-bit register for signal <REG09>.

Found 16-bit register for signal <REG10>.

Found 16-bit register for signal <REG11>.

Found 16-bit register for signal <REG12>.

Found 16-bit register for signal <REG13>.

Found 16-bit register for signal <REG14>.

Found 16-bit register for signal <REG15>.

Found 16-bit register for signal <REG16>.

Found 16-bit register for signal <REG17>.

Found 16-bit register for signal <REG18>.

Found 16-bit register for signal <REG19>.

Found 16-bit register for signal <REG20>.

Found 16-bit register for signal <REG21>.

Found 16-bit register for signal <REG22>.

Found 16-bit register for signal <REG23>.

Found 16-bit register for signal <REG24>.

Found 16-bit register for signal <REG25>.

Found 16-bit register for signal <REG26>.

Found 16-bit register for signal <REG27>.

Found 16-bit register for signal <REG28>.

Found 16-bit register for signal <REG29>.

Found 16-bit register for signal <REG30>.

Found 16-bit register for signal <REG31>.

Found 16-bit register for signal <REG01>.

Found 16-bit 32-to-1 multiplexer for signal <A> created at line 30.

Found 16-bit 32-to-1 multiplexer for signal <B> created at line 31.

Summary:

inferred 496 D-type flip-flop(s).

inferred 2 Multiplexer(s).

Unit <regbank> synthesized.

=========================================================================

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors : 1

16-bit addsub : 1

# Registers : 31

16-bit register : 31

# Comparators : 2

16-bit comparator greater : 2

# Multiplexers : 21

1-bit 2-to-1 multiplexer : 6

16-bit 2-to-1 multiplexer : 13

16-bit 32-to-1 multiplexer : 2

# Logic shifters : 4

16-bit shifter arithmetic right : 1

16-bit shifter logical left : 1

16-bit shifter rotate left : 1

16-bit shifter rotate right : 1

# Xors : 1

16-bit xor2 : 1

=========================================================================

# **Architecture C**

**DataPathC.vhd**

**library** IEEE**;**

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

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.ALL;**

**entity** DataPath\_C **is**

**GENERIC(**

data\_size **:** natural **:=** 16**;**

num\_registers **:** natural **:=** 32

**);**

**Port** **(**

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

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

en **:** **in** STD\_LOGIC**;**

-- Inputs

R\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address A

R\_B **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address B

W\_EN **:** **in** STD\_LOGIC**;** -- Register write enable

W\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Write address

IMM **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Itermediate value

M\_A **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

M\_in **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory input (read)

PC **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);** -- Current Program Counter

S **:** **in** STD\_LOGIC\_VECTOR **(**4 **downto** 1**);** -- Selector control

AL **:** **in** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);** -- ALU control

SH **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**data\_size**)-**1 **downto** 0**);** -- Shift amount

-- Outputs

PC\_plus **:** **out** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);** -- Next Program Counter

Flags **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** -- ALU flags

M\_DA **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

M\_out **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**)** -- Memory output (write)

**);**

**end** DataPath\_C**;**

**architecture** Behavioral **of** DataPath\_C **is**

-- Data to write to the registers

**signal** reg\_in **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- Outputs of the registers

**signal** A\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- Output of the register on the A and B buses

**signal** A\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The Outputs of the two Muxes on the input to the ALU

**signal** A\_mux **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_mux **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The output of the combined ALU and shifter

**signal** ALU\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** ALU\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The output of the memory in register

**signal** M\_in\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**begin**

-- The two multiplexers on the input to the ALU

A\_mux **<=** A\_reg\_out **when** S**(**2**)** **=** '0' **else** PC**;**

B\_mux **<=** B\_reg\_out **when** S**(**1**)** **=** '0' **else** IMM**;**

-- The address multiplexer for the memory

M\_DA **<=** M\_A **when** s**(**3**)** **=** '1' **else** ALU\_reg\_out**;**

M\_out **<=** B\_reg\_out**;**

PC\_plus **<=** ALU\_reg\_out**;**

-- The register write multiplexer

reg\_in **<=** ALU\_reg\_out **when** s**(**4**)** **=** '0' **else** M\_in\_reg\_out**;**

-- The register on the A bus

A\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** A\_data**,**

data\_out **=>** A\_reg\_out

**);**

-- The register on the B bus

B\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_data**,**

data\_out **=>** B\_reg\_out

**);**

-- The Register on the out put of the ALU

ALU\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_out**,**

data\_out **=>** ALU\_reg\_out

**);**

-- The register on the memory read bus

M\_in\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** M\_in**,**

data\_out **=>** M\_in\_reg\_out

**);**

-- The ALU and shifter from Lab 1

ALU**:** **entity** work**.**ALU\_param **GENERIC** **MAP(** N **=>** data\_size **)**

**PORT** **MAP(**

A **=>** A\_mux**,**

B **=>** B\_mux**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

rst **=>** '0'

**);**

**end** Behavioral**;**

**(Continued.)**

-- The register write multiplexer

reg\_in **<=** ALU\_reg\_out **when** s**(**4**)** **=** '0' **else** M\_in\_reg\_out**;**

-- The register on the A bus

A\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** A\_data**,**

data\_out **=>** A\_reg\_out

**);**

-- The register on the B bus

B\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_data**,**

data\_out **=>** B\_reg\_out

**);**

-- The Register on the out put of the ALU

ALU\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_out**,**

data\_out **=>** ALU\_reg\_out

**);**

-- The register on the memory read bus

M\_in\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** M\_in**,**

data\_out **=>** M\_in\_reg\_out

**);**

-- The ALU and shifter from Lab 1

ALU**:** **entity** work**.**ALU\_param **GENERIC** **MAP(** N **=>** data\_size **)**

**PORT** **MAP(**

A **=>** A\_mux**,**

B **=>** B\_mux**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

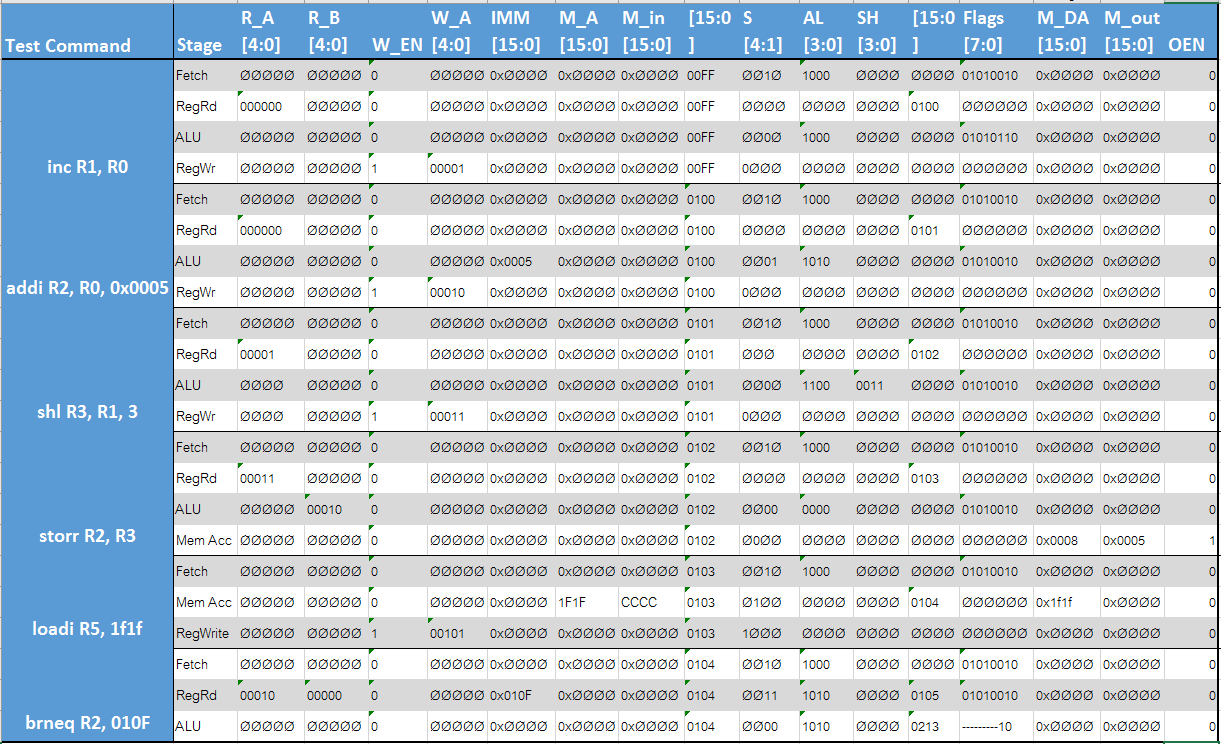
rst **=>** '0'

**);**

**end** Behavioral**;**

**Test Data for DataPath C**

Here is the control signals and test data fro datapath C



**DataPathC-TB.vhd**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** work**.**DigEng**.ALL;**

**USE** work**.**easyprint**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**ENTITY** DataPath\_C\_TB **IS**

**END** DataPath\_C\_TB**;**

**ARCHITECTURE** behavior **OF** DataPath\_C\_TB **IS**

-- Constants

**constant** data\_size **:** NATURAL **:=** 16**;**

**constant** num\_registers **:** NATURAL **:=** 32**;**

-- Clock period definitions

**constant** clk\_period **:** time **:=** 10 ns**;**

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

**COMPONENT** DataPath\_C

**GENERIC(**

data\_size **:** natural**;**

num\_registers **:** natural

**);**

**PORT(**

clk **:** **IN** std\_logic**;**

rst **:** **IN** std\_logic**;**

en **:** **IN** std\_logic**;**

R\_A **:** **IN** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** **IN** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_EN **:** **IN** std\_logic**;**

W\_A **:** **IN** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** **IN** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

M\_A **:** **IN** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

M\_in **:** **IN** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

PC **:** **IN** std\_logic\_vector **(**15 **downto** 0**);**

S **:** **IN** std\_logic\_vector **(**4 **downto** 1**);**

AL **:** **IN** std\_logic\_vector **(**3 **downto** 0**);**

SH **:** **IN** std\_logic\_vector **(**log2**(**data\_size**)-**1 **downto** 0**);**

PC\_plus **:** **OUT** std\_logic\_vector **(**15 **downto** 0**);**

Flags **:** **OUT** std\_logic\_vector **(**7 **downto** 0**);**

M\_DA **:** **OUT** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

M\_out **:** **OUT** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)**

**);**

**END** **COMPONENT;**

--Inputs

**signal** clk **:** std\_logic **:=** '0'**;**

**signal** rst **:** std\_logic **:=** '1'**;**

**signal** en **:** std\_logic **:=** '0'**;**

**signal** R\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** R\_B **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** W\_EN **:** std\_logic **:=** '0'**;**

**signal** W\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0 **)** **:=** **(others** **=>** '0'**);**

**signal** IMM **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** M\_A **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** M\_in **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

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

**signal** S **:** std\_logic\_vector **(**4 **downto** 1**)** **:=** **(others** **=>** '0'**);**

**signal** AL **:** std\_logic\_vector **(**3 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** SH **:** std\_logic\_vector **(**log2**(**data\_size**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

--Outputs

**signal** PC\_plus **:** std\_logic\_vector **(**15 **downto** 0**);**

**signal** Flags **:** std\_logic\_vector **(**7 **downto** 0**);**

**signal** M\_DA **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** M\_out **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** OEN **:** std\_logic **:=** '0'**;**

-- Test data definitions

**type** TEST\_VECTOR **is** **RECORD**

R\_A **:** STD\_LOGIC\_VECTOR**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_EN **:** std\_logic**;**

W\_A **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

PC **:** std\_logic\_vector**(**15 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

AL **:** std\_logic\_vector**(**3 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

PC\_plus**:** std\_logic\_vector**(**15 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_out **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

-- Test Data

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

--R\_A, R\_B, W\_EN, W\_A, IMM, M\_A, M\_in, PC, S, AL, SH, PC\_plus, flags, M\_DA, M\_out

-- inc R1, R0

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00000"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "----"**,** "----"**,** "----"**,** X"0100"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "--0-"**,** "1000"**,** "----"**,** "----------------"**,** "01010110"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00001"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- addi R2, R0, 0x0005

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00000"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "----"**,** "----"**,** "----"**,** X"0101"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** X"0005"**,** "----------------"**,** "----------------"**,** X"0100"**,** "--01"**,** "1010"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00010"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- shl R3, R1, 3

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00001"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "----"**,** "----"**,** "----"**,** X"0102"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "--0-"**,** "1100"**,** "0011"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00011"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- storr R2, R3

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00011"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "----"**,** "----"**,** "----"**,** X"0103"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "00010"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--00"**,** "0000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "-0--"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** X"0008"**,** X"0005" **,** '1' **),**

-- loadi R5, 1f1f

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** X"1F1F"**,** X"CCCC"**,** X"0103"**,** "-1--"**,** "----"**,** "----"**,** X"0104"**,** "--------"**,** X"1f1f"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00101"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "1---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- brneq R2, 010F

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00010"**,** "00000"**,** '0'**,** "-----"**,** X"010f"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--11"**,** "1010"**,** "----"**,** X"0105"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--00"**,** "1010"**,** "----"**,** X"0213"**,** "------10"**,** "----------------"**,** "----------------"**,** '0' **)**

**);**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_C

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

clk **=>** clk**,**

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

en **=>** en**,**

-- Inputs

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

IMM **=>** IMM**,**

M\_A **=>** M\_A**,**

M\_in **=>** M\_in**,**

PC **=>** PC**,**

S **=>** S**,**

AL **=>** AL**,**

SH **=>** SH**,**

-- Outputs

PC\_plus **=>** PC\_plus**,**

Flags **=>** Flags**,**

M\_DA **=>** M\_DA**,**

M\_out **=>** M\_out

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- enable the system

rst **<=** '0'**;**

en **<=** '1'**;**

**wait** **for** 2**\***clk\_period**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

**wait** **until** **rising\_edge(**clk**);**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

PC **<=** test\_vectors**(**i**).**PC**;**

S **<=** test\_vectors**(**i**).**S**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **falling\_edge(**clk**);**

-- Check to see if the out put was what we were expecting

-- Have to use std\_match() instead of = when comparing meta values like '-'

**assert** **std\_match(**test\_vectors**(**i**).**flags**,** flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_out**,** M\_out**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_out**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_out**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual program counter did not equal expected program counter."**&**

" Actual [ " **&** u\_tostr**(**PC\_plus**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**PC\_plus**)** **&** " ]" **&** lf

**severity** error**;**

-- if there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_out**,** test\_vectors**(**i**).**M\_out**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**)** and

**std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**)**

**report** lf **&** " [ OK ] Test " **&** integer'**image(**i**)&** " was successful!" **&** lf

**severity** note**;**

**end** **loop;**

-- End of test

**wait;**

**end** **process;**

**END;**

**(Continued.)**

W\_EN **:** std\_logic**;**

W\_A **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

PC **:** std\_logic\_vector**(**15 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

AL **:** std\_logic\_vector**(**3 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

PC\_plus**:** std\_logic\_vector**(**15 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_out **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

-- Test Data

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

--R\_A, R\_B, W\_EN, W\_A, IMM, M\_A, M\_in, PC, S, AL, SH, PC\_plus, flags, M\_DA, M\_out,

OEN --

-- inc R1, R0

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00000"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "----"**,** "----"**,** "----"**,** X"0100"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "--0-"**,**  "1000"**,** "----"**,** "----------------"**,** "01010110"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00001"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"00FF"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- addi R2, R0, 0x0005

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00000"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "----"**,** "----"**,** "----"**,**  X"0101"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** X"0005"**,** "----------------"**,** "----------------"**,** X"0100"**,** "--01"**,** "1010"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00010"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0100"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- shl R3, R1, 3

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00001"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "----"**,** "----"**,** "----"**,** X"0102"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "--0-"**,** "1100"**,** "0011"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00011"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "0---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- storr R2, R3

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00011"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "----"**,** "----"**,** "----"**,** X"0103"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "00010"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--00"**,** "0000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "-0--"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** X"0008"**,** X"0005" **,** '1' **),**

-- loadi R5, 1f1f

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** X"1F1F"**,** X"CCCC"**,** X"0103"**,** "-1--"**,** "----"**,** "----"**,** X"0104"**,** "--------"**,** X"1f1f"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00101"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "1---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- brneq R2, 010F

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00010"**,** "00000"**,** '0'**,** "-----"**,** X"010f"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--11"**,** "1010"**,** "----"**,** X"0105"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--00"**,** "1010"**,** "----"**,** X"0213"**,** "------10"**,** "----------------"**,** "----------------"**,** '0' **)**

**);**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_C

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

clk **=>** clk**,**

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

en **=>** en**,**

-- Inputs

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

IMM **=>** IMM**,**

M\_A **=>** M\_A**,**

M\_in **=>** M\_in**,**

PC **=>** PC**,**

S **=>** S**,**

AL **=>** AL**,**

SH **=>** SH**,**

-- Outputs

PC\_plus **=>** PC\_plus**,**

Flags **=>** Flags**,**

M\_DA **=>** M\_DA**,**

M\_out **=>** M\_out

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- enable the system

rst **<=** '0'**;**

en **<=** '1'**;**

**wait** **for** 2**\***clk\_period**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

**wait** **until** **rising\_edge(**clk**);**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

PC **<=** test\_vectors**(**i**).**PC**;**

S **<=** test\_vectors**(**i**).**S**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **falling\_edge(**clk**);**

-- Check to see if the out put was what we were expecting

-- Have to use std\_match() instead of = when comparing meta values like '-'

**assert** **std\_match(**test\_vectors**(**i**).**flags**,** flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_out**,** M\_out**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_out**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_out**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual program counter did not equal expected program counter."**&**

" Actual [ " **&** u\_tostr**(**PC\_plus**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**PC\_plus**)** **&** " ]" **&** lf

**severity** error**;**

-- if there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_out**,** test\_vectors**(**i**).**M\_out**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**)** and

**std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**)**

**report** lf **&** " [ OK ] Test " **&** integer'**image(**i**)&** " was successful!" **&** lf

**severity** note**;**

**end** **loop;**

-- End of test

**wait;**

**end** **process;**

**END;**

**(Continued.)**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "--0-"**,** "1100"**,** "0011"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00011"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0101"**,** "0---"**,** "----"**,** "----"**,**  "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- storr R2, R3

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00011"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "----"**,** "----"**,**  "----"**,** X"0103"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "00010"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "--00"**,** "0000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0102"**,** "-0--"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** X"0008"**,** X"0005"**,** '1' **),**

-- loadi R5, 1f1f

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** X"1F1F"**,** X"CCCC"**,** X"0103"**,** "-1--"**,** "----"**,** "----"**,** X"0104"**,** "--------"**,** X"1f1f"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '1'**,** "00101"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0103"**,** "1---"**,** "----"**,** "----"**,** "----------------"**,** "--------"**,** "----------------"**,** "----------------"**,** '0' **),**

-- brneq R2, 010F

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--1-"**,** "1000"**,** "----"**,** "----------------"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "00010"**,** "00000"**,** '0'**,** "-----"**,** X"010f"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--11"**,** "1010"**,** "----"**,** X"0105"**,** "01010010"**,** "----------------"**,** "----------------"**,** '0' **),**

**(** "-----"**,** "-----"**,** '0'**,** "-----"**,** "----------------"**,** "----------------"**,** "----------------"**,** X"0104"**,** "--00"**,** " 1010"**,** "----"**,** X"0213"**,** "------10"**,** "----------------"**,** "----------------"**,** '0' **)**

**);**

**(Continued.)**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_C

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

clk **=>** clk**,**

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

en **=>** en**,**

-- Inputs

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

IMM **=>** IMM**,**

M\_A **=>** M\_A**,**

M\_in **=>** M\_in**,**

PC **=>** PC**,**

S **=>** S**,**

AL **=>** AL**,**

SH **=>** SH**,**

-- Outputs

PC\_plus **=>** PC\_plus**,**

Flags **=>** Flags**,**

M\_DA **=>** M\_DA**,**

M\_out **=>** M\_out

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- enable the system

rst **<=** '0'**;**

en **<=** '1'**;**

**wait** **for** 2**\***clk\_period**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

**wait** **until** **rising\_edge(**clk**);**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

PC **<=** test\_vectors**(**i**).**PC**;**

S **<=** test\_vectors**(**i**).**S**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **falling\_edge(**clk**);**

-- Check to see if the out put was what we were expecting

-- Have to use std\_match() instead of = when comparing meta values like '-'

**assert** **std\_match(**test\_vectors**(**i**).**flags**,** flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_out**,** M\_out**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_out**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_out**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual program counter did not equal expected program counter."**&**

" Actual [ " **&** u\_tostr**(**PC\_plus**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**PC\_plus**)** **&** " ]" **&** lf

**severity** error**;**

-- if there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_out**,** test\_vectors**(**i**).**M\_out**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**)** and

**std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**)**

**report** lf **&** " [ OK ] Test " **&** integer'**image(**i**)&** " was successful!" **&** lf

**severity** note**;**

**end** **loop;**

-- End of test

**wait;**

**end** **process;**

**END;**

**(Continued.)**

**wait** **until** **falling\_edge(**clk**);**

-- Check to see if the out put was what we were expecting

-- Have to use std\_match() instead of = when comparing meta values like '-'

**assert** **std\_match(**test\_vectors**(**i**).**flags**,** flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_out**,** M\_out**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_out**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_out**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]" **&** lf

**severity** error**;**

**assert** **std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&**

" Actual program counter did not equal expected program counter."**&**

" Actual [ " **&** u\_tostr**(**PC\_plus**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**PC\_plus**)** **&** " ]" **&** lf

**severity** error**;**

-- if there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_out**,** test\_vectors**(**i**).**M\_out**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**)** and

**std\_match(**PC\_plus**,** test\_vectors**(**i**).**PC\_plus**)**

**)**

**report** lf **&** " [ OK ] Test " **&** integer'**image(**i**)&** " was successful!" **&** lf

**severity** note**;**

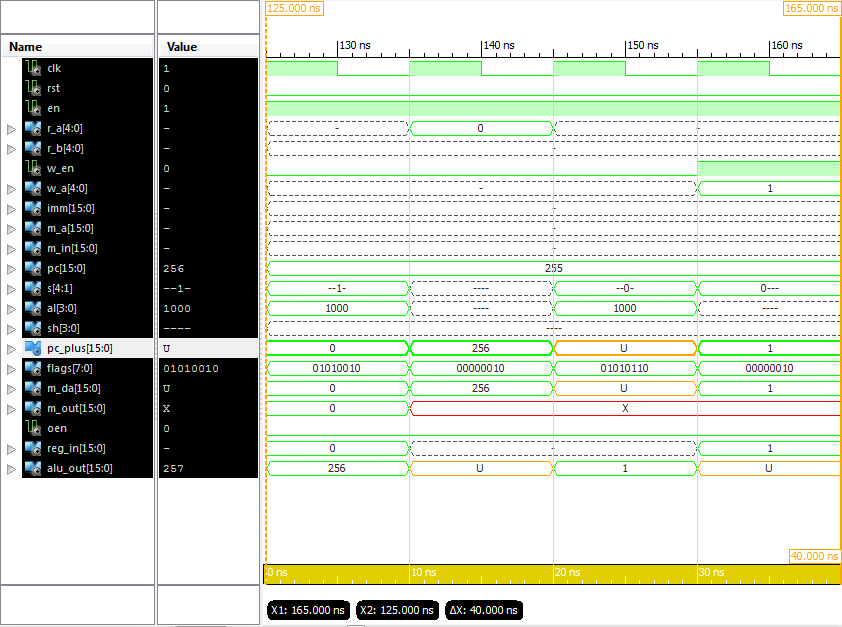
**end** **loop;**

-- End of test

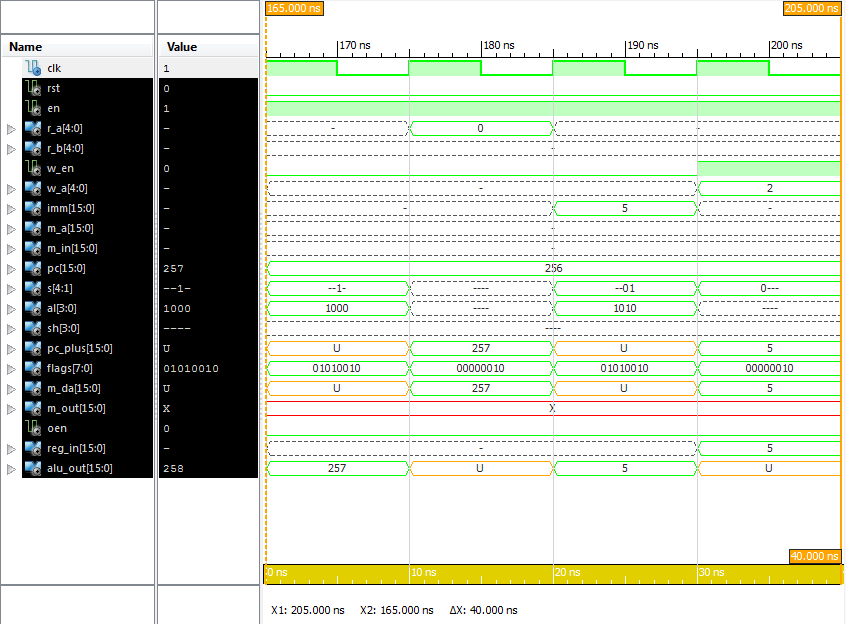
**wait;**

**end** **process;**

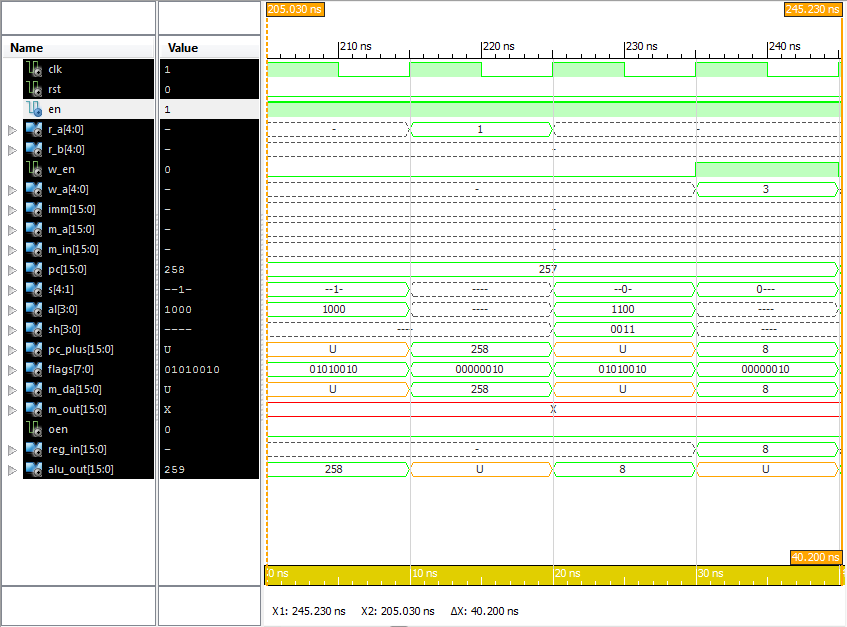
**END;**

**Simulations**

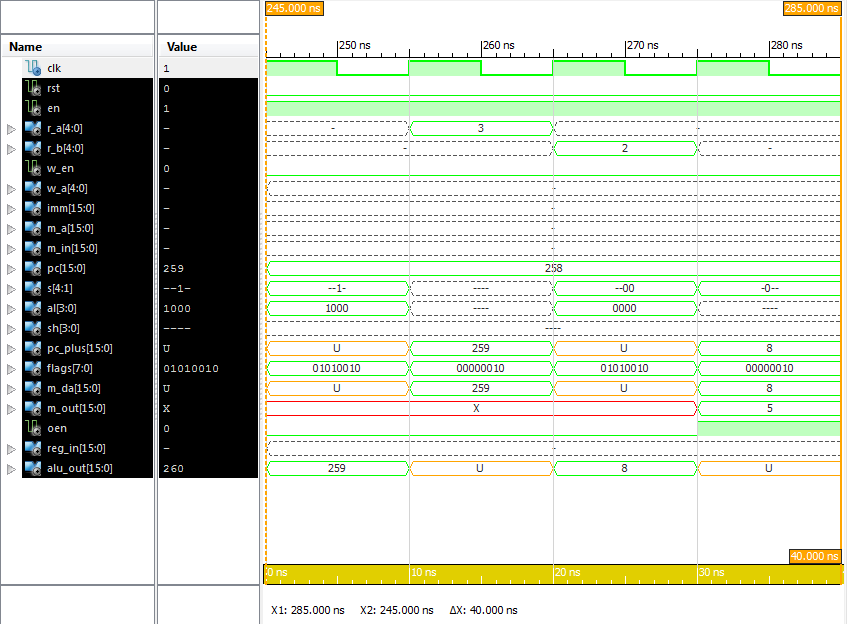
Screenshot showing the ‘inc’ instruction



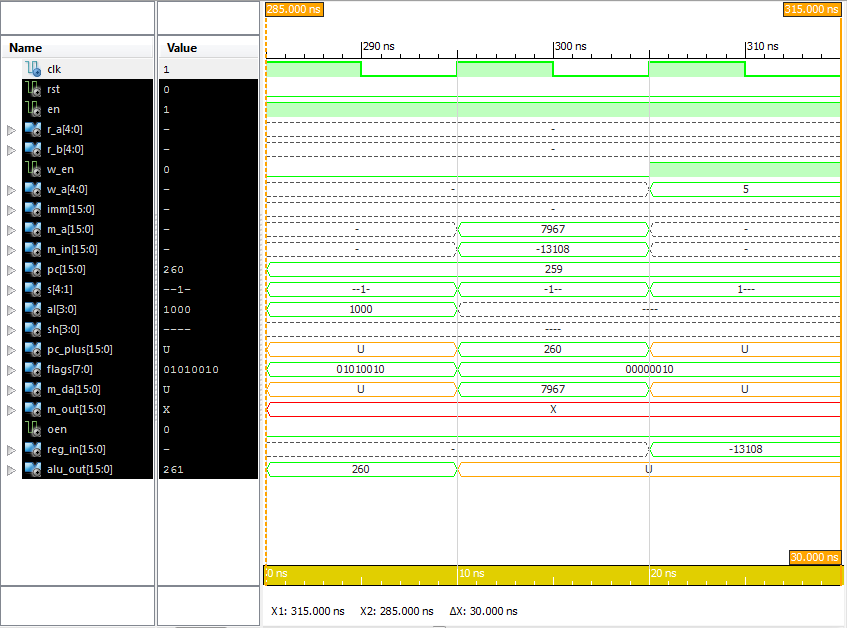
Screenshot showing the ‘addi’ instruction



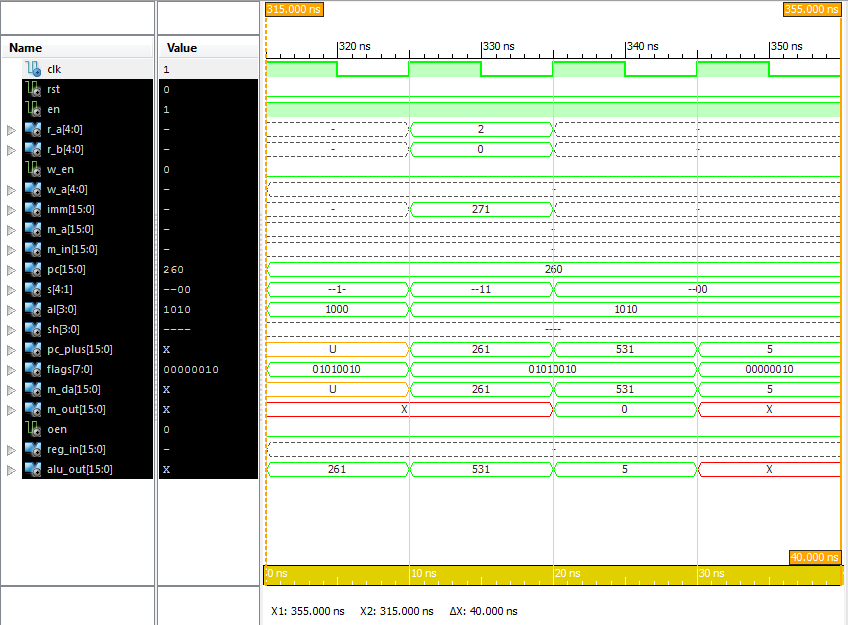
Screenshot showing the ‘shl’ instruction



Screenshot showing the ‘storr’ instruction



Screenshot showing the ‘loadi’ instruction



Screenshot showing the ‘brneq’ instruction

**iSim Console Output**

ISim P.28xd (signature 0xa0883be4)

This is a Full version of ISim.

Time resolution is 1 ps

WARNING: Simulation object /datapath\_c\_tb/test\_vectors was not traceable in the design for the following reason:

ISim does not yet support tracing of constant and generic multi-dimensional arrays.

Simulator is doing circuit initialization process.

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD."/=": metavalue detected, returning TRUE

at 0 ps, Instance /datapath\_c\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

Finished circuit initialization process.

at 130 ns(1): Note: [ OK ] Test 0 was successful! (/datapath\_c\_tb/).

at 140 ns(1): Note: [ OK ] Test 1 was successful! (/datapath\_c\_tb/).

at 150 ns(1): Note: [ OK ] Test 2 was successful! (/datapath\_c\_tb/).

at 160 ns(1): Note: [ OK ] Test 3 was successful! (/datapath\_c\_tb/).

at 170 ns(1): Note: [ OK ] Test 4 was successful! (/datapath\_c\_tb/).

at 180 ns(1): Note: [ OK ] Test 5 was successful! (/datapath\_c\_tb/).

at 190 ns(1): Note: [ OK ] Test 6 was successful! (/datapath\_c\_tb/).

at 200 ns(1): Note: [ OK ] Test 7 was successful!(/datapath\_c\_tb/).

at 210 ns(1): Note: [ OK ] Test 8 was successful!(/datapath\_c\_tb/).

at 220 ns(1): Note: [ OK ] Test 9 was successful! (/datapath\_c\_tb/).

at 230 ns(1): Note: [ OK ] Test 10 was successful! (/datapath\_c\_tb/).

at 240 ns(1): Note: [ OK ] Test 11 was successful! (/datapath\_c\_tb/).

at 250 ns(1): Note: [ OK ] Test 12 was successful! (/datapath\_c\_tb/).

at 260 ns(1): Note: [ OK ] Test 13 was successful! (/datapath\_c\_tb/).

at 270 ns(1): Note: [ OK ] Test 14 was successful! (/datapath\_c\_tb/).

at 280 ns(1): Note: [ OK ] Test 15 was successful! (/datapath\_c\_tb/).

at 290 ns(1): Note: [ OK ] Test 16 was successful! (/datapath\_c\_tb/).

at 300 ns(1): Note: [ OK ] Test 17 was successful! (/datapath\_c\_tb/).

at 310 ns(1): Note: [ OK ] Test 18 was successful! (/datapath\_c\_tb/).

at 320 ns(1): Note: [ OK ] Test 19 was successful! (/datapath\_c\_tb/).

at 330 ns(1): Note: [ OK ] Test 20 was successful! (/datapath\_c\_tb/).

at 340 ns(1): Note: [ OK ] Test 21 was successful! (/datapath\_c\_tb/).

ISim>

***All but the first meta value warnings have been removed to improve readability.***

**HDL Synthesis**

=========================================================================

\* HDL Synthesis \*

=========================================================================

Synthesizing Unit <DataPath\_C>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\DataPath\_C.vhd".

data\_size = 16

num\_registers = 32

Summary:

inferred 4 Multiplexer(s).

Unit <DataPath\_C> synthesized.

Synthesizing Unit <Reg>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\Reg.vhd".

data\_size = 16

Found 16-bit register for signal <data\_out>.

Summary:

inferred 16 D-type flip-flop(s).

Unit <Reg> synthesized.

Synthesizing Unit <ALU\_param>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\ALU\_param.vhd".

N = 16

Found 16-bit adder for signal <A\_itrn[15]\_B\_itrn[15]\_add\_27\_OUT> created at line 69.

Found 16-bit adder for signal <A\_itrn[15]\_GND\_7\_o\_add\_31\_OUT> created at line 1253.

Found 16-bit subtractor for signal <A\_itrn[15]\_B\_itrn[15]\_sub\_26\_OUT<15:0>> created at line 70.

Found 16-bit subtractor for signal <A\_itrn[15]\_GND\_7\_o\_sub\_30\_OUT<15:0>> created at line 1320.

Found 16-bit shifter rotate right for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_right\_17\_OUT> created at line 3021

Found 16-bit shifter rotate left for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_left\_19\_OUT> created at line 3012

Found 16-bit shifter arithmetic right for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_right\_21\_OUT> created at line 2982

Found 16-bit shifter logical left for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_left\_23\_OUT> created at line 2973

Found 16-bit 13-to-1 multiplexer for signal <O\_itrn> created at line 42.

Found 16-bit comparator greater for signal <flags<3>> created at line 99

Found 16-bit comparator greater for signal <flags<4>> created at line 100

Summary:

inferred 1 Adder/Subtractor(s).

inferred 2 Comparator(s).

inferred 16 Multiplexer(s).

inferred 4 Combinational logic shifter(s).

Unit <ALU\_param> synthesized.

Synthesizing Unit <regbank>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\otherRegBank.vhd".

Found 16-bit register for signal <REG02>.

Found 16-bit register for signal <REG03>.

Found 16-bit register for signal <REG04>.

Found 16-bit register for signal <REG05>.

Found 16-bit register for signal <REG06>.

Found 16-bit register for signal <REG07>.

Found 16-bit register for signal <REG08>.

Found 16-bit register for signal <REG09>.

Found 16-bit register for signal <REG10>.

Found 16-bit register for signal <REG11>.

Found 16-bit register for signal <REG12>.

Found 16-bit register for signal <REG13>.

Found 16-bit register for signal <REG14>.

Found 16-bit register for signal <REG15>.

Found 16-bit register for signal <REG16>.

Found 16-bit register for signal <REG17>.

Found 16-bit register for signal <REG18>.

Found 16-bit register for signal <REG19>.

Found 16-bit register for signal <REG20>.

Found 16-bit register for signal <REG21>.

Found 16-bit register for signal <REG22>.

Found 16-bit register for signal <REG23>.

Found 16-bit register for signal <REG24>.

Found 16-bit register for signal <REG25>.

Found 16-bit register for signal <REG26>.

Found 16-bit register for signal <REG27>.

Found 16-bit register for signal <REG28>.

Found 16-bit register for signal <REG29>.

Found 16-bit register for signal <REG30>.

Found 16-bit register for signal <REG31>.

Found 16-bit register for signal <REG01>.

Found 16-bit 32-to-1 multiplexer for signal <A> created at line 30.

Found 16-bit 32-to-1 multiplexer for signal <B> created at line 31.

Summary:

inferred 496 D-type flip-flop(s).

inferred 2 Multiplexer(s).

Unit <regbank> synthesized.

=========================================================================

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors : 1

16-bit addsub : 1

# Registers : 35

16-bit register : 35

# Comparators : 2

16-bit comparator greater : 2

# Multiplexers : 22

1-bit 2-to-1 multiplexer : 6

16-bit 2-to-1 multiplexer : 14

16-bit 32-to-1 multiplexer : 2

# Logic shifters : 4

16-bit shifter arithmetic right : 1

16-bit shifter logical left : 1

16-bit shifter rotate left : 1

16-bit shifter rotate right : 1

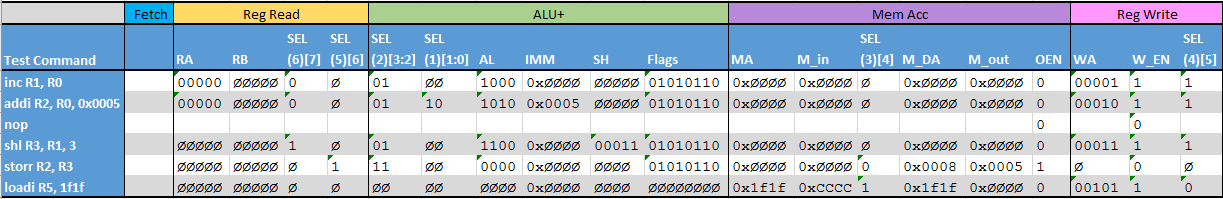
# Xors : 1

16-bit xor2 : 1

=========================================================================

**Test Data for Architecture C**

Here is our control logic for Arch C as well as the expected outputs.

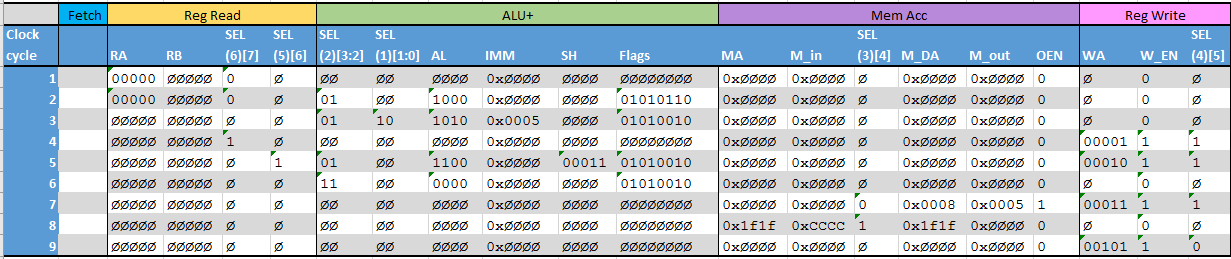


The instruction time line below shows how we used a nop instruction as a pipeline stall between the addi instruction and the shl instruction. We had to use a pipeline stall in this case rather forwarding, as there was no way to use any of the forwarding paths to get the value to be in the correct location at the time when we needed it.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test command** | **TimeLine** | | | | | | | | | |
| **inc R1, R0** | Fetch | Reg Read | ALU+ | Mem Acc | Reg Write |  |  |  |  |  |
| **addi R2, R0, 0x0005** |  | Fetch | Reg Read | ALU+ | Mem Acc | Reg Write |  |  |  |  |
| **nop** |  |  |  |  |  |  |  |  |  |  |
| **shl R3, R1, 3** |  |  |  | Fetch | Reg Read | ALU+ | Mem Acc | Reg Write |  |  |
| **storr R2, R3** |  |  |  |  | Fetch | Reg Read | ALU+ | Mem Acc | Reg Write |  |
| **loadi R5, 1f1f** |  |  |  |  |  | Fetch | Reg Read | ALU+ | Mem Acc | Reg Write |

We also used forwarding in our control logic. In fact, the only time we take a value directly from a register is when the value of R Zero is used for the first two instruction. Both the shift left and the store instruction use forwarding in order to execute sooner than they would of other wise.

You can see in the table below that the only time RA has a definite value is during the first two clock cylces. You can also see use of select 5 and 6 to forward data ( that has not yet been writte to the register) into the A and B registers. Select 2 has also been used to forward data from the output of the alu back to the input of the alu.



**Architecture D**

**DataPathD.vhd**

**library** IEEE**;**

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

**use** IEEE**.**NUMERIC\_STD**.ALL;**

**use** work**.**DigEng**.All;**

**entity** DataPath\_D **is**

**Generic(**

data\_size **:** natural **:=** 16**;**

num\_registers **:** natural **:=** 32

**);**

**Port** **(**

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

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

en **:** **in** STD\_LOGIC**;**

R\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address A

R\_B **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Read address B

W\_EN **:** **in** STD\_LOGIC**;** -- Register write enable

W\_A **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**num\_registers**)-**1 **downto** 0**);** -- Write address

IMM **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Itermediate value

M\_A **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

M\_in **:** **in** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory input (read)

SEL **:** **in** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** -- Selector control

AL **:** **in** STD\_LOGIC\_VECTOR **(**3 **downto** 0**);** -- ALU control

SH **:** **in** STD\_LOGIC\_VECTOR **(**log2**(**data\_size**)-**1 **downto** 0**);** -- Shift amount

Flags **:** **out** STD\_LOGIC\_VECTOR **(**7 **downto** 0**);** -- ALU flags

M\_DA **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);** -- Memory address

M\_out **:** **out** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**)** -- Memory output (write)

**);**

**end** DataPath\_D**;**

**architecture** Behavioral **of** DataPath\_D **is**

-- Data to write to the registers

**signal** reg\_in **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- Outputs of the registers

**signal** A\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_data **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The two multiplexers for A and B

**signal** A\_mux **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_mux **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- Output of the register on the A and B buses

**signal** A\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The inputs to the ALU

**signal** A\_ALU **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** B\_ALU **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The output of the combined ALU and shifter

**signal** ALU\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**signal** ALU\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- The ALU output after it passes through a second register

**signal** ALU\_reg\_out\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

-- Signals for memory registers

**signal** M\_in\_reg\_out **:** STD\_LOGIC\_VECTOR **(**data\_size**-**1 **downto** 0**);**

**begin**

-- The two multiplexers for A and B

A\_mux **<=** A\_data **when** SEL**(**7**)** **=** '0' **else** reg\_in**;**

B\_mux **<=** B\_data **when** SEL**(**6**)** **=** '0' **else** reg\_in**;**

-- The multiplexers on the input of the ALU

A\_ALU **<=**

ALU\_reg\_out **when** SEL**(**3 **downto** 2**)** **=** "11" **else**

**(others** **=>** 'U'**)** **when** SEL**(**3 **downto** 2**)** **=** "10" **else**

A\_reg\_out **when** SEL**(**3 **downto** 2**)** **=** "01" **else**

reg\_in **when** SEL**(**3 **downto** 2**)** **=** "00" **else**

**(others** **=>** 'U'**);**

B\_ALU **<=**

ALU\_reg\_out **when** SEL**(**1 **downto** 0**)** **=** "11" **else**

IMM **when** SEL**(**1 **downto** 0**)** **=** "10" **else**

B\_reg\_out **when** SEL**(**1 **downto** 0**)** **=** "01" **else**

reg\_in **when** SEL**(**1 **downto** 0**)** **=** "00" **else**

**(others** **=>** 'U'**);**

-- The address multiplexer for the memory

M\_DA **<=** M\_A **when** SEL**(**4**)** **=** '1' **else** ALU\_reg\_out**;**

-- The register write multiplexer

reg\_in **<=** ALU\_reg\_out\_reg\_out **when** SEL**(**5**)** **=** '1' **else** M\_in\_reg\_out**;**

-- The register on the A bus

A\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** A\_mux**,**

data\_out **=>** A\_reg\_out

**);**

-- The register on the B bus

B\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_mux**,**

data\_out **=>** B\_reg\_out

**);**

-- The Register on the out put of the ALU

ALU\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_out**,**

data\_out **=>** ALU\_reg\_out

**);**

-- The register on the memory read bus

M\_in\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,** data\_in **=>** M\_in**,**

data\_out **=>** M\_in\_reg\_out

**);**

-- The second register on the ALU out

ALU\_reg\_out\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_reg\_out**,**

data\_out **=>** ALU\_reg\_out\_reg\_out

**);**

-- The register on the memory read bus

M\_out\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_reg\_out**,**

data\_out **=>** M\_out

**);**

-- The ALU

ALU**:** **entity** work**.**ALU\_param **GENERIC** **MAP(** N **=>** data\_size **)**

**PORT** **MAP(**

A **=>** A\_ALU**,**

B **=>** B\_ALU**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

rst **=>** '0'

**);**

**end** Behavioral**;**

**(Continued.)**

-- The multiplexers on the input of the ALU

A\_ALU **<=**

ALU\_reg\_out **when** SEL**(**3 **downto** 2**)** **=** "11" **else**

**(others** **=>** 'U'**)** **when** SEL**(**3 **downto** 2**)** **=** "10" **else**

A\_reg\_out **when** SEL**(**3 **downto** 2**)** **=** "01" **else**

reg\_in **when** SEL**(**3 **downto** 2**)** **=** "00" **else**

**(others** **=>** 'U'**);**

B\_ALU **<=**

ALU\_reg\_out **when** SEL**(**1 **downto** 0**)** **=** "11" **else**

IMM **when** SEL**(**1 **downto** 0**)** **=** "10" **else**

B\_reg\_out **when** SEL**(**1 **downto** 0**)** **=** "01" **else**

reg\_in **when** SEL**(**1 **downto** 0**)** **=** "00" **else**

**(others** **=>** 'U'**);**

-- The address multiplexer for the memory

M\_DA **<=** M\_A **when** SEL**(**4**)** **=** '1' **else** ALU\_reg\_out**;**

-- The register write multiplexer

reg\_in **<=** ALU\_reg\_out\_reg\_out **when** SEL**(**5**)** **=** '1' **else** M\_in\_reg\_out**;**

-- The register on the A bus

A\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** A\_mux**,**

data\_out **=>** A\_reg\_out

**);**

-- The register on the B bus

B\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_mux**,**

data\_out **=>** B\_reg\_out

**);**

-- The Register on the out put of the ALU

ALU\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_out**,**

data\_out **=>** ALU\_reg\_out

**);**

-- The register on the memory read bus

M\_in\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,** data\_in **=>** M\_in**,**

data\_out **=>** M\_in\_reg\_out

**);**

-- The second register on the ALU out

ALU\_reg\_out\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** ALU\_reg\_out**,**

data\_out **=>** ALU\_reg\_out\_reg\_out

**);**

-- The register on the memory read bus

M\_out\_reg**:** **entity** work**.**Reg **Generic** **Map** **(**data\_size **=>** data\_size**)**

**PORT** **MAP(**

clk **=>** clk**,**

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

en **=>** en**,**

data\_in **=>** B\_reg\_out**,**

data\_out **=>** M\_out

**);**

-- The ALU

ALU**:** **entity** work**.**ALU\_param **GENERIC** **MAP(** N **=>** data\_size **)**

**PORT** **MAP(**

A **=>** A\_ALU**,**

B **=>** B\_ALU**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

rst **=>** '0'

**);**

**end** Behavioral**;**

**(Continued.)**

-- The ALU

ALU**:** **entity** work**.**ALU\_param **GENERIC** **MAP(** N **=>** data\_size **)**

**PORT** **MAP(**

A **=>** A\_ALU**,**

B **=>** B\_ALU**,**

X **=>** SH**,**

ctrl **=>** AL**,**

O **=>** ALU\_out**,**

flags **=>** flags

**);**

-- The register bank

Registers**:** **entity** work**.**regbank

**PORT** **MAP(**

RSELA **=>** R\_A**,**

RSELB **=>** R\_B**,**

WSEL **=>** W\_A**,**

D **=>** reg\_in**,**

WEN **=>** W\_EN**,**

clk **=>** clk**,**

A **=>** A\_data**,**

B **=>** B\_data**,**

rst **=>** '0'

**);**

**end** Behavioral**;**

**DataPathD-TB.vhd**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** work**.**DigEng**.ALL;**

**USE** work**.**easyprint**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**ENTITY** DataPathB\_TB **IS**

**END** DataPathB\_TB**;**

**ARCHITECTURE** behavior **OF** DataPathB\_TB **IS**

-- Constants

**constant** data\_size **:** NATURAL **:=** 16**;**

**constant** num\_registers **:** NATURAL **:=** 32**;**

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

**COMPONENT** DataPath\_B

**GENERIC(**

data\_size **:** natural**;**

num\_registers **:** natural

**);**

**PORT(**

R\_A **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_EN **:** **IN** std\_logic**;**

W\_A **:** **IN** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

clk **:** **IN** std\_logic**;**

IMM **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

AL **:** **IN** std\_logic\_vector**(**3 **downto** 0**);**

SH **:** **IN** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**);**

M\_A **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** **IN** std\_logic\_vector**(**4 **downto** 1**);**

flags **:** **OUT** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** **OUT** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** **OUT** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_in **:** **IN** std\_logic\_vector**(**data\_size**-**1 **downto** 0**)**

**);**

**END** **COMPONENT;**

--Inputs

**signal** R\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** R\_B **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** W\_EN **:** std\_logic **:=** '0'**;**

**signal** W\_A **:** std\_logic\_vector **(**log2**(**num\_registers**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** clk **:** std\_logic **:=** '0'**;**

**signal** IMM **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** AL **:** std\_logic\_vector **(**3 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** SH **:** std\_logic\_vector **(**log2**(**data\_size**)-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** M\_A **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

**signal** S **:** std\_logic\_vector **(**4 **downto** 1**)** **:=** **(others** **=>** '0'**);**

**signal** M\_in **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**)** **:=** **(others** **=>** '0'**);**

--Outputs

**signal** flags **:** std\_logic\_vector **(**7 **downto** 0**);**

**signal** M\_B **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** M\_DA **:** std\_logic\_vector **(**data\_size**-**1 **downto** 0**);**

**signal** OEN **:** std\_logic **:=** '0'**;**

-- Clock period definitions

**constant** clk\_period **:** time **:=** 10 ns**;**

**constant** wait\_time **:** time **:=** clk\_period**;**

-- Test data for self checking test bench

**type** TEST\_VECTOR **is** **RECORD**

W\_EN **:** std\_logic**;**

AL **:** std\_logic\_vector**(**3 **downto** 0**);**

R\_A **:** STD\_LOGIC\_VECTOR**(**log2**(**num\_registers**)-**1 **downto** 0**);**

R\_B **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

W\_A **:** std\_logic\_vector**(**log2**(**num\_registers**)-**1 **downto** 0**);**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

--W\_EN, AL, R\_A, R\_B, W\_A, IMM, SH,

--M\_A, S, M\_in, flags, M\_B,

--M\_DA, OEN

**(** '1'**,** "1000"**,** "00000"**,** "-----"**,** "00001"**,** "----------------"**,** "----"**,**

"----------------"**,** "0---"**,** "----------------"**,** "01010110"**,** "----------------"**,**

"----------------"**,** '0'**),**

**(** '1'**,** "1010"**,** "00000"**,** "-----"**,** "00010"**,** X"0005"**,** "----"**,**

"----------------"**,** "0--1"**,** "----------------"**,** "01010010"**,** "----------------"**,**

"----------------"**,** '0'**),**

**(** '1'**,** "1100"**,** "00001"**,** "-----"**,** "00011"**,** "----------------"**,** "0011"**,**

"----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** "----------------"**,**

"----------------"**,** '0'**),**

**(** '0'**,** "0000"**,** "00011"**,** "00010"**,** "-----"**,** "----------------"**,** "----"**,**

"----------------"**,** "0---"**,** "----------------"**,** "01010010"**,** X"0005"**,**

X"0008"**,** '1'**),**

**(** '1'**,** "----"**,** "-----"**,** "-----"**,** "00101"**,** "----------------"**,** "----"**,**

X"1f1f"**,** "11--"**,** X"CCCC"**,** "--------"**,** "----------------"**,**

X"1f1f"**,** '0'**)**

**);**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_B

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

clk **=>** clk**,**

IMM **=>** IMM**,**

AL **=>** AL**,**

SH **=>** SH**,**

M\_A **=>** M\_A**,**

S **=>** S**,**

flags **=>** flags**,**

M\_B **=>** M\_B**,**

M\_DA **=>** M\_DA**,**

M\_in **=>** M\_in

**);**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

S **<=** test\_vectors**(**i**).**S**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **rising\_edge(**clk**);**

-- Check that the actual outputs are the same as were expecting

-- Have to use std\_match when comparing meta values like '-'

**assert** **std\_match(**flags**,** test\_vectors**(**i**).**flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_B**,** M\_B**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_B**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_B**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]"

**severity** error**;**

-- If there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_B**,** test\_vectors**(**i**).**M\_B**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**))**

**report** lf **&**" [ OK ] Test " **&** integer'**image(**i**)&** " was successful!"

**severity** note**;**

**wait** **until** **falling\_edge(**clk**);**

**end** **loop;**

**wait;**

**end** **process;**

**END;**

**(Continued.)**

IMM **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

SH **:** std\_logic\_vector**(**log2**(**data\_size**)-**1 **downto** 0**)** **;**

M\_A **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

S **:** std\_logic\_vector**(**4 **downto** 1**);**

M\_in **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

flags **:** std\_logic\_vector**(**7 **downto** 0**);**

M\_B **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

M\_DA **:** std\_logic\_vector**(**data\_size**-**1 **downto** 0**);**

OEN **:** std\_logic**;**

**end** **RECORD;**

**type** TEST\_VECTOR\_ARRAY **is** **ARRAY(**NATURAL **RANGE** **<>)** **of** TEST\_VECTOR**;**

**constant** test\_vectors **:** TEST\_VECTOR\_ARRAY **:=** **(**

-- W\_EN, AL, R\_A, R\_B, W\_A, IMM, SH, M\_A, S, M\_in, flags, M\_B, M\_DA,OEN

**(** '1'**,** "1000"**,** "00000"**,** "-----"**,** "00001"**,** "----------------"**,**

"----"**,** "----------------"**,** "0---"**,** "----------------"**,**

"01010110"**,** "----------------"**,** "----------------"**,**

'0'**),**

**(** '1'**,** "1010"**,** "00000"**,** "-----"**,** "00010"**,** X"0005"**,**

"----"**,** "----------------"**,** "0--1"**,** "----------------"**,**

"01010010"**,** "----------------"**,** "----------------"**,**

'0'**),**

**(** '1'**,** "1100"**,** "00001"**,** "-----"**,** "00011"**,** "----------------"**,**

"0011"**,** "----------------"**,** "0---"**,** "----------------"**,**

"01010010"**,** "----------------"**,** "----------------"**,**

'0'**),**

**(** '0'**,** "0000"**,** "00011"**,** "00010"**,** "-----"**,** "----------------"**,**

"----"**,** "----------------"**,** "0---"**,** "----------------"**,**

"01010010"**,** X"0005"**,** X"0008"**,**

'1'**),**

**(** '1'**,** "----"**,** "-----"**,** "-----"**,** "00101"**,** "----------------"**,**

"----"**,** X"1f1f"**,** "11--"**,** X"CCCC"**,**

"--------"**,** "----------------"**,** X"1f1f"**,**

'0'**)**

**);**

**BEGIN**

-- Instantiate the Unit Under Test (UUT)

uut**:** DataPath\_B

**Generic** **Map(**

data\_size **=>** data\_size**,**

num\_registers **=>** num\_registers

**)**

**PORT** **MAP** **(**

R\_A **=>** R\_A**,**

R\_B **=>** R\_B**,**

W\_EN **=>** W\_EN**,**

W\_A **=>** W\_A**,**

clk **=>** clk**,**

IMM **=>** IMM**,**

AL **=>** AL**,**

SH **=>** SH**,**

M\_A **=>** M\_A**,**

S **=>** S**,**

flags **=>** flags**,**

M\_B **=>** M\_B**,**

M\_DA **=>** M\_DA**,**

M\_in **=>** M\_in

**);**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

S **<=** test\_vectors**(**i**).**S**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **rising\_edge(**clk**);**

-- Check that the actual outputs are the same as were expecting

-- Have to use std\_match when comparing meta values like '-'

**assert** **std\_match(**flags**,** test\_vectors**(**i**).**flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_B**,** M\_B**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_B**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_B**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]"

**severity** error**;**

-- If there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_B**,** test\_vectors**(**i**).**M\_B**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**))**

**report** lf **&**" [ OK ] Test " **&** integer'**image(**i**)&** " was successful!"

**severity** note**;**

**wait** **until** **falling\_edge(**clk**);**

**end** **loop;**

**wait;**

**end** **process;**

**END;**

**(Continued.)**

-- Clock process definitions

clk\_process **:process**

**begin**

clk **<=** '0'**;**

**wait** **for** clk\_period**/**2**;**

clk **<=** '1'**;**

**wait** **for** clk\_period**/**2**;**

**end** **process;**

-- Stimulus process

stim\_proc**:** **process**

**begin**

-- hold reset state for 100 ns.

**wait** **for** 100 ns**;**

-- run the test for every set of data

**for** i **in** test\_vectors'**range** **loop**

-- assign test inputs

R\_A **<=** test\_vectors**(**i**).**R\_A**;**

R\_B **<=** test\_vectors**(**i**).**R\_B**;**

W\_EN **<=** test\_vectors**(**i**).**W\_EN**;**

W\_A **<=** test\_vectors**(**i**).**W\_A**;**

IMM **<=** test\_vectors**(**i**).**IMM**;**

AL **<=** test\_vectors**(**i**).**AL**;**

SH **<=** test\_vectors**(**i**).**SH**;**

M\_A **<=** test\_vectors**(**i**).**M\_A**;**

S **<=** test\_vectors**(**i**).**S**;**

M\_in **<=** test\_vectors**(**i**).**M\_in**;**

OEN **<=** test\_vectors**(**i**).**OEN**;**

**wait** **until** **rising\_edge(**clk**);**

-- Check that the actual outputs are the same as were expecting

-- Have to use std\_match when comparing meta values like '-'

**assert** **std\_match(**flags**,** test\_vectors**(**i**).**flags**)**

**report** lf **&** " [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual flags did not equal expected flags."**&**

" Actual [ " **&** to\_bstring**(**flags**)** **&** " ]" **&**

" Expected [ " **&** to\_bstring**(**test\_vectors**(**i**).**flags**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**test\_vectors**(**i**).**M\_B**,** M\_B**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual value to memory did not equal expected value to memory."**&**

" Actual [ " **&** u\_tostr**(**M\_B**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_B**)** **&** " ]"

**severity** error**;**

**assert** **std\_match(**M\_DA **,** test\_vectors**(**i**).**M\_DA**)**

**report** lf **&**" [ERR!] Test " **&** integer'**image(**i**)&** lf **&**

" Actual memory address did not equal expected memory address."**&**

" Actual [ " **&** u\_tostr**(**M\_DA**)** **&** " ]" **&**

" Expected [ " **&** u\_tostr**(**test\_vectors**(**i**).**M\_DA**)** **&** " ]"

**severity** error**;**

-- If there were no isses report that the test was successful

**assert** not **(**

**std\_match(**flags**,** test\_vectors**(**i**).**flags**)** and

**std\_match(**M\_B**,** test\_vectors**(**i**).**M\_B**)** and

**std\_match(**M\_DA**,** test\_vectors**(**i**).**M\_DA**))**

**report** lf **&**" [ OK ] Test " **&** integer'**image(**i**)&** " was successful!"

**severity** note**;**

**wait** **until** **falling\_edge(**clk**);**

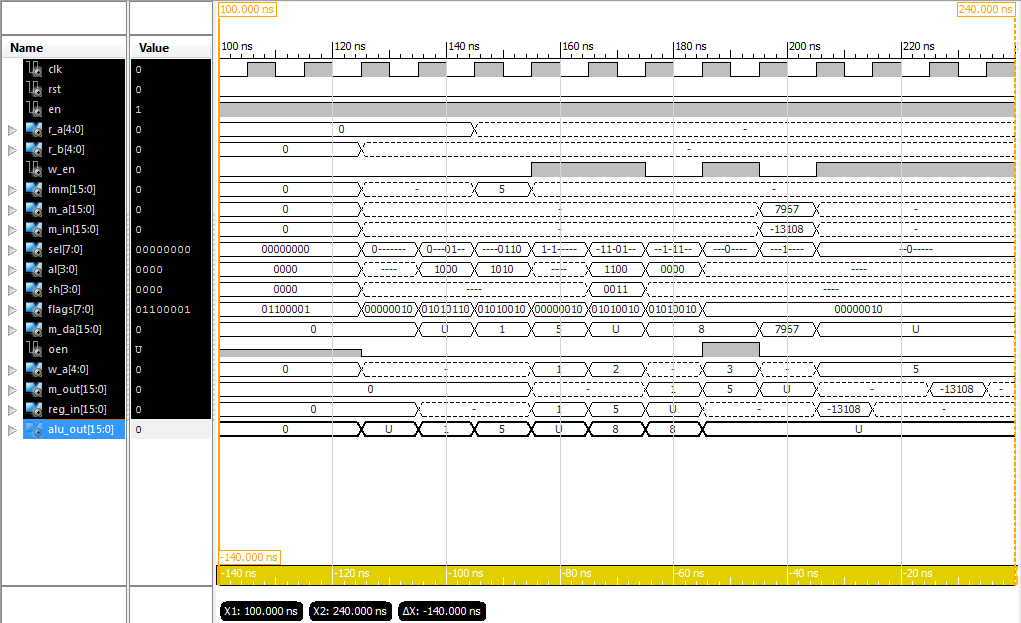
**end** **loop;**

**wait;**

**end** **process;**

**END;**

**Simulations**

****

Screenshot showing entire simulation of Architecture D. The signal reg\_in is the data data that will be written to the register bank.

**iSim Console Output**

ISim P.28xd (signature 0xa0883be4)

This is a Full version of ISim.

Time resolution is 1 ps

WARNING: Simulation object /datapath\_d\_tb/test\_vectors was not traceable in the design for the following reason:

ISim does not yet support tracing of constant and generic multi-dimensional arrays.

Simulator is doing circuit initialization process.

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD.">": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD."<": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD."/=": metavalue detected, returning TRUE

at 0 ps, Instance /datapath\_d\_tb/uut/ALU/ : Warning: NUMERIC\_STD."=": metavalue detected, returning FALSE

Finished circuit initialization process.

at 130 ns(1): Note: [ OK ] Test 0 was successful! (/datapath\_d\_tb/).

at 140 ns(1): Note: [ OK ] Test 1 was successful! (/datapath\_d\_tb/).

at 150 ns(1): Note: [ OK ] Test 2 was successful! (/datapath\_d\_tb/).

at 160 ns(1): Note: [ OK ] Test 3 was successful! (/datapath\_d\_tb/).

at 170 ns(1): Note: [ OK ] Test 4 was successful! (/datapath\_d\_tb/).

at 180 ns(1): Note: [ OK ] Test 5 was successful! (/datapath\_d\_tb/).

at 190 ns(1): Note: [ OK ] Test 6 was successful! (/datapath\_d\_tb/).

at 200 ns(1): Note: [ OK ] Test 7 was successful! (/datapath\_d\_tb/).

at 210 ns(1): Note: [ OK ] Test 8 was successful! (/datapath\_d\_tb/).

ISim>

***All but the first meta value warnings have been removed to improve readability.***

**HDL Synthesis**

=========================================================================

\* HDL Synthesis \*

=========================================================================

Synthesizing Unit <DataPath\_D>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\DataPath\_D.vhd".

data\_size = 16

num\_registers = 32

Found 16-bit 3-to-1 multiplexer for signal <A\_ALU> created at line 52.

Found 16-bit 4-to-1 multiplexer for signal <B\_ALU> created at line 53.

Summary:

inferred 6 Multiplexer(s).

Unit <DataPath\_D> synthesized.

Synthesizing Unit <Reg>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\Reg.vhd".

data\_size = 16

Found 16-bit register for signal <data\_out>.

Summary:

inferred 16 D-type flip-flop(s).

Unit <Reg> synthesized.

Synthesizing Unit <ALU\_param>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\ALU\_param.vhd".

N = 16

Found 16-bit adder for signal <A\_itrn[15]\_B\_itrn[15]\_add\_27\_OUT> created at line 69.

Found 16-bit adder for signal <A\_itrn[15]\_GND\_7\_o\_add\_31\_OUT> created at line 1253.

Found 16-bit subtractor for signal <A\_itrn[15]\_B\_itrn[15]\_sub\_26\_OUT<15:0>> created at line 70.

Found 16-bit subtractor for signal <A\_itrn[15]\_GND\_7\_o\_sub\_30\_OUT<15:0>> created at line 1320.

Found 16-bit shifter rotate right for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_right\_17\_OUT> created at line 3021

Found 16-bit shifter rotate left for signal <A\_itrn[15]\_X\_itrn[30]\_rotate\_left\_19\_OUT> created at line 3012

Found 16-bit shifter arithmetic right for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_right\_21\_OUT> created at line 2982

Found 16-bit shifter logical left for signal <A\_itrn[15]\_X\_itrn[30]\_shift\_left\_23\_OUT> created at line 2973

Found 16-bit 13-to-1 multiplexer for signal <O\_itrn> created at line 42.

Found 16-bit comparator greater for signal <flags<3>> created at line 99

Found 16-bit comparator greater for signal <flags<4>> created at line 100

Summary:

inferred 1 Adder/Subtractor(s).

inferred 2 Comparator(s).

inferred 16 Multiplexer(s).

inferred 4 Combinational logic shifter(s).

Unit <ALU\_param> synthesized.

Synthesizing Unit <regbank>.

Related source file is "E:\University\\_Second Year\Computer Architectures\assesment\MyMicroProccessor\Lab\_2\DataPaths - RegBankAlt\otherRegBank.vhd".

Found 16-bit register for signal <REG02>.

Found 16-bit register for signal <REG03>.

Found 16-bit register for signal <REG04>.

Found 16-bit register for signal <REG05>.

Found 16-bit register for signal <REG06>.

Found 16-bit register for signal <REG07>.

Found 16-bit register for signal <REG08>.

Found 16-bit register for signal <REG09>.

Found 16-bit register for signal <REG10>.

Found 16-bit register for signal <REG11>.

Found 16-bit register for signal <REG12>.

Found 16-bit register for signal <REG13>.

Found 16-bit register for signal <REG14>.

Found 16-bit register for signal <REG15>.

Found 16-bit register for signal <REG16>.

Found 16-bit register for signal <REG17>.

Found 16-bit register for signal <REG18>.

Found 16-bit register for signal <REG19>.

Found 16-bit register for signal <REG20>.

Found 16-bit register for signal <REG21>.

Found 16-bit register for signal <REG22>.

Found 16-bit register for signal <REG23>.

Found 16-bit register for signal <REG24>.

Found 16-bit register for signal <REG25>.

Found 16-bit register for signal <REG26>.

Found 16-bit register for signal <REG27>.

Found 16-bit register for signal <REG28>.

Found 16-bit register for signal <REG29>.

Found 16-bit register for signal <REG30>.

Found 16-bit register for signal <REG31>.

Found 16-bit register for signal <REG01>.

Found 16-bit 32-to-1 multiplexer for signal <A> created at line 30.

Found 16-bit 32-to-1 multiplexer for signal <B> created at line 31.

Summary:

inferred 496 D-type flip-flop(s).

inferred 2 Multiplexer(s).

Unit <regbank> synthesized.

=========================================================================

HDL Synthesis Report

Macro Statistics

# Adders/Subtractors : 1

16-bit addsub : 1

# Registers : 37

16-bit register : 37

# Comparators : 2

16-bit comparator greater : 2

# Multiplexers : 24

1-bit 2-to-1 multiplexer : 6

16-bit 2-to-1 multiplexer : 14

16-bit 3-to-1 multiplexer : 1

16-bit 32-to-1 multiplexer : 2

16-bit 4-to-1 multiplexer : 1

# Logic shifters : 4

16-bit shifter arithmetic right : 1

16-bit shifter logical left : 1

16-bit shifter rotate left : 1

16-bit shifter rotate right : 1

# Xors : 1

16-bit xor2 : 1

=========================================================================