Efficient Coding Methods

Dr. Cyril Prasanna Raj P.

Director – Angstromers Engg. Services

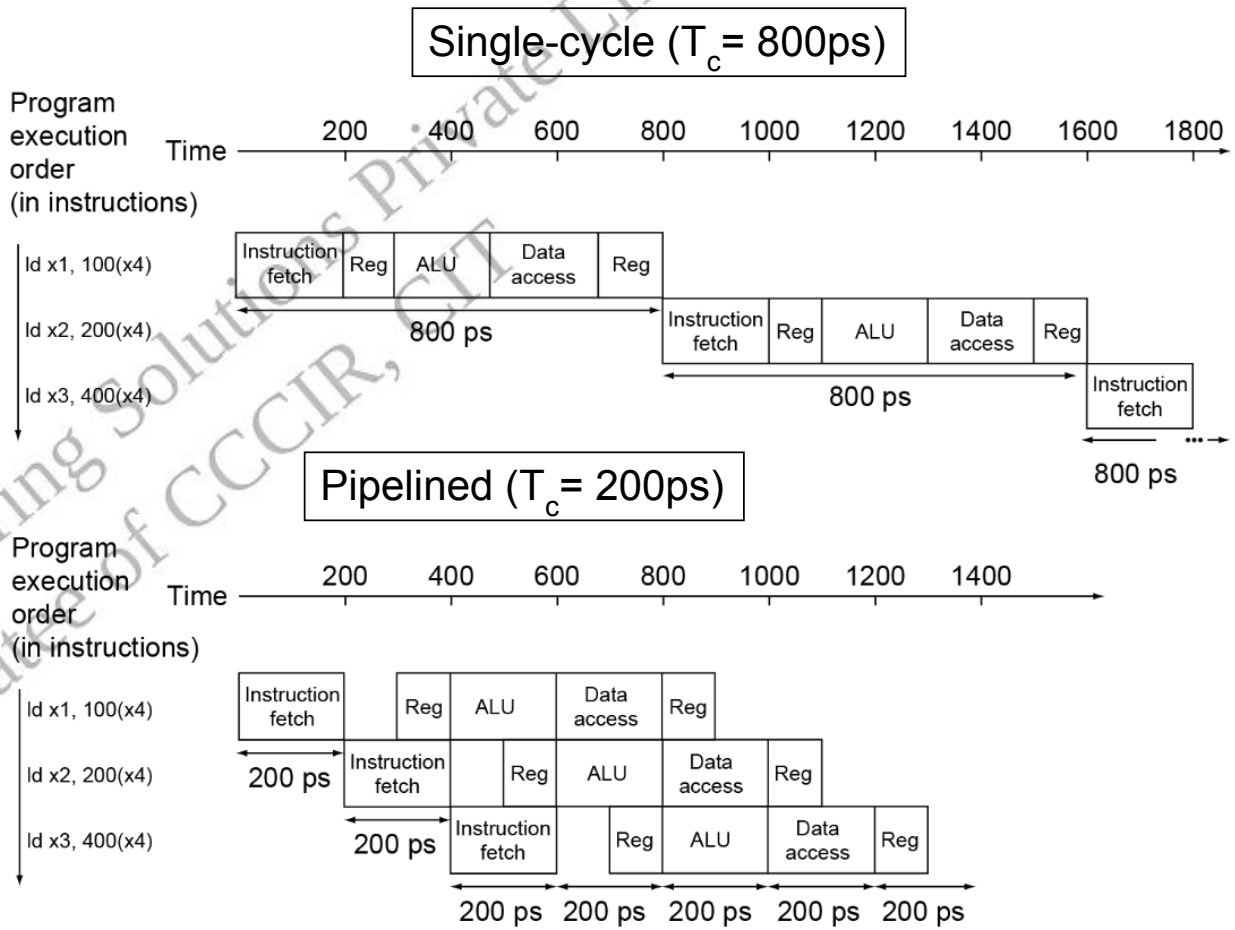
Professor – Dept. of ECE

CIT, Bangalore

RISC-V Pipeline

- Five stages, one step per stage
  1. IF: Instruction fetch from memory
  2. ID: Instruction decode & register read
  3. EX: Execute operation or calculate address
  4. MEM: Access memory operand
  5. WB: Write result back to register
- Pipeline Performance
- Assume time for stages is
  - 100ps for register read or write
  - 200ps for other stages

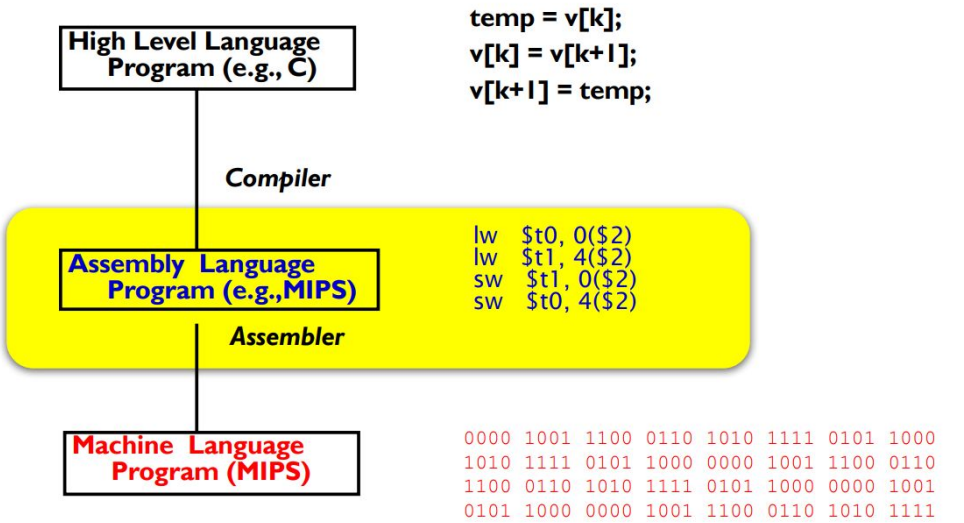
Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
ld	200ps	100 ps	200ps	200ps	100 ps	800ps
sd	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

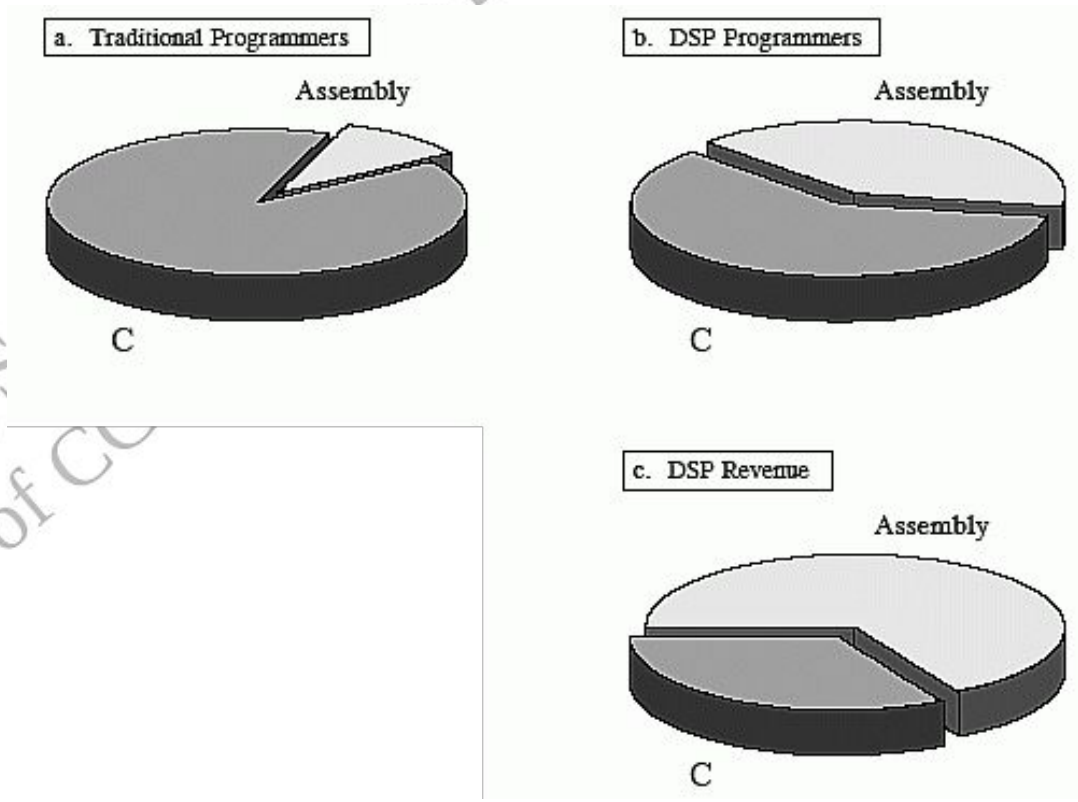


Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Levels of optimization

Levels of Program Code





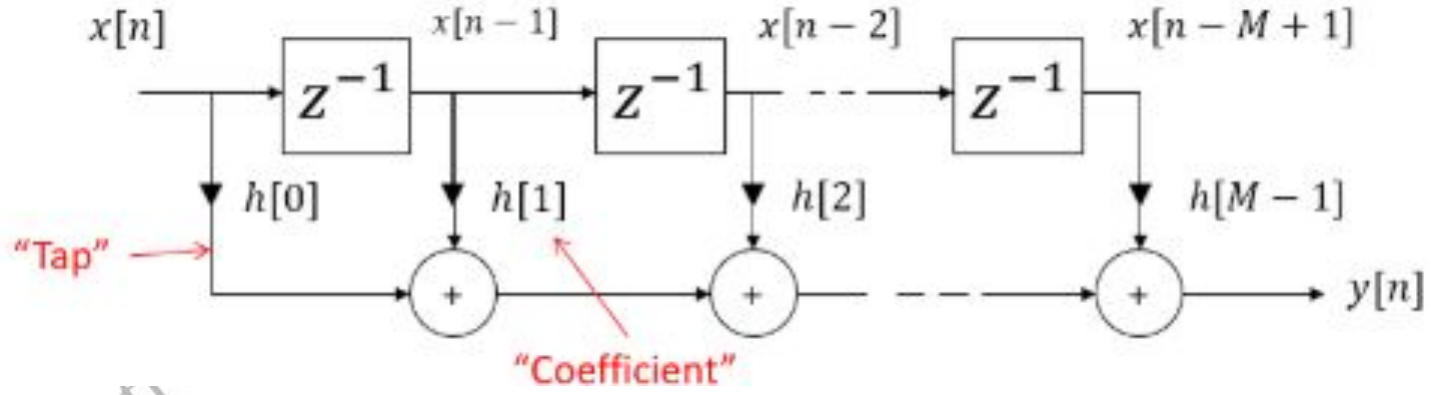
Limited
ngstromers E
Incuba
e of C

Simple example – vector multiplications

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k]$$

$$\underbrace{\begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[n] \end{bmatrix}}_{\mathbf{y}} \underbrace{\begin{bmatrix} h[0] & 0 & 0 & \dots & 0 \\ h[1] & h[0] & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ h[n] & h[n-1] & \dots & \dots & h[0] \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[n] \end{bmatrix}}_{\mathbf{x}}$$

Sum the products □ $x[0]\times y[0] + x[1] \times y[1] + x[2] \times y[2] + \dots. x[19] \times y[19]$



Simple Example – vector multiplications

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k]$$

Sum the products □ $x[0]\times y[0] + x[1]\times y[1] + x[2]\times y[2] + \ldots. x[19]\times y[19]$

C – Program

```
001    #define LEN 20
002     float dm x[LEN];
003     float pm y[LEN];
004     float result;
005
006    main()
007
008    {
009     int n;
010     float s;
011     for (n=0;n<LEN;n++)
012       s += x[n]*y[n];
013     result = s
014    }
```

Assembly Program

```
001    i12 = _y;                    /* i12 points to beginning of y[ ] */
002    i4 = _x;                     /* i4 points to beginning of x[ ] */
003
004    lcntr = 20, do (pc,4) until lce;   /* loop for the 20 array entries */
005      f2 = dm(i4,m6);            /* load the x[ ] value into register f2 */
006      f4 = pm(i12,m14);          /* load the y[ ] value into register f4 */
007      f8 = f2*f4;                /* multiply the two values, store in f8 */
008      f12 = f8 + f12;            /* add the product to the accumulator in f12 */
009
010    dm(_result) = f12;           /* write the accumulator to memory */
```

Optimized ASM code

Generic ASM Code

```
001        i12 = _y;
002        i4 = _x;
003
004        lcntr = 20, do (pc,4) until lce;
005          f2 = dm(i4,m6);
006          f4 = pm(i12,m14);
007          f8 = f2*f4;
008          f12 = f8 + f12;
009
010        dm(_result) = f12;
```

Optimum ASM Code

```
001        i12 = _y;                          /* i12 points to beginning of y[ ] */
002        i4 = _x;                           /* i4 points to beginning of x[ ] */
003
004        f2 = dm(i4,m6), f4 = pm(i12,m14)   /* prime the registers */
005        f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
006
007        lcntr = 18, do (pc,1) until lce;   /* highly efficient main loop */
008        f12 = f8 + f12, f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
009
010        f12 = f8 + f12, f8 = f2*f4;        /* complete the last loop */
011        f12 = f8 + f12;
012
013        dm(_result) = f12;                 /* store the result in memory */
```

Loop is reduced from 20 to 18

Single loop multiple instructions

Load data lines 004 and compute the first operation in line 005
Performing operation on last data in line 010 and 011

ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Optimized ASM code

```
001    i12 = _y;                                   /* i12 points to beginning of y[ ] */
002    i4 = _x;                                    /* i4 points to beginning of x[ ] */
003
004    f2 = dm(i4,m6), f4 = pm(i12,m14)            /* prime the registers */
005    f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
006
007    lcntr = 18, do (pc,1) until lce;            /* highly efficient main loop */
008    f12 = f8 + f12, f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
009
010    f12 = f8 + f12, f8 = f2*f4;                 /* complete the last loop */
011    f12 = f8 + f12;
012
013    dm(_result) = f12;                          /* store the result in memory */
```

- *Line 008 - Four* operations are being carried out in parallel:
  - (1) the value for *x*[ ] is moved from a circular buffer in program memory and placed in f2;
  - (2) the value for *y*[ ] is being moved from a circular buffer in data memory and placed in f4;
  - (3) the previous values of f2 and f4 are multiplied and placed in f8; and
  - (4) the previous value in f8 is added to the accumulator in f12.

No. of clock cycles:

Un optimized code:

- 20 loops, with four actions being required in each loop □ 80 clock cycles
- 5 clock cycles of overhead
- Total of 85 clock

Optimized code:

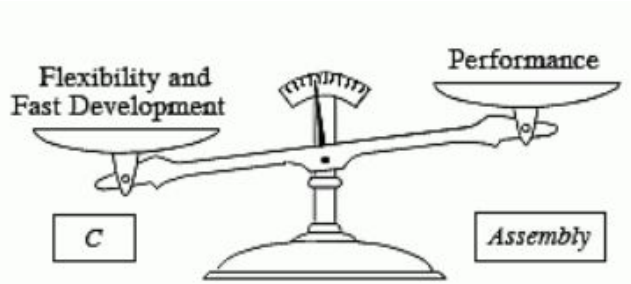
- 18 loops in 18 clock cycles
- 11 clock cycles of overhead
- Total of 29 clocks
- Or ~3x faster

For example, the fifth time that line 008 is executed, x[7] and y[7] are fetched from memory and stored in f2 and f4.
At the same time, the values for x[6] and y[6] (that were in f2 and f4 at the start of this cycle) are multiplied and placed in f8.
In addition, the value of x[5] × y[5] (that was in f8 at the start of this cycle) is added to the value of f12.

ANGSTROMERS CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution Chips to Startup Programme

The Industry approach

- Write the program in C, but use assembly for the critical sections that must execute quickly
- Things you should consider:
  - How complicated is the program? If it is large and intricate, you will probably want to use C. If it is small and simple, assembly may be a good choice.
  - Are you pushing the maximum speed of the DSP? If so, assembly will give you the last drop of performance from the device. For less demanding applications, assembly has little advantage, and you should consider using C.
  - How many programmers will be working together? If the project is large enough for more than one programmer, lean toward C and use in-line assembly only for time critical segments.
  - Which is more important, *product cost* or *development cost*? If it is product cost, choose assembly; if it is development cost, choose C.
  - What is your background? If you are experienced in assembly (on other microprocessors), choose assembly for your DSP. If your previous work is in C, choose C for your DSP.



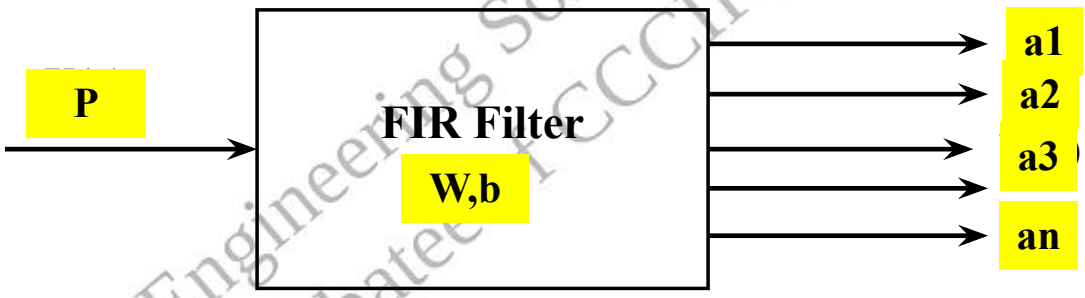
Angstromers Engineering Solutions Private Limited
Incubated @ CCCIR, CIT

Case study 2

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

ANN and Filter

- Filter has input **x(n)**, filter coefficient **h(n),** output **y(n)**
- ANN has input **P**, filter coefficient **W, b** and output **a**
- **Output in filter is Y = X*H □ realized using multiplier and adder**
- **Output in ANN is a = P.W □ realized using multiplier and adder**
  - **Additional macro unit is network activation function f(x)**



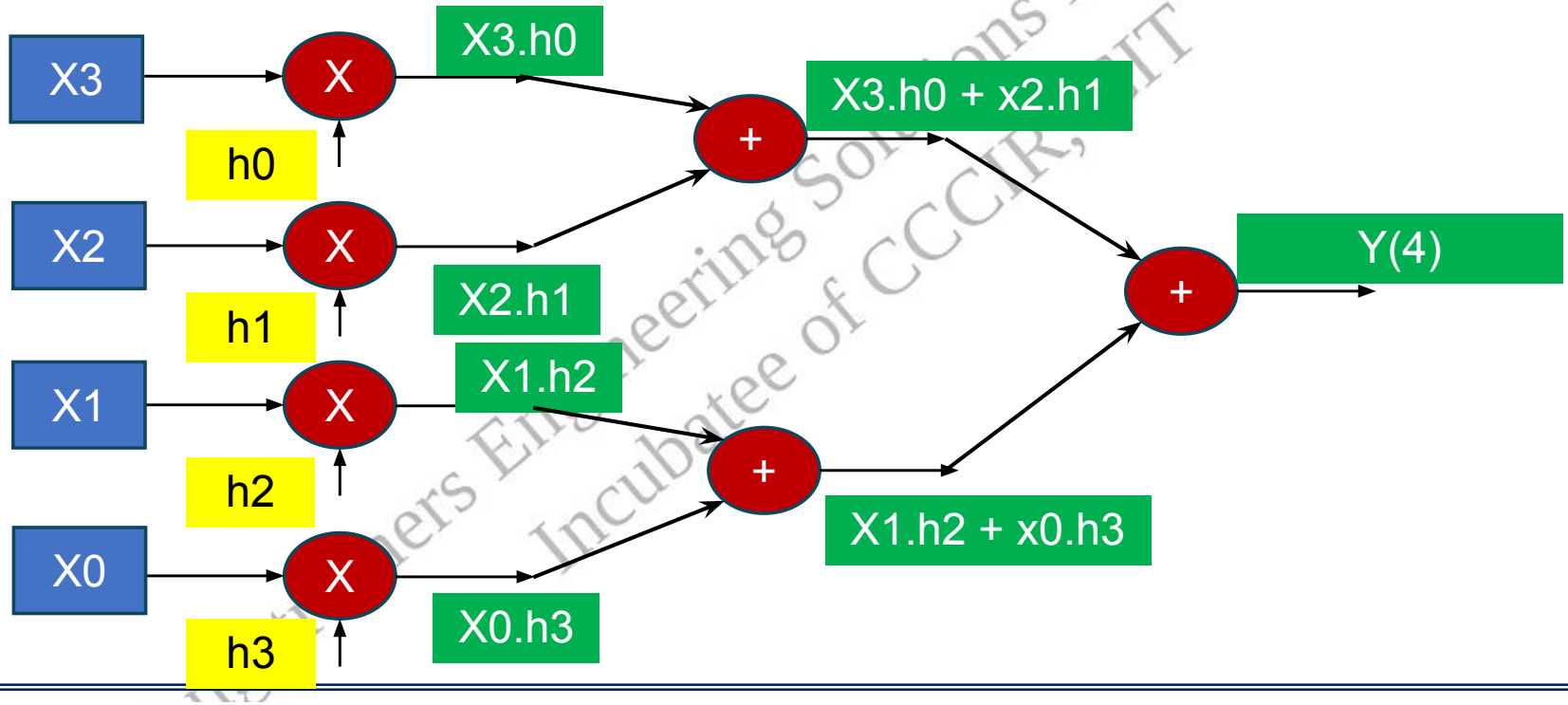
$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} x[m] \cdot h[n-m] = \sum_{m=-\infty}^{\infty} h[m] \cdot x[n-m]$$

Engineering Solutions Private Limited
Incubatee of CCCIR, CITI

Filter/ANN basics

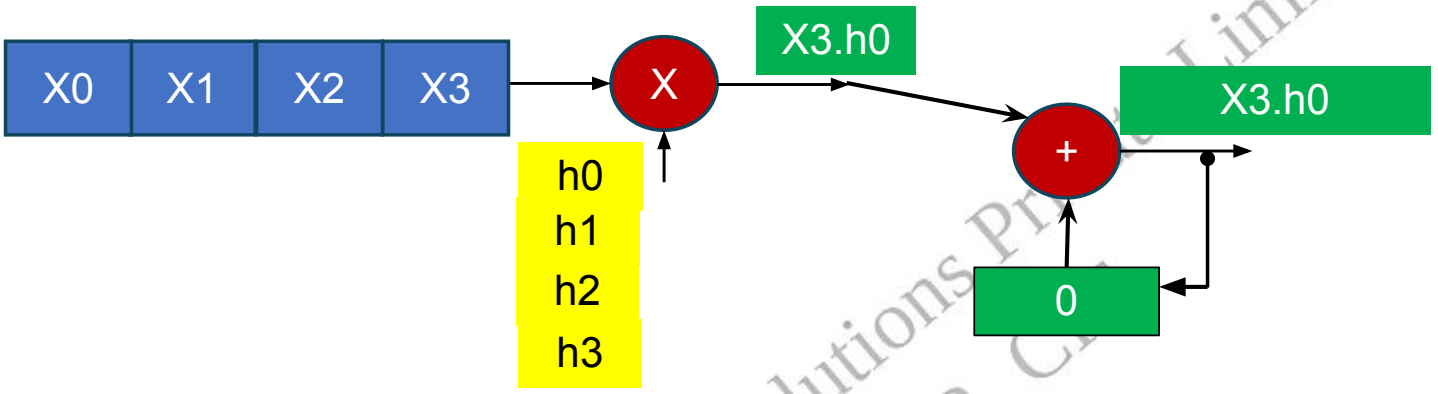
- To compute

$$y(3)= h(0).x(3) + h(1).x(2) + h(2).x(1) + h(3).x(0)$$



Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, IIT

Fully Serial



Clock 1,2 output = x3.h0+0

Clock 3,4 output = x3.h0 + x2h1

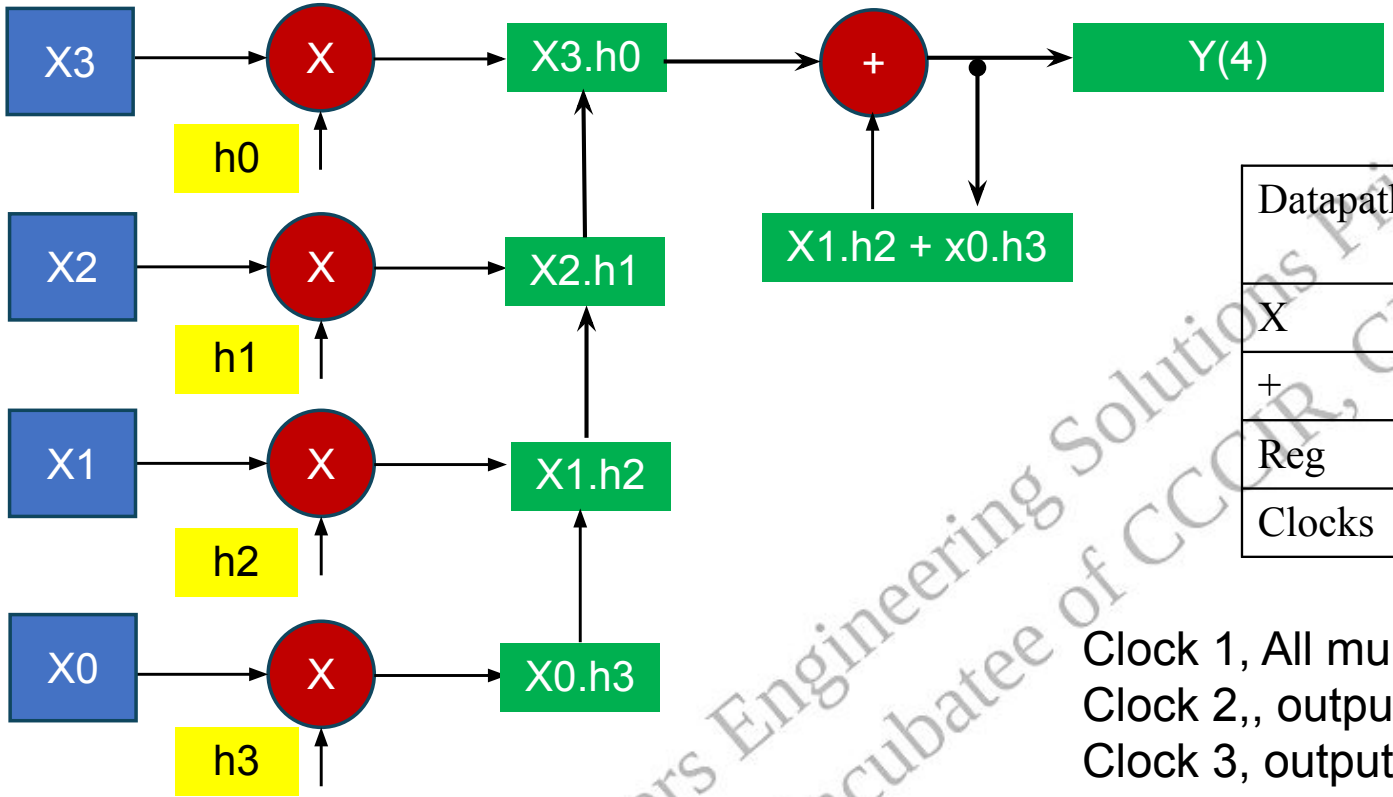
Clock 5,6 output = x3.h0 + x2h1 + x1h2

Clock 7,8 output = x3.h0 + x2h1 + x1h2 + x0h3

Datapath	Fully parallel	Fully serial
X	4	1
+	3	1
Reg	15	11
Clocks	3	8

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Partially Serial

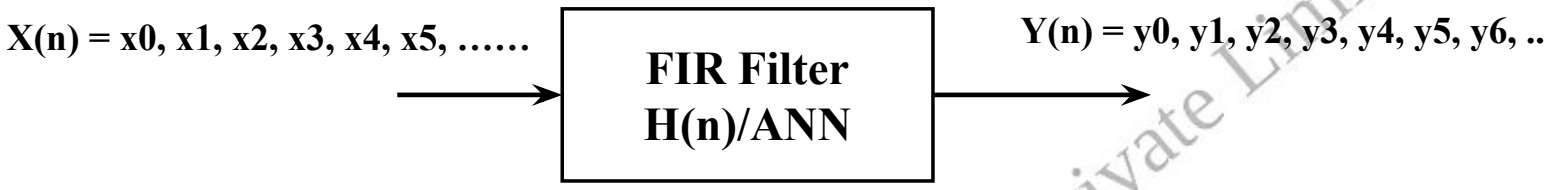


Datapath	Fully parallel	Fully serial	Partially Serial
X	4	1	4
+	3	1	1
Reg	15	11	14
Clocks	3	8	5

Clock 1, All multiplications
Clock 2,, output = x3.h0 + 0
Clock 3, output = x3.h0 + x2h1
Clock 4, output = x3.h0 + x2h1 + x1h2
Clock 5, output = x3.h0 + x2h1 + x1h2 + x0h3

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Case Study 2 - Optimization



Order h(n)	Filter coefficient
Ho	18
H1	18
H2	24
H3	24
H4	56
H5	31
H6	31
H7	56

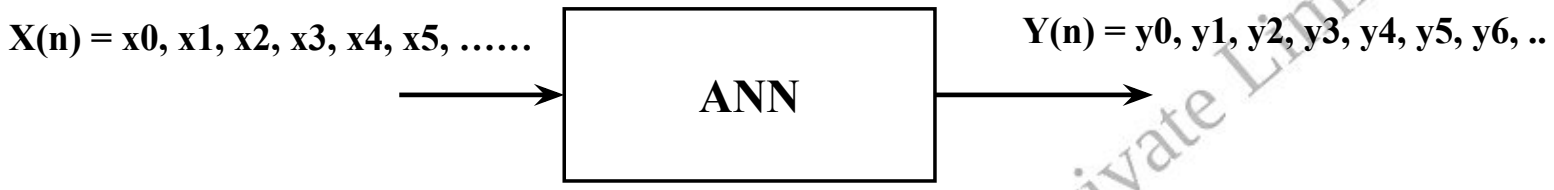
$$y(0)= h(0).x(0) + h(1).x(1) + h(2).x(2) + h(3).x(3)+ h(4).x(4) + h(5).x(5) + h(6).x(6) + h(7).x(7)+$$

$$y(1)= h(0).x(1) + h(1).x(2) + h(2).x(3) + h(3).x(4)+ h(4).x(5) + h(5).x(6) + h(6).x(7) + h(7).x(8)+$$

Datapath	Fully parallel	Fully serial	Partially Serial
X	8	1	8
+	7	1	1
Reg	27	19	26
Clocks	4	18	9

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Case Study 2



Order h(n)	Filter coefficient
Ho	18
H1	18
H2	24
H3	24
H4	56
H5	31
H6	31
H7	56

$$y(0)= h(0).x(0) + h(1).x(1) + h(2).x(2) + h(3).x(3)+ h(4).x(4) + h(5).x(5) + h(6).x(6) + h(7).x(7)+$$

Substituting filter coefficients

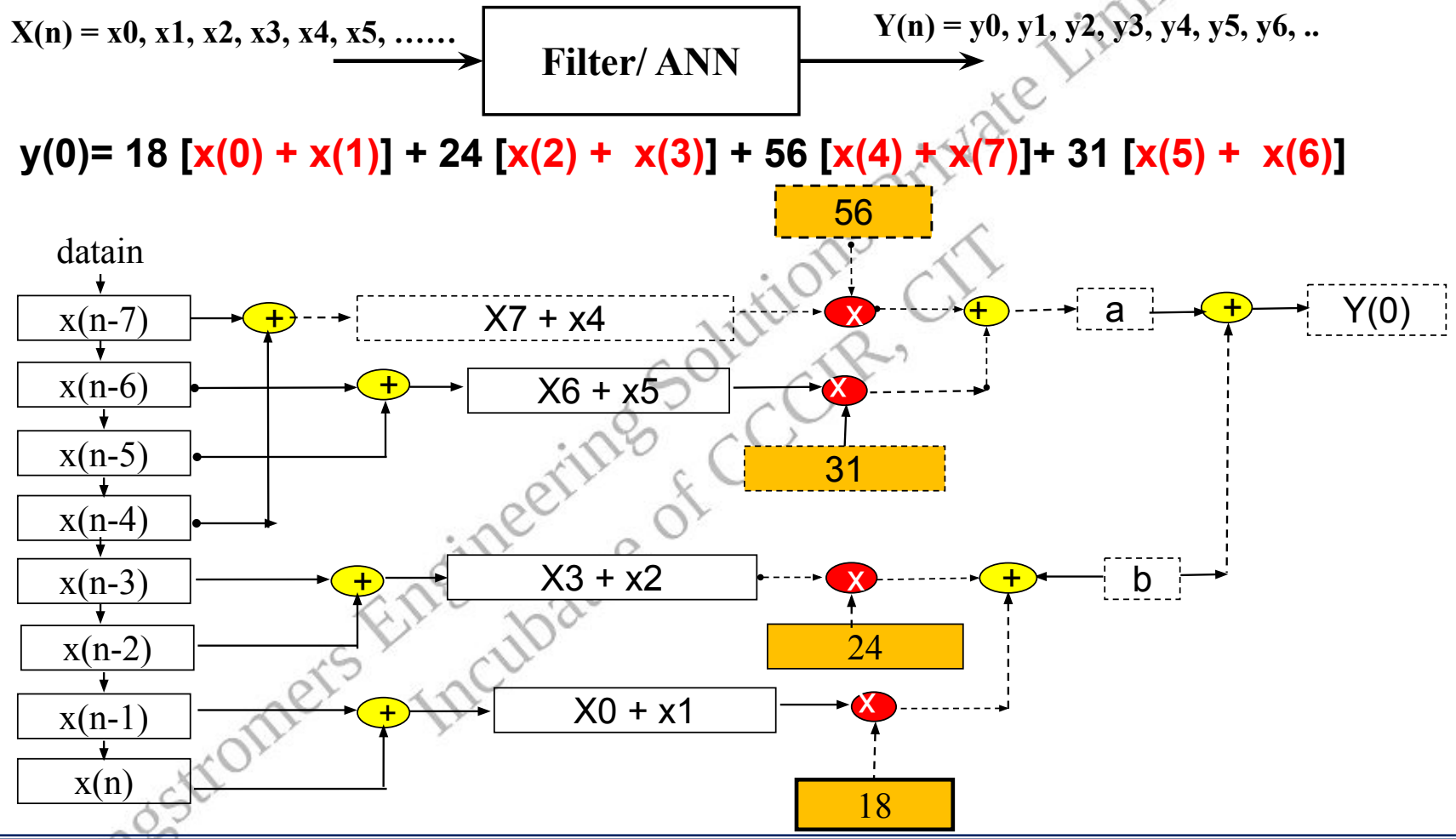
$$y(0)= 18.\ x(0) + 18\ .x(1) + 24\ .x(2) + 24\ .x(3)+ 56\ .x(4) + 31\ .x(5) + 31\ .x(6) + 56\ .x(7)+$$

Rearranging the terms

$$y(0)= 18\ [x(0) + x(1)] + 24\ [x(2) + x(3)] + 56\ [x(4) + x(7)]+ 31\ [x(5) + x(6)]$$

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Case Study 2



ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Comparison of all Designs

Datapath	Fully parallel	Fully serial	Partially Serial	New design
X	8	1	8	4
+	7	1	1	7
Reg	27	19	26	19
Clocks	4	18	9	4

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Further Optimization

$$y(0)= 18 [x(0) + x(1)] + 24 [x(2) + x(3)] + 56 [x(4) + x(7)]+ 31 [x(5) + x(6)]$$

$$y(0) = 18 [a1] + 24 [a2] + 56 [a3]+ 31 [a4]$$

Decimal No.	Binary No.					
	32	16	8	4	2	1
18	0	1	0	0	1	0
24	0	1	1	0	0	0
56	1	1	1	0	0	0
31	0	1	1	1	1	1
32	1	0	0	0	0	0

- Lets assume a1 = 2, a2 = 7, a3 = 1, a4 = 4
- y(0) = 36 + 168 + 56 + 124 = 384
- y(0) = 36 + 168 + 56 + 128 = 388

ngstrom... Engineering Solutions Private Limited ...tee of CCIR, CIT

Further Optimization – ZERO PAD METHOD

y(0) = 18 [2] + 24 [7] + 56 [1]+ 32 [4]

Decimal No.	Binary No.					
	32	16	8	4	2	1
2	0	0	0	0	1	0
7	0	0	0	1	1	1
1	0	0	0	0	0	1
4	0	0	0	1	0	0

Decimal No.	Binary No.					
	32	16	8	4	2	1
18	0	1	0	0	1	0
24	0	1	1	0	0	0
56	1	1	1	0	0	0
32	1	0	0	0	0	0

$2 \times 18 = 36 \rightarrow 2 \times (1x2^4 + 1x2^1) \rightarrow 2 \times (1x2^4) + 2 \times (1x2^1)$

2 = 00 00 10

2 x 2 = 00 01 00

2 x 4 = 00 10 00

2 x 8 = 01 00 00

2 x 16 = 10 00 00

00 00 10 **00 00**

00 00 10 **0**

0	0	0	0	0	1	0	**0**	**0**	**0**	**0**
+										
	0	0	0	0	0	1	0	**0**		
0	0	0	0	0	1	0	**0**	**1**	**0**	**0**
1024	512	256	128	64	32	16	**8**	**4**	**2**	**1**

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, IIT

Further Optimization – ZERO PAD METHOD

y(0) = 18 [2] + 24 [7] + 56 [1]+ 32 [4]

Decimal No.	Binary No.					
	32	16	8	4	2	1
2	0	0	0	0	1	0
7	0	0	0	1	1	1
1	0	0	0	0	0	1
4	0	0	0	1	0	0

Decimal No.	Binary No.					
	32	16	8	4	2	1
18	0	1	0	0	1	0
24	0	1	1	0	0	0
56	1	1	1	0	0	0
32	1	0	0	0	0	0

$7 \times 24 = 168 \square 7 \times (1x2^4 + 1x2^3) \square 7 \times (1x2^4) + 7 \times (1x2^3)$

00 01 11

00 01 11 **00**
00

0 00 11 1
000

	0	0	0	0	1	1	1	**0**	**0**	**0**	**0**
+	0	0	0	0	0	1	1	1	**0**	**0**	**0**
	0	0	0	1	0	1	0	**1**	**0**	**0**	**0**
	1024	512	256	128	64	32	16	**8**	**4**	**2**	**1**

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, SRMIST

Further Optimization – ZERO PAD METHOD

y(0) = 18 [2] + 24 [7] + 56 [1]+ 32 [4]

Decimal No.	Binary No.					
	32	16	8	4	2	1
2	0	0	0	0	1	0
7	0	0	0	1	1	1
1	0	0	0	0	0	1
4	0	0	0	1	0	0

Decimal No.	Binary No.					
	32	16	8	4	2	1
18	0	1	0	0	1	0
24	0	1	1	0	0	0
56	1	1	1	0	0	0
32	1	0	0	0	0	0

$1 \times 56 = 56 \square 1 \times (1 \times 2^4 + 1 \times 2^4 + 1 \times 2^3) \square 1 \times (1 \times 2^5) + 1 \times (1 \times 2^4) + 1 \times (1 \times 2^3)$

00 00 01

00 00 01 **00 00 0**

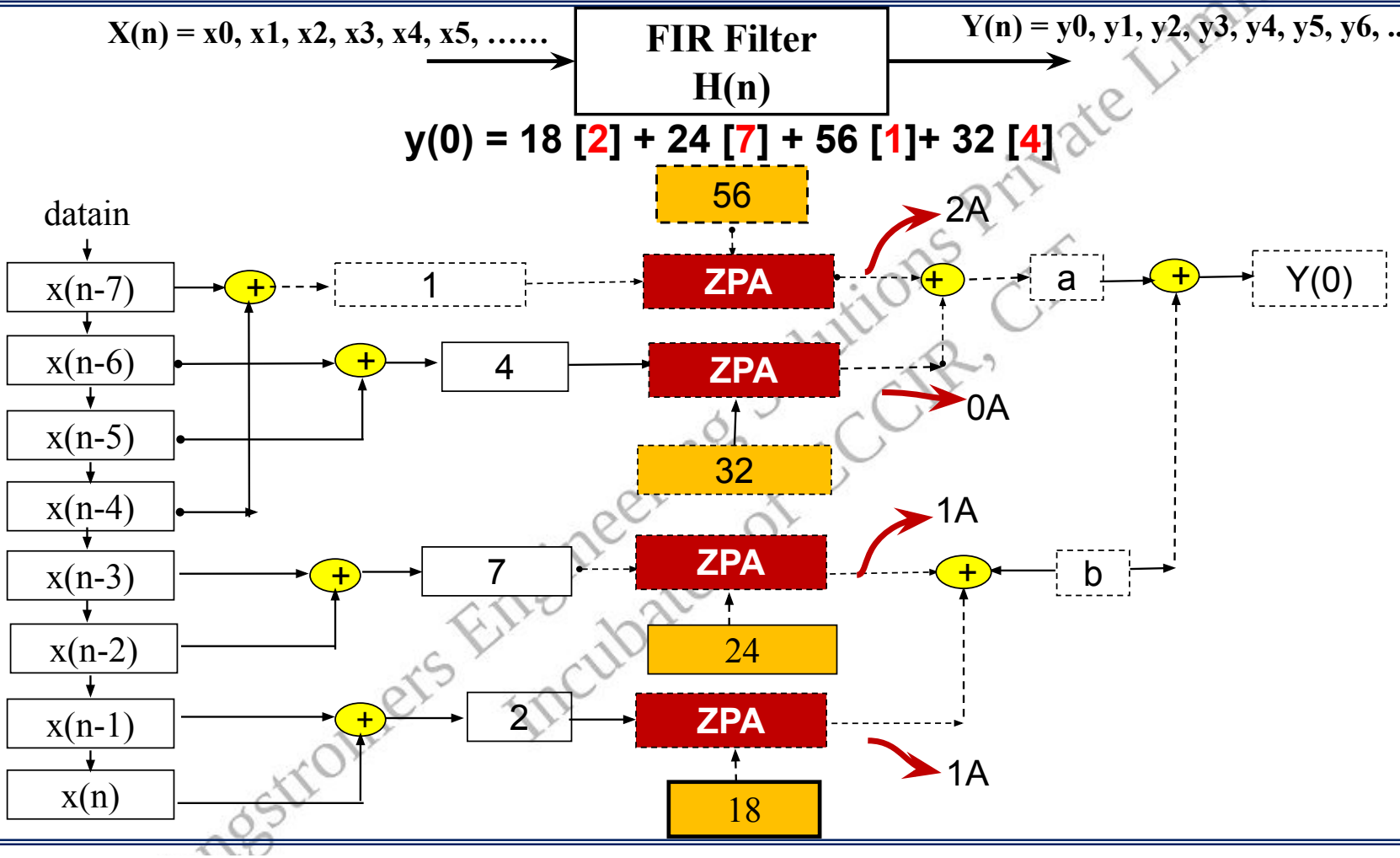
0 00 00 1 **0000**

0 00 00 1 **000**

	0	0	0	0	0	0	1	**0**	**0**	**0**	**0**	**0**
	0	0	0	0	0	0	0	1	**0**	**0**	**0**	**0**
+	0	0	0	0	0	0	0	0	1	**0**	**0**	**0**
	0	0	0	0	0	0	1	1	1	**0**	**0**	**0**
	1024	512	256	128	64	32	16	**8**	**4**	**2**	**1**	

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Further Optimization using ZERO PAD



Angstroms Engineering Solutions Private Limited
Incubated at CCIR, CET

Comparison of all Designs

Data path	Fully parallel	Fully serial	Partially Serial	New design	Improved Design
X	8	1	8	4	0
+	7	1	1	7	11
Reg	27	19	26	19	19
Clocks	4	18	9	4	4

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Case study 3

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Realizing convolution algorithm

$$X_m = \sum_{n=0}^{N-1} x_n w^{nm}, \qquad \text{where } N \text{ is the size of the vectors, } w = e^{2i\pi/N}$$

$$X_m = \sum_{n=0}^{N/2-1} x_n w^{nm} + w^{mN/2} \sum_{n=0}^{N/2-1} x_{n+N/2} w^{nm},$$

$$X_m = \sum_{n=0}^{N/2-1} x_{2n} w^{2nm} + w^{m} \sum_{n=0}^{N/2-1} x_{2n+1} w^{2nm}.$$

Angstromers Private Limited Incubating CCCIR

Matrix representation

$$X_m = \sum_{n=0}^{N-1} x_n w^{nm},$$

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^2 & w^4 & w^6 & w^8 & w^{10} & w^{12} & w^{14} \\ 1 & w^3 & w^6 & w^9 & w^{12} & w^{15} & w^{18} & w^{21} \\ 1 & w^4 & w^8 & w^{12} & w^{16} & w^{20} & w^{24} & w^{28} \\ 1 & w^5 & w^{10} & w^{15} & w^{20} & w^{25} & w^{30} & w^{35} \\ 1 & w^6 & w^{12} & w^{18} & w^{24} & w^{30} & w^{36} & w^{42} \\ 1 & w^7 & w^{14} & w^{21} & w^{28} & w^{35} & w^{42} & w^{49} \end{bmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Matix representation – arranging in even terms

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} = \left[\begin{array}{cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^2 & w^4 & w^6 & w & w^3 & w^5 & w^7 \\ 1 & w^4 & w^8 & w^{12} & w^2 & w^6 & w^{10} & w^{14} \\ 1 & w^6 & w^{12} & w^{18} & w^3 & w^9 & w^{15} & w^{21} \\ \hline 1 & w^8 & w^{16} & w^{24} & w^4 & w^{12} & w^{20} & w^{28} \\ 1 & w^{10} & w^{20} & w^{30} & w^5 & w^{15} & w^{25} & w^{35} \\ 1 & w^{12} & w^{24} & w^{36} & w^6 & w^{18} & w^{30} & w^{42} \\ 1 & w^{14} & w^{28} & w^{42} & w^7 & w^{21} & w^{35} & w^{49} \end{array}\right] \begin{pmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \\ \hline x_1 \\ x_3 \\ x_5 \\ x_7 \end{pmatrix}$$

ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Matix representation – arranging in even terms

$$
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} =
\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & w^4 & w^2 & w^6 & w & w^5 & w^3 & w^7 \\
\hline
1 & w^8 & w^4 & w^{12} & w^2 & w^{10} & w^6 & w^{14} \\
1 & w^{12} & w^6 & w^{18} & w^3 & w^{15} & w^9 & w^{21} \\
\hline
1 & w^{16} & w^8 & w^{24} & w^4 & w^{20} & w^{12} & w^{28} \\
1 & w^{20} & w^{10} & w^{30} & w^5 & w^{25} & w^{15} & w^{35} \\
\hline
1 & w^{24} & w^{12} & w^{36} & w^6 & w^{30} & w^{18} & w^{42} \\
1 & w^{28} & w^{14} & w^{42} & w^7 & w^{35} & w^{21} & w^{49}
\end{array}\right]
\begin{pmatrix} x_0 \\ x_4 \\ \hline x_2 \\ x_6 \\ \hline x_1 \\ x_5 \\ \hline x_3 \\ x_7 \end{pmatrix}
$$

Limited

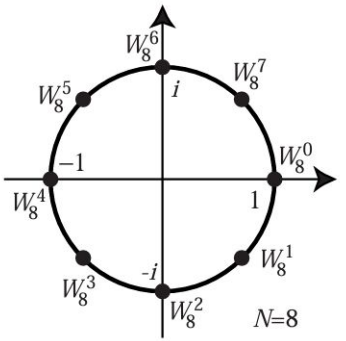
Replacing with twiddle factors (mod 8)

$$\left[\begin{array}{cc|cc|cc|cc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^4 & w^2 & w^6 & w & w^5 & w^3 & w^7 \\ \hline 1 & w^8 & w^4 & w^{12} & w^2 & w^{10} & w^6 & w^{14} \\ 1 & w^{12} & w^6 & w^{18} & w^3 & w^{15} & w^9 & w^{21} \\ \hline 1 & w^{16} & w^8 & w^{24} & w^4 & w^{20} & w^{12} & w^{28} \\ 1 & w^{20} & w^{10} & w^{30} & w^5 & w^{25} & w^{15} & w^{35} \\ \hline 1 & w^{24} & w^{12} & w^{36} & w^6 & w^{30} & w^{18} & w^{42} \\ 1 & w^{28} & w^{14} & w^{42} & w^7 & w^{35} & w^{21} & w^{49} \end{array}\right]$$

$$\begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^4 & W^2 & W^6 & W^1 & W^5 & W^3 & W^7 \\ W^0 & W^0 & W^4 & W^4 & W^2 & W^2 & W^6 & W^6 \\ W^0 & W^4 & W^6 & W^2 & W^3 & W^7 & W^1 & W^5 \\ W^0 & W^0 & W^0 & W^0 & W^4 & W^4 & W^4 & W^4 \\ W^0 & W^4 & W^2 & W^6 & W^5 & W^1 & W^7 & W^3 \\ W^0 & W^0 & W^4 & W^4 & W^6 & W^6 & W^2 & W^2 \\ W^0 & W^4 & W^6 & W^2 & W^7 & W^3 & W^5 & W^1 \end{pmatrix}$$

ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

FFT algorithm



$$\begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^4 & W^2 & W^6 & W^1 & W^5 & W^3 & W^7 \\ W^0 & W^0 & W^4 & W^4 & W^2 & W^2 & W^6 & W^6 \\ W^0 & W^4 & W^6 & W^2 & W^3 & W^7 & W^1 & W^5 \\ W^0 & W^0 & W^0 & W^0 & W^4 & W^4 & W^4 & W^4 \\ W^0 & W^4 & W^2 & W^6 & W^5 & W^1 & W^7 & W^3 \\ W^0 & W^0 & W^4 & W^4 & W^6 & W^6 & W^2 & W^2 \\ W^0 & W^4 & W^6 & W^2 & W^7 & W^3 & W^5 & W^1 \end{pmatrix}$$

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} = \left[\begin{array}{cc|cc|cc|cc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & w^2 & -w^2 & w & -w & w^3 & -w^3 \\ \hline 1 & 1 & -1 & -1 & w^2 & w^2 & -w^2 & -w^2 \\ 1 & -1 & -w^2 & w^2 & w^3 & -w^3 & w & -w \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & w^2 & -w^2 & -w & w & -w^3 & w^3 \\ \hline 1 & 1 & -1 & -1 & -w^2 & -w^2 & w^2 & w^2 \\ 1 & -1 & -w^2 & w^2 & -w^3 & w^3 & -w & w \end{array}\right] \begin{pmatrix} x_0 \\ x_4 \\ x_2 \\ x_6 \\ x_1 \\ x_5 \\ x_3 \\ x_7 \end{pmatrix}$$

ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Recursive matrix multiplications

The recursive sum can be represented as a sequence of matrix transformations:

$$(X) = [A_2][A_1][A_0][P](x),$$

At any level l (from 0 to 2), we can define a $2^{l+1} \times 2^{l+1}$ matrix template: $[T]_l = \begin{bmatrix} [I] & [\Omega] \\ [I] & [-\Omega] \end{bmatrix}$

where $[\Omega] = \text{diag}(w^0, w^{N/2^{l+1}}, w^{2 \cdot N/2^{l+1}}, w^{3 \cdot N/2^{l+1}}, \cdots)$

$[I]$ is a $2^l \times 2^l$ identity matrix block

ngstromers Engineering Solutions
Incubatee of C IR,
ate Limited

Recursive computing (l=0)

$$(X) = [A_2][A_1][A_0][P](x), \qquad [T]_l = \begin{bmatrix} [I] & [\Omega] \\ [I] & [-\Omega] \end{bmatrix}$$

where $[\Omega] = \text{diag}(w^0, w^{N/2^{l+1}}, w^{2\cdot N/2^{l+1}}, w^{3\cdot N/2^{l+1}}, \cdots)$

$[I]$ is a $2^l \times 2^l$ identity matrix block

$$[A_0] = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Recursive computing (l=1)

$$(X) = [A_2][A_1][A_0][P](x), \qquad [T]_l = \begin{bmatrix} [I] & [\Omega] \\ [I] & [-\Omega] \end{bmatrix} \qquad \text{where } [\Omega] = \text{diag}(w^0, w^{N/2^{l+1}}, w^{2\cdot N/2^{l+1}}, w^{3\cdot N/2^{l+1}}, \cdots)$$

$[I]$ is a $2^l \times 2^l$ identity matrix block

$$[A_1] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & w^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -w^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & w^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -w^2 \end{bmatrix}$$

ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Recursive computing (l=2)

$$(X) = [A_2][A_1][A_0][P](x), \quad [T]_l = \begin{bmatrix} [I] & [\Omega] \\ [I] & [-\Omega] \end{bmatrix} \quad \text{where } [\Omega] = \text{diag}(w^0, w^{N/2^{l+1}}, w^{2\cdot N/2^{l+1}}, w^{3\cdot N/2^{l+1}}, \cdots)$$

$[I]$ is a $2^l \times 2^l$ identity matrix block

$$[A_2] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & w^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & w^3 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -w^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -w^3 \end{bmatrix}$$

Limited

Recursive matrix multiplications

The recursive sum can be represented as a sequence of matrix transformations:

$$(X) = [A_2][A_1][A_0][P](x),$$

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix} = \begin{pmatrix} 1 & \cdot & \cdot & \cdot & W^0 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & W^1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & W^2 & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & W^3 \\ 1 & \cdot & \cdot & \cdot & W^4 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & W^5 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & W^6 & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & W^7 \end{pmatrix} \begin{pmatrix} 1 & \cdot & W^0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & W^2 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & W^4 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & W^6 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & W^0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & W^2 \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & W^4 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & W^6 \end{pmatrix} \begin{pmatrix} 1 & W^0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & W^4 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & W^0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & W^4 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & W^0 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & W^4 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & W^0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & W^4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_4 \\ \hline x_2 \\ x_6 \\ \hline x_1 \\ x_5 \\ \hline x_3 \\ x_7 \end{pmatrix}$$

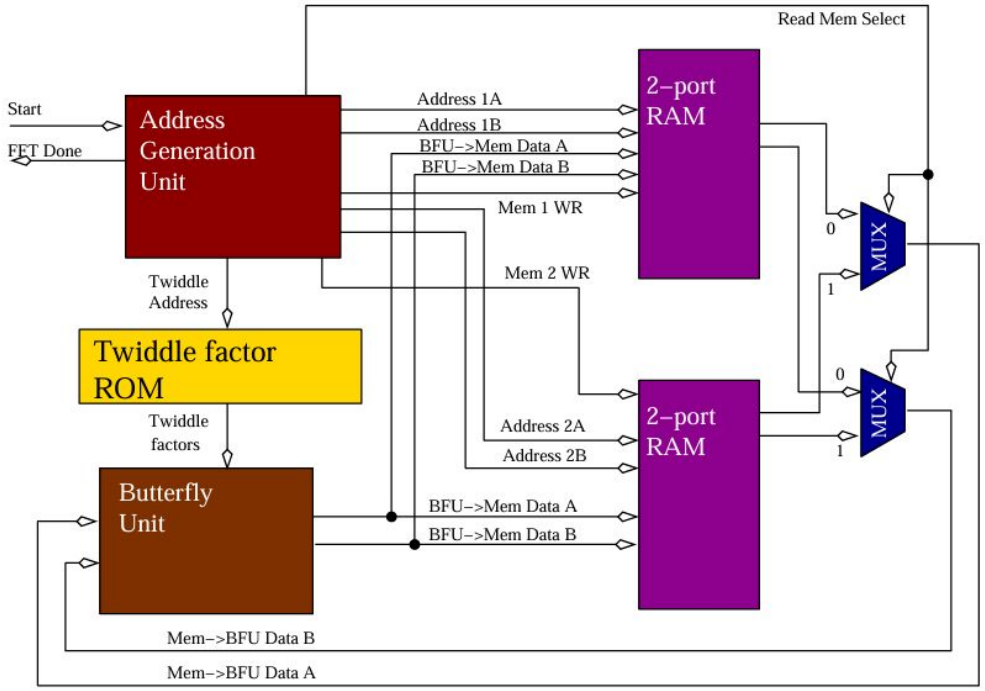
ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

The full transform requires – step 1

1) an address generator

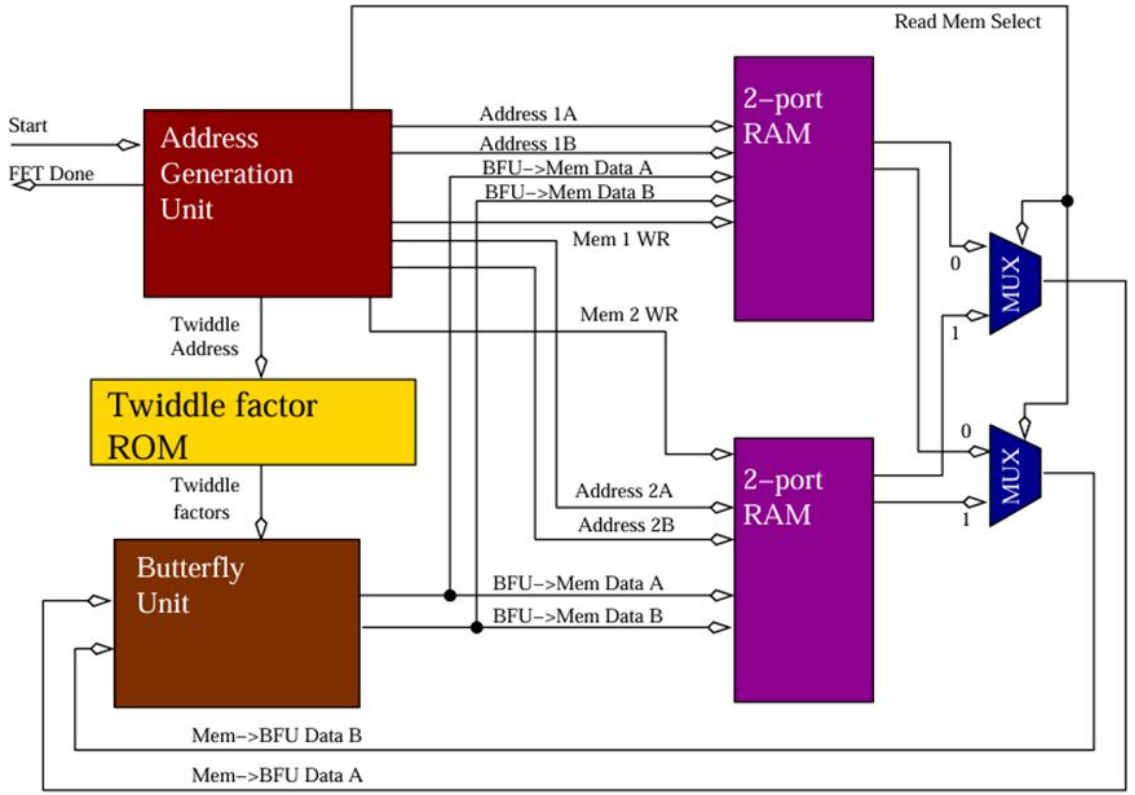
ILLUSTRATION OF THE BIT-REVERSED INDICES.

Index	binary	Bit reversed index	binary
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111



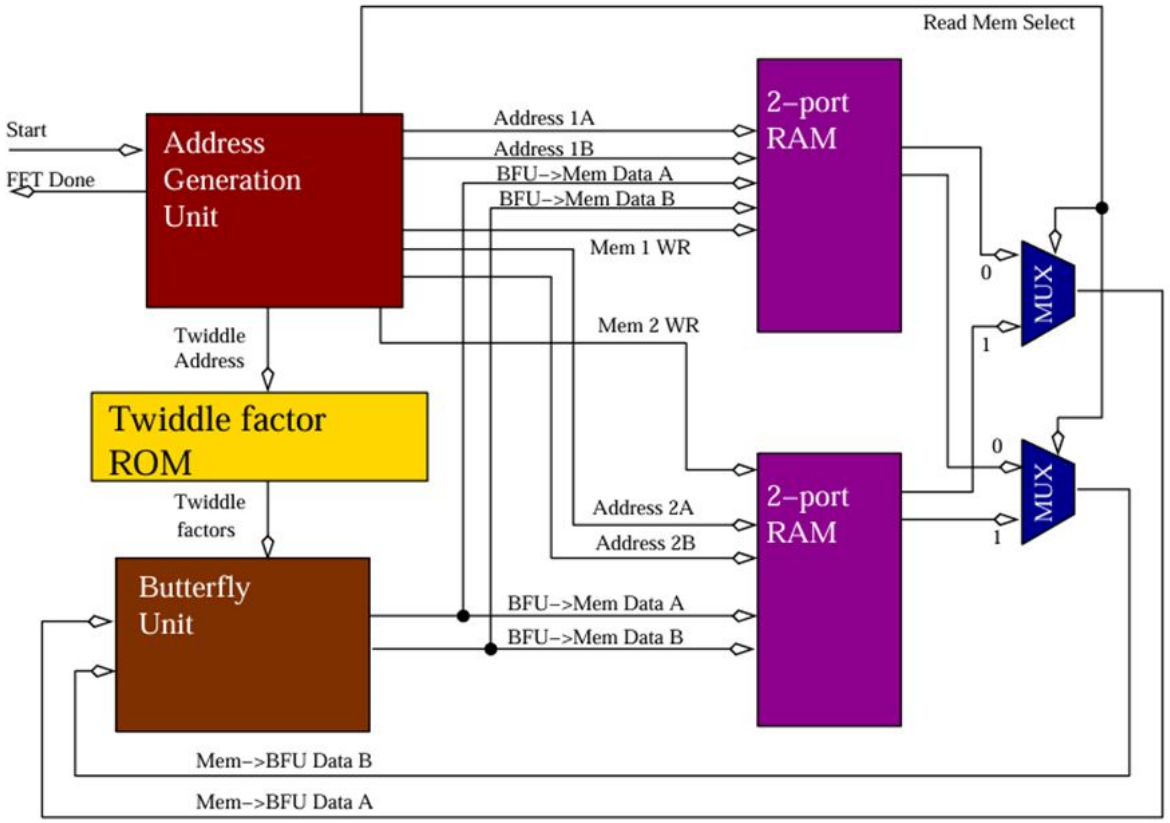
The full transform requires – step 2

2) a “butterfly” operator to do the complex multiply/add,



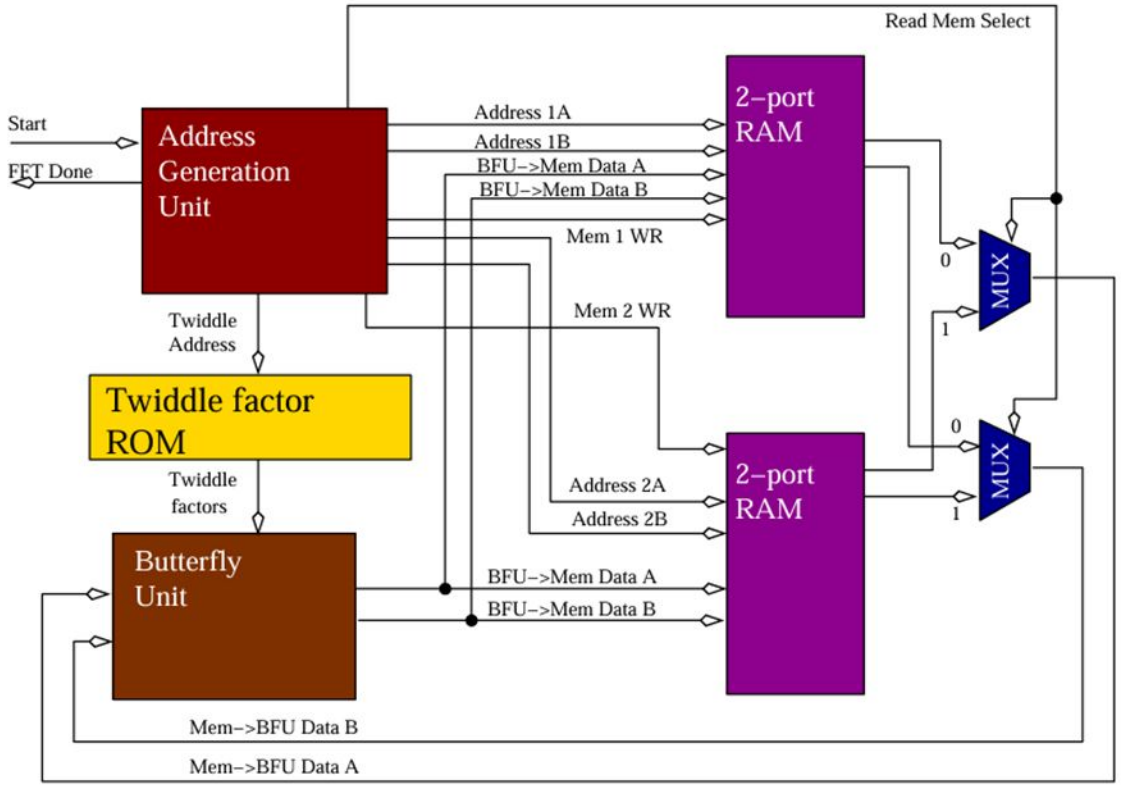
The full transform requires – step 3

3) A memory and



The full transform requires – step 4

4) roots-of-unity(twiddle factor) generator



Limited

Computation time

N	1000	10^6	10^9
N^2	10^6	10^{12}	10^{18}
$Nlog_2N$	10^4	$20\,x\,10^6$	$30\,x\,10^9$

$$10^{18} ns \rightarrow 31.2\, years$$

$$30\,x\,10^9 ns \rightarrow 30\, seconds$$

N	$r = \log_2 N$	**BRUTE FORCE** $4N^2$	**FFT** $2N\log_2 N$	speedup
2	1	16	4	4
4	2	64	16	4
8	3	256	48	5
1,024	10	4,194,304	20,480	205
65,536	16	$1.7 \cdot 10^{10}$	$2.1 \cdot 10^6$	~10^4

Case study 4

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Research problem and Innovation

- Linear interpolation expression:

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l_{pi}) + \frac{\hat{\boldsymbol{H}}(k,l_{p(i+1)}) - \hat{\boldsymbol{H}}(k,l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}),$$

Implementing this expression:

- Step 1 – Develop C or python program
- Step 2 – Simulate the program and verify functionality
- Step 3 – Convert this C code to Assembly code or Op code
- Step 4 – Implement the code on the hardware
- Step 5 – Estimate the processing time, latency, throughput, power
- Step 6 – Optimize

ngstromers Engineering Solutions Limited
IncuBatee of CCUR, CIT

Operations required

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k, l_{pi}) + \frac{\hat{\boldsymbol{H}}(k, l_{p(i+1)}) - \hat{\boldsymbol{H}}(k, l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}),$$

k ∈ {1, 2,...,N}, and column index l ∈ { lpi + 1,lpi + 2,...,lp(i+1) − 1}.

- We see that there are many operations per data item,
  - a division (div),
  - 3 subtraction (sub),
  - a multiplication (mpy),
  - an addition (add) and
  - at least 2 fixed point type cast (cast).

Optimization Techniques

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Optimization process – human intervention

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l_{pi}) + \frac{\hat{\boldsymbol{H}}(k,l_{p(i+1)}) - \hat{\boldsymbol{H}}(k,l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}),$$

k ∈ {1, 2,...,N}, and column index l ∈ { lpi + 1,lpi + 2,...,lp(i+1) − 1}.

- Step 1
  - lpi = 0 is the time index of the left known column and lp(i+1) = Rps is the time index of the right known column, where Rps is the row pilot spacing for that block.
  - l ∈ { lpi + 1,lpi + 2,...,lp(i+1) − 1} □ l ∈ {1, 2,..., Rps − 1}
- Step 2
  - the column index changes to l ∈ 1,2,...,Rps − 1

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l_{pi}) + \frac{\hat{\boldsymbol{H}}(k,l_{p(i+1)}) - \hat{\boldsymbol{H}}(k,l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}), \quad \Rightarrow \quad \hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,0) + \left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right) \cdot \frac{l}{R_{ps}}$$

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, SIT

Optimization process – human intervention

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l_{pi}) + \frac{\hat{\boldsymbol{H}}(k,l_{p(i+1)}) - \hat{\boldsymbol{H}}(k,l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}),$$

k ∈ {1, 2,...,N}, and column index l ∈ { lpi + 1,lpi + 2,...,lp(i+1) − 1}.

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,0) + \left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right) \cdot \frac{l}{R_{ps}}$$

l ∈ {1, 2,..., Rps − 1}

- Step 3
  - The terms l/Rps (Rps is constant and l is variable)
  - For example: for l = 1, 2, 3, 4, 5. …. and Rps = 10
    - l/Rps = 0.1, 0.2, 0.3 …..

l	l/Rps
1	0.1
2	0.2
3	0.3
.	.

LUT (x(l))

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,0) + \left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right) \cdot \boldsymbol{x}(l),$$

Operations required are
- a sub,
- a mpy,
- an add and
- a cast per data item

Angstroms Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Optimization process – human intervention

- Step 4
  - Recursive expression

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,0) + \left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right)\cdot \boldsymbol{x}(l),$$

sub, a mpy, an add and a cast per data item

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l-1) + \left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right)\cdot \dot{\boldsymbol{x}}(l),$$

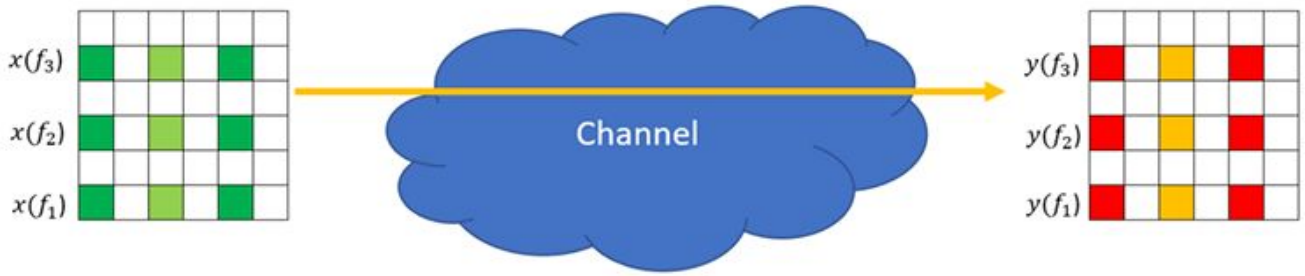
$\dot{\boldsymbol{x}}$ is a new LUT

$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,0) + \sum_{i=1}^{l}\left(\hat{\boldsymbol{H}}(k,R_{ps}) - \hat{\boldsymbol{H}}(k,0)\right)\cdot \dot{\boldsymbol{x}}(i),$$

Requires □ multiply-accumulate (mac) operation efficiently

ngstromers Engineering Solutions Private Limited
Incubatee of C CIT

Channel Estimation



Which can be written as

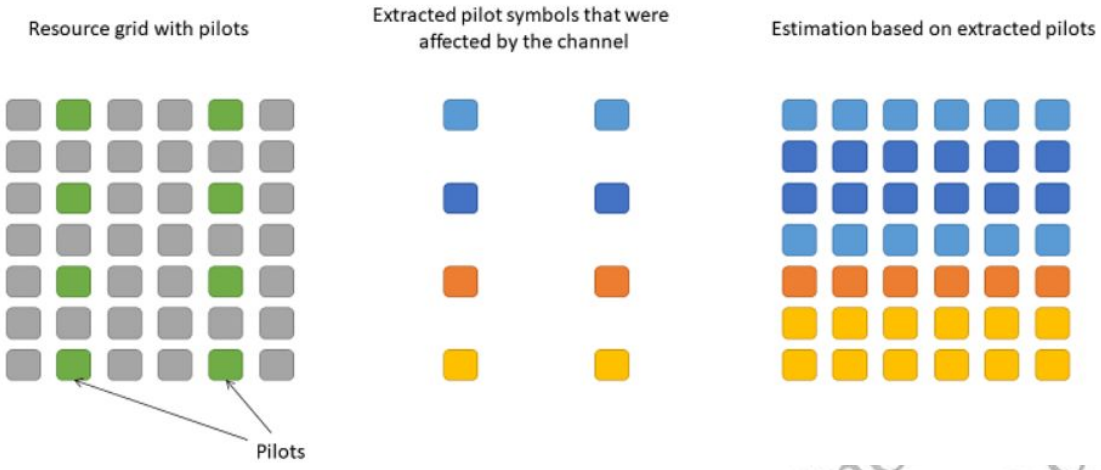
$$y(f_1) = h(f_1) \cdot x(f_1)$$
$$y(f_2) = h(f_2) \cdot x(f_2)$$
$$y(f_3) = h(f_3) \cdot x(f_3)$$

From these equations, $x(f)$ and $y(f)$ are known thus $h(f)$ can be calculated.

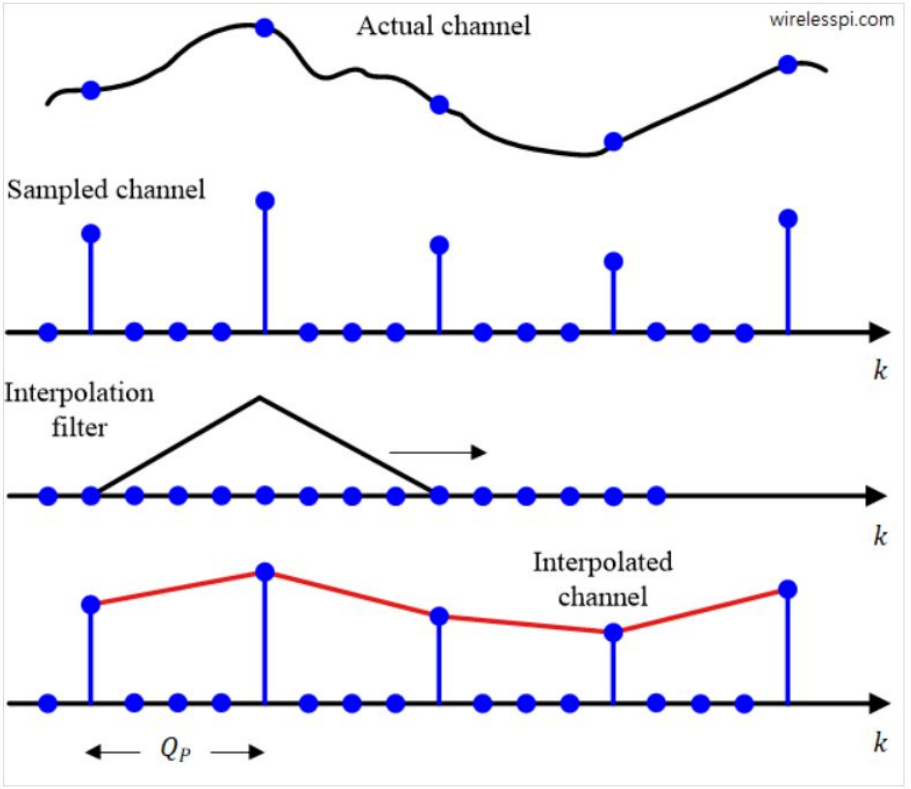
$$h(f_1) = y(f_1) \cdot x^H(f_1)$$
$$h(f_2) = y(f_2) \cdot x^H(f_2)$$
$$h(f_3) = y(f_3) \cdot x^H(f_3)$$

Here $x^H(f)$ is the Hermitian of $x(f)$.

Channel Estimation

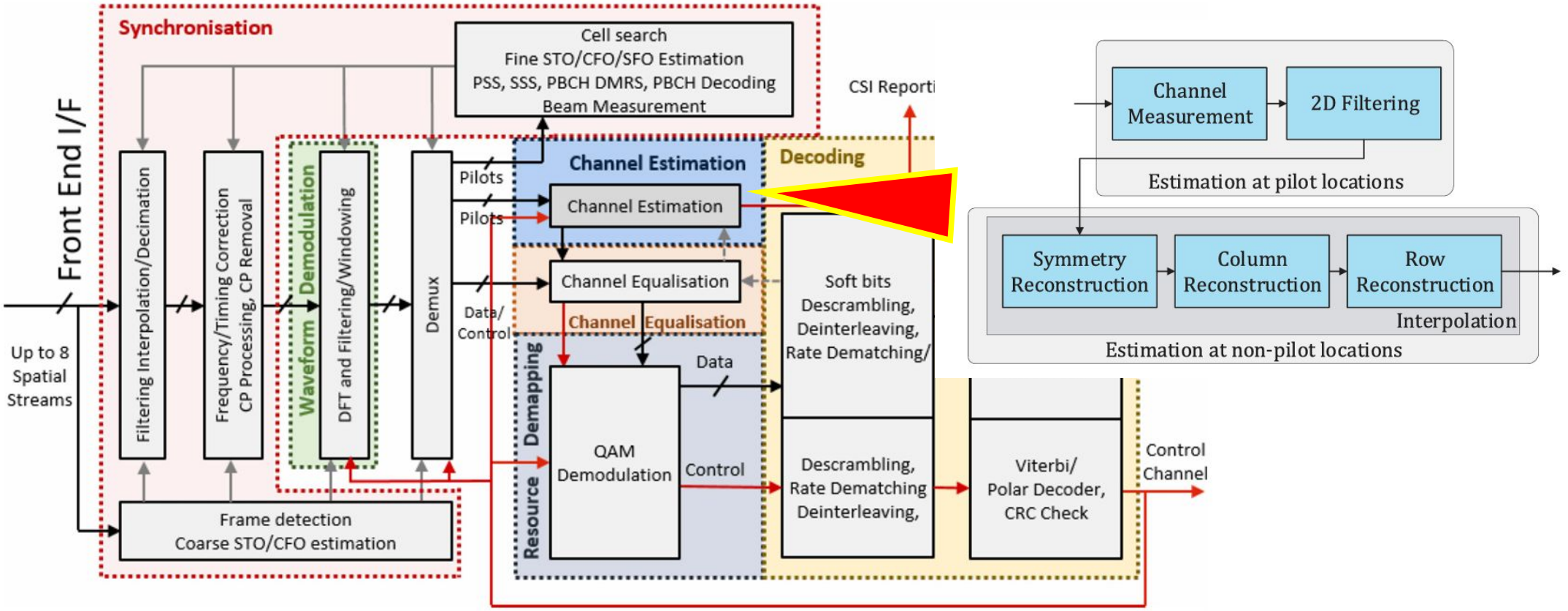


$$\hat{\boldsymbol{H}}(k,l) = \hat{\boldsymbol{H}}(k,l_{pi}) + \frac{\hat{\boldsymbol{H}}(k,l_{p(i+1)}) - \hat{\boldsymbol{H}}(k,l_{pi})}{l_{p(i+1)} - l_{pi}} \cdot (l - l_{pi}),$$



ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

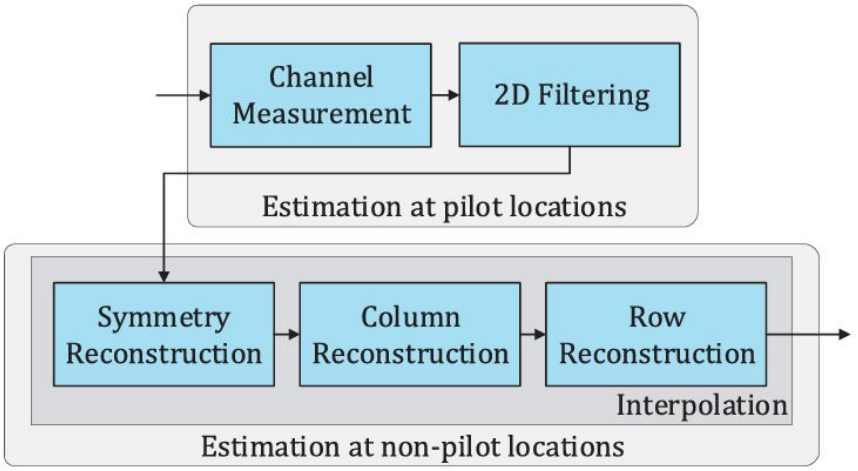
Scope of this work - 3GPP Downlink Receiver DBB PHY System Diagram



ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

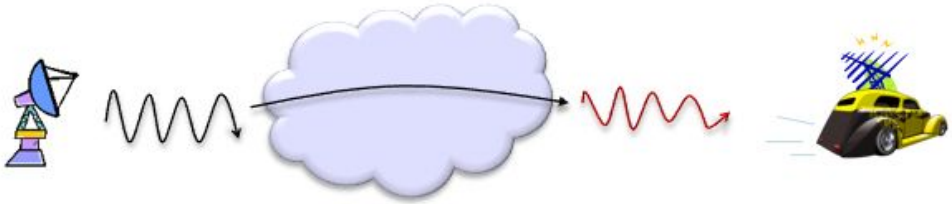
Channel Estimation for Advanced 5G/6G

The process of figuring out channel characteristics is called Channel Estimation.



Channel do the following to the signal going through

- attenuate
- phase-shift
- add noise



How much of the followings are added to the signal at a specific frequency and a specific time ?

- attenuate
- phase-shift
- add noise

What if the frequency and time changes ?

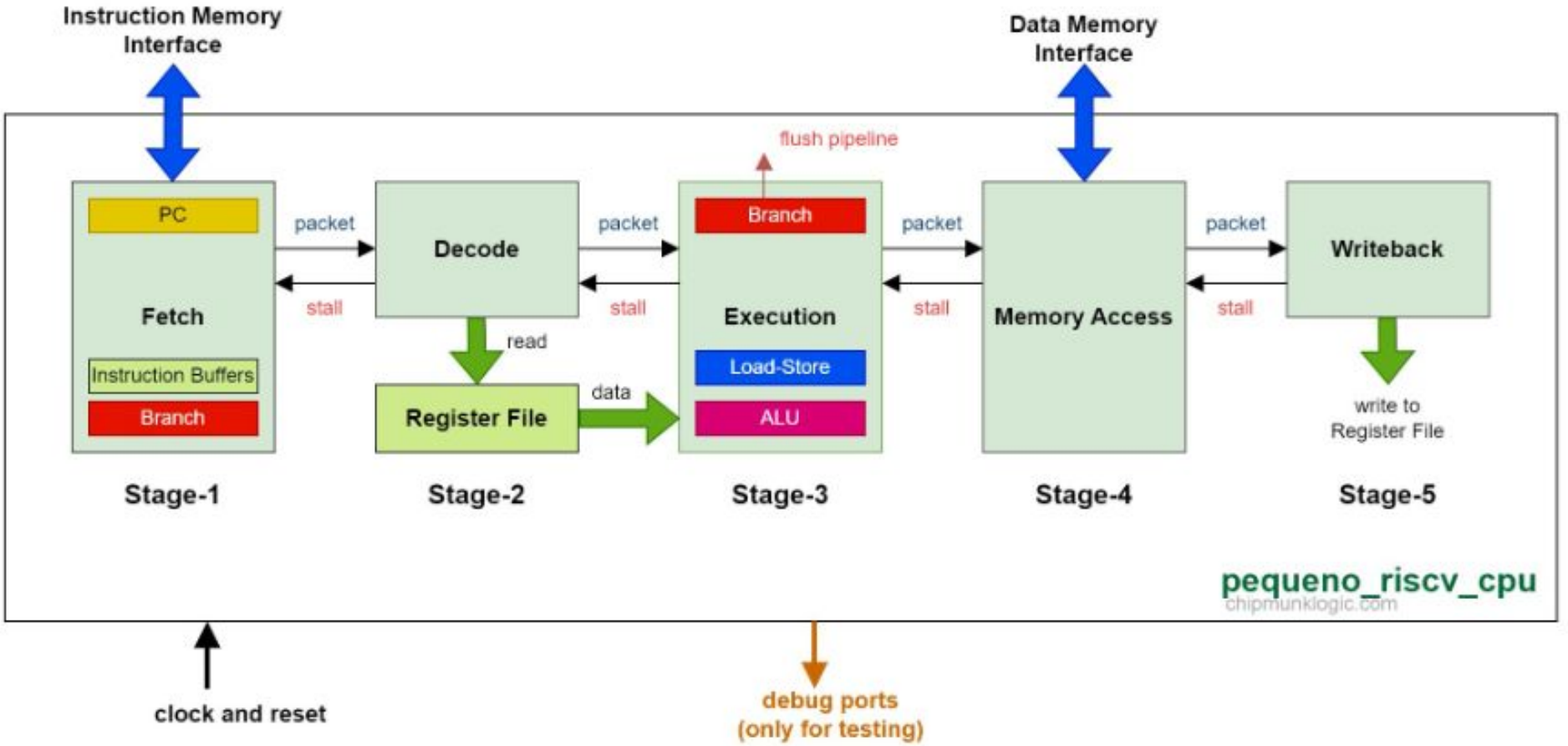
Processing of finding the answer to these questions is called

Channel Estimation

Summary

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

RISCV


Instruction Memory Interface
Data Memory Interface
flush pipeline
PC
Fetch
Instruction Buffers
Branch
packet
stall
Decode
read
Register File
data
Branch
Execution
Load-Store
ALU
Memory Access
Writeback
write to Register File
Stage-1
Stage-2
Stage-3
Stage-4
Stage-5
pequeno_riscv_cpu
chipmunklogic.com
clock and reset
debug ports (only for testing)

RISCV Overview – Five stages of RISCV

Fetch

- Fetch 32-bit instruction from memory
- Increment PC = PC + 4

Decode

- Gather data from the instruction
- Read opcode; determine instruction type, field lengths
- Read in data from register file

Execute

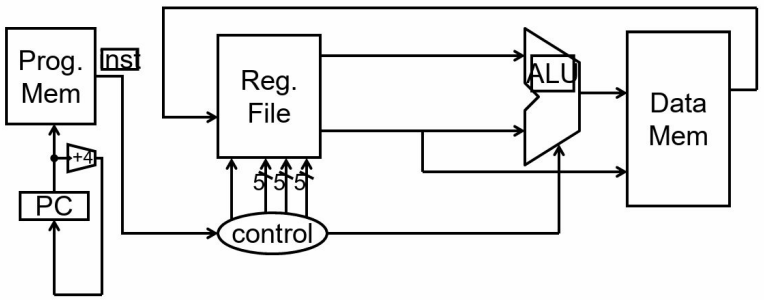
- Useful work done here (+, -, *, /), shift, logic operation, comparison (slt)

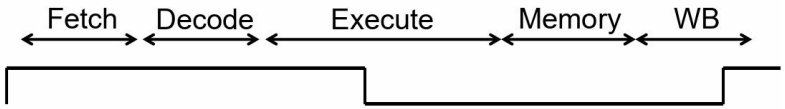
Memory

- Used by load and store instructions only
- Other instructions will skip this stage

Write Back

- Write to register file
- Update PC





A single cycle processor – this diagram is not 100% spatial

Types of Code

- Five types of code

Instruction	Example		Fetch	Decode	Execute	Memory	Write Back
Load	`R1 <- [1]`	address of 1 is loaded to R1	✓	✓	✓	✓	✓
Store	`[A] <- R1`	R1 is stored in address A	✓	✓	✓	✓	X
Branch	`BNE R1, R2, Loop`	R1 ≠ R2, not equal go to Loop	✓	✓	✓	X	X
Jump	`JMP Loop`	directly to the place stated in the code	✓	✓	X	X	X
R type*	`R1 <- R2 + R3`	R2 and R3 are added and stored in R1	✓	✓	✓	X	✓

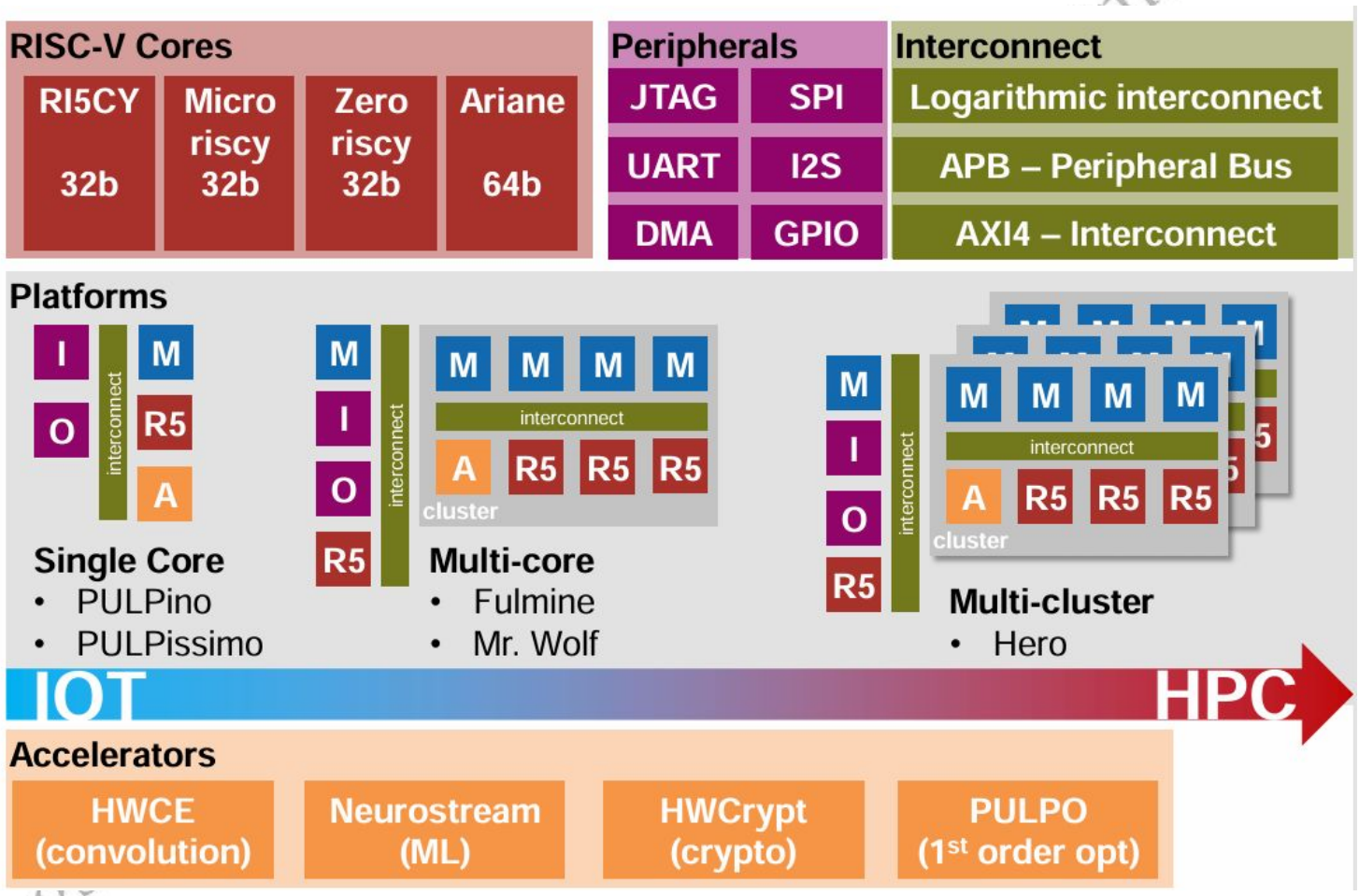
Limited
ngstromers E
Incub

Spike RISC-V ISA Simulator

- Spike is the golden reference functional RISC-V ISA C++ sofware simulator. It provides full system emulation, serves as a starting point for running software on a RISC-V target. Spikes main features:
  - Multiple ISAs: RV32IMAFDQCV extensions
  - Multiple memory models: Weak Memory Ordering (WMO) and Total Store Ordering (TSO)
  - Privileged Spec: Machine, Supervisor, User modes (v1.11)
  - Debug Spec
  - Single-step debugging with support for viewing memory/register contents
  - Multiple CPU support
  - JTAG support
  - Highly extensible (add and test new instructions)

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Parallel Ultra Low Power (PULP)



ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Programming Techniques

- Finite state machines
- Pipelining
- Recursive
- Human interventions for optimizations

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

What Next ?

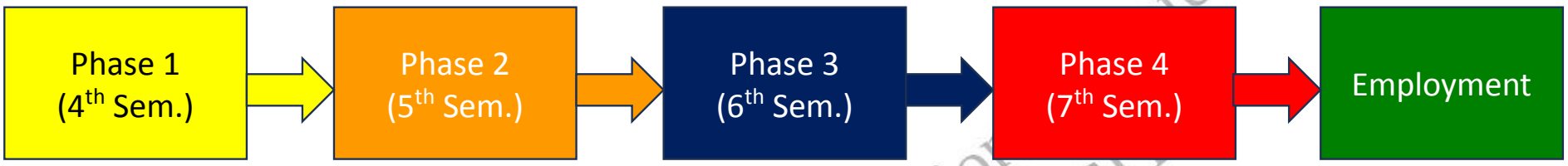
ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Mini Projects

- Individual projects will be assigned
- Total duration will be 4 weeks
  - Every week we will have project tracking and progress work reviews
- Deliverables at the end of 4th week:
  - Project completion report
  - Presentation
  - Poster
  - Github link
- Evaluation by external experts & Certifications – August 1st week
- Commencement of Phase 2 Training
  - RISCV architecture design and FPGA Implementation

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

RISCV Journey @ BMSIT



RISCV Software

RISCV & FPGA

RISCV & ASIC

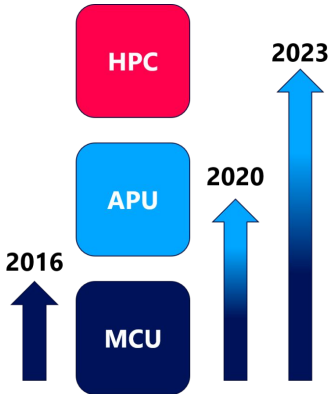
Internship (onboarding) @
- Angstromers
- Gravitas Cynergy
- & other companies

Expected outcomes:
1. 30 projects to be completed in 3 semesters
2. Min. of 2 patent per team
3. Min. of 4 publications per team (two)
4. One product or IP per team

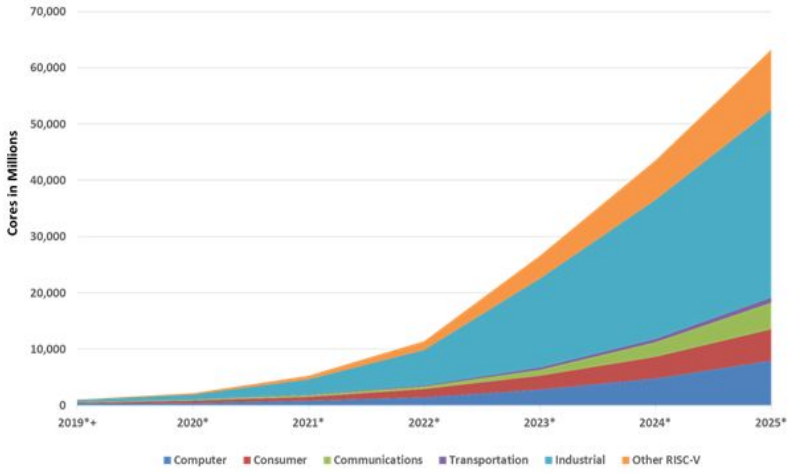
Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CI

Is RISC-V The Future?

- Trend 1: RISC-V is shifting up in performance
  - more and more suppliers will create complex RISC-V cores for high-performance computing in the future
- Trend 2: Barriers between processor types are breaking down
  - RISC-V ISA, having a minimalist base integer instruction set and providing for custom extensions, it is an ideal starting point for creating special accelerators.



- Trend 3: Customers want to avoid a monopoly supplier
  - microprocessors have been dominated by the Intel/AMD X86 duopoly - 1980
  - Arm became the de-facto standard in the mobile phone processor market monopoly- 1990
  - “Arm fatigue” and disquiet with the monopolist position and vendor lock-in in key markets
  - acquisition by SoftBank, that neutrality was seriously eroded, and the possible Nvidia merger may mean that Arm, having singled one of their former customers out for strategic partnership, won’t eventually be considered a safe choice at all.



Source: Semico Research Corp.

Angstromers Engineering Solutions Private Limited
Incubatee of C

Trends in India

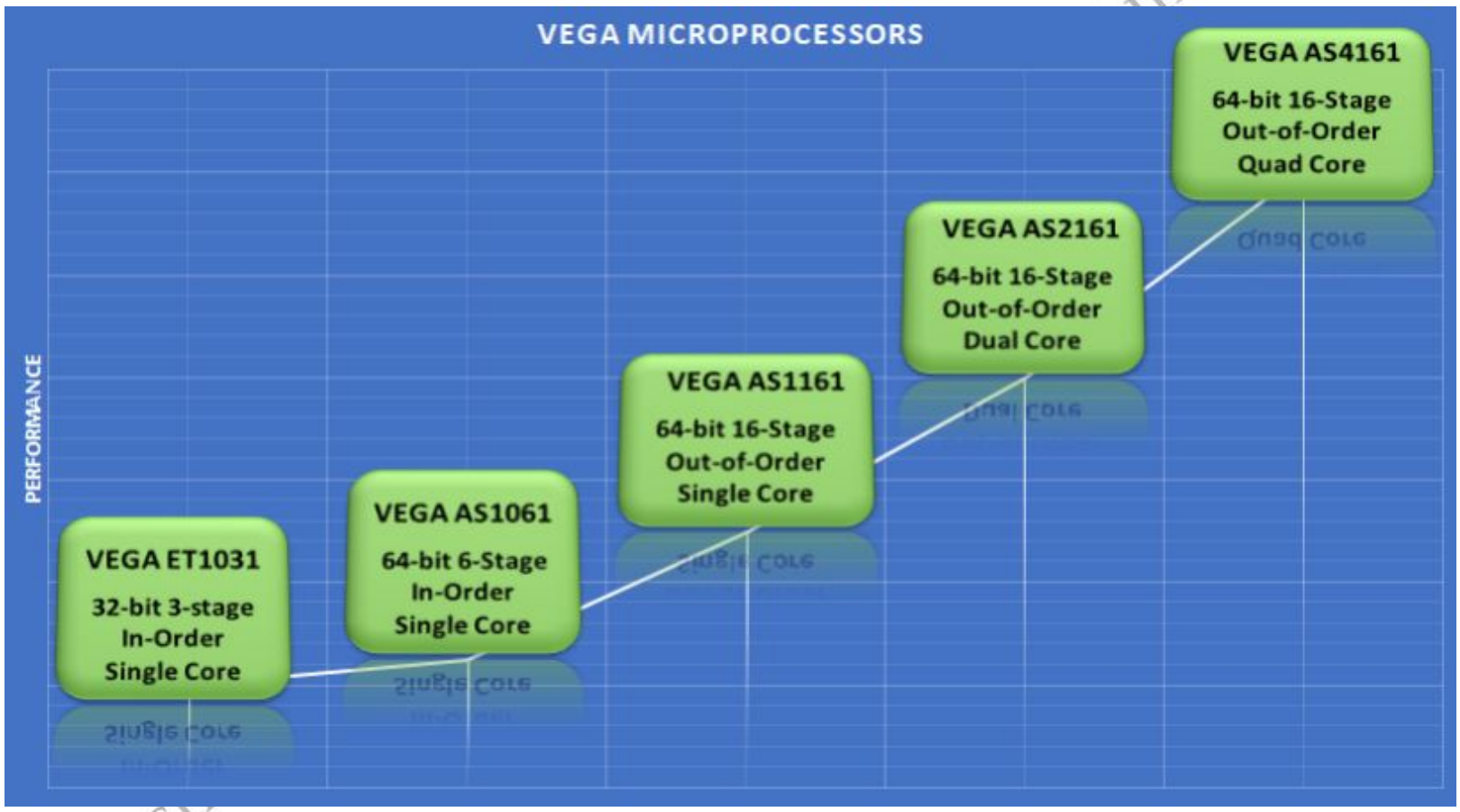
The DIR-V program rests on three pillars that promise to revolutionize India's semiconductor landscape:

1. **Research and Development**: Recognizing the importance of advancing RISC-V design, verification, and testing, the program pledges to support cutting-edge research and development initiatives. By nurturing innovation in this open-source ISA, the DIR-V program aims to drive India towards self-sufficiency in semiconductor technology and establish it as a pioneering force in RISC-V development.
2. **Education and Training**: To harness the full potential of RISC-V, the program emphasizes the need for a skilled workforce well-versed in this architecture. Through targeted education and training programs, students, engineers, and researchers will gain expertise in RISC-V, empowering them to contribute significantly to India's semiconductor industry and drive technological innovation.
3. **Industry Collaboration**: Recognizing the power of collaboration, the DIR-V program seeks to foster strong ties between academia and industry. By promoting the adoption of RISC-V in India's tech sector, the program aims to create a thriving environment that encourages innovation, bolsters economic growth, and solidifies India's position as a global hub for electronics system design and manufacturing.

India's RISC-V Revolution: DIR-V to Create 10,000 Jobs - techovedas

ngstromers Engineering Solutions Limited
Incubatee of CCCIR, CI

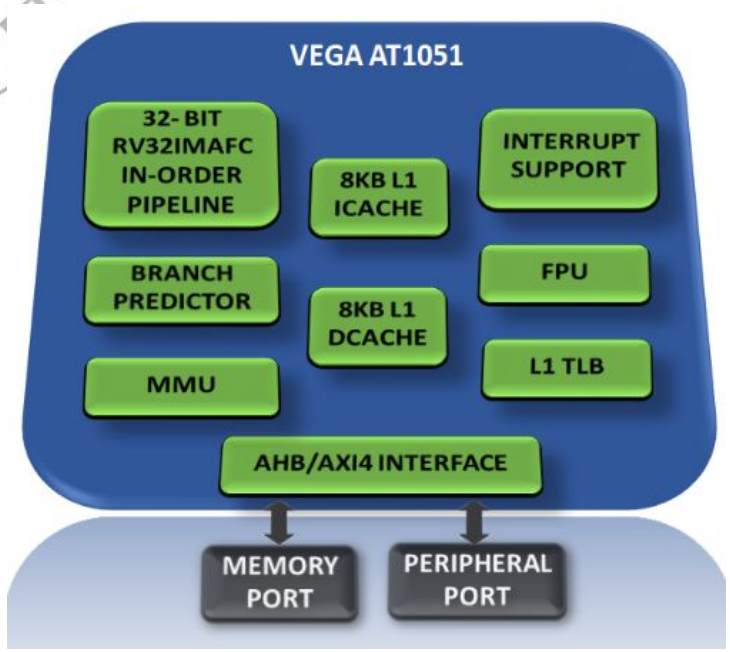
CDAC-VEGA MICROPROCESSORS



ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Key Features	VEGA ET1031	VEGA AS1061	VEGA AS1161	VEGA AS2161	VEGA AS4161
RISC-V ISA	32-bit RV32IM	64- bit RV64IMAFDC	64-bit RV64IMAFD	64-bit RV64IMAFD	64-bit RV64IMAFD
No of cores	1	1	1	2	4
Pipeline	In-order	In-order	Out-of-Order	Out-of-Order	Out-of-Order
Pipeline stages	3-Stage	6-Stage	13-16 Stage	13-16 Stage	13-16 Stage
Superscalar	No	No	Yes	Yes	Yes
Processor modes	Machine	Machine/ Supervisor/User	Machine/ Supervisor/User	Machine/ Supervisor/User	Machine/ Supervisor/User
MMU	Optional	Yes	Yes	Yes	Yes
Debug	Optional	Yes	Yes	Yes	Yes
Branch Predictor	No	Yes	Yes	Yes	Yes
L1 ICache	TIM	8KB	32KB	32KB	32KB
L1 DCache	TIM	8KB	32KB	32KB	32KB
L2 Caches	No	No	No	512KB	1024KB
Bus Interface	AHB/AXI4	AHB/AXI4	AHB/AXI4/ACE	AHB/AXI4/ACE	AHB/AXI4/ACE
IEEE 754-2008 compliant FPU	No	Single and Double precision	Single and Double precision	Single and Double precision	Single and Double precision
Availability	Now	Now	Now	Now	Now

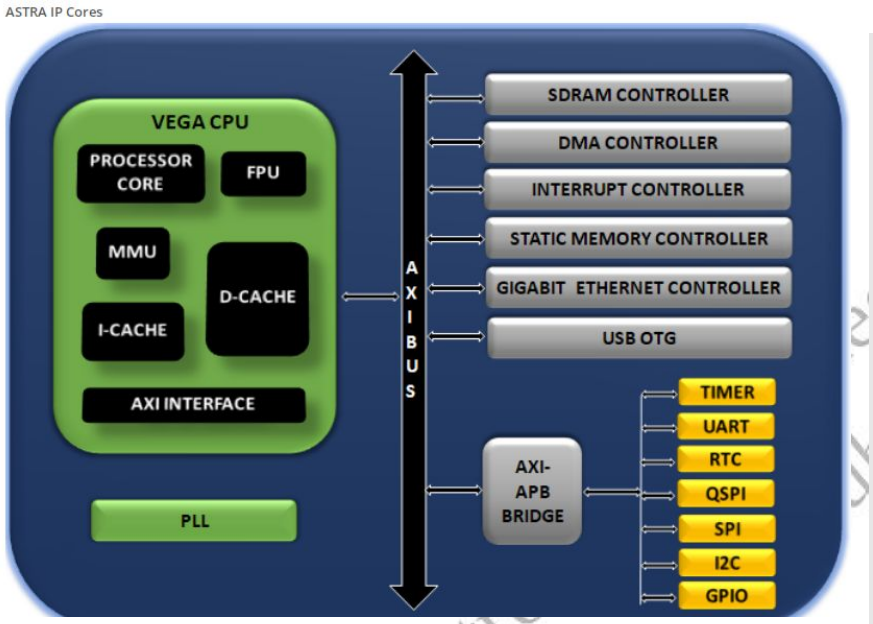
The complete ecosystem for embedded design with the VEGA processors, comprises of Board Support Packages, SDK with integrated tool chain, IDE plug-ins and Debugger for the development, testing and debugging. Linux and FreeRTOS & ZephyrOS Operating Systems have been ported and are available as part of the ecosystem.



ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

IP CORES FROM CDAC

ASTRA SYSTEM & PERIPHERAL IPs



ASTRA IP CORES

- EROTG1
- ERUSBHC
- ERUSB2
- ERPCIe
- ERSATAII
- ERMAC
- ERGMAC
- ER15530
- ERVIC
- ER146818
- ERTIMER
- ER16C450

1.Processor IPs

- VEGA ET1031 (32-bit single core 3-stage in-order RISC-V processor)
- VEGA AS1061 (64-bit single core 6-stage in-order RISC-V processor)
- VEGA AS1161 (64-bit single core 16-stage pipeline out-of-order RISC-V processor)
- VEGA AS2161 (64-bit dual core 16-stage pipeline out-of-order RISC-V processor)
- VEGA AS4161 (64-bit quad core 16-stage pipeline out-of-order RISC-V processor)

2. System IPs

- ERPLIC - Interrupt Controller
- ERDMA - Direct Memory Access Controller

3. Peripheral IPs

- EROTG1 – USB2.0 On-the-Go Controller
- ERUSBHC – USB2.0 Host Controller
- ERUSB2 – USB2.0 Function Controller
- ERMAC - Ethernet Media Access Controller (10/100 Mbps)
- ERGMAC - Gigabit Ethernet Media Access Controller
- ERTIMER - Configurable Timer
- ER16C450 – UART
- ERSPI - SPI Controller
- ERI2C - I2C Controller
- ERPWM - PWM Controller
- ERGPIO - GPIO Controller
- ERSDHOST - SD Host Controller
- ERQSPI - QSPI Controller
- ERSMC - Static Memory Controller (SRAM/NOR)

Angstroms Private Limited
CCCIR CIT

Products

Development Boards

ARIES v2.0

ARIES MICRO v1.0

ARIES IOT v2.0

- **THEJAS32 SoC -** is based on the VEGA ET1031 processor, which is a 32 bit single core in-order, 3-stage pipeline processor.

ngstromers Engineering Solutions Private Limited
Incubated at CCCIR, CIT

Thank You

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT