

Flowgraph for Matrix Multiply

$$\begin{pmatrix} \mathbf{T00} & \mathbf{T01} & \mathbf{T02} & \mathbf{T03} \\ \mathbf{T10} & \mathbf{T11} & \mathbf{T12} & \mathbf{T13} \\ \mathbf{T20} & \mathbf{T21} & \mathbf{T22} & \mathbf{T23} \\ \mathbf{T30} & \mathbf{T31} & \mathbf{T32} & \mathbf{T33} \end{pmatrix} \begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \\ \mathbf{W} \end{pmatrix}$$

$$\mathbf{X}' = \mathbf{X} * \mathbf{T00} + \mathbf{Y} * \mathbf{T01} + \mathbf{Z} * \mathbf{T02} + \mathbf{W} * \mathbf{T03}$$

$$\mathbf{Y}' = \mathbf{X} * \mathbf{T10} + \mathbf{Y} * \mathbf{T11} + \mathbf{Z} * \mathbf{T12} + \mathbf{W} * \mathbf{T13}$$

$$\mathbf{Z}' = \mathbf{X} * \mathbf{T20} + \mathbf{Y} * \mathbf{T21} + \mathbf{Z} * \mathbf{T22} + \mathbf{W} * \mathbf{T23}$$

$$\mathbf{W}' = \mathbf{X} * \mathbf{T30} + \mathbf{Y} * \mathbf{T31} + \mathbf{Z} * \mathbf{T32} + \mathbf{W} * \mathbf{T33}$$

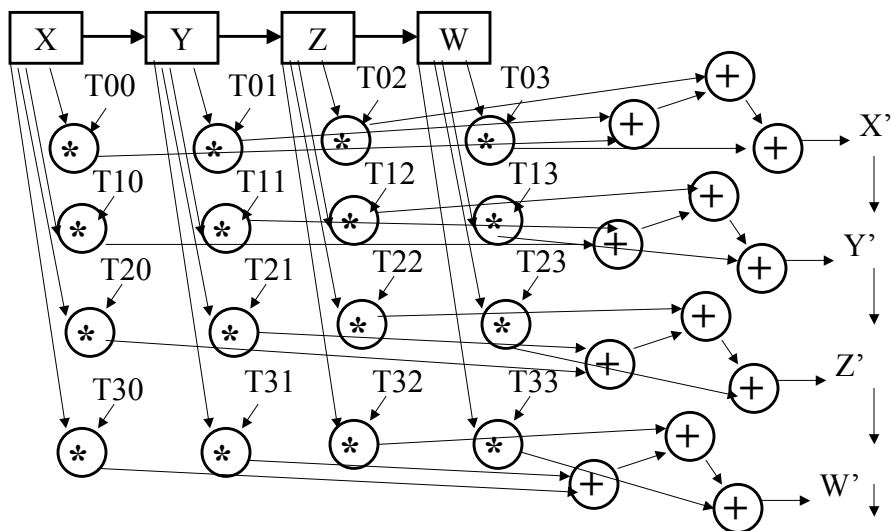
IO Constraint: Single input bus, single output bus

7/11/01

BR

1

Flowgraph for Matrix Multiply (cont)



7/11/01

BR

2

Comments on MM Flowgraph

- **The main thing to notice about the graph is that you don't have to wait until you have X,Y,Z,W before you begin operations**
 - Once you have X, you can do four multiply operations
- **Another thing to note is the symmetry and parallelism available**
 - You could have four parallel datapaths, each one containing a multiplier and an adder, and produce X', Y', Z', W' from these four datapaths

Entity Declaration

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.coordxform_defs.all;

entity coordxform is
port (din: in signed(INWIDTH-1 downto 0);
      clk: in std_logic;
      reset: in std_logic;
      input_rdy : out std_logic; -- ready for a new sample
      tmatrix_rdy : out std_logic; -- ready for a new coeff
      output_rdy : out std_logic;
      dout: out signed(OUTWIDTH-1 downto 0)
);
end coordxform;
```

tmatrix_rdy is
handshaking line
used to input
coefficients.

input_rdy for input
values, *output_rdy*
for output values.

INWIDTH =
OUTWIDTH = 12
bits for example.

architecture behv of coordxfom is

--for this design, it does not make any sense to have different widths,
--so the Internal Width, Output Width is forced to be Input WIDTH

```
constant INTERNALW : integer := 12;  
subtype elemtype is signed(INWIDTH-1 downto 0);  
subtype tmptype is signed(INTERNALW-1 downto 0);  
type matrixarray is array ( natural range <>) of elemtype;  
type tmparray is array (natural range <>) of tmptype;
```

```
begin  
main:process  
  variable coeffs : matrixarray(0 to 15) ;  
  variable products : tmparray(0 to 3);  
  variable colsums : tmparray(0 to 3);  
  variable coordinate : elemtype;  
  variable final_sum : tmptype;
```

```
begin  
  reset_loop: loop  
    .....
```

coefficient registers

tmp regs for products

tmp regs for column
sums

7/11/01

BR

5

Model Structure

- Reset loop will be used to load 16 coefficient values
 - tmatrix_rdy handshaking line toggled for each coefficient
- Main loop consists of 16 superstates
 - 8 superstates for inputting X,Y,Z,W values over *din* bus – two superstates for each variable so that input_rdy can be toggled
 - 8 superstates for outputting X', Y', Z', W' over *dout* bus – two superstates for each variable so that output_rdy can be toggled
 - Matrix calculations spread throughout superstates

7/11/01

BR

6

```

begin
  reset_loop: loop
    --## RESET SuperState
    -- initialize variables
    output_rdy <= '0'; input_rdy <= '0'; tmatrix_rdy <= '0';
    wait until clk'event and clk = '1';
    if (reset = '1') then exit reset_loop; end if;
    tmatrix_rdy <= '1';
    wait until clk'event and clk = '1';
    if (reset = '1') then exit reset_loop; end if;
    init_loop: for j in 0 to 14 loop
      coeffs(j) := din; tmatrix_rdy <= '0';
      wait until clk'event and clk='1';
      if (reset = '1') then exit reset_loop; end if;
      tmatrix_rdy <= '1';
      wait until clk'event and clk='1';
      if (reset = '1') then exit reset_loop; end if;
    end loop;
    -- now input last coefficient
    coeffs(15) := din; tmatrix_rdy <= '0';
    wait until clk'event and clk='1';
    if (reset = '1') then exit reset_loop; end if;
    input_rdy <= '1';
    wait until clk'event and clk='1';
    if (reset = '1') then exit reset_loop; end if;
  l1: loop -- sample loop

```

Reset Loop

loop for inputting
Coefficients, toggling
tmatrix_rdy

Assert *input_rdy* for
first input value

7/11/01 BR 7

```

l1: loop -- sample loop
  -- Super State #0 GET X
  coordinate := din; input_rdy <= '0'; output_rdy <= '0';
  products(0) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(0)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));
  products(1) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(4)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));
  products(2) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(8)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));
  products(3) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(12)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));

  colsums(0) := products(0);
  colsums(1) := products(1);
  colsums(2) := products(2);
  colsums(3) := products(3);

  wait until clk'event and clk='1';
  if (reset = '1') then exit reset_loop; end if;
  -- Super State #0a -- raise handshaking line
  input_rdy <= '1';
  wait until clk'event and clk='1';
  if (reset = '1') then exit reset_loop; end if;

```

Get X value, do X value products

Input values are assumed in
fixed point notation and
between (1.0, 0.0].

Multiplies are most significant
6 bits x 6 bits = 12 bit result

7/11/01 BR 8

Get Y value, do Y value products

```
-- Super State #1 GET Y
coordinate := din; input_rdy <= '0';
products(0) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(1)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));

products(1) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(5)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));

products(2) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(9)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));

products(3) := coordinate(INWIDTH-1 downto (INWIDTH -
INTERNALW/2)) * coeffs(13)(INWIDTH-1 downto (INWIDTH - INTERNALW/2));

colsums(0) := colsums(0) + products(0);
colsums(1) := colsums(1) + products(1);
colsums(2) := colsums(2) + products(2);
colsums(3) := colsums(3) + products(3);

wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;

-- Super State #1a -- raise handshaking line
input_rdy <= '1';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
```

Accumulate column sums for X', Y', Z', W' values.

Superstates for Z, W values are similar, not shown.

7/11/01

BR

9

Output X', Y' values

```
-- Super State #4 OUTPUT X'
output_rdy <= '1';
dout <= colsums(0)(INTERNALW-1 downto (INTERNALW - OUTWIDTH));
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;

-- Super State #4a -- lower handshaking line
output_rdy <= '0';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;

-- Super State #5 OUTPUT Y'
output_rdy <= '1';
dout <= colsums(1)(INTERNALW-1 downto (INTERNALW - OUTWIDTH));
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;

-- Super State #5a -- lower handshaking line
output_rdy <= '0';
wait until clk'event and clk='1';
if (reset = '1') then exit reset_loop; end if;
```

Toggle *output_rdy*

Toggle *output_rdy*

Code for outputting Z', W' is similar, not shown.

7/11/01

BR

10

Minimum Resource Implementation (I0_p0)

Clock period 1000.00

Loop timing information:

```
main.....55 cycles (cycles 0 - 55)
reset_loop.....55 cycles (cycles 0 - 55)
l1.....21 cycles (cycles 34 - 55)
```

Register Types

```
=====
6-bit register.....11
12-bit register.....5
192-bit register.....1
```

Operator Types

```
=====
(6_6->12)-bit DW02_mult.....1
(12_12->12)-bit DW01_add.....1
```

I/O Ports

```
=====
1-bit registered output port.....3
12-bit input port.....1
12-bit registered output port.....1
```

7/11/01

BR

11

Constrain Loop time to 16 cycles (I16_p16)

Clock period 1000.00

Loop timing information:

```
main.....50 cycles (cycles 0 - 50)
reset_loop.....50 cycles (cycles 0 - 50)
l1.....16 cycles (cycles 34 - 50)
```

Register Types

```
=====
6-bit register.....10
12-bit register.....4
192-bit register.....1
```

Operator Types

```
=====
(6_6->12)-bit DW02_mult.....2
(12_12->12)-bit DW01_add.....2
```

I/O Ports

```
=====
1-bit registered output port.....3
12-bit input port.....1
12-bit registered output port.....1
```

7/11/01

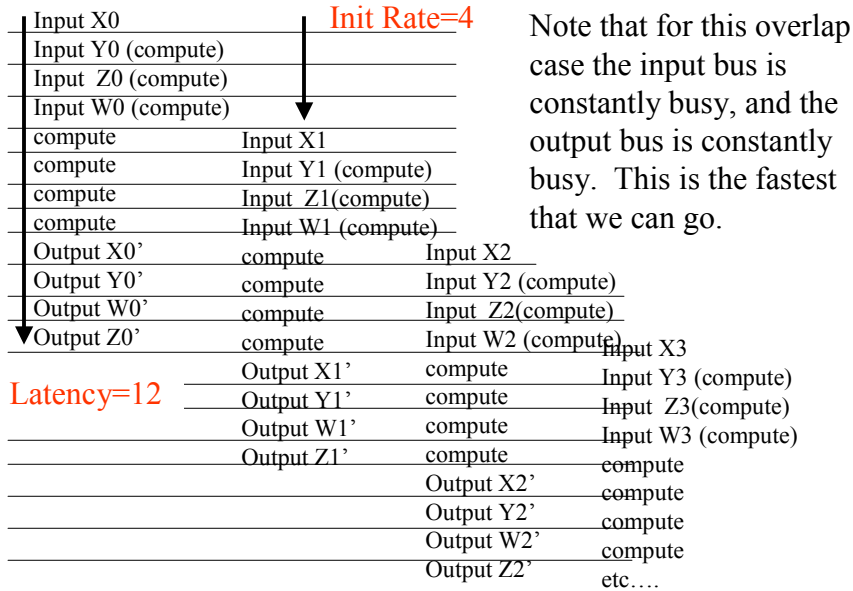
BR

12

Additional multiplier
and adder required.

Because we toggling
handshake lines, can't
do this any faster.

Drop toggling of handshake lines and add pipelining



No toggling of handshake lines, Lat = 12, Init Rate = 4

Clock period 1000.00
 Loop timing information:
 main.....46 cycles (cycles 0 - 46)
 reset_loop.....46 cycles (cycles 0 - 46)
 l1.....(initiation interval)..4 cycles
 (pipeline latency)..12 cycles (cycles 34 - 46)

Register Types
 =====
 6-bit register.....6
 12-bit register.....8
 192-bit register.....1

Operator Types
 =====
 (6_6->12)-bit DW02_mult.....4
 (12_12->12)-bit DW01_add.....3

I/O Ports
 =====
 1-bit registered output port.....3
 12-bit input port.....1
 12-bit registered output port.....1

Input/Output Busses are now the limiting factor, can't do this with any faster initiation rate because input/output busses are constantly busy.