Lab A

Digital Filter: 10.1-10.3

Team1 吳靖崴 施承志 范勝禹

https://github.com/strongshih/ACA2021_Lab1_Team1

Outline

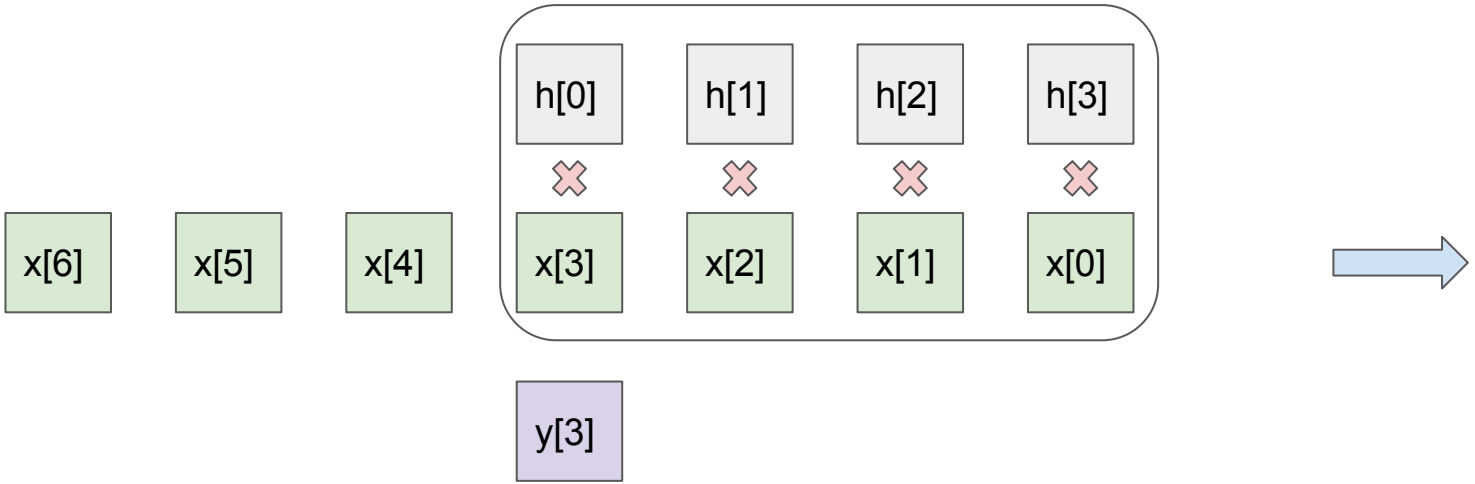
- FIR filter
- Illustrate filter with rolled/unrolled MAC loop
- Demonstrate various filter design in the example code
- Design and explain the structure of the Systolic array filter

Outline

- ***FIR filter***
- Illustrate filter with rolled/unrolled MAC loop
- Demonstrate various filter design in the example code
- Design and explain the structure of the Systolic array filter

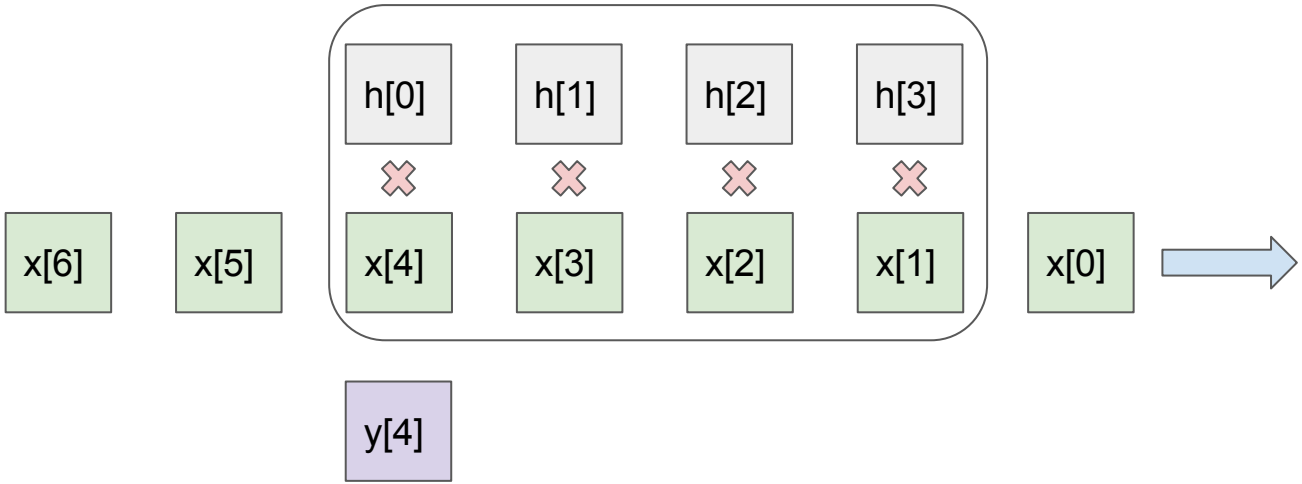
FIR filter

- Finite Impulse Response filter
- $y[n] = \Sigma_{k=0}^{N} h[k]x[n-k]$
- Here, 4 coefficients, h[0], h[1], h[2], h[3]



FIR filter

- Finite Impulse Response filter
- $y[n] = \Sigma_{k=0}^{N} h[k]x[n-k]$
- Here, 4 coefficients, h[0], h[1], h[2], h[3]



Outline

- FIR filter
- ***Illustrate filter with rolled/unrolled MAC loop***
- Demonstrate various filter design in the example code
- Design and explain the structure of the Systolic array filter

FIR filter

```
void fir_filter ( ap_fixed<8,1> *x,
                  ap_fixed<8,1> h[4],
                  ap_fixed<19,4> *y){
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

Rolled FIR filter

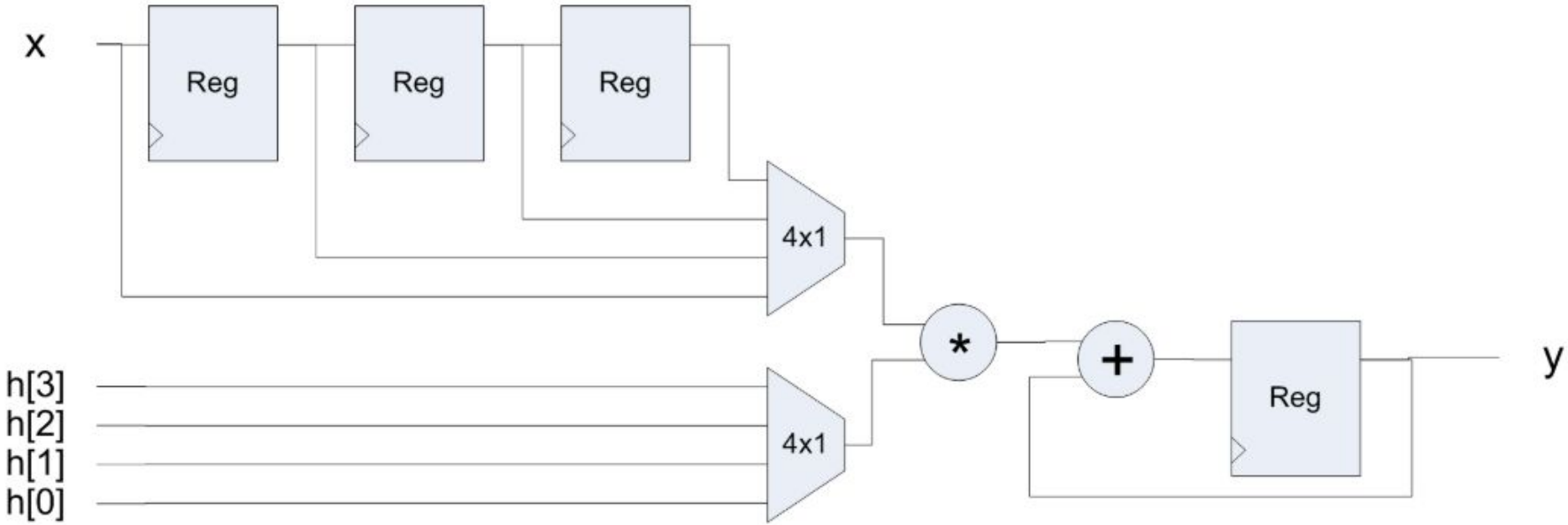


Illustration 138: FIR Filter with External Coefficients

Rolled FIR filter

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
r_V_2_fu_206_p2	*	0	0	41	8	8
i_fu_140_p2	+	0	0	12	3	1
temp_V_fu_224_p2	+	0	0	26	19	19

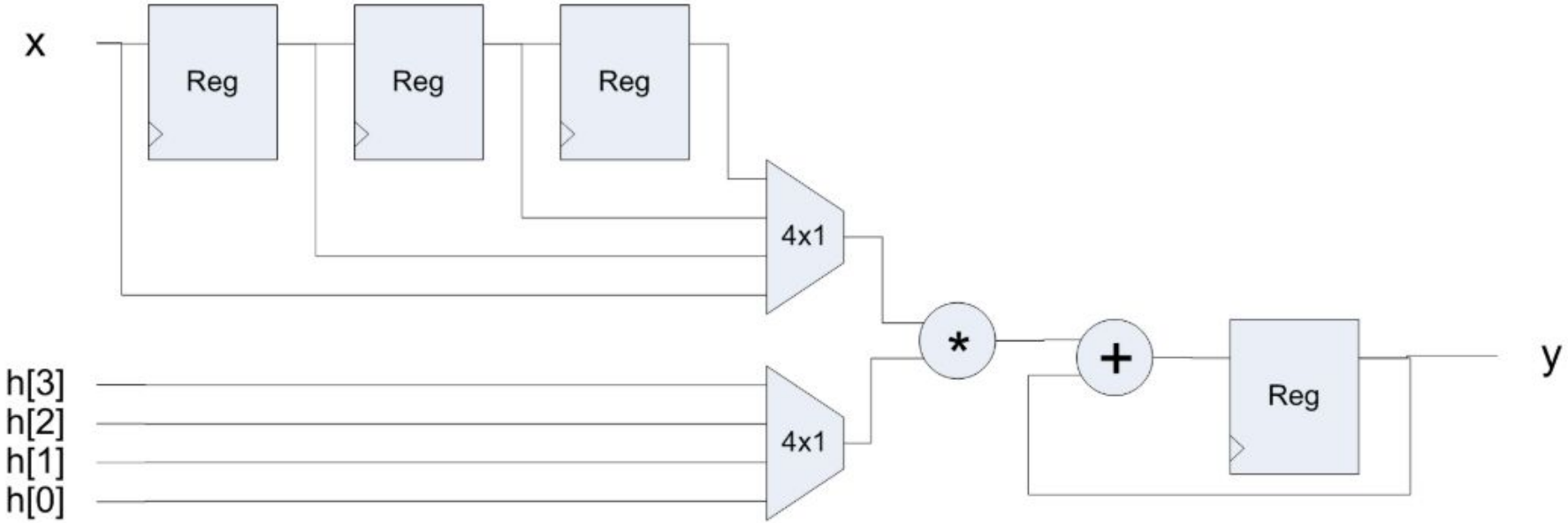


Illustration 138: FIR Filter with External Coefficients

Unrolled FIR filter

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
mul_ln1118_1_fu_150_p2	*	0	0	41	8	8
mul_ln1118_2_fu_179_p2	*	0	0	41	8	8
mul_ln1118_3_fu_208_p2	*	0	0	41	8	8
mul_ln1118_fu_121_p2	*	0	0	41	8	8
add_ln703_1_fu_236_p2	+	0	0	25	18	18
add_ln703_fu_226_p2	+	0	0	25	18	18
y_V	+	0	0	26	19	19

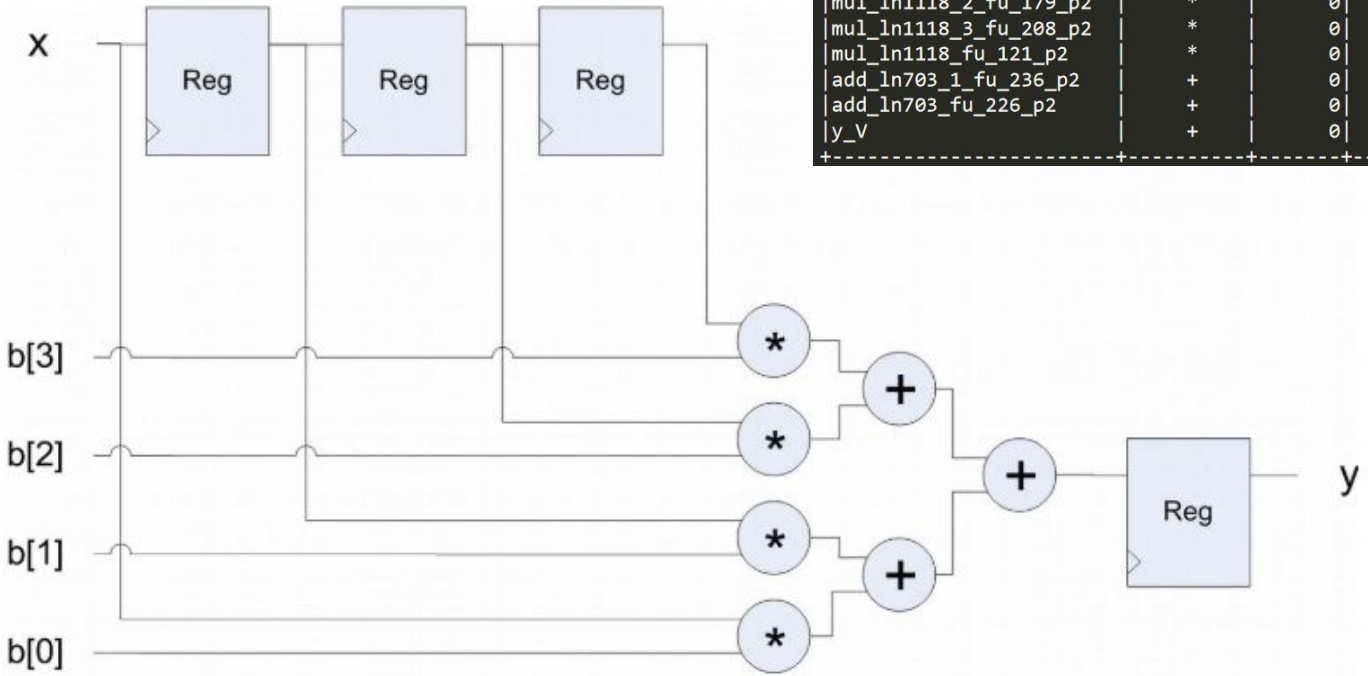


Illustration 139: Fully Parallel FIR Filter with External Coefficients

Rolled v.s. Unrolled

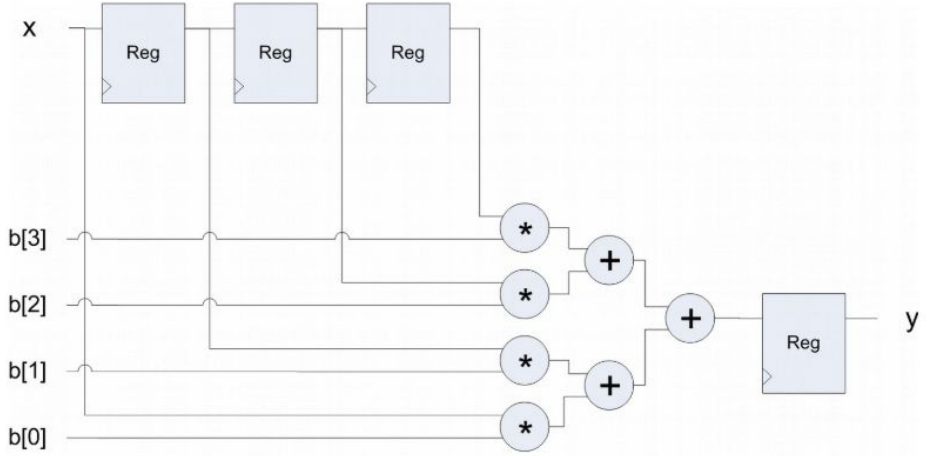


Illustration 139: Fully Parallel FIR Filter with External Coefficients

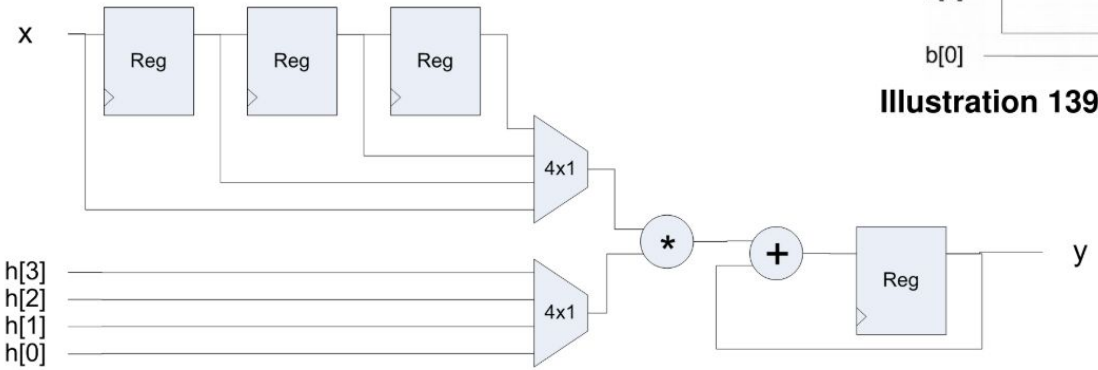


Illustration 138: FIR Filter with External Coefficients

	Latency (cycle)	FF	LUT
Rolled	16	108	295
Unrolled	8	74	402

Outline

- FIR filter
- Illustrate filter with rolled/unrolled MAC loop
- ***Demonstrate various filter design in the example code***
- Design and explain the structure of the Systolic array filter

FIR filter

* Expression:

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
mul_ln1118_1_fu_150_p2	*	0	0	41	8	8
mul_ln1118_2_fu_179_p2	*	0	0	41	8	8
mul_ln1118_3_fu_208_p2	*	0	0	41	8	8
mul_ln1118_fu_121_p2	*	0	0	41	8	8
add_ln703_1_fu_236_p2	+	0	0	25	18	18
add_ln703_fu_226_p2	+	0	0	25	18	18
y_V	+	0	0	26	19	19

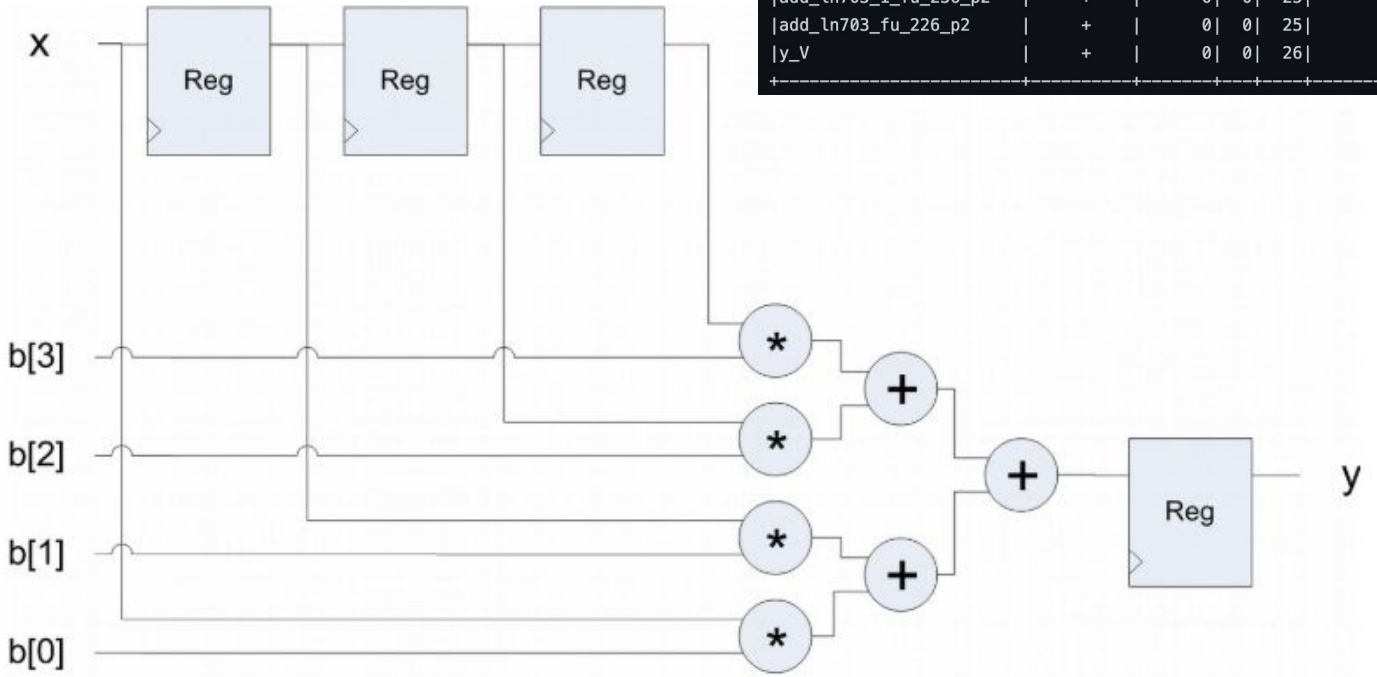


Illustration 139: Fully Parallel FIR Filter with External Coefficients

Constant Coefficient FIR filter

```
void fir_filter ( ap_fixed<8,1> *x,
                  ap_fixed<8,1> h[4],
                  ap_fixed<19,4> *y){
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

```
#include "fir_filter.h"

void fir_filter_const (ap_fixed<8,1> *x,
                       ap_fixed<19,4> *y){
  const ap_fixed<8,1> h[4] = {0.1, 0.2, 0.3, 0.4};
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

Constant Coefficient FIR filter

```
* DSP48E:
+-------------------------+-----------------------+--------------+
|         Instance        |         Module        |  Expression  |
+-------------------------+-----------------------+--------------+
|fir_filter_const_bkb_U6  |fir_filter_const_bkb   | i0 + i1 * i2 |
|fir_filter_const_cud_U7  |fir_filter_const_cud   | i0 + i1 * i2 |
+-------------------------+-----------------------+--------------+
```

```
* Expression:
+-------------------------+-----------+-------+---+----+------------+------------+
|     Variable Name       | Operation| DSP48E| FF| LUT| Bitwidth P0| Bitwidth P1|
+-------------------------+-----------+-------+---+----+------------+------------+
|mul_ln703_2_fu_77_p2     |     *     |      0|  0|  41|           8|           7|
|add_ln703_2_fu_154_p2    |     +     |      0|  0|  24|          17|          17|
|sub_ln1118_fu_111_p2     |     -     |      0|  0|  17|          13|          13|
+-------------------------+-----------+-------+---+----+------------+------------+
```

```
void fir_filter ( ap_fixed<8,1> *x,
                  ap_fixed<8,1> h[4],
                  ap_fixed<19,4> *y){
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

```
#include "fir_filter.h"

void fir_filter_const (ap_fixed<8,1> *x,
                       ap_fixed<19,4> *y){
  const ap_fixed<8,1> h[4] = {0.1, 0.2, 0.3, 0.4};
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

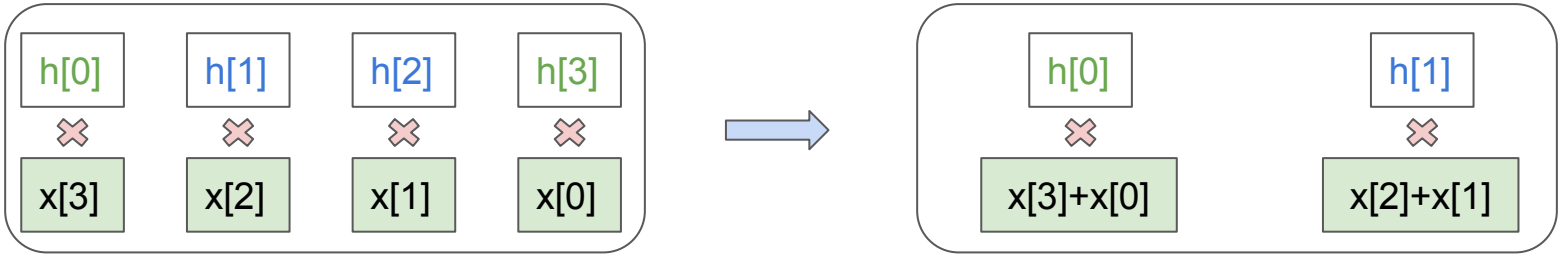
Loadable Coefficients FIR filter

```
void fir_filter ( ap_fixed<8,1> *x,
                  ap_fixed<8,1> h[4],
                  ap_fixed<19,4> *y){
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h[i]*regs[i];
  }
  *y = temp;
}
```

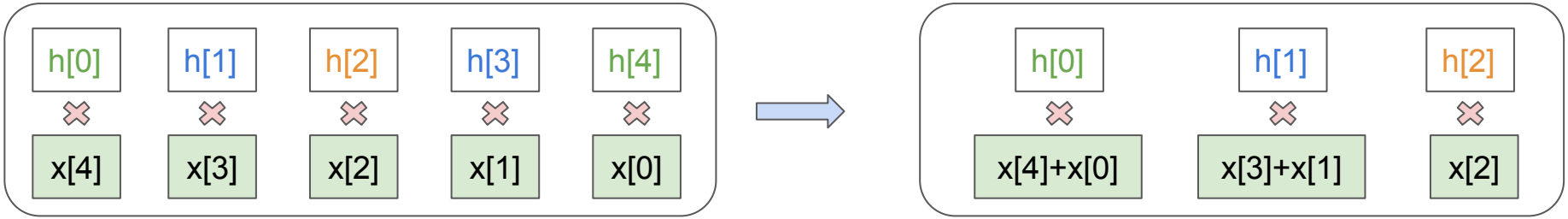
```
void fir_filter_ld (ap_fixed<8,1> *x,
                    ap_fixed<8,1> h[4],
                    ap_fixed<19,4> *y,
                    bool &ld){
  static shift_class<ap_fixed<8,1>,4> regs;
  ap_fixed<19,4> temp = 0;
  static ap_fixed<8,1> h_int[4];
  if(ld==true)
    for(int i=0;i<4;i++)
      h_int[i] = h[i];
  regs << *x;
  MAC:for (int i = 0; i<4; i++) {
    temp += h_int[i]*regs[i];
  }
  *y = temp;
}
```

Symmetric Cofficient FIR filter

- Even Symmetric: h[4] = {0.3, 0.9, 0.9, 0.3}



- Odd Symmetric: h[5] = {0.3, 0.9, 1.0, 0.9, 0.3}



Symmetric Cofficient FIR filter - Even Symmetric

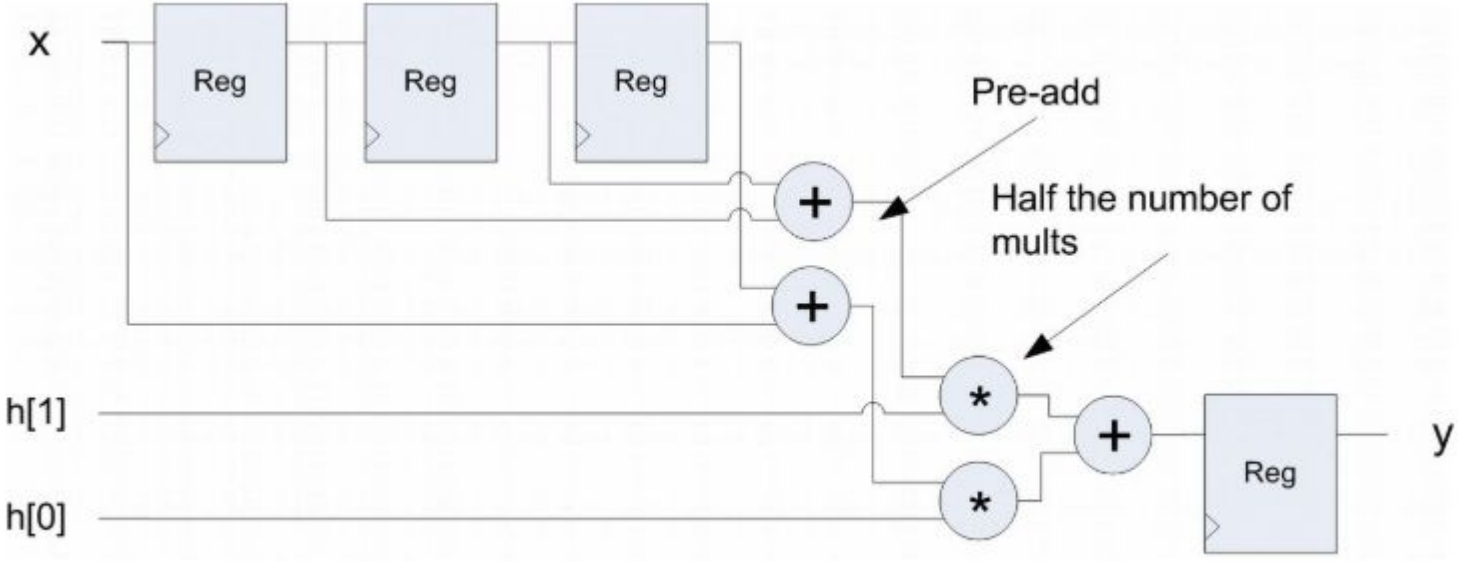


Illustration 140: FIR Filter with Even Symmetry

Symmetric Cofficient FIR filter - Even Symmetric

```
* DSP48E:
+-------------------------+---------------------+----------------+
|         Instance        |        Module       |   Expression   |
+-------------------------+---------------------+----------------+
|fir_filter_even_abkb_U6  |fir_filter_even_abkb | i0 * (i1 + i2) |
|fir_filter_even_abkb_U7  |fir_filter_even_abkb | i0 * (i1 + i2) |
+-------------------------+---------------------+----------------+
```

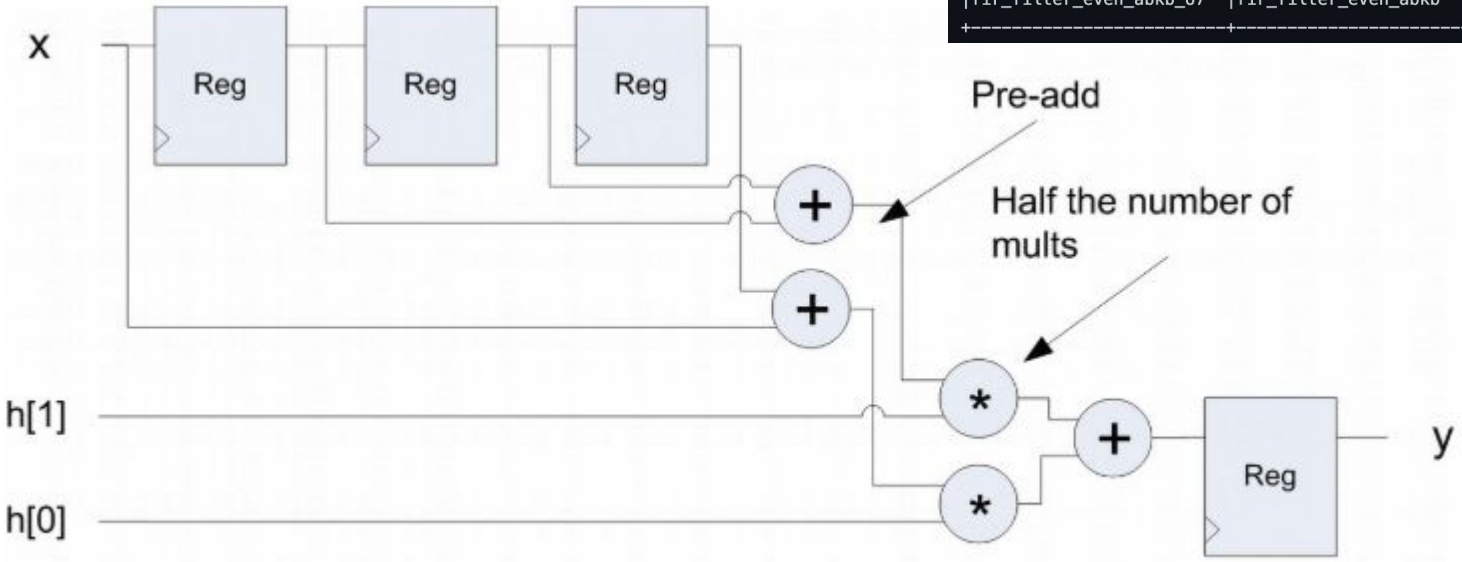


Illustration 140: FIR Filter with Even Symmetry

Symmetric Cofficient FIR filter - Odd Symmetric

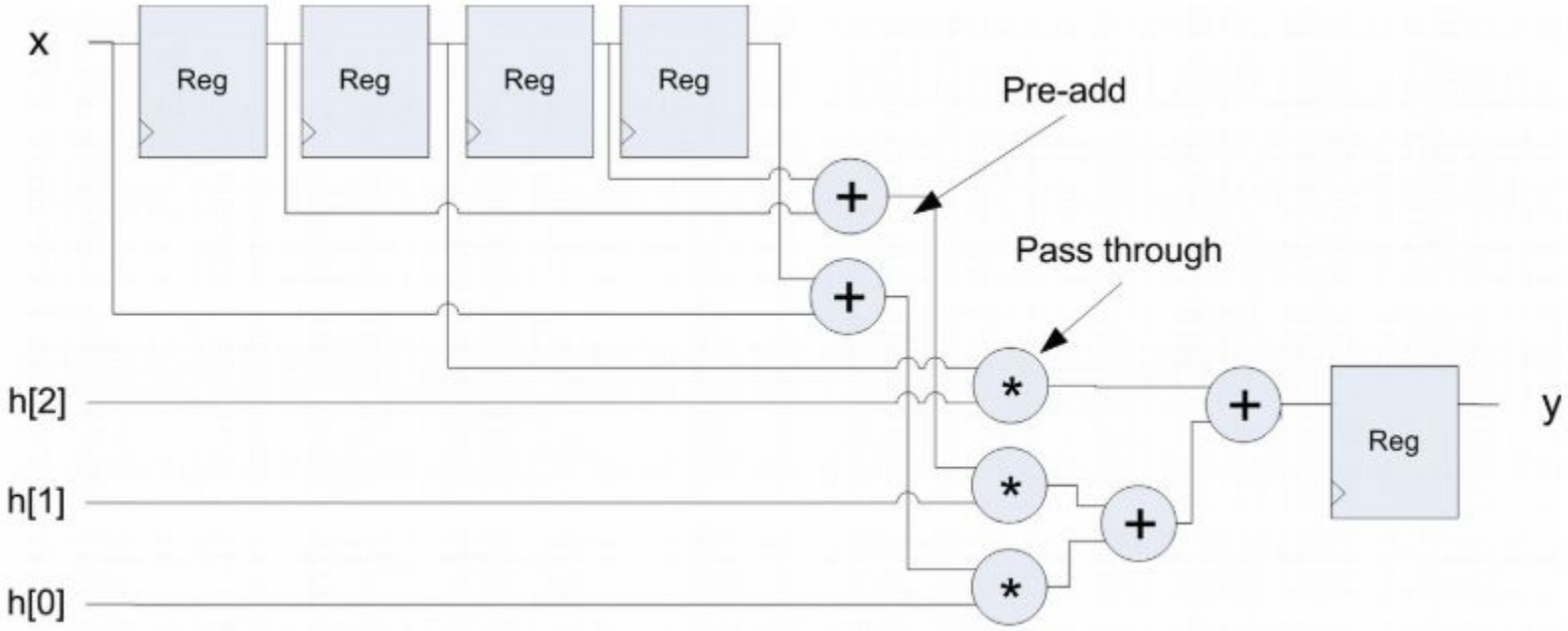


Illustration 141: FIR Filter with Odd Symmetry

Transposed FIR filter

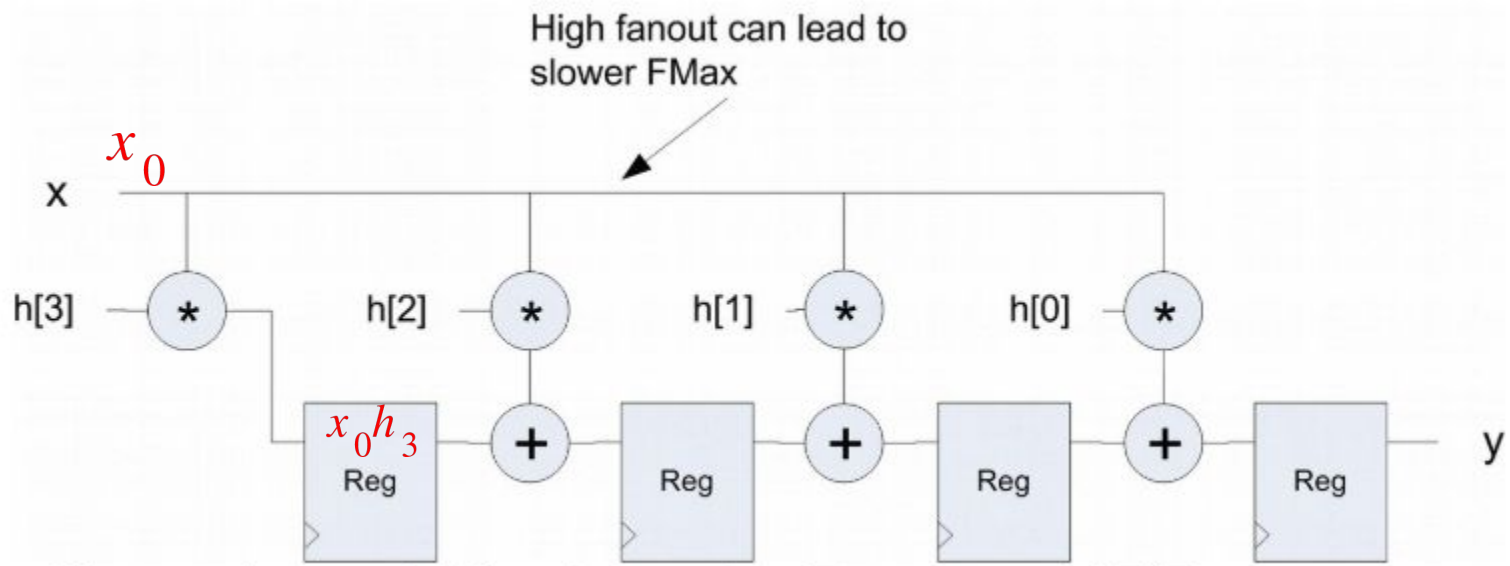


Illustration 143: Hardware for Transposed FIR

Transposed FIR filter

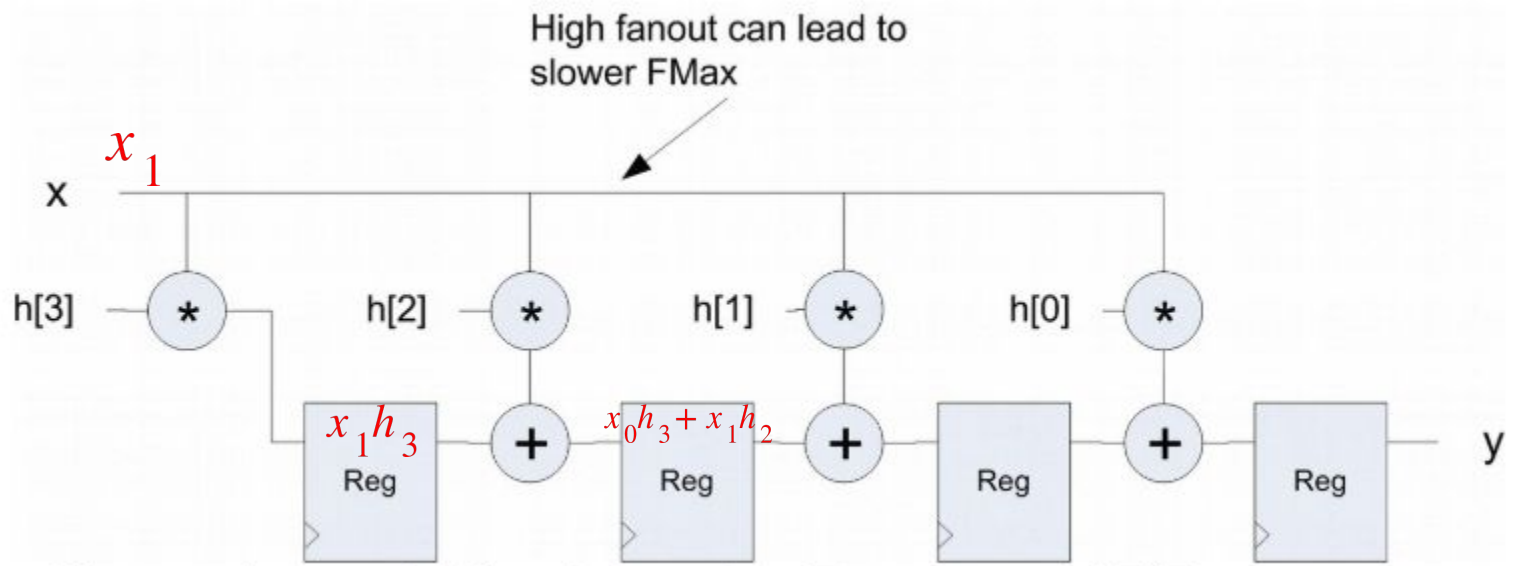


Illustration 143: Hardware for Transposed FIR

Transposed FIR filter

* Expression:

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
mul_ln1118_1_fu_139_p2	*	0	0	41	8	8
mul_ln1118_2_fu_181_p2	*	0	0	41	8	8
mul_ln1118_3_fu_215_p2	*	0	0	41	8	8
mul_ln1118_fu_107_p2	*	0	0	41	8	8
add_ln703_1_fu_157_p2	+	0	0	25	18	18
add_ln703_2_fu_199_p2	+	0	0	25	18	18
y_V	+	0	0	26	19	19

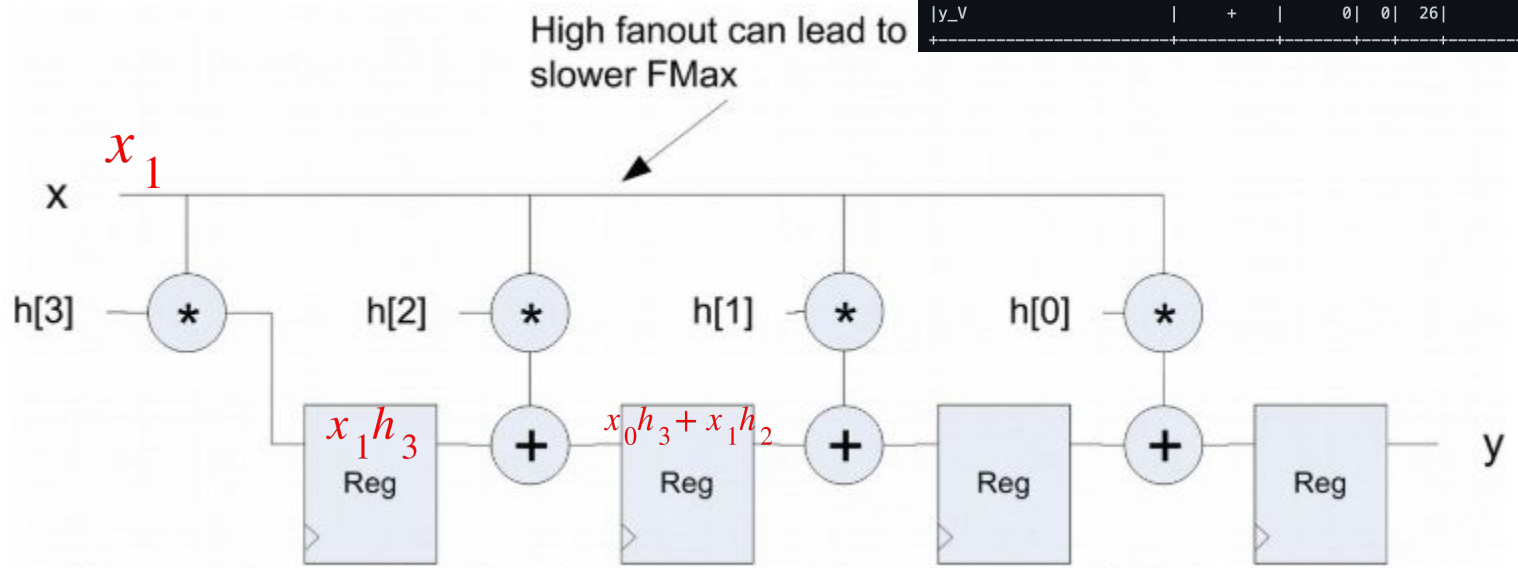


Illustration 143: Hardware for Transposed FIR

Comparison

```
+-----------------------+-------------+---------+-------+---+-----+-----+
|        Instance       |    Module   | BRAM_18K| DSP48E| FF| LUT | URAM|
+-----------------------+-------------+---------+-------+---+-----+-----+
|grp_operator_ls_fu_104 |operator_ls  |        0|      0|  5|  105|    0|
+-----------------------+-------------+---------+-------+---+-----+-----+
```

```
static shift_class<ap_fixed<8,1>,4>
```

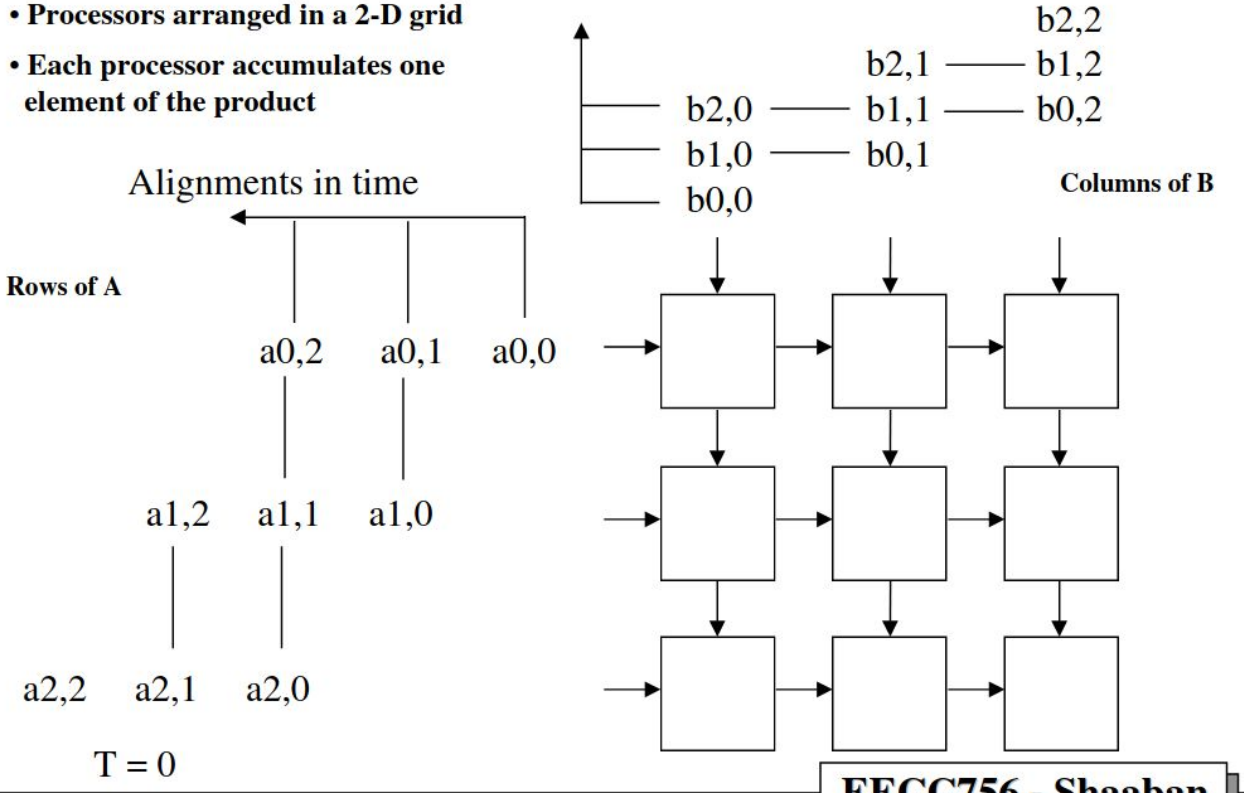
	Latency(ns)	FF	LUT	DSP48E
baseline	80	74	402	0
const coef	80	72	239	2
load coef	80	83	408	0
even symmetric	80	76	158	2
odd symmetric	150	110	267	2
transposed	20	69	291	0

Outline

- FIR filter
- Illustrate filter with rolled/unrolled MAC loop
- Demonstrate various filter design in the example code
- ***Design and explain the structure of the Systolic array filter***

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

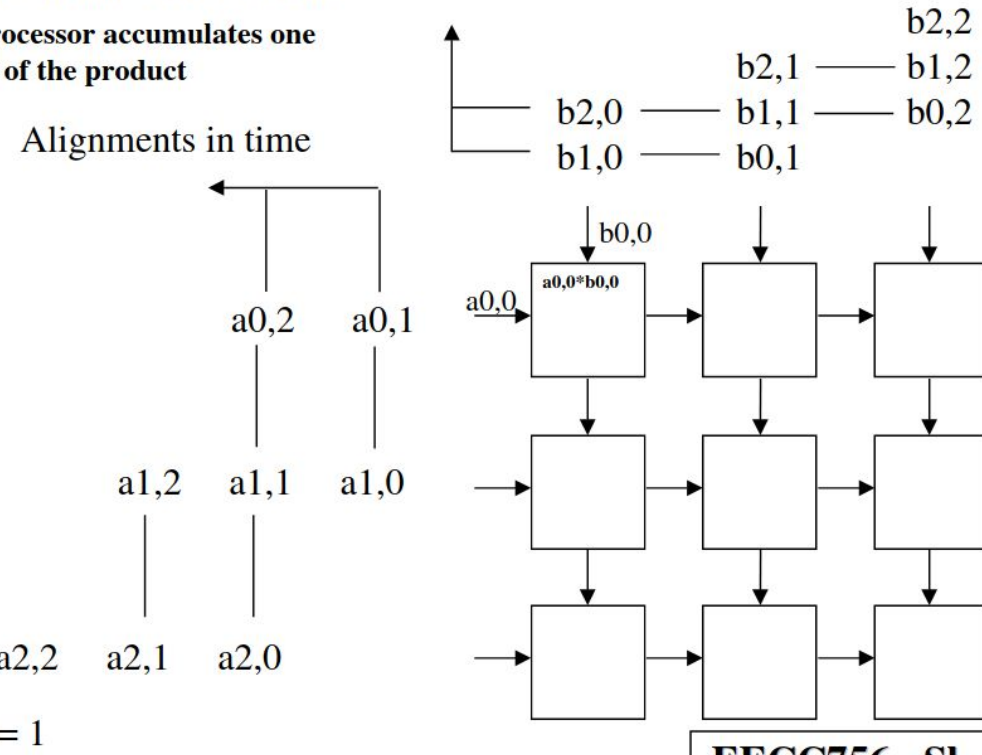


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

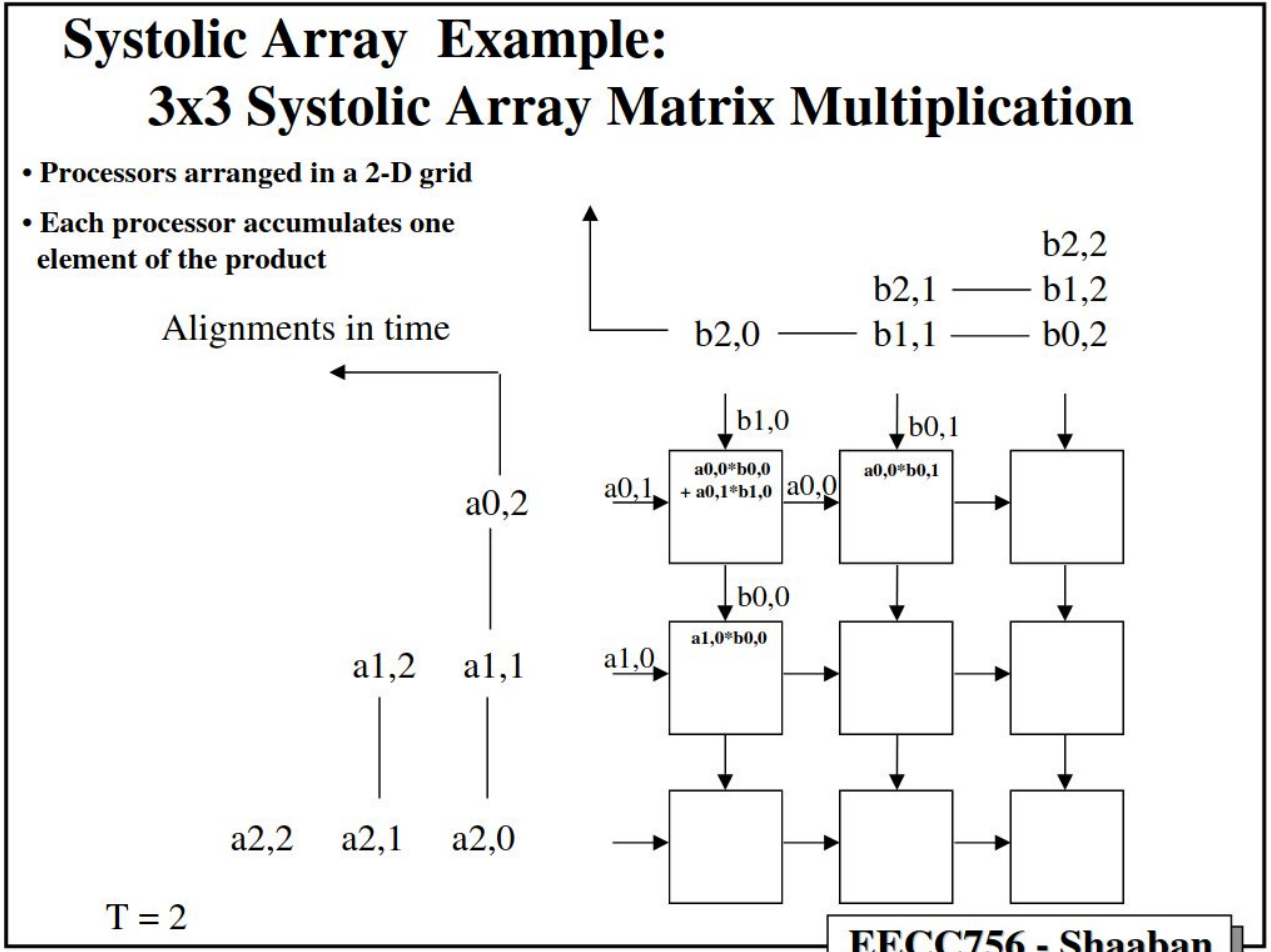


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

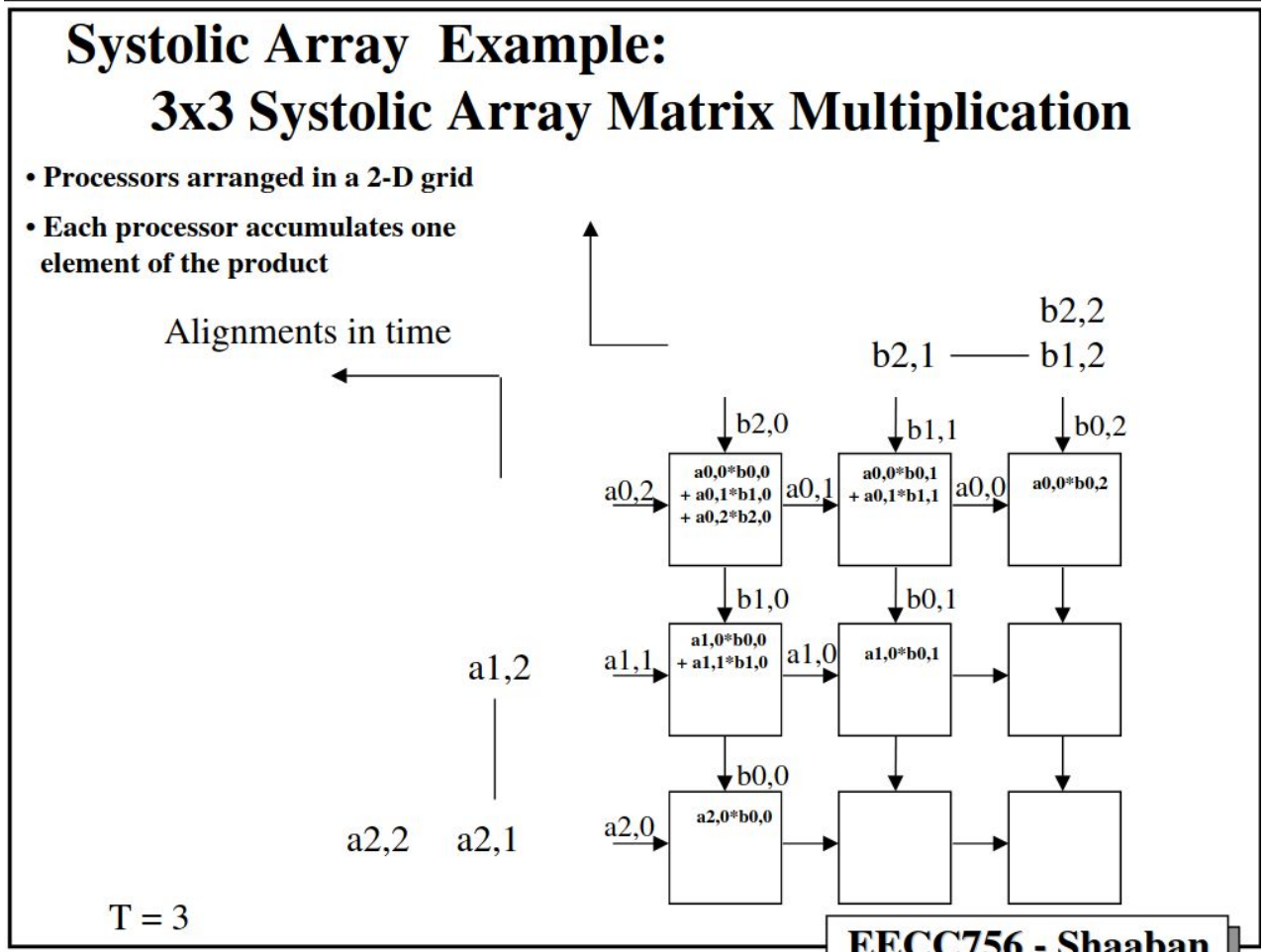


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

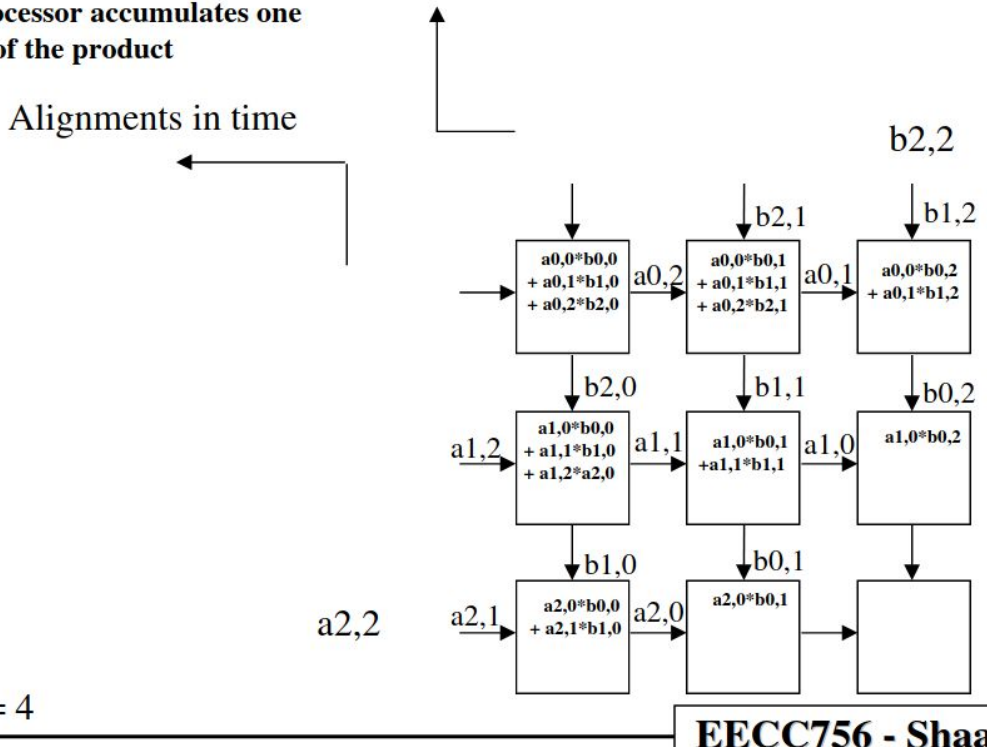


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

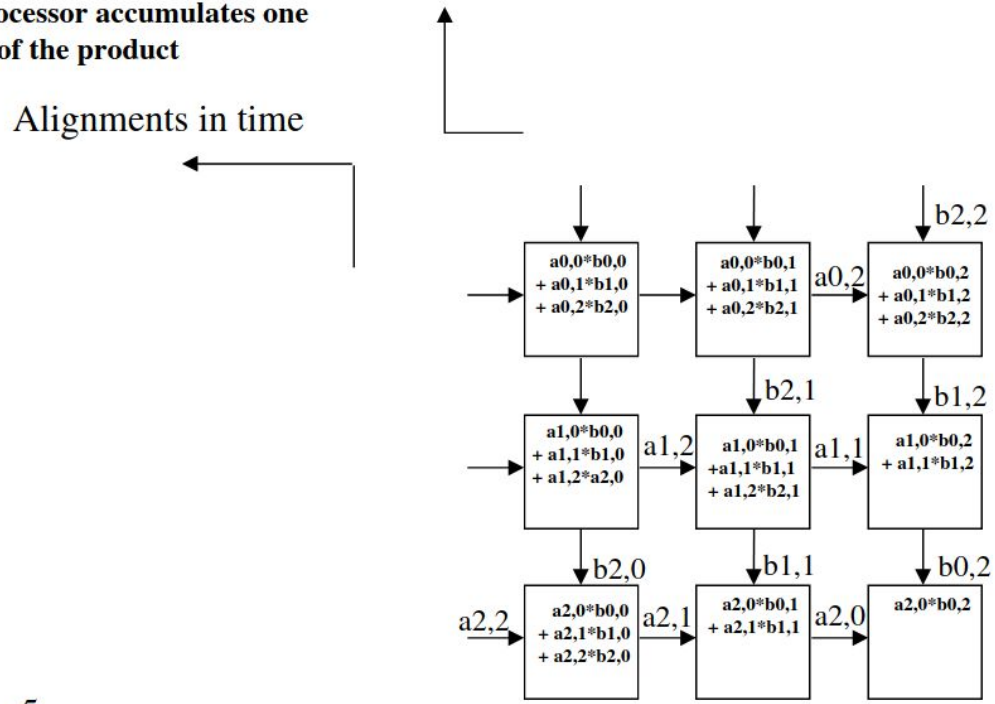


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

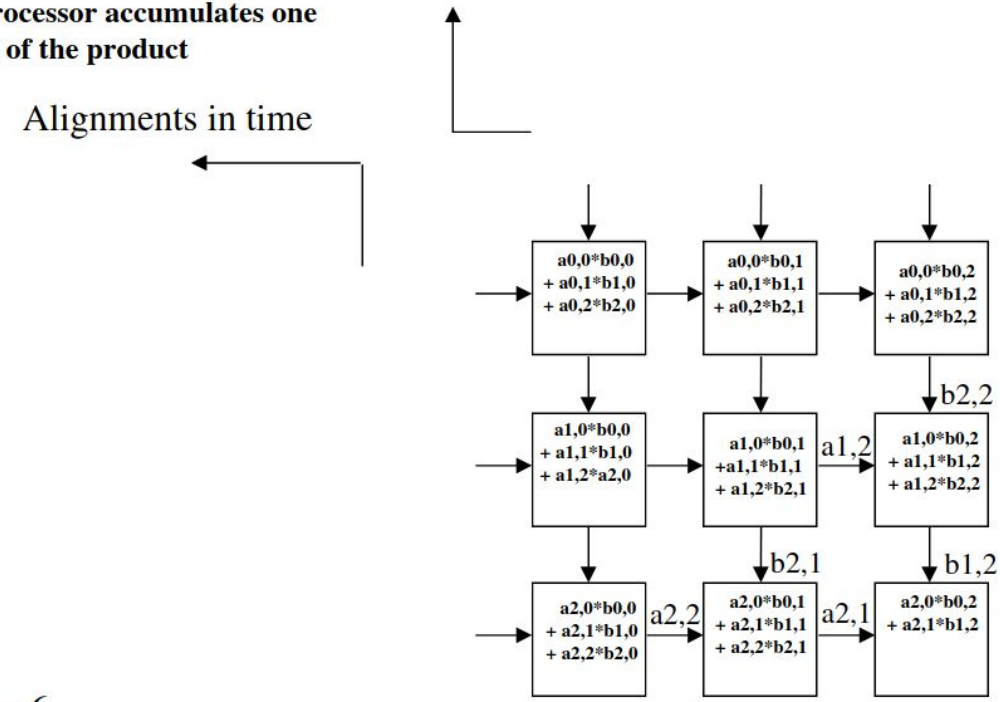


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product

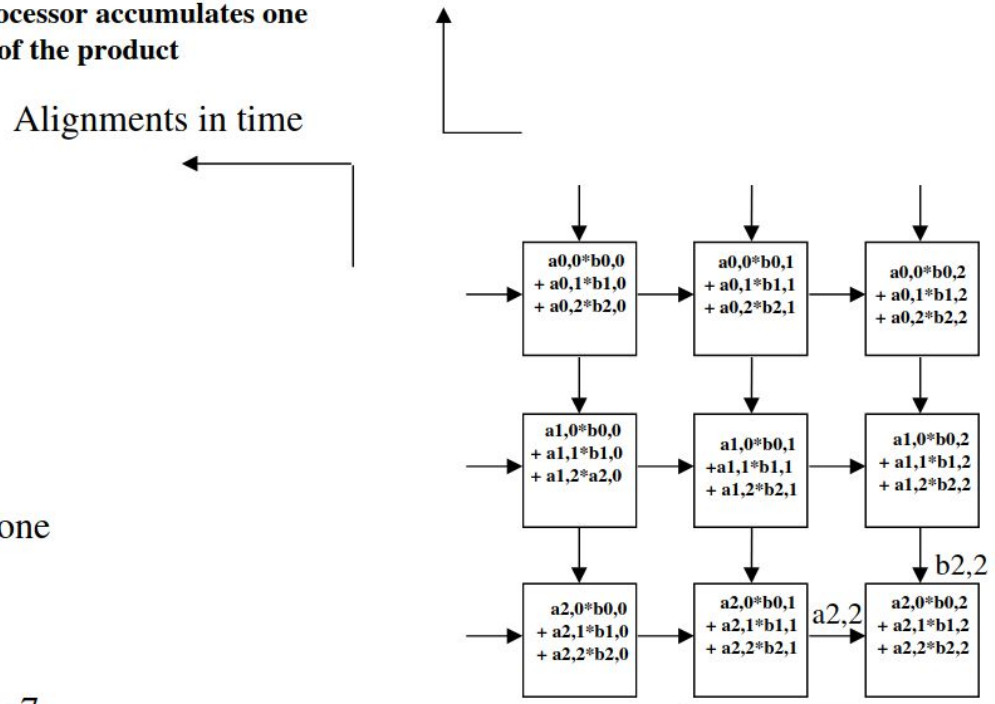


Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systolic Array Example:
3x3 Systolic Array Matrix Multiplication

- Processors arranged in a 2-D grid
- Each processor accumulates one element of the product



Example source: http://www.cs.hmc.edu/courses/2001/spring/cs156/

ref : http://web.cecs.pdx.edu/~mperkows/temp/May22/0020.Matrix-multiplication-systolic.pdf

Systotic FIR Implementation Overview

- previous FIR definition
- FIR definition here

$$y[n] = \Sigma_{k=0}^{N} h[k]x[n-k]$$

$$y[n] = \sum_{k=0}^{N} h[k]x[n+k]$$

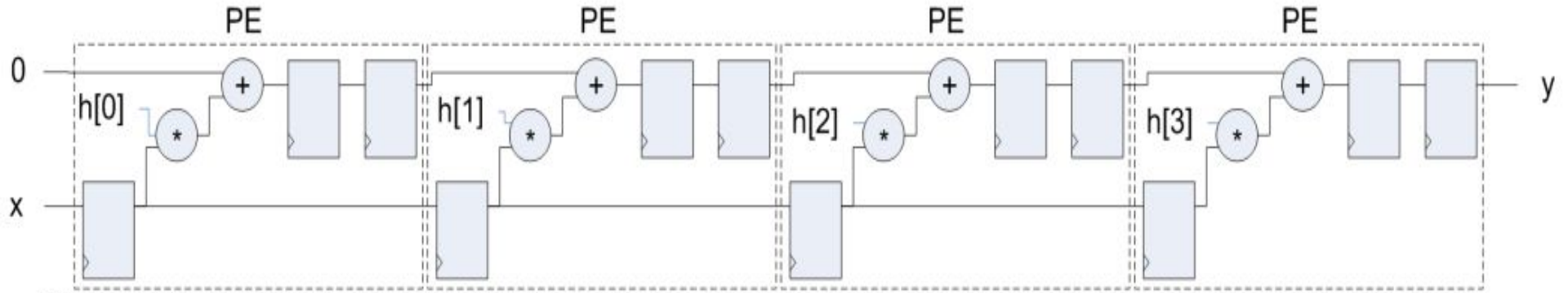
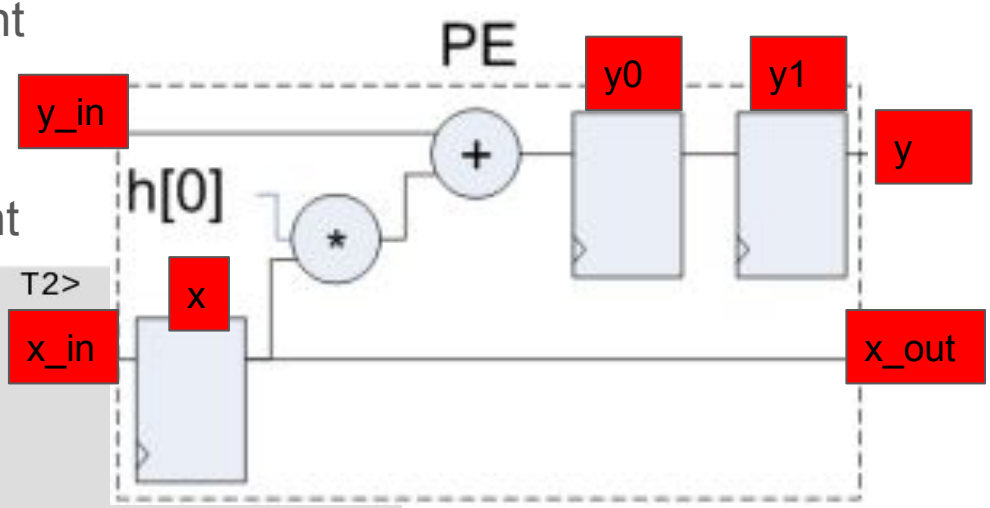


Illustration 144: Systolic Architecture

Systotic FIR : Processing Element

- y_in x_in : input of Processing Element
- $y0 = x * h + y_{in}$
- x y0 y1 : registers
- y x_out : output of Processing Element




```
 1 template<typename T0, typename T1, typename T2>
 2 class pe_class{
 3 private:
 4   T0 x;
 5   T2 y0;
 6   T2 y1;
 7 public:
 8   void exec(T0 &x_in, T1 &h, T2 &y_in,T0 &x_out, T2 &y){
 9     y = y1;
10     x_out = x;
11     y1 = y0;
12     y0 = x * h + y_in;
13     x = x_in;
14   }
15 };
```

Systotic iteration=1

-

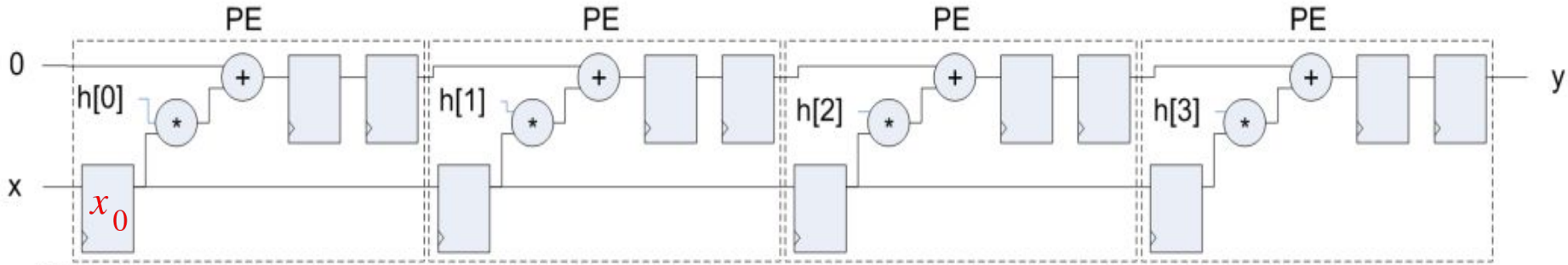


Illustration 144: Systolic Architecture

Systotic iteration=2

-

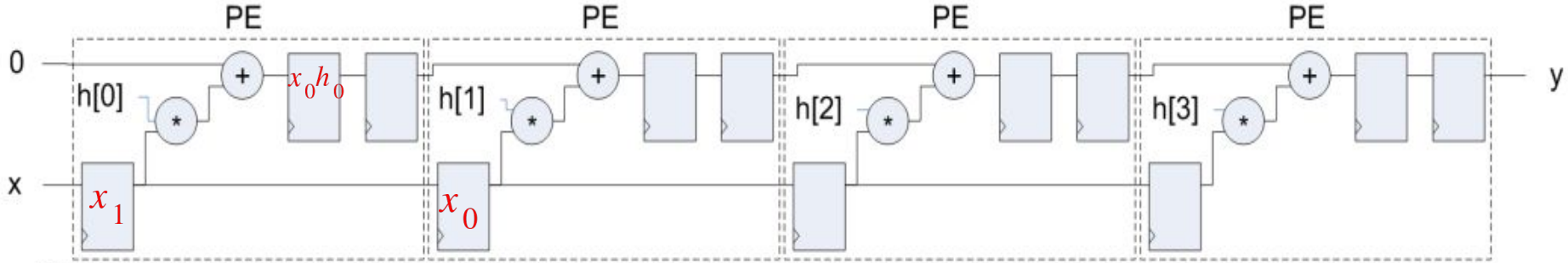


Illustration 144: Systolic Architecture

Systotic iteration=3

-

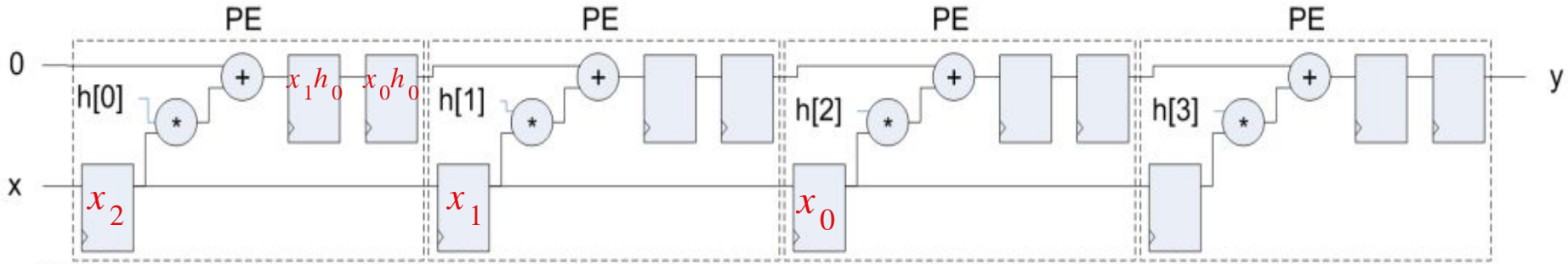


Illustration 144: Systolic Architecture

Systotic iteration=4

-

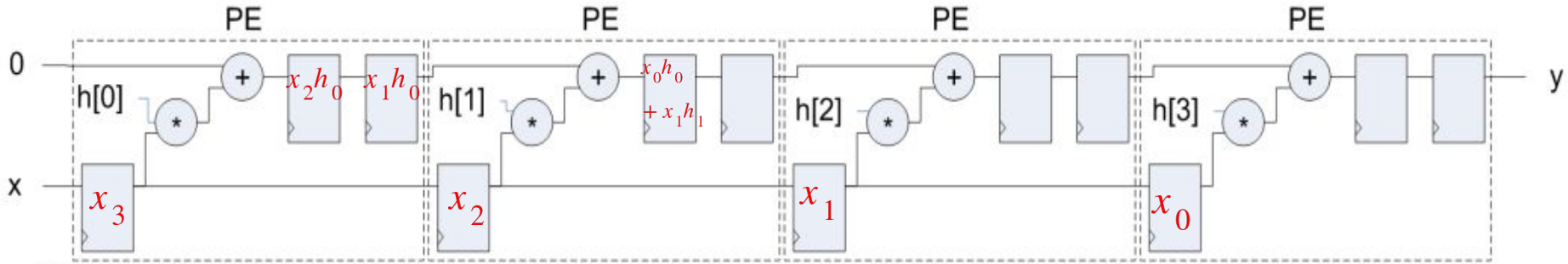


Illustration 144: Systolic Architecture

Systotic iteration=5

-

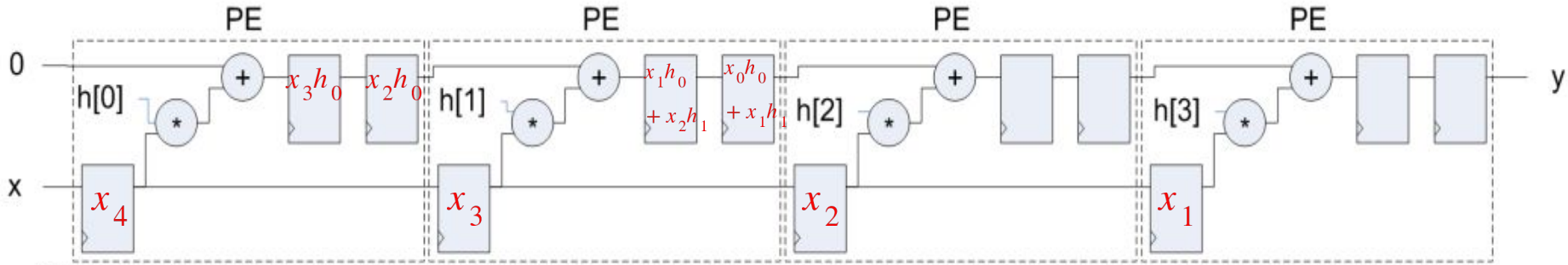


Illustration 144: Systolic Architecture

Systotic iteration=6

-

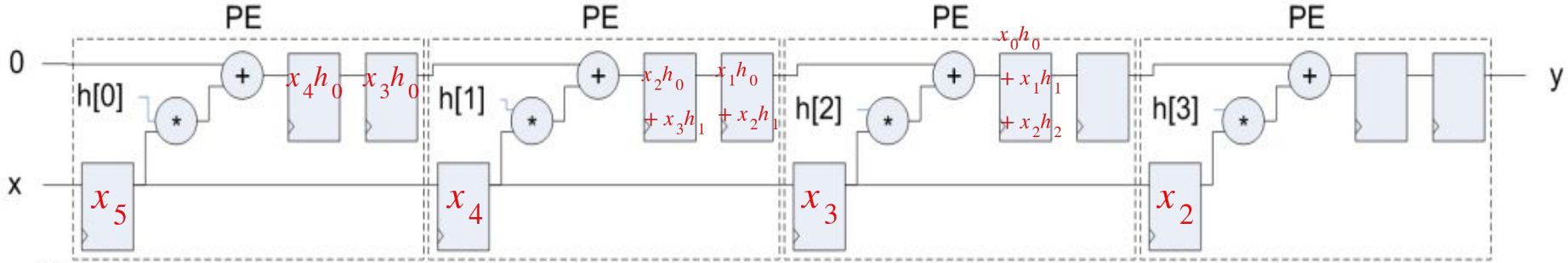


Illustration 144: Systolic Architecture

Systotic iteration=7

-

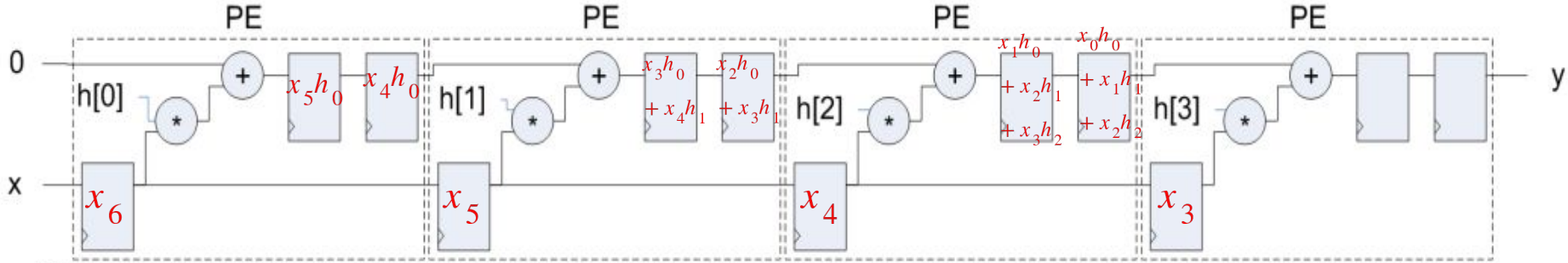


Illustration 144: Systolic Architecture

Systotic iteration=8

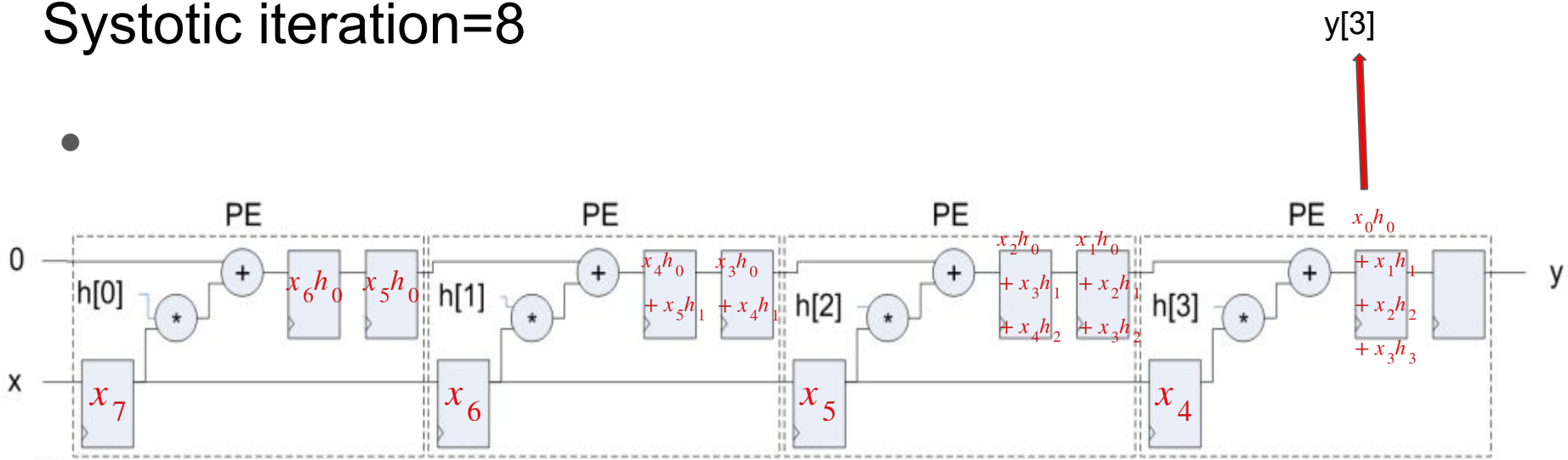


Illustration 144: Systolic Architecture

Systotic FIR Implementation

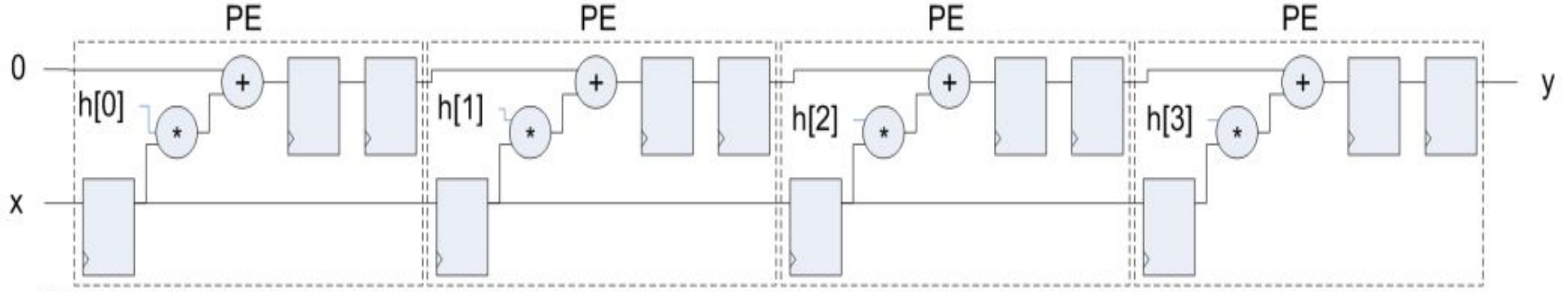
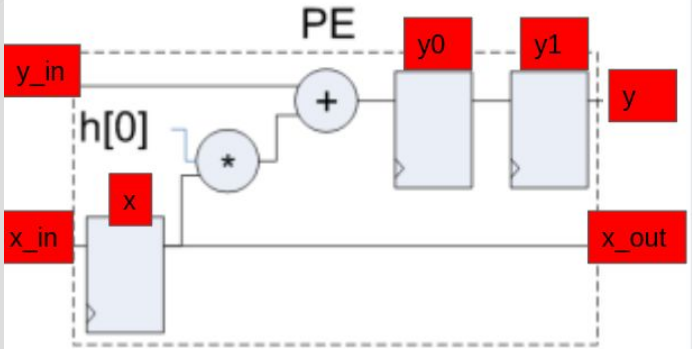


Illustration 144: Systolic Architecture

```
 1 #include "pe_class.hpp"
 2 #include "fir_filter.h"
 3 void fir_filter (ac_fixed<8,1> *x,
 4                  ac_fixed<8,1> h[4],
 5                  ac_fixed<19,4> *y){
 6   static pe_class<ac_fixed<8,1>,ac_fixed<8,1>,ac_fixed<19,4> > pe[4];
 7   ac_fixed<8,1> x_int[4];
 8   ac_fixed<19,4> y_int[4];
 9   ac_fixed<19,4> tmp = 0;
10
11  CONN:for(int i=0;i<4;i++)
12     if(i==0)
13       pe[0].exec(*x, h[i],tmp,x_int[i],y_int[i]);
14     else
15       pe[i].exec(x_int[i-1], h[i],y_int[i-1],x_int[i],y_int[i]);
16   *y = y_int[3];
17 }
```

```
8   void exec(T0 &x_in, T1 &h, T2 &y_in,T0 &x_out, T2 &y){
```



Comparison

	Latency(ns)	FF	LUT	DSP48E
baseline	80	74	402	0
const coef	80	72	239	2
load coef	80	83	408	0
even symmetric	80	76	158	2
odd symmetric	150	110	267	2
transposed	20	69	291	0
systolic	20	289	292	0