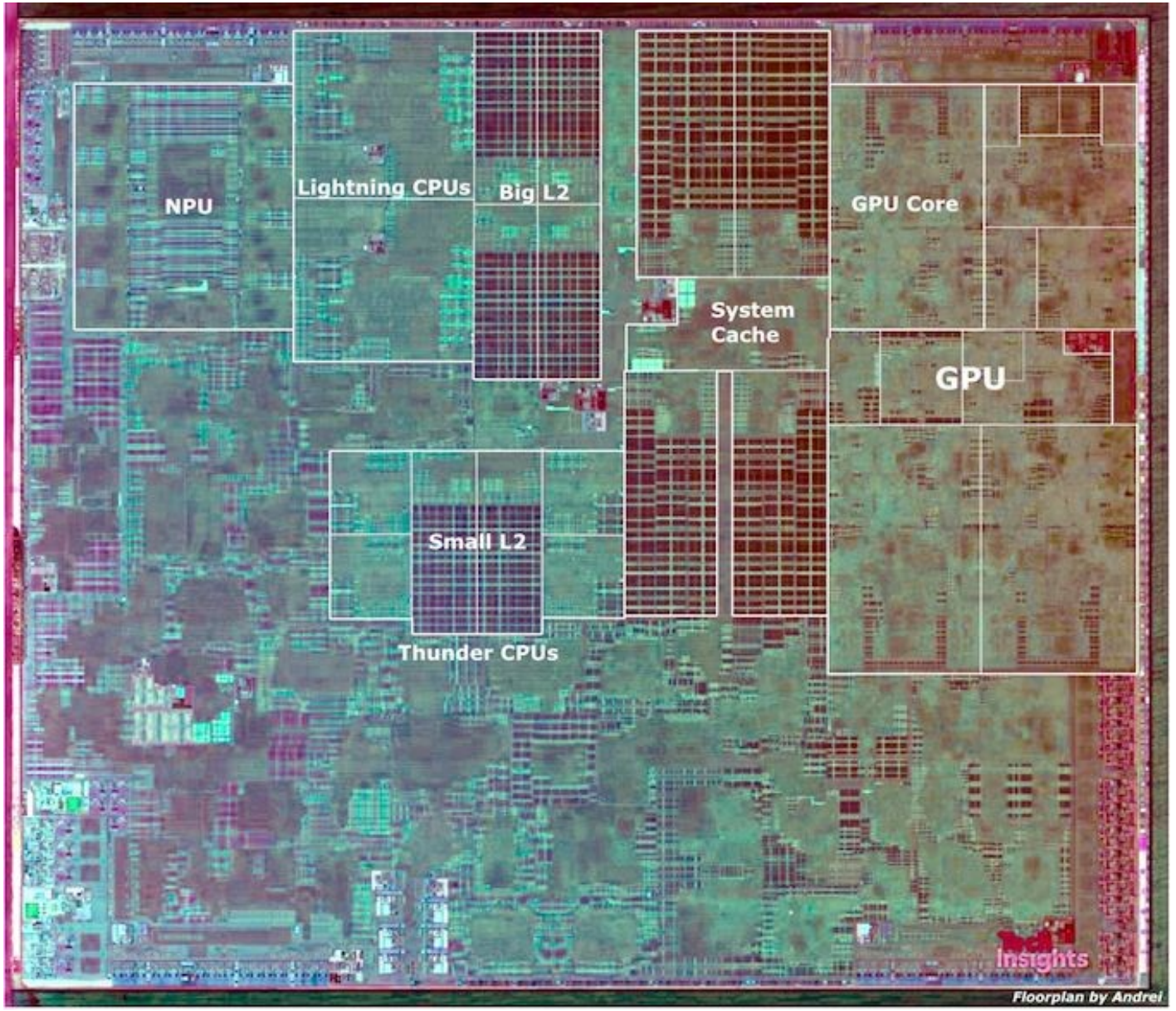
ENS L3 : "Systèmes numériques : de l'algorithme aux circuits"

Opérateurs spéciaux

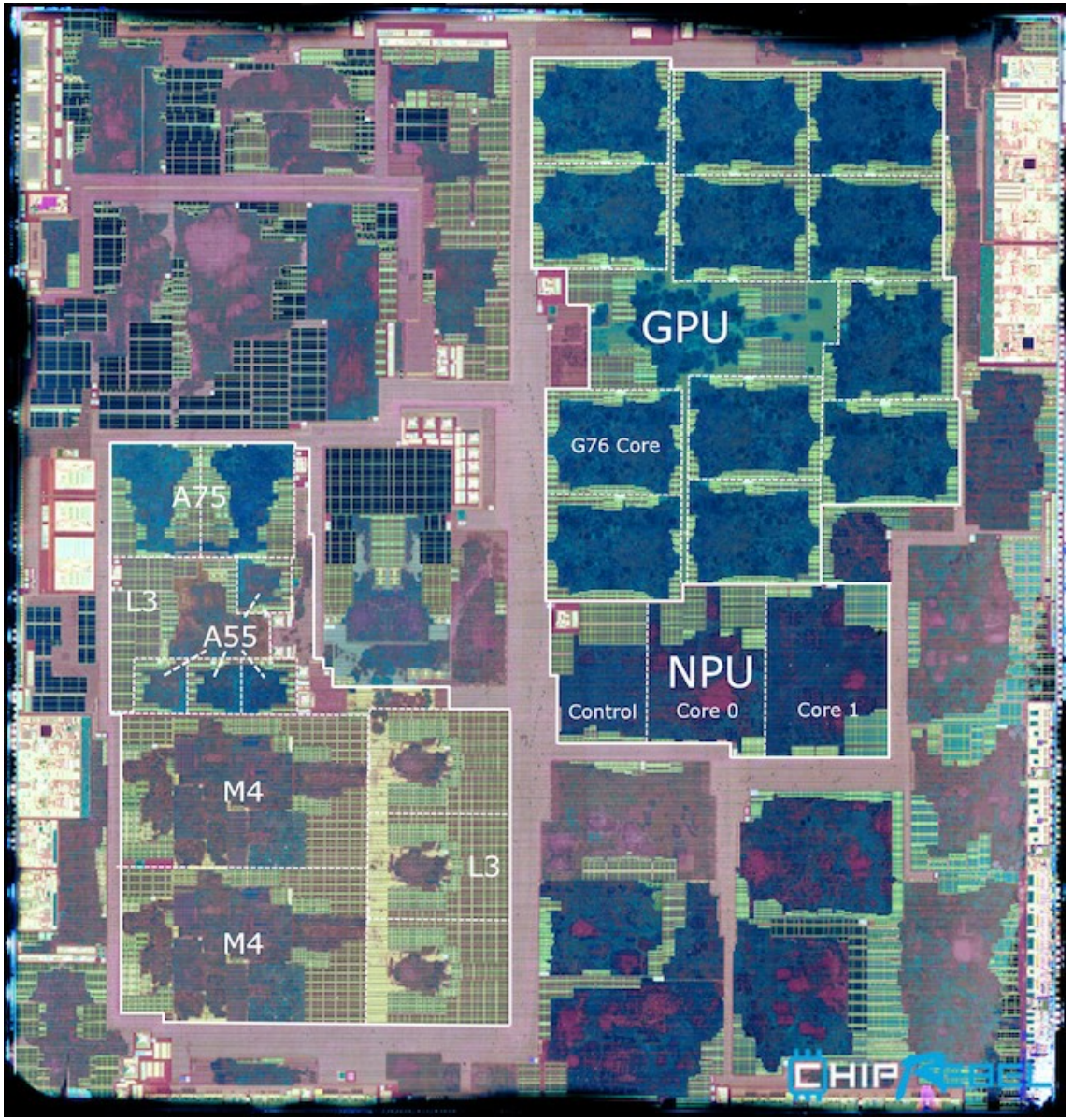
Sumanta Chaudhuri

28 Nov. 2023

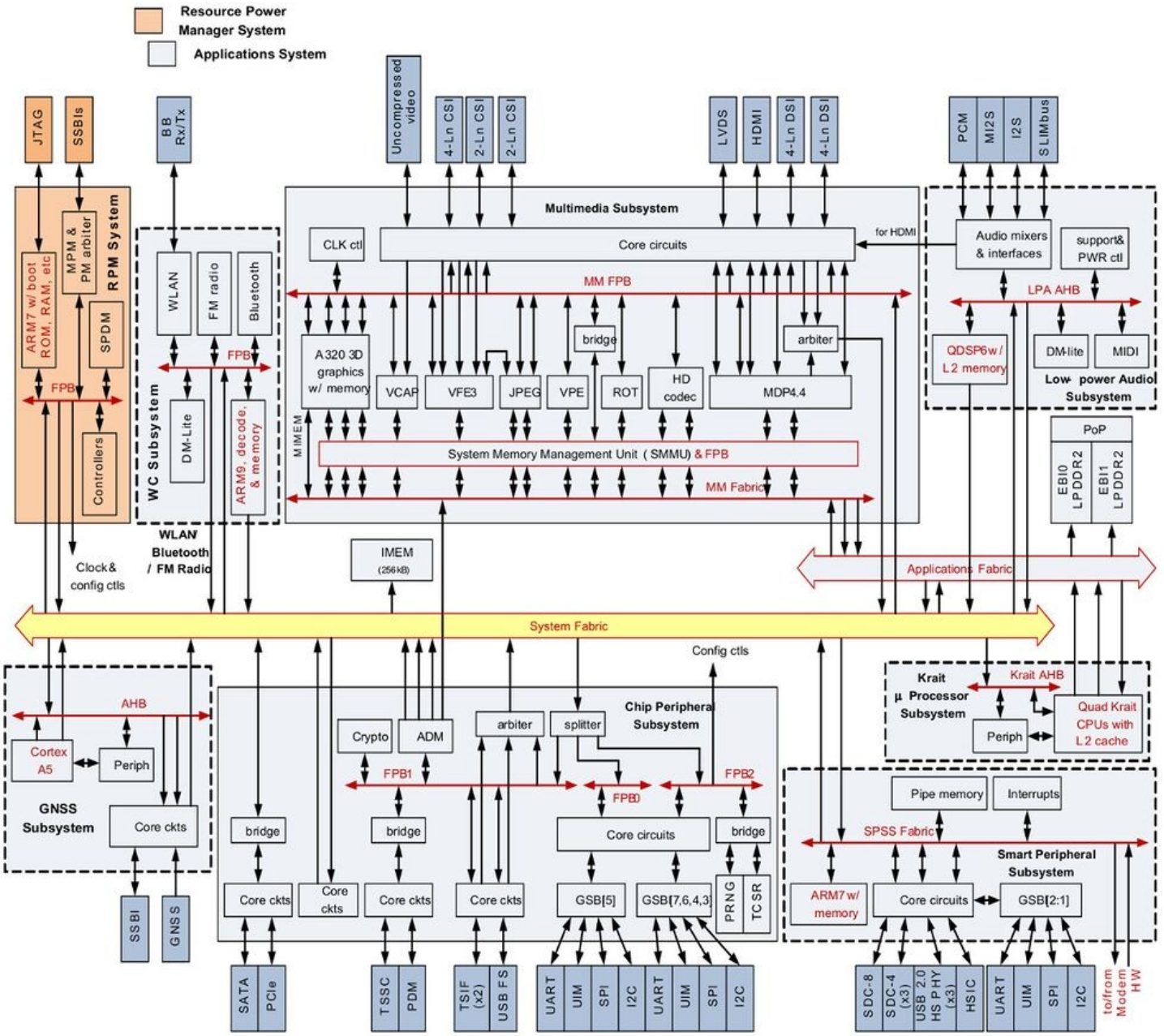
Apple A13 Chip



Samsung Exynos 9820



Qualcomm SnapDragon



Agenda

- CPU
  - Micro-Controllers
  - Multi Cores
- GPU
- FPGAs
- ASICs
  - Most popular functions can aspire to become ASIC.
  - Example ??

Agenda

- ASIC Domains
  - Telecommunications ‘60s -now
  - Cryptography ‘60s - now
  - Computer Graphics ‘80s- now
  - Neural Networks ‘2012 - now

Agenda

- Speicific Operator examples
  - Telecom
    - CRC
    - CORDIC
    - FFT
    - FIR
  - Graphics
    - Bresenham Algorithm
  - AI
    - Convolutional Neural Networks
- Alternative Logic Styles
  - Asynchronous Logic
  - Multi-Valued Logic

CRC : Cyclic Redundancy Checksum

- Detection of errors in transmission
- A cyclic polynomial code.
- Every word is a polynomial in GF(2).
- e.g code word 11000101 is $X^7+X^6+0\cdot X^5+0\cdot X^4+0\cdot X^3+X^2+0\cdot X+1$
- Every valid code is a multiple of the generator polynomial. (prime polynomial)
- All arithmetic is done in GF(2)

CRC

- Given a message polynomial M(x) deg. m
- And a generator polynomial G(x) deg. m
- $M(x).x^n=G(x)P(x)+r(x)$ r(x) : remainder polynomial
- Transmit $M(x).x^n$- r(x)
  - M(x) is m significant bits , r(x) is in least significant bits

- Verify : divide by G(x) and verify that remainder is 0

CRC

- GSM X^3+X+1
- Ethernet/Wifi : $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^8+X^7+X^5+X^4+X^2+X+1$
- WCDMA: $X^{24}+X^{23}+X^6+X^5+X+1$
- Bluetooth $X^{16}+X^{12}+X^5+1$
- RFID X^5+X^3+1
- Implementation Efficace ?

Polynomial Division

- Let’s divide

$$X^7+X^6+0\cdot X^5+0\cdot X^4+0\cdot X^3+X^2+0\cdot X+1$$

  By X+1.

- What is the method ?

CRC : Cyclic Redundancy Checksum

$(1x^3 + 0x^2 + 1x + 1)$ → 1011

```
11010011101100 000 <--- input right padded by 3 bits
1011               <--- divisor
01100011101100 000 <--- result (note the first four bits are the XOR with the divisor
beneath, the rest of the bits are unchanged)
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000 <--- note that the divisor moves over to align with the next 1 in the
dividend (since quotient for that step was zero)
       1011            (in other words, it doesn't necessarily move one bit per
iteration)
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------
00000000000000 100 <--- remainder (3 bits).  Division algorithm stops here as dividend is
equal to zero.
```

Nota bene : 1-bit CRC = parity bit

CRC : Cyclic Redundancy Checksum

$(1x^3 + 0x^2 + 1x + 1)$ → polynomic division remainer

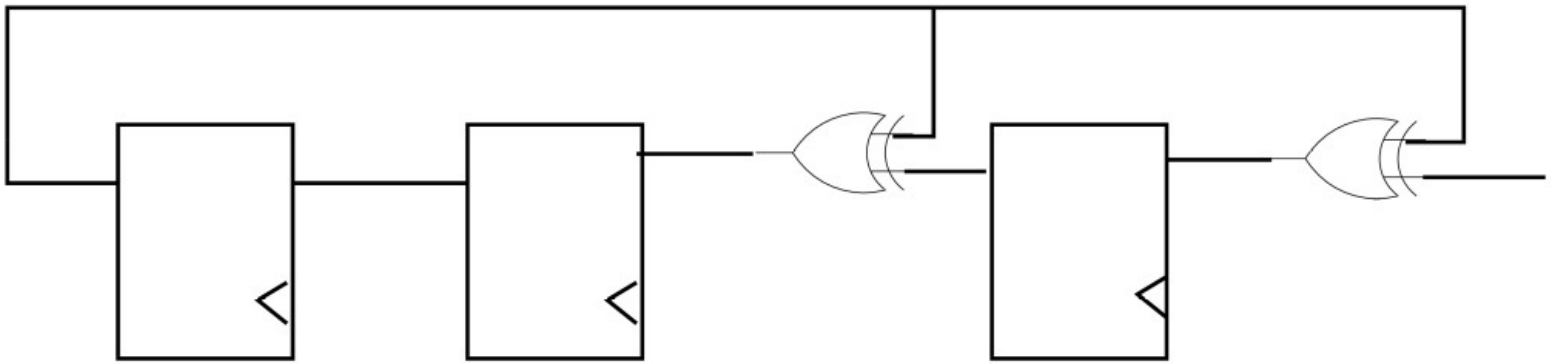
```
> R<X> := PolynomialRing(GF(2));

> P := X^3 + X + 1;
> IsIrreducible(P);
true
```

← magma code

```
11010011101100 100 <--- input with check value
1011               <--- divisor
01100011101100 100 <--- result
 1011              <--- divisor ...
00111011101100 100


......

00000000001110 100
          1011
00000000000101 100
           101 1
-----------------
               0 <--- remainder
```

Nota bene : 1-bit CRC = parity bit

CRC Hardware



- CRC est un division polynomial GF(2)
  Décalage a Gauche
  Quand MSB=1, faite XOR avec le diviseur
- LFSR (Linear Feedback Shift Registers)
- Galois LFSR for Polynomial division in GF(2)
- X ^ 1 = ~X ; X^0= X

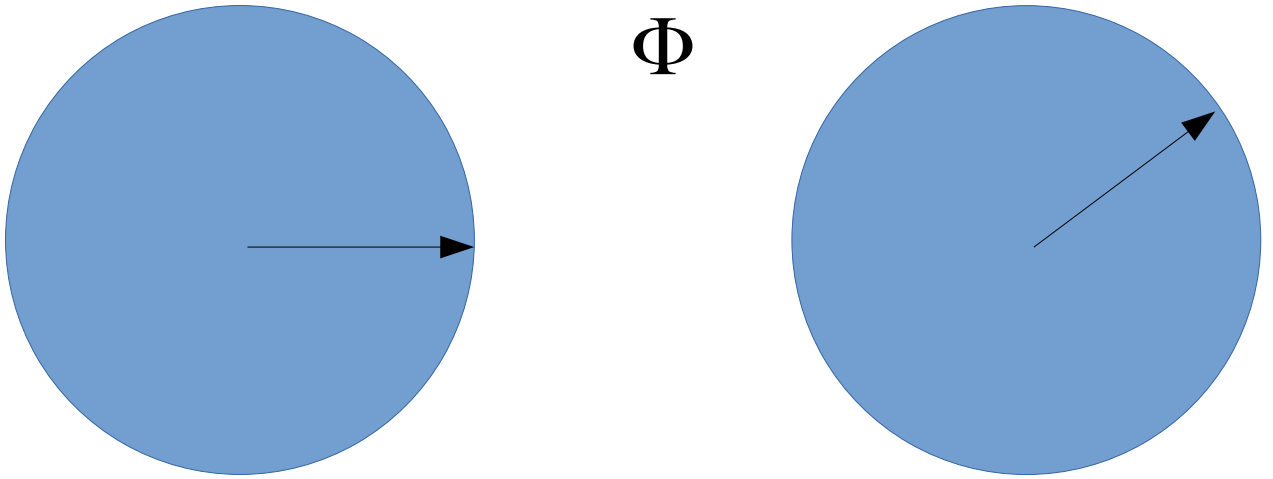
```
11010011101100 000
1011
0110011101100 000
 1011
 0111011101100 000
  1011
  010111101100 000
```

CORDIC

- Comment calculer le valeur de cos(x) et sin(x) ?
- Comment calculer le valeur de cos(x) et sin(x) sans multiplication?

CORDIC : Coordinate Rotation Digital Computer

- Deprettere, E., Dewilde, P., and Udo, R., "Pipelined CORDIC Architecture for Fast VLSI Filtering and Array Processing," Proc. ICASSP'84, 1984, pp. 41.A.6.1-41.A.6.4
- In 2D plane (x,y) → (x',y')
-



CORDIC

$$x' = x\cos\phi - y\sin\phi$$
$$y' = y\cos\phi + x\sin\phi$$

$$x' = \cos\phi \cdot [x - y\tan\phi]$$
$$y' = \cos\phi \cdot [y + x\tan\phi]$$

$$\tan(\phi) = \pm 2^{-i}$$

Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations. If the decision at each iteration, i, is which direction to rotate rather than whether or not to rotate, then the $\cos(\delta_i)$ term becomes a constant (because $\cos(\delta_i) = \cos(-\delta_i)$). The iterative rotation can now be expressed as:

$$x_{i+1} = K_i\left[x_i - y_i \cdot d_i \cdot 2^{-i}\right]$$
$$y_{i+1} = K_i\left[y_i + x_i \cdot d_i \cdot 2^{-i}\right]$$

where:

$$K_i = \cos\left(\tan^{-1} 2^{-i}\right) = 1/\sqrt{1+2^{-2i}}$$
$$d_i = \pm 1$$

CORDIC

Lemma 1.

$$\cos(\arctan x) = \frac{1}{\sqrt{1+x^2}}.$$

CORDIC

Lemma 1.

$$\cos(\arctan x) = \frac{1}{\sqrt{1+x^2}}.$$

Proof.

$$\begin{aligned} \cos^2 y + \sin^2 y &= 1 \\ \implies 1 + \tan^2 y &= \frac{1}{\cos^2 y} \\ \implies \cos^2 y &= \frac{1}{1+\tan^2 y} \end{aligned}$$

apply formula for $y = \arctan x$. Notice that when $x \geq 0$, $\arctan x \geq 0$, hence $\cos(\arctan x) \geq 0$. □

CORDIC

CORDIC equations are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}\left(2^{-i}\right)$$

where

$$d_i = -1 \text{ if } z_i < 0, \ +1 \text{ otherwise}$$

which provides the following result:

$$x_n = A_n\left[x_0 \cos z_0 - y_0 \sin z_0\right]$$

$$y_n = A_n\left[y_0 \cos z_0 + x_0 \sin z_0\right]$$

$$z_n = 0$$

$$A_n = \prod_n \sqrt{1 + 2^{-2i}}$$

```python
import math

A = 1
for i in range(0,10):
  A *= math.sqrt( 1+2**(-2*i) )
  print( "A_{i} = {A:.3f}".format( i=i, A=A ))
```

```
A_0 = 1.414
A_1 = 1.581
A_2 = 1.630
A_3 = 1.642
A_4 = 1.646
A_5 = 1.646
A_6 = 1.647
A_7 = 1.647
A_8 = 1.647
A_9 = 1.647
```

Cordic : Structure

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}\left(2^{-i}\right)$$

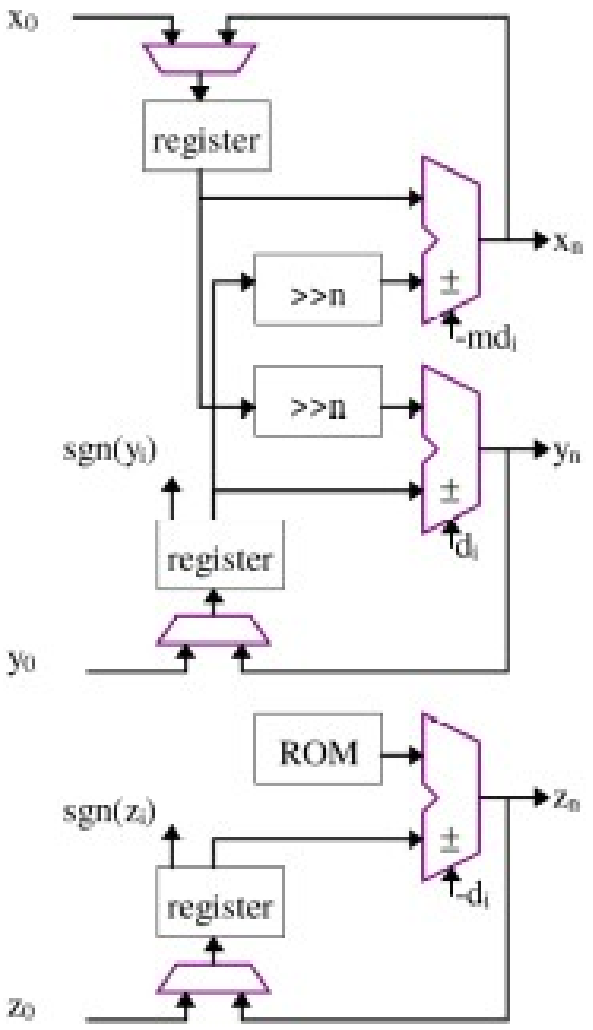


Figure 1. Iterative CORDIC structure

FFT

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

where N is the size of the vectors, $w = e^{2i\pi/N}$ are the "roots-of-unity" (twiddle factors), and $0 \leq m < N$.

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

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

FFT (Coût Multiplication)

$$X[k]=\sum_{n=0}^{N-1} x(n).W_N^{kn} \qquad k=0,1,2....N \qquad W_N=e^{\frac{-j2\Pi}{N}}$$

DFT à N point, Chaque point (X[k], k= 0,1,2…. N) requiers N multiplications.

Nombre de Multiplications Nécessaire : $O(N^2)$

Idée d'optimisation : (Cooley-Tukey)

Diviser un DFT à N point en Partie Paire et Impaire.

Chaque partie peut être réprésenté comme DFT à N/2 point.

FFT (DIT)

$$X[k]=\sum_{m=0}^{\frac{N}{2}-1} x(2m).W_N^{k2m}+\sum_{m=0}^{\frac{N}{2}-1} x(2m+1).W_N^{k(2m+1)}$$

$$X[k]=\sum_{m=0}^{\frac{N}{2}-1} s_1(m)W_N^{k2m}+\sum_{m=0}^{\frac{N}{2}-1} s_2(m).W_N^{k(2m+1)}$$

$$X[k]=\sum_{m=0}^{\frac{N}{2}-1} s_1(m)W_{\frac{N}{2}}^{km}+W_N^k\sum_{m=0}^{\frac{N}{2}-1} s_2(m).W_{\frac{N}{2}}^{k(m)}$$

$$X[k]=S_1[k]+W_N^k S_2[k] \qquad k=0,1,2....N$$

Nombre totale de multiplications= $2\left(\frac{N}{2}\right)^2+\frac{N}{2}$

~50 % d'economie sur les multiplieurs.

$$W_N^{k+N/2}=-W_N^k$$
$$W_N^{K+N}=W_N^K$$
$$W_N^2=W_{\frac{N}{2}}$$

FFT (DIT)

$$X[k] = S_1[k] + W_N^k S_2[k] \qquad k = 0,1,2.... \frac{N}{2} - 1$$

$$X[k + N/2] = S_1[k] - W_N^k S_2[k] \qquad k = 0,1,2.... N$$

Appliquons cette technique récursivement.

$$N, \frac{N}{2}, \frac{N}{4}, \frac{N}{8} ....2$$

Nombre. De Multiplicaitons : $O(N \log(N))$

FFT example on 8 bits

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

FFT example on 8 bits

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

FFT example on 8 bits

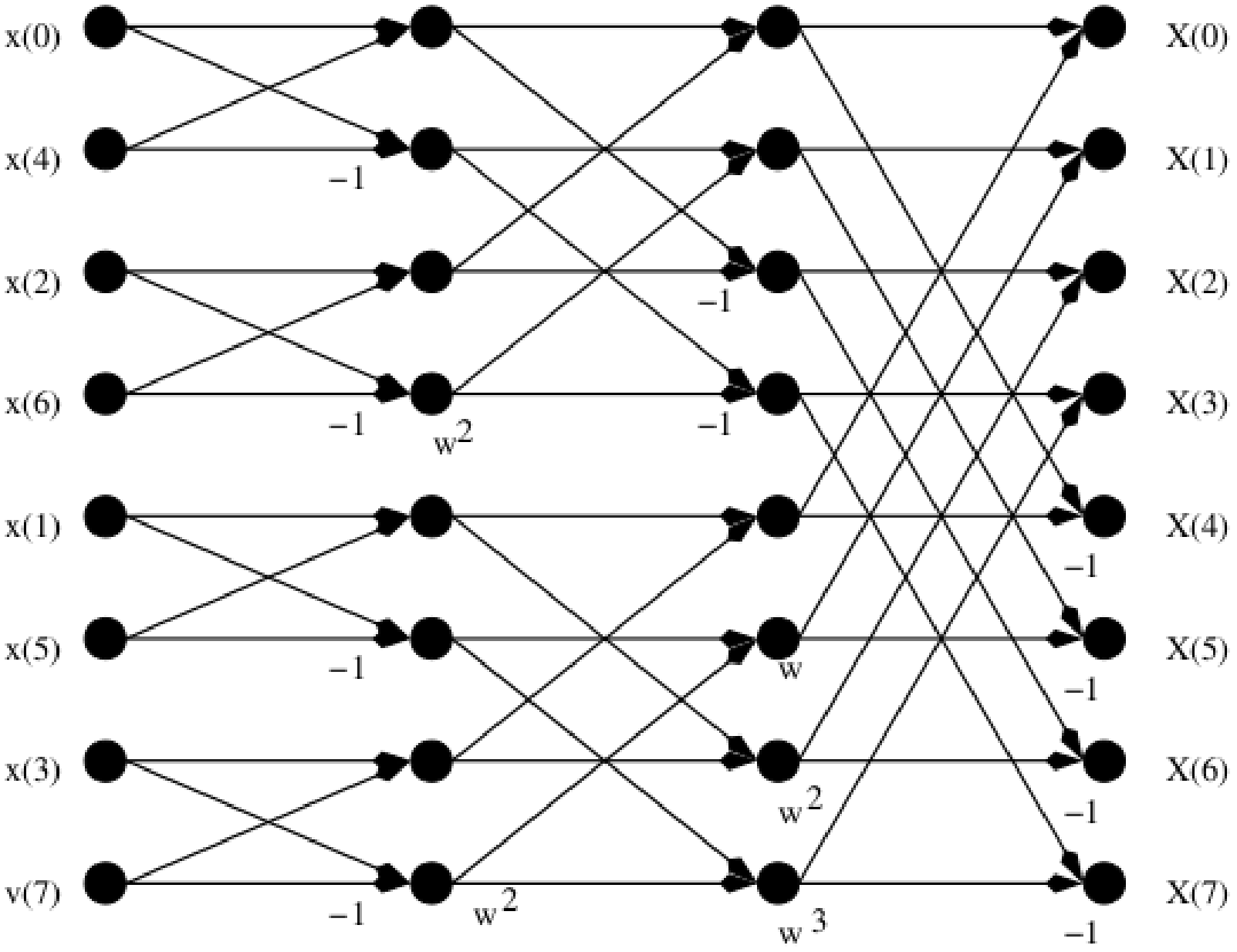
$$
\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{pmatrix}
=
\left[\begin{array}{cc|cc|cc|cc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & w^4 & w^2 & w^6 & w & w^5 & w^3 & w^7 \\
\hline
1 & w^8 & w^4 & w^{12} & w^2 & w^{10} & w^6 & w^{14} \\
1 & w^{12} & w^6 & w^{18} & w^3 & w^{15}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

FFT example on 8 bits

$$
\begin{aligned}
w^n &= w^{n+Nk}, N = 8, k = 0, 1, 2, \cdots \\
w^n &= -w^{n+N/2} \\
w^{Nk} &= 1.
\end{aligned}
$$

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

Final architecture



FFT example on 8 bits

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

FFT example on 8 points

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

$$[A_0] = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad [A_1] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & w^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -w^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & w^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -w^2 \end{bmatrix} \quad [A_2] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & w^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & w^3 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -w & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -w^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -w^3 \end{bmatrix}$$

See also: https://svn.comelec.enst.fr/trusted_computing/courses/SG_ENS/20161122/pdf/fft.sage

DCT : Discrete Cosine Transform

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \ldots, N-1.$$

Used in JPEG image coding.

FIR Filter

- Discrete Convolution.
- where:
-
- x[n] is the input signal,
- y[n] is the output signal,
- N is the filter order;
- b_{i} is the value of the impulse response at the i'th instant for {\textstyle 0\leq i\leq N}{\textstyle 0\leq i\leq N} of an {\textstyle N^{\text{th}}}{\textstyle N^{\text{th}}}-order FIR filter. If the filter is a direct form FIR filter then {\textstyle b_{i}}{\textstyle b_{i}} is also a coefficient of the filter.

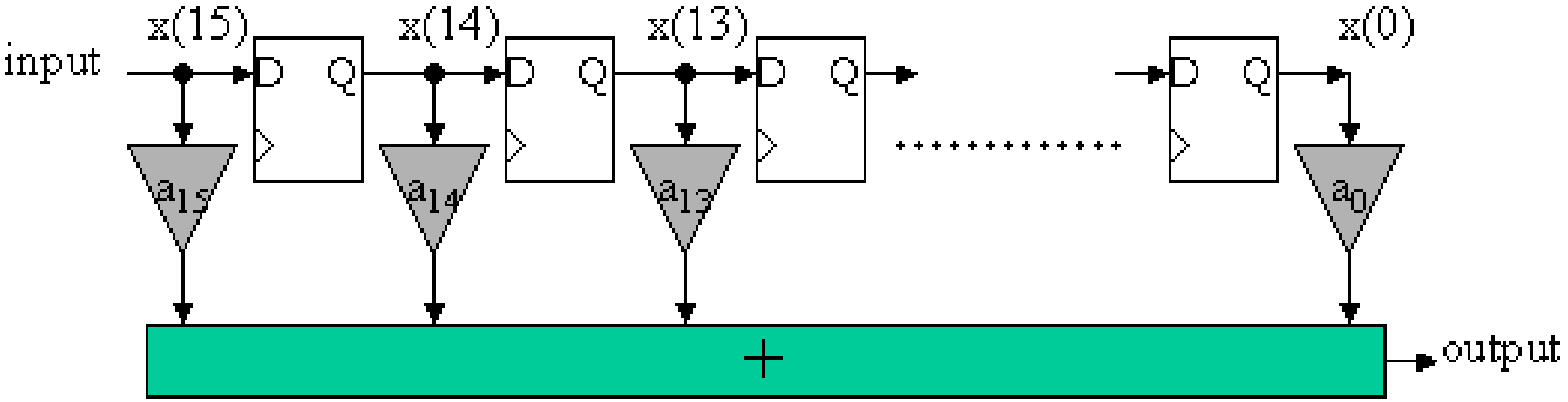
Finite Impulse Response Filter

$$output(n) = \sum_{i=0}^{15} a_i \cdot x(n-15+i)$$

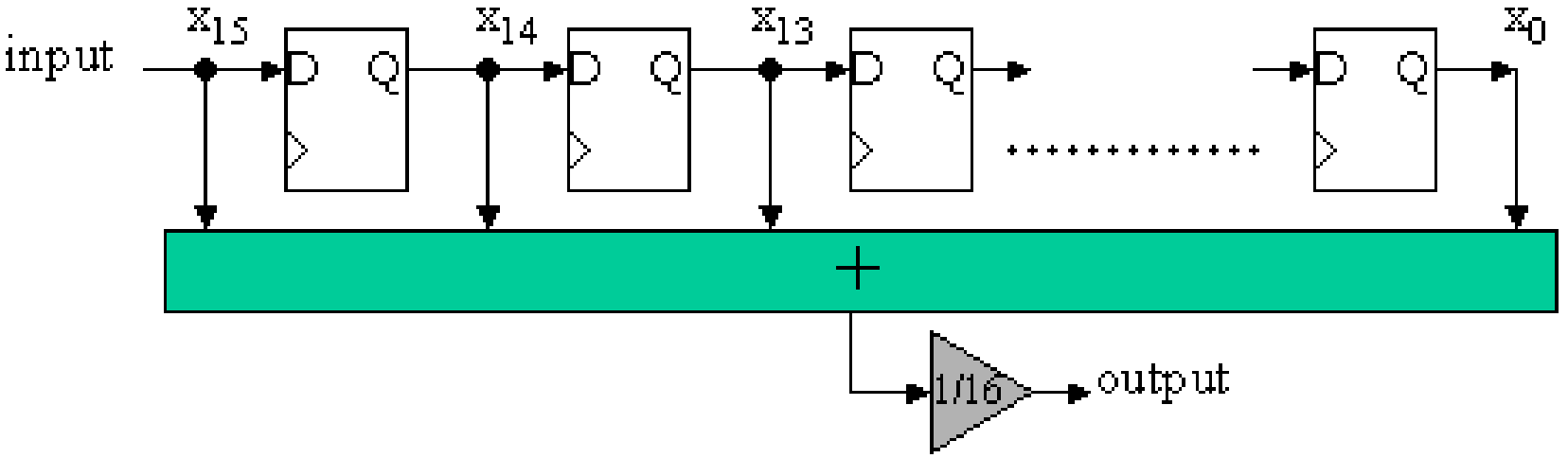
$$output(15) = \sum_{i=0}^{15} a_i \cdot x(i)$$

$$= a_0 \cdot x(0) + a_1 \cdot x(1) + \cdots + a_{15} \cdot x(15)$$

Finite Impulse Response Filter



(a) 16 Tap FIR Filter



(b) 16 Tap Averaging Filter

Bresenham's Algorithm

$$y = mx + b$$

Can be re-written as

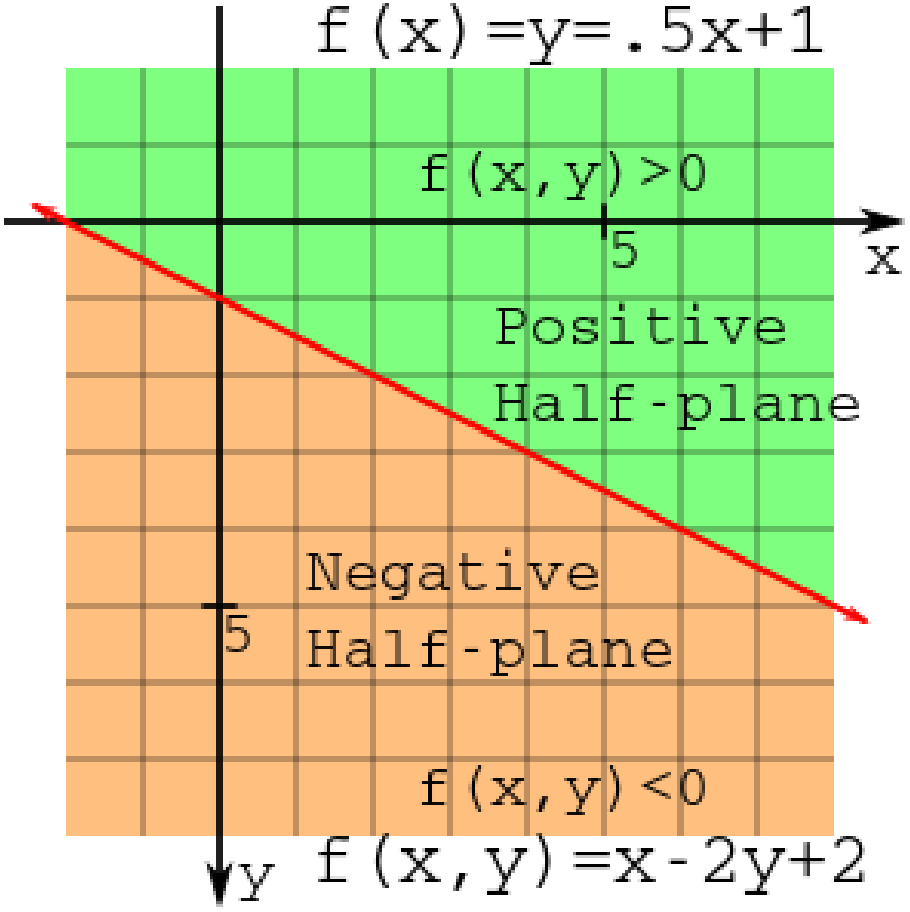
$$f(x, y) \equiv Ax + By + C = 0$$

Where

$$A = Y_1 - Y_0$$

$$B = -(X_1 - X_0)$$

$$C = (X_1 - X_0)b$$



Bresenham's Algorithm

$$D = f\left(x_0+1, y_0+\frac{1}{2}\right) - f\left(x_0, y_0\right)$$

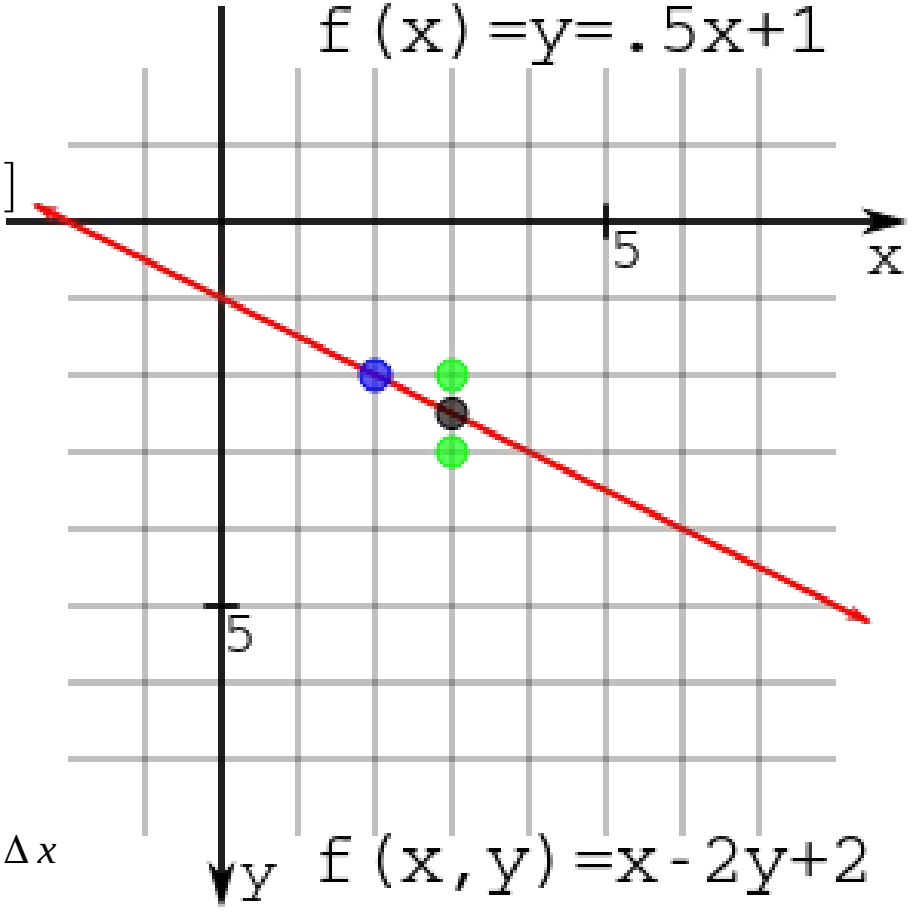
$$D = \left[A\left(x_0+1\right)+B\left(y_0+\frac{1}{2}\right)+C\right] - \left[Ax_0+By_0+C\right]$$

$$D = \left[Ax_0+By_0+C+A+\frac{1}{2}B\right] - \left[Ax_0+By_0+C\right]$$

$$D = A+\frac{1}{2}B = \Delta y - \frac{1}{2}\Delta x$$

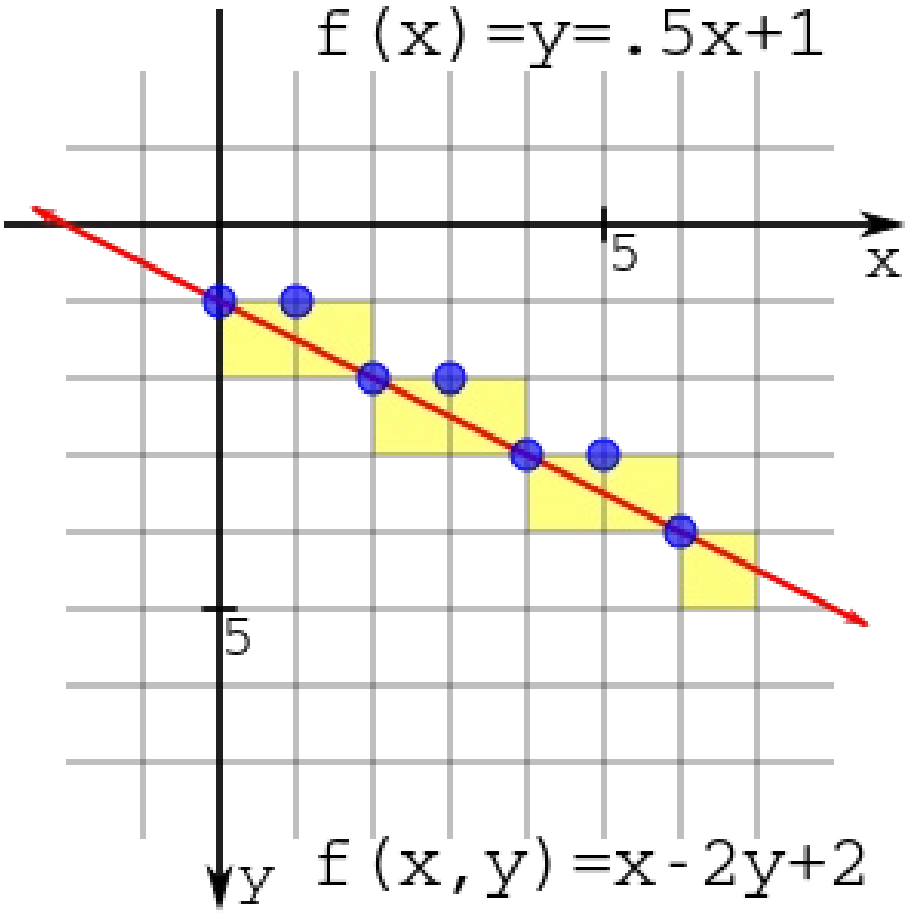
$$\Delta D = f\left(x_0+2, y_0+\frac{1}{2}\right) - f\left(x_0+1, y_0+\frac{1}{2}\right) = A = \Delta y$$

$$\Delta D = f\left(x_0+2, y_0+\frac{3}{2}\right) - f\left(x_0+1, y_0+\frac{1}{2}\right) = A+B = \Delta y - \Delta x$$

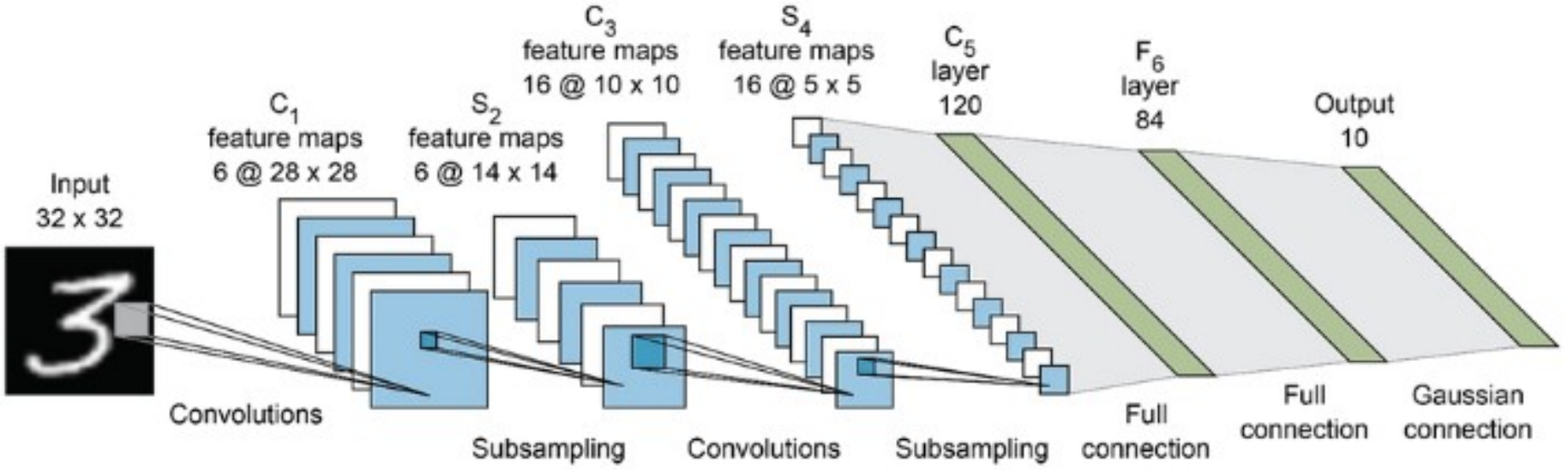


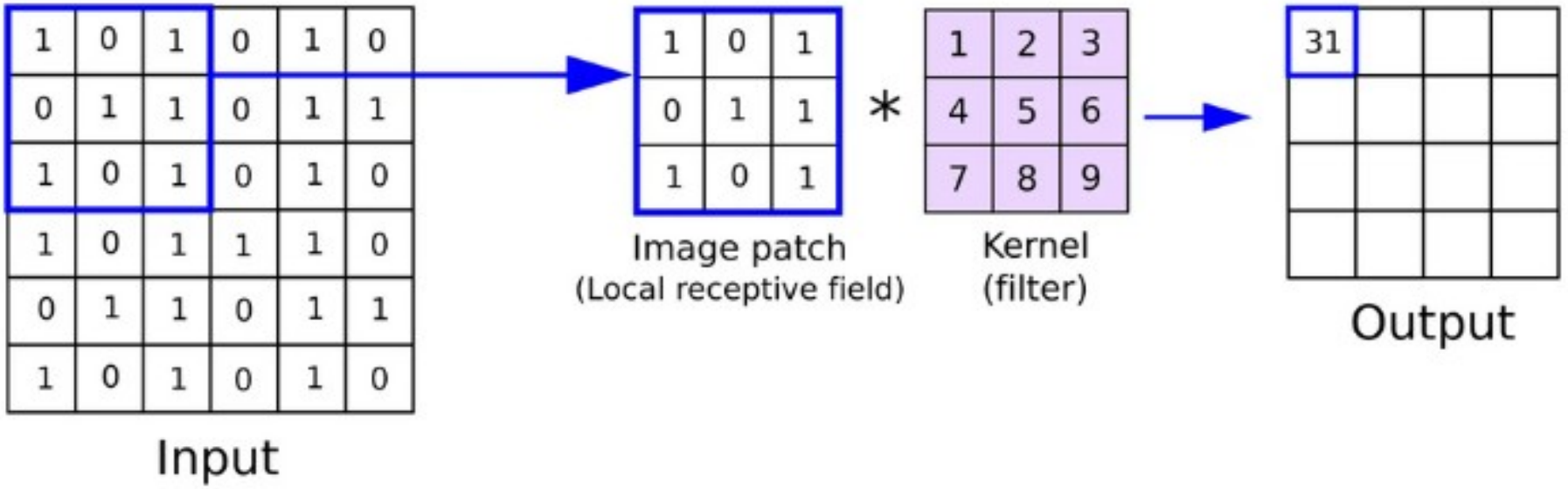
Bresenham's Algorithm

```
plotLine(x0, y0, x1, y1)

     dx = x1 - x0
     dy = y1 - y0
     D = 2*dy - dx
     y = y0

     for x from x0 to x1
         plot(x, y)
         if D > 0
             y = y + 1
             D = D - 2*dx
         end if
         D = D + 2*dy
```

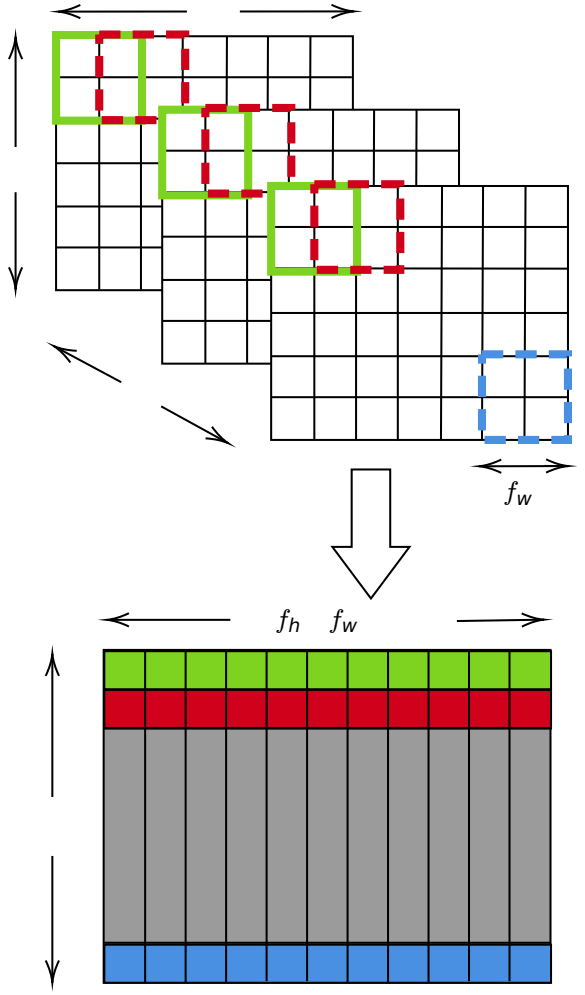


AI: CNNs

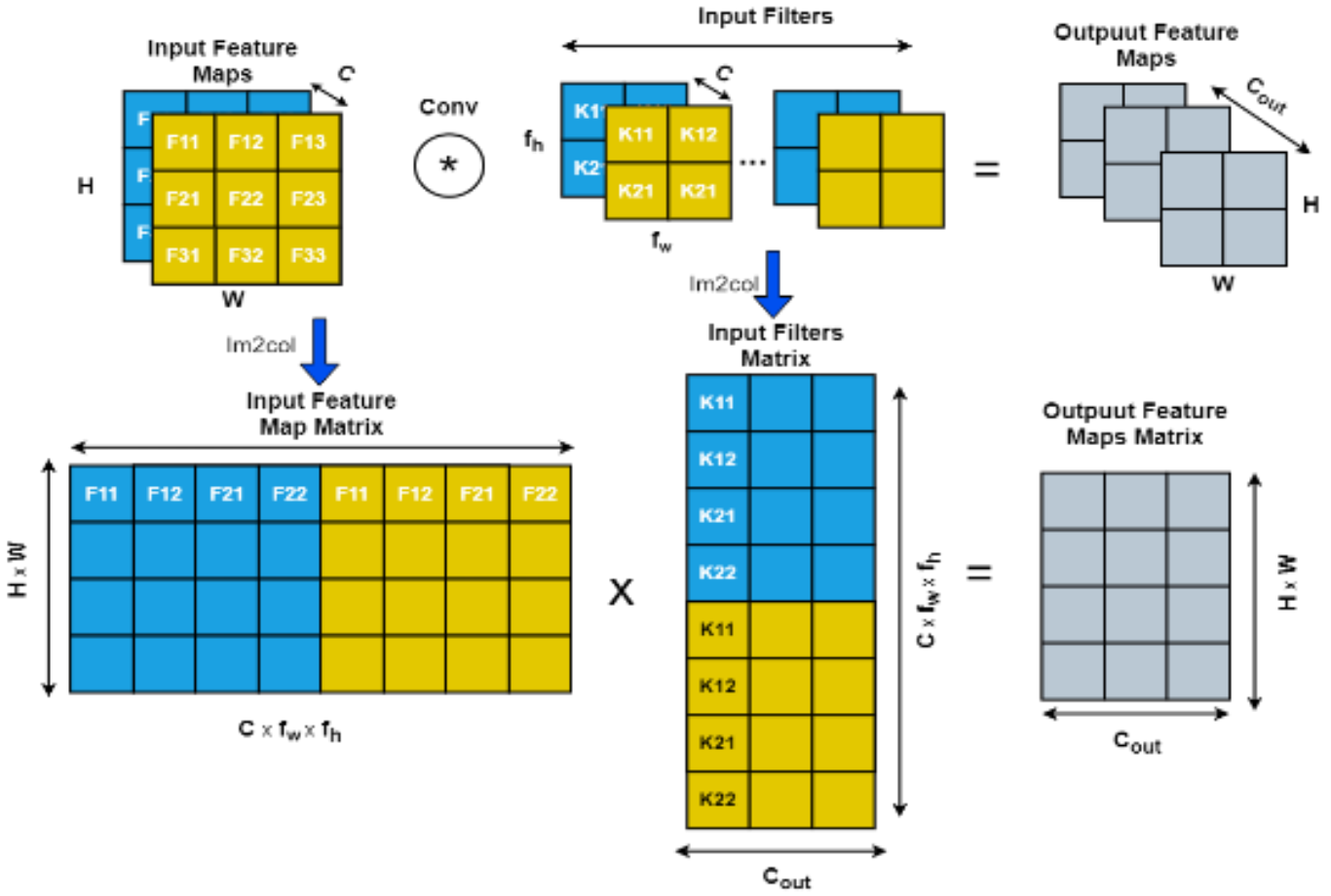

Input
32 x 32
C1
feature maps
6 @ 28 x 28
S2
feature maps
6 @ 14 x 14
C3
feature maps
16 @ 10 x 10
S4
feature maps
16 @ 5 x 5
C5
layer
120
F6
layer
84
Output
10
Convolutions
Subsampling
Convolutions
Subsampling
Full
connection
Full
connection
Gaussian
connection


1 0 1 0 1 0
0 1 1 0 1 1
1 0 1 0 1 0
1 0 1 1 1 0
0 1 1 0 1 1
1 0 1 0 1 0
Input
1 0 1
0 1 1
1 0 1
Image patch
(Local receptive field)
*
1 2 3
4 5 6
7 8 9
Kernel
(filter)
31
Output

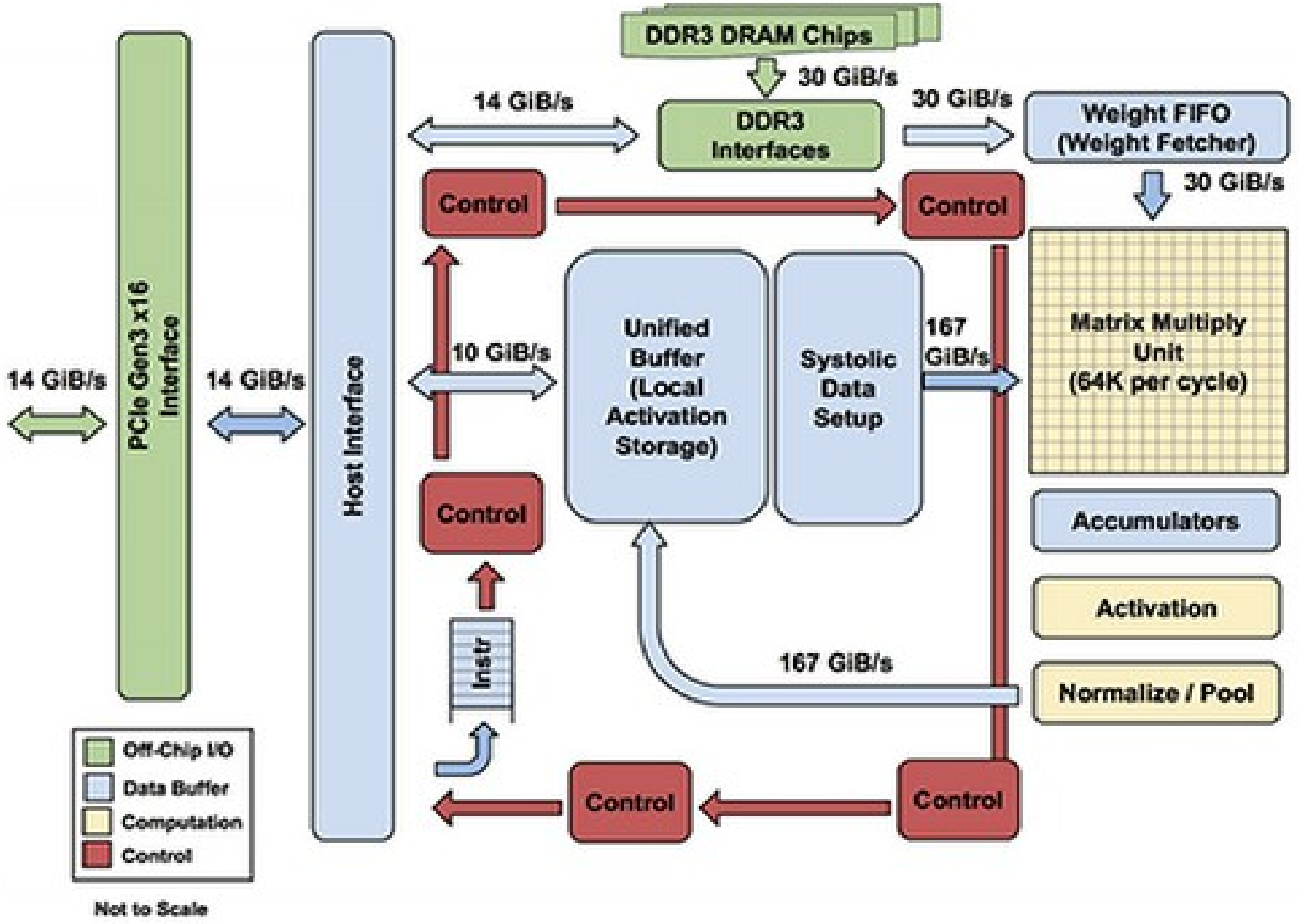
AI: IM2COL



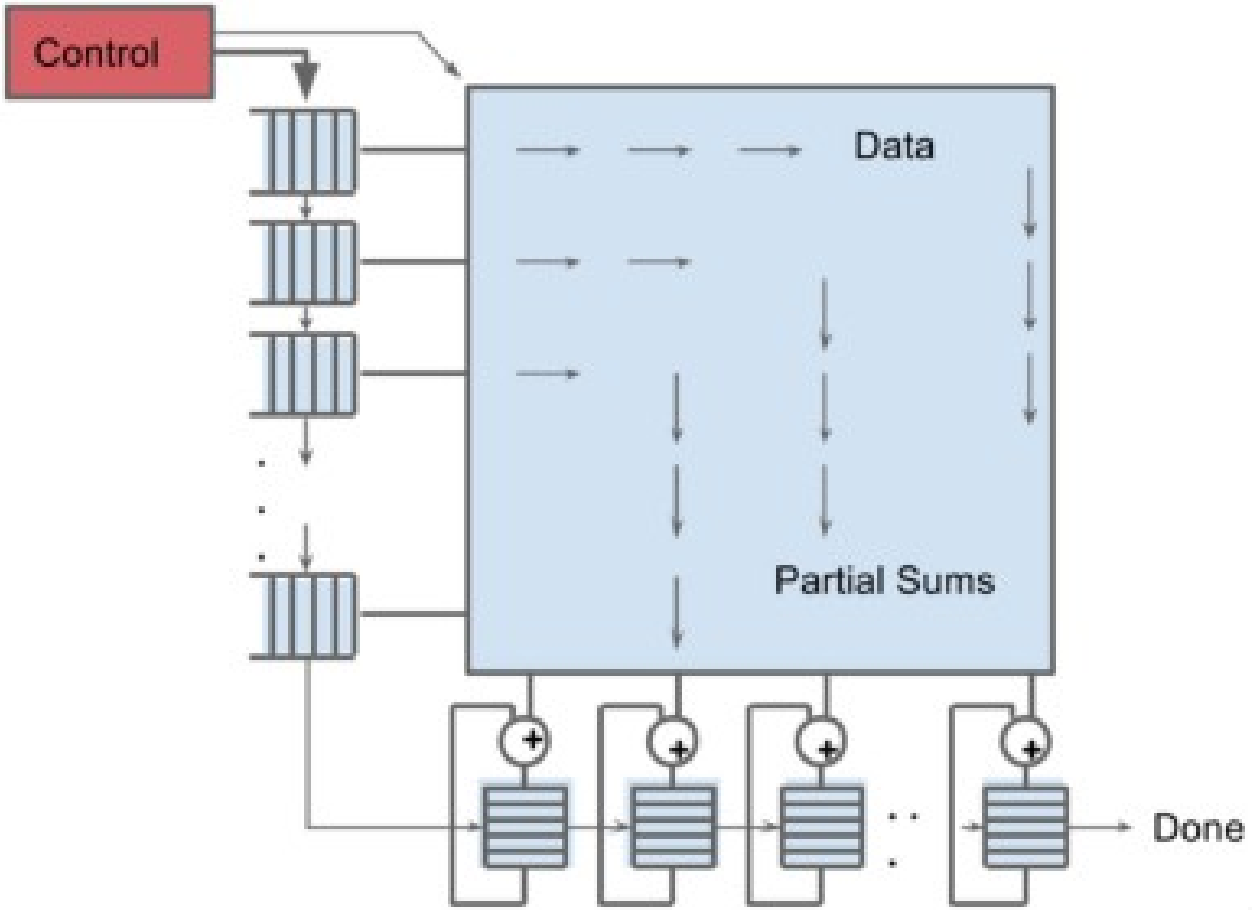
AI: IM2COL + MatMult



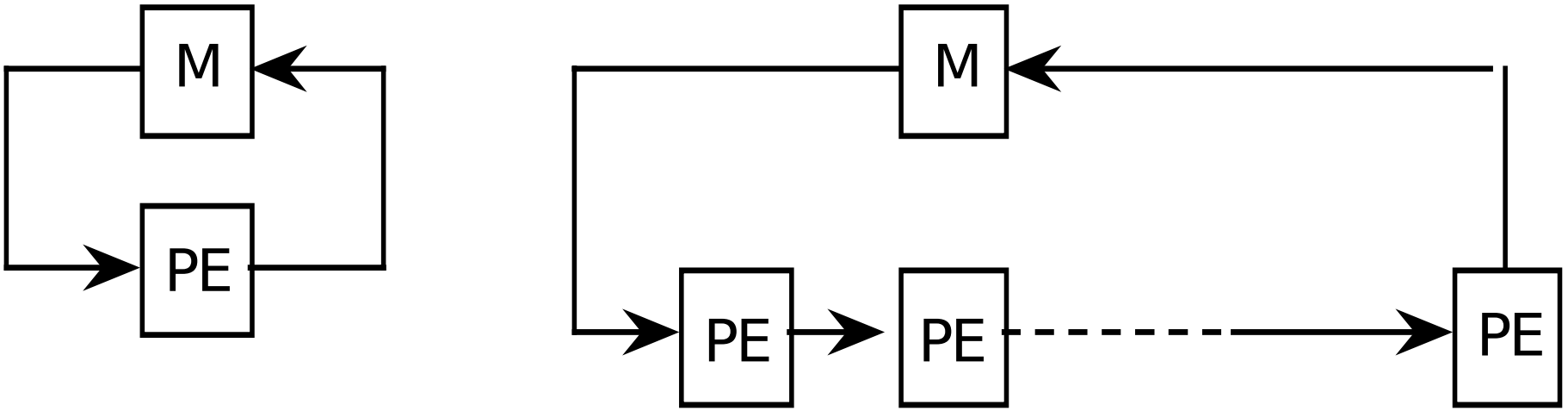
AI: Google TPU



TPU: Systolic Array

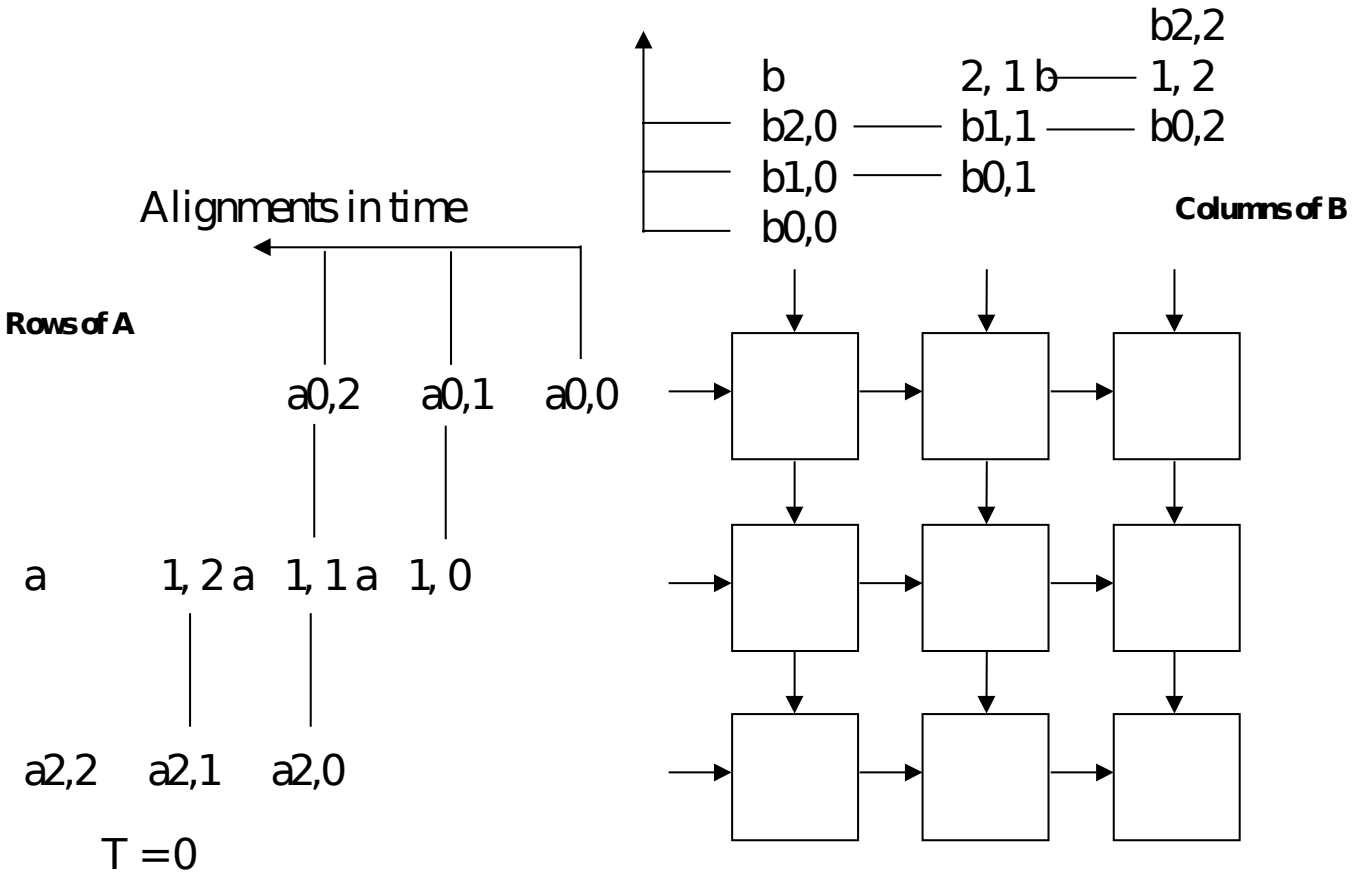


Systolic Arrays

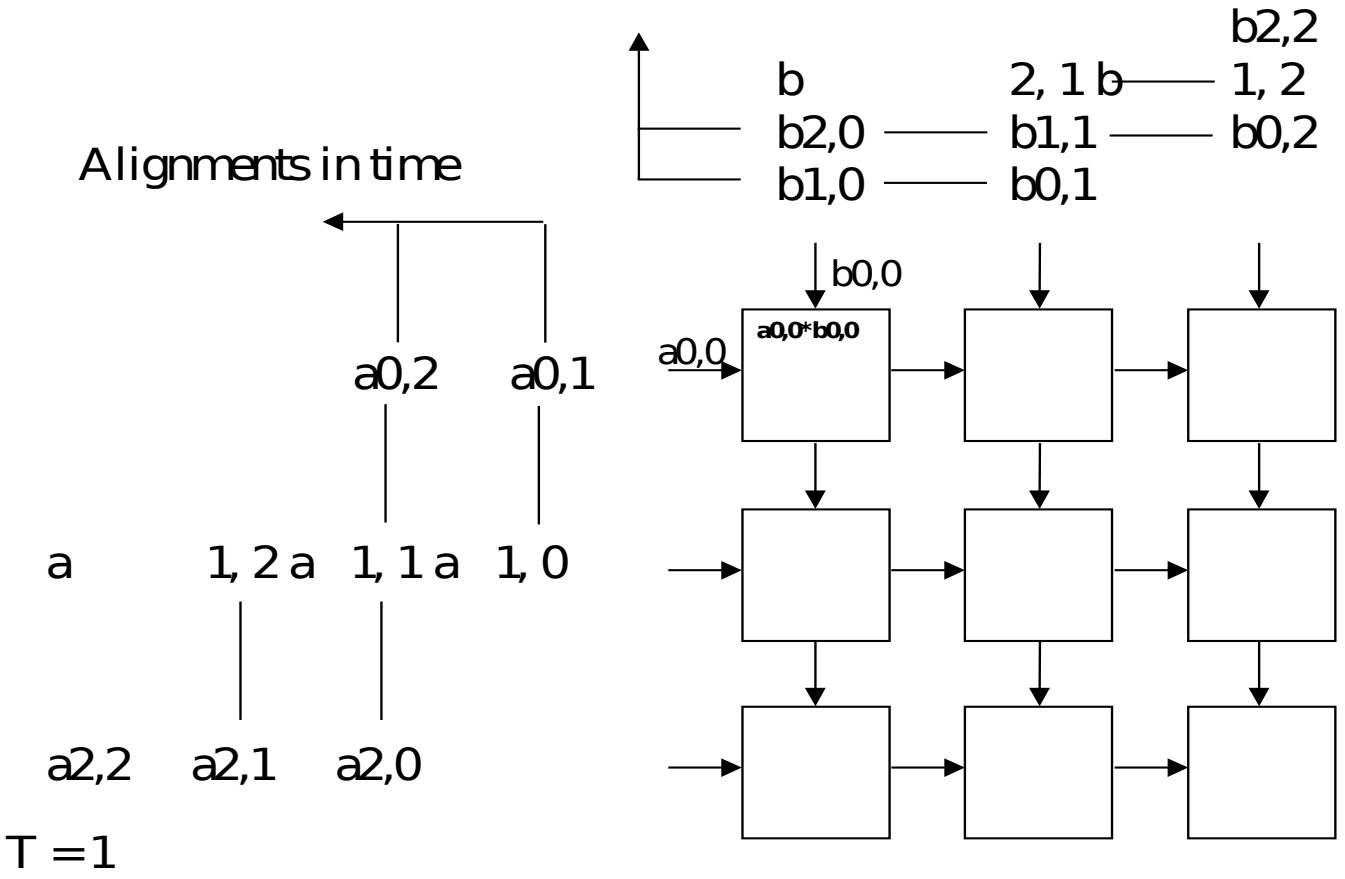


Why Systolic Architectures? Computer 15(1): 37-46 (1982) H. T. Kung

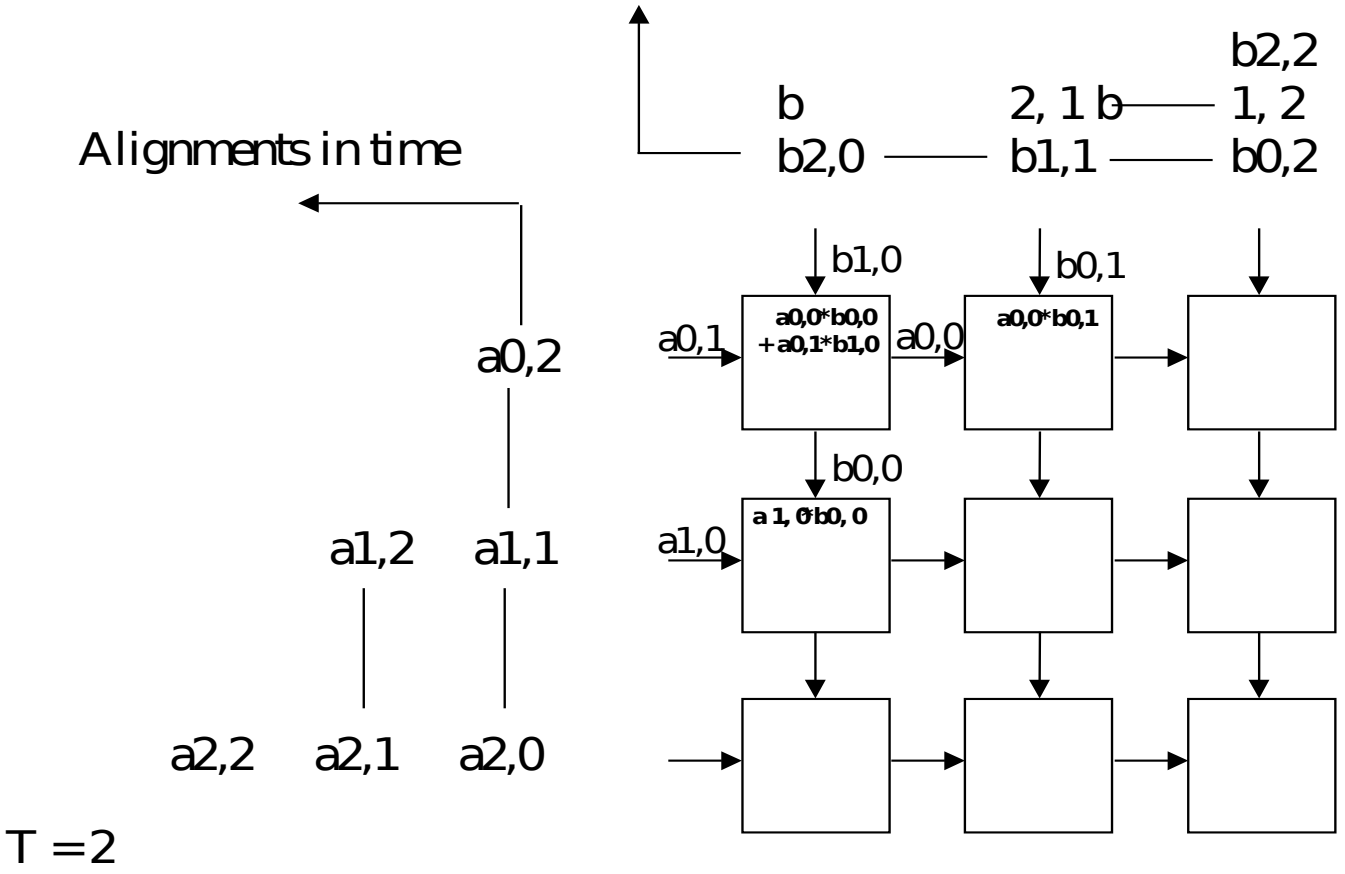
Systolic MatMul



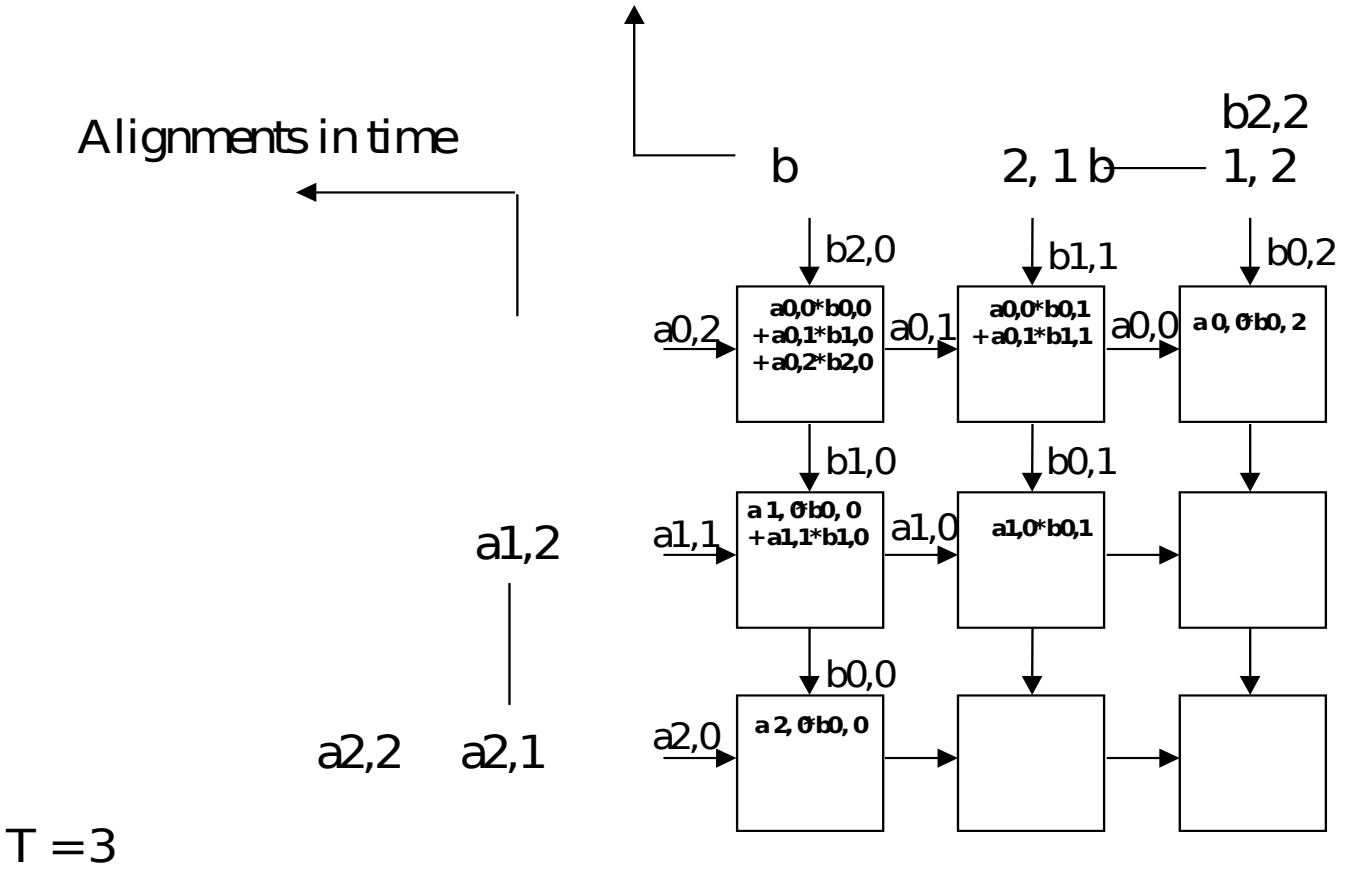
Systolic MatMul



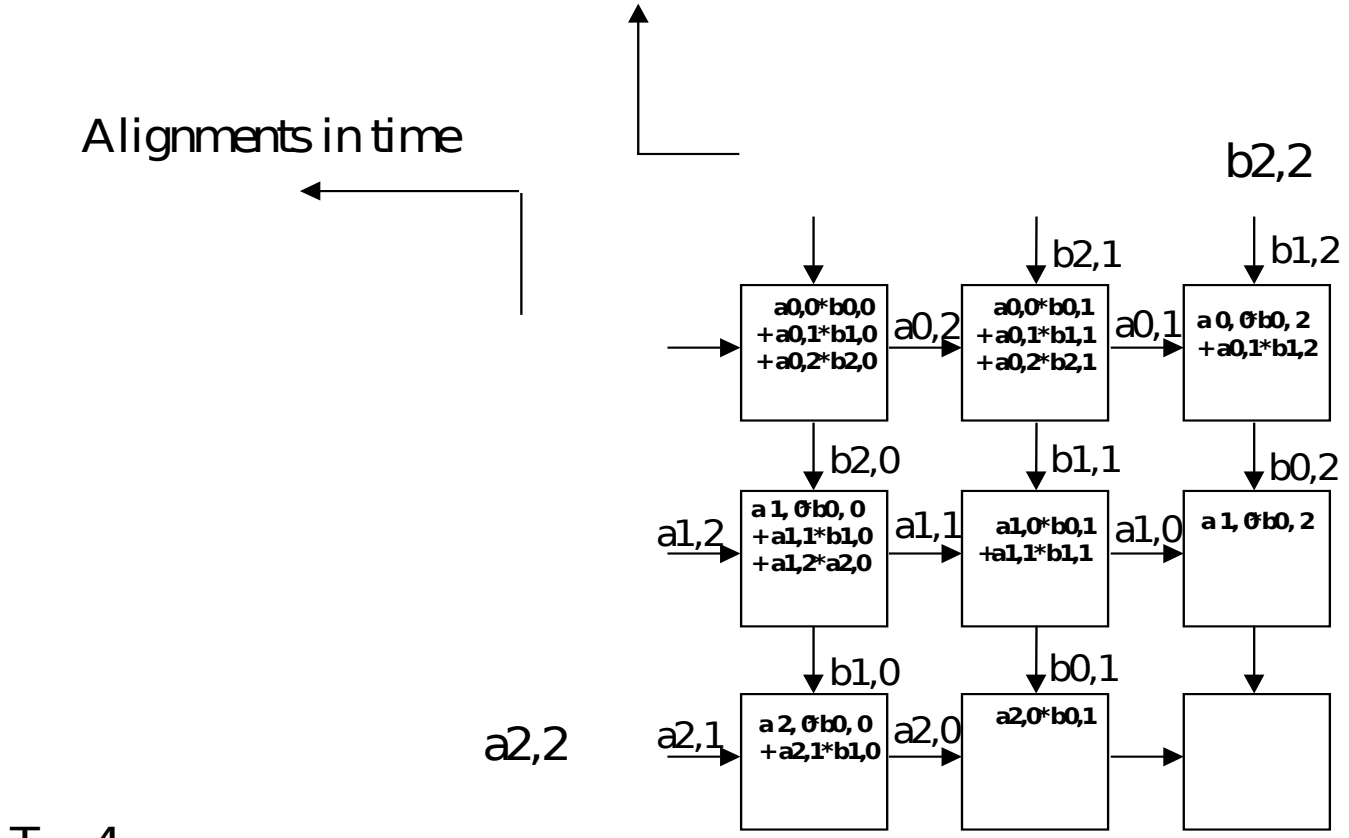
Systolic MatMul



Systolic MatMul

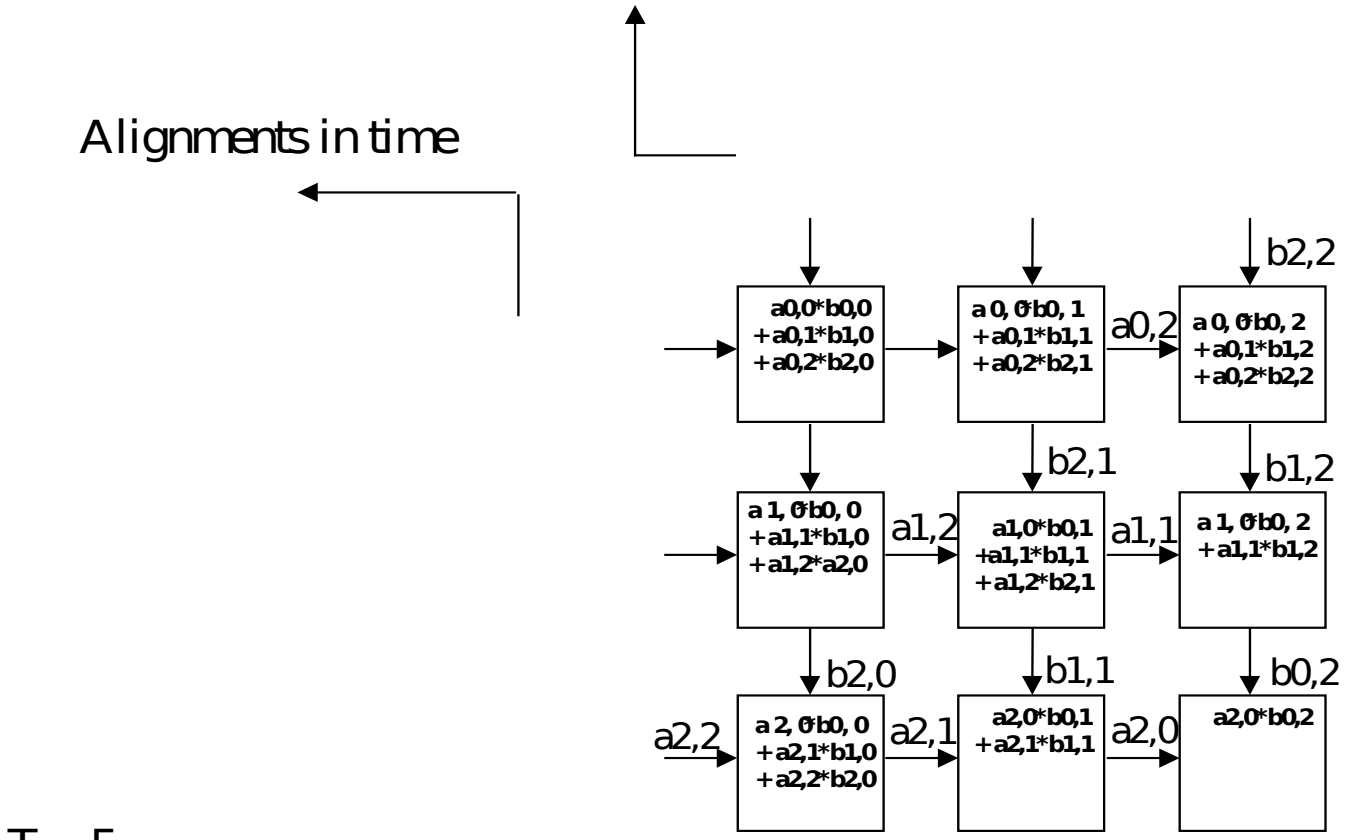


Systolic MatMul

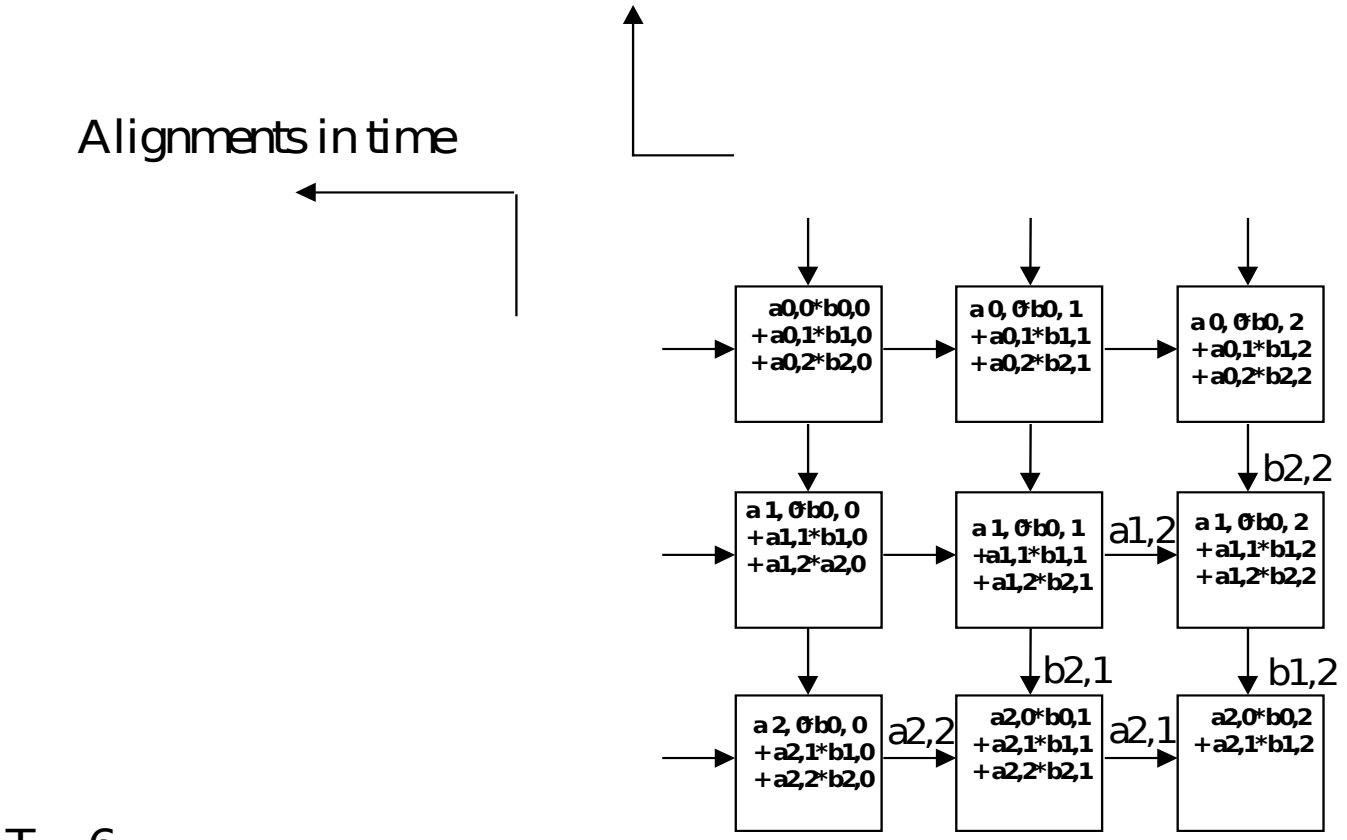


T = 4

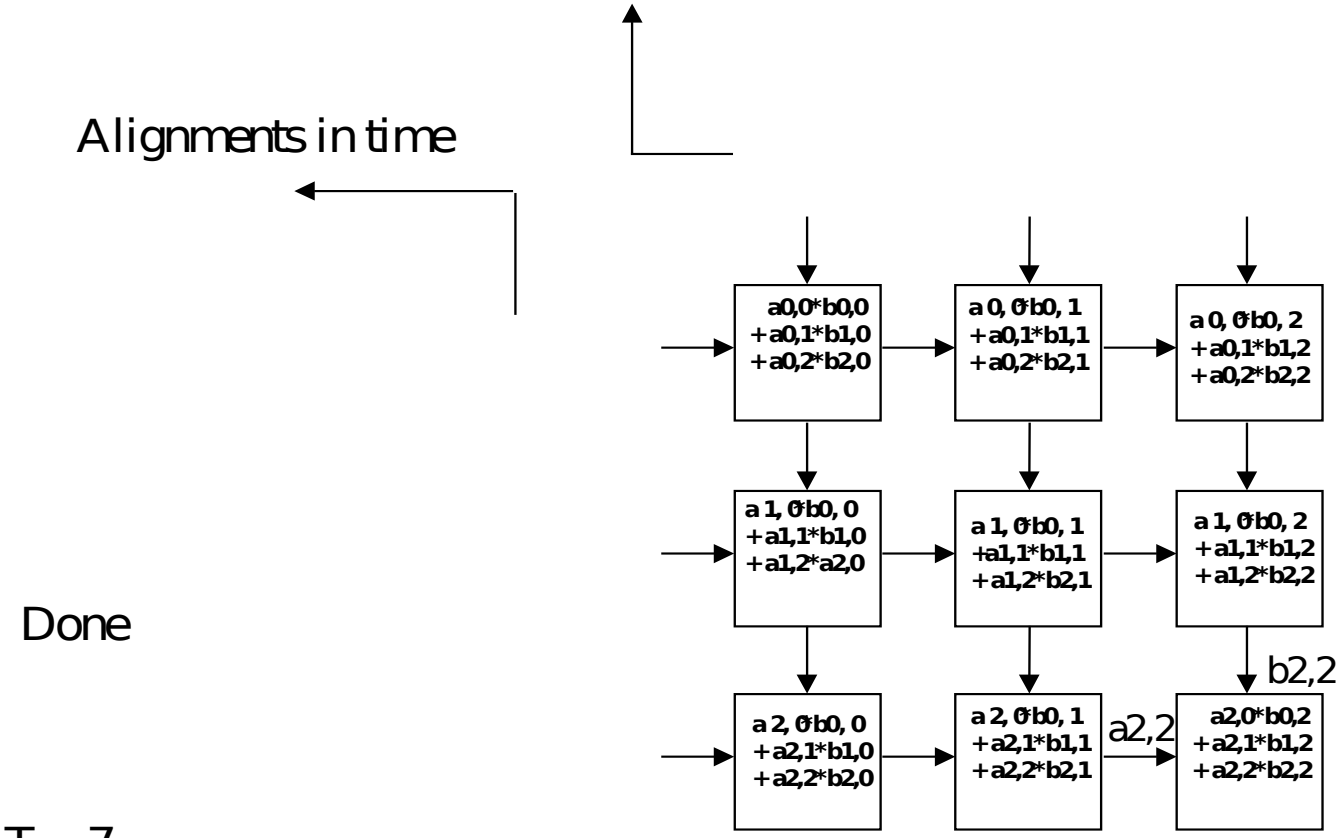
Systolic MatMul



Systolic MatMul

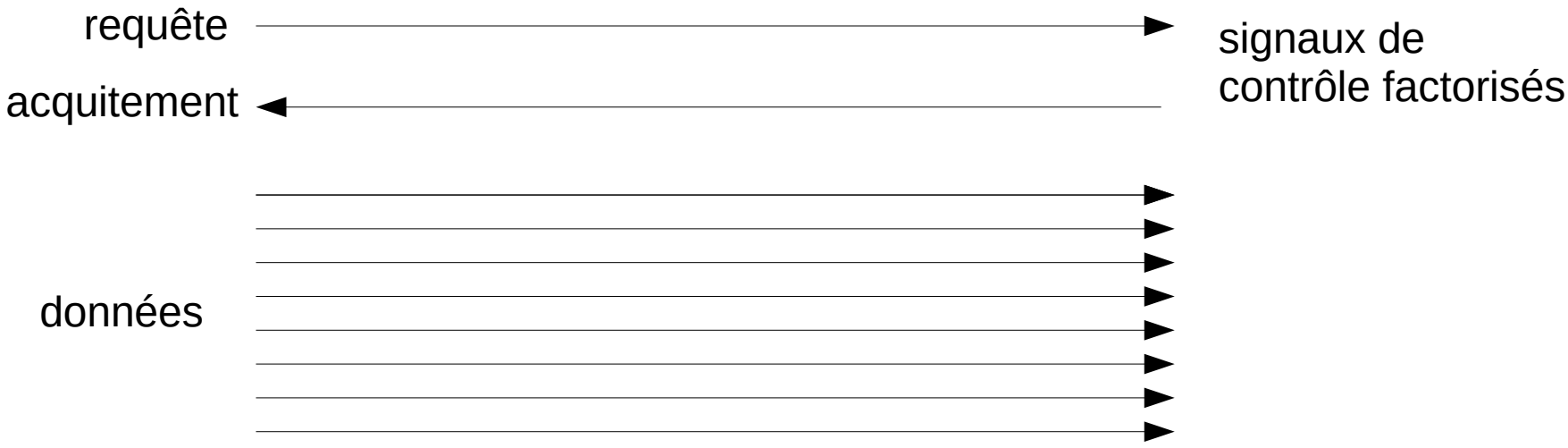


Systolic MatMul



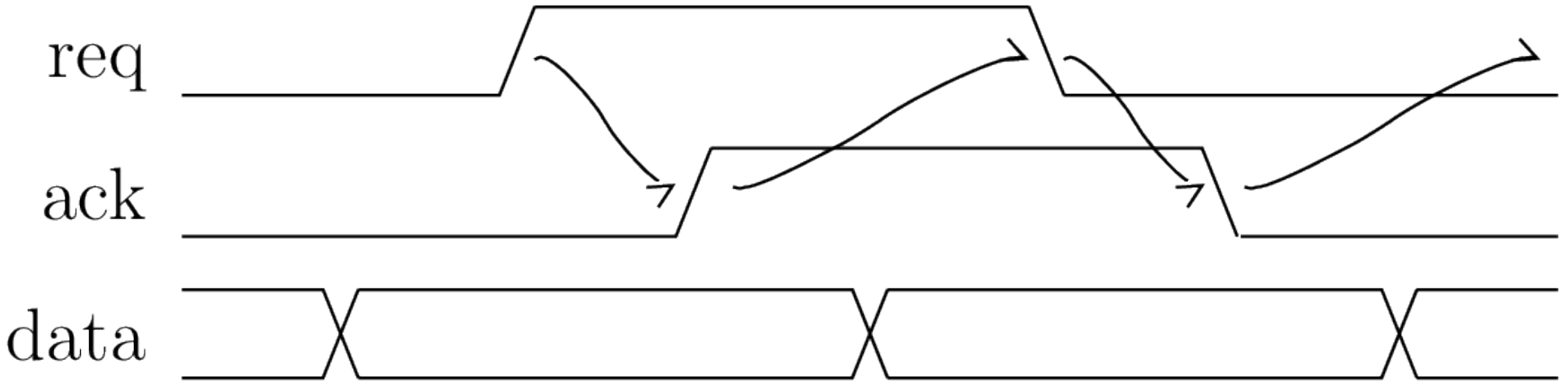
Logique asynchrone

- Illustration de logique séquentielle sans horloge
- Mise en place d'une signalisation



Logique asynchrone

- Illustration de logique séquentielle sans horloge
- Mise en place d'une signalisation



Logique asynchrone

- Illustration de logique séquentielle sans horloge
- Mise en place d'une signalisation

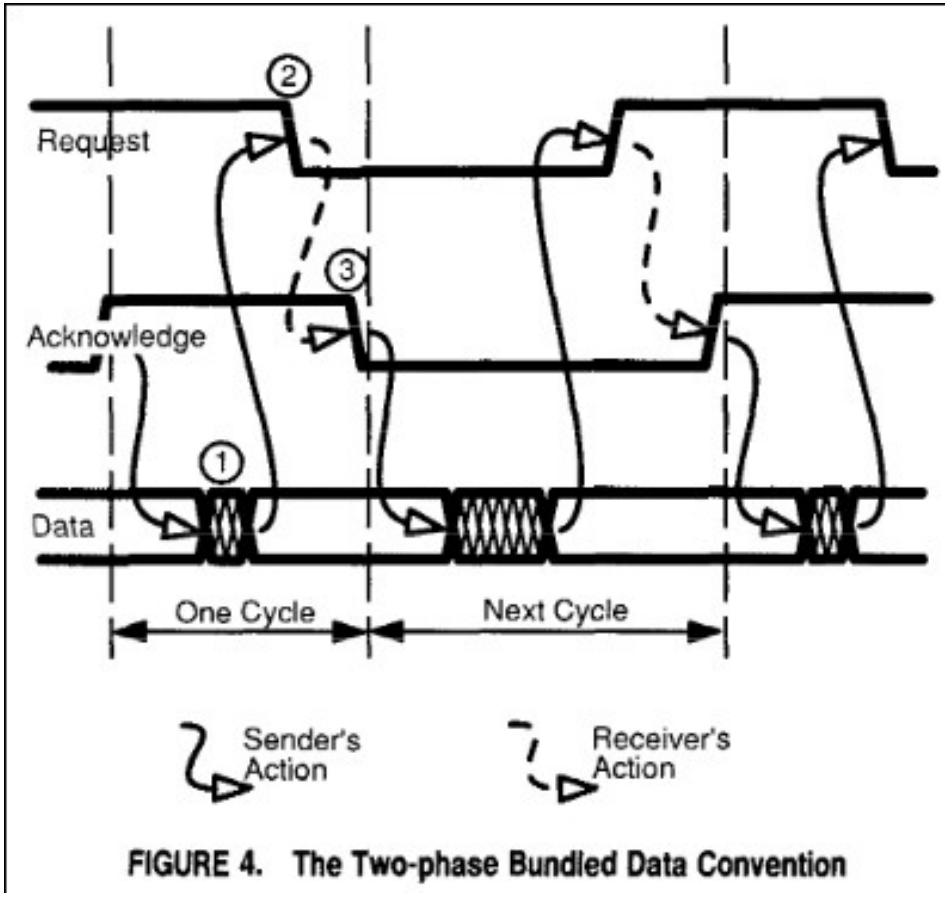


FIGURE 4. The Two-phase Bundled Data Convention

Ivan Sutherland, Turing award 1988

Communications of the ACM, June 1989, Volume 32, Number 6.

Rendez-vous

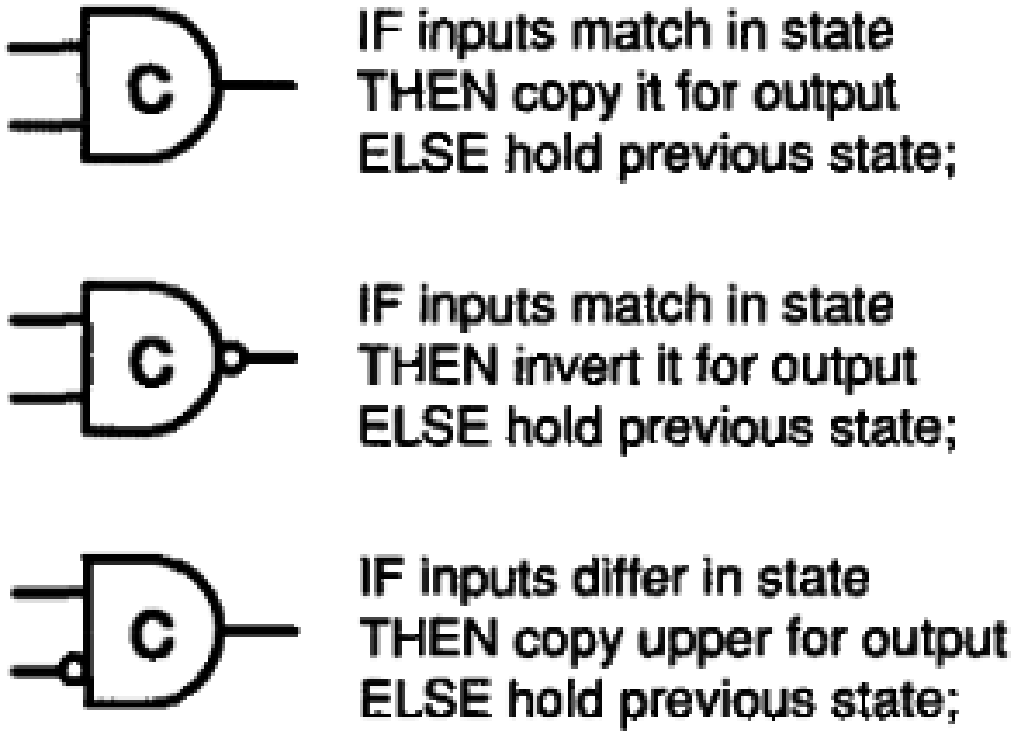
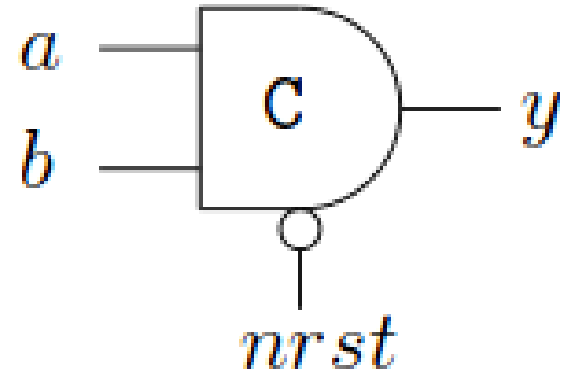


FIGURE 8. Muller C-Elements with Inverters

Muller C-elements contain storage to hold a previous state on some input conditions. When inverters are included in input or output wires, as indicated by the bubbles in this figure, the actions are as listed. Muller C-elements provide the AND function for events.

C-Element : comportemental

TAB. 1. **Le C-Element à deux entrées accompagné de son comportement décrit en VHDL.**




```
if (nrst = 0) then
    y <= 0;
elsif (a = b) then
    y <= a;
end if;
```

C-Element : structures en transistors

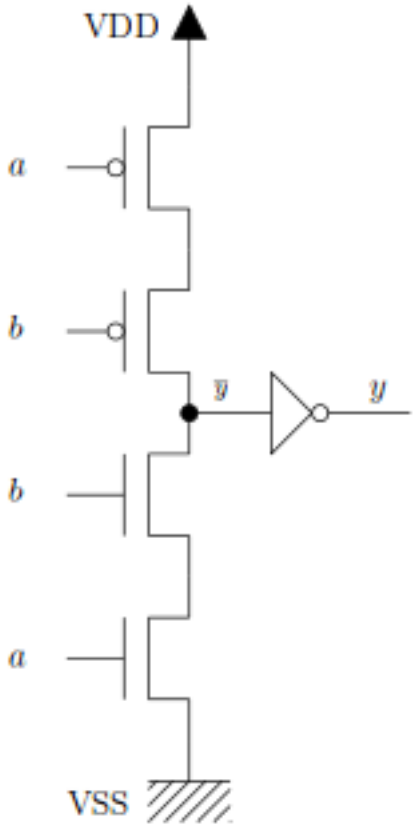


FIG. **1a. C-Element dynamique.**

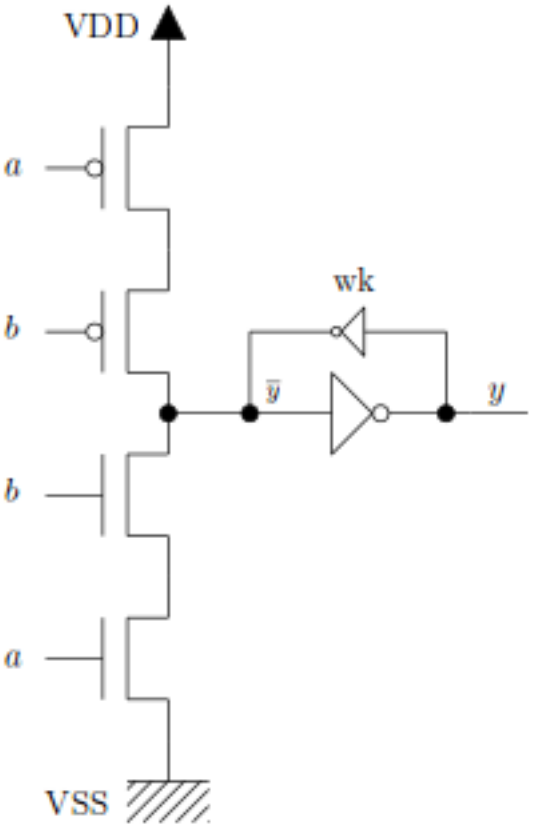


FIG. **1b. C-Element à rebouclage faible.**

C-Element : structures en transistors

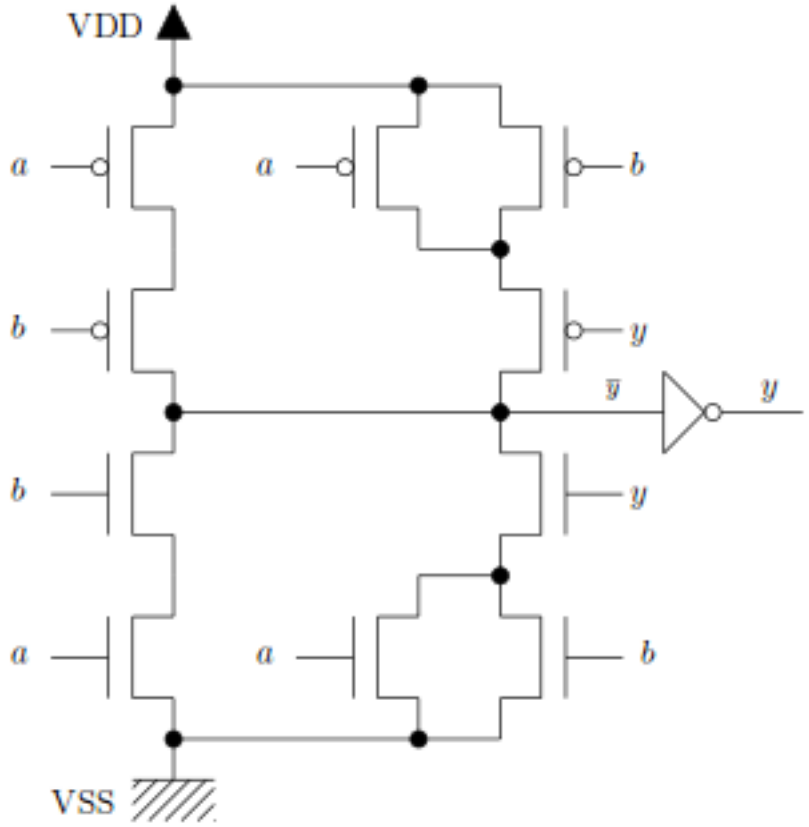


FIG. **1c. C-Element conventionnel.**

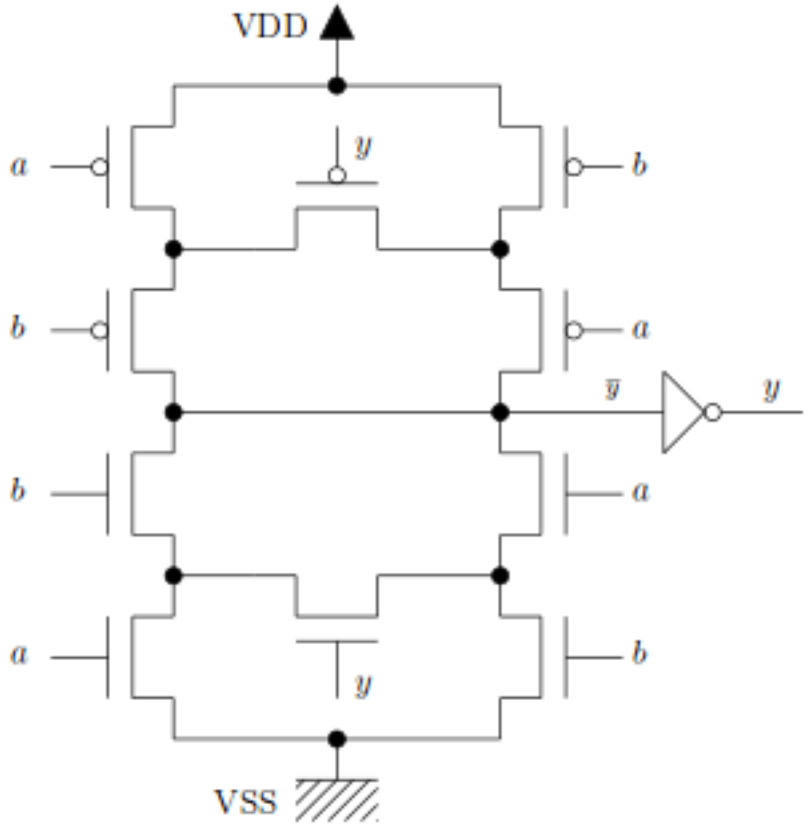


FIG. **1d. C-Element symétrique.**

C-Element : structures en transistors

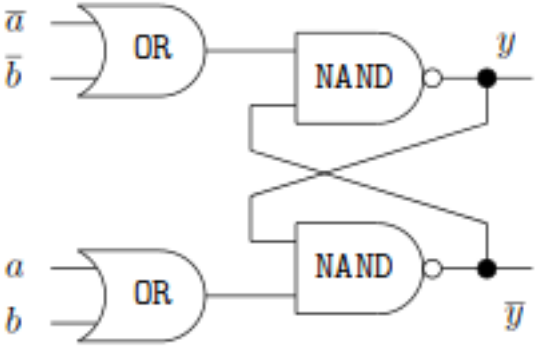


FIG. **2a. Structure en portes du C-Element à bascule RS.**

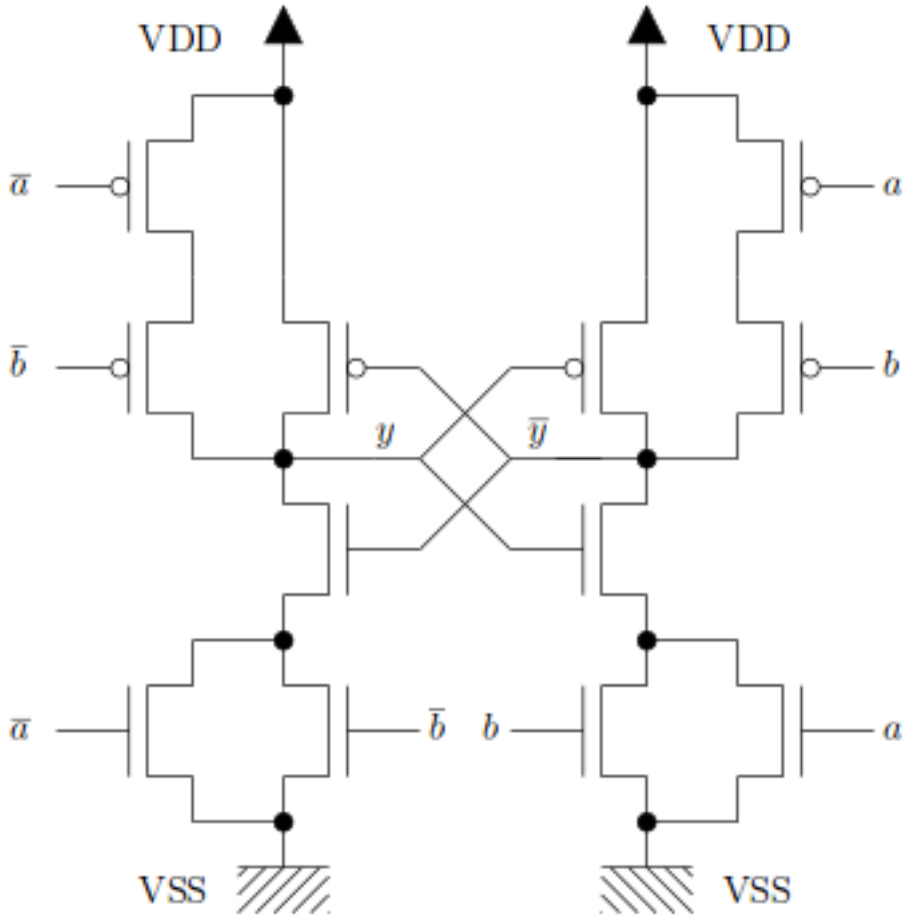


FIG. **2b. Structure en transistors du C-Element à bascule RS de la Fig. 2a.**

Micro-pipeline

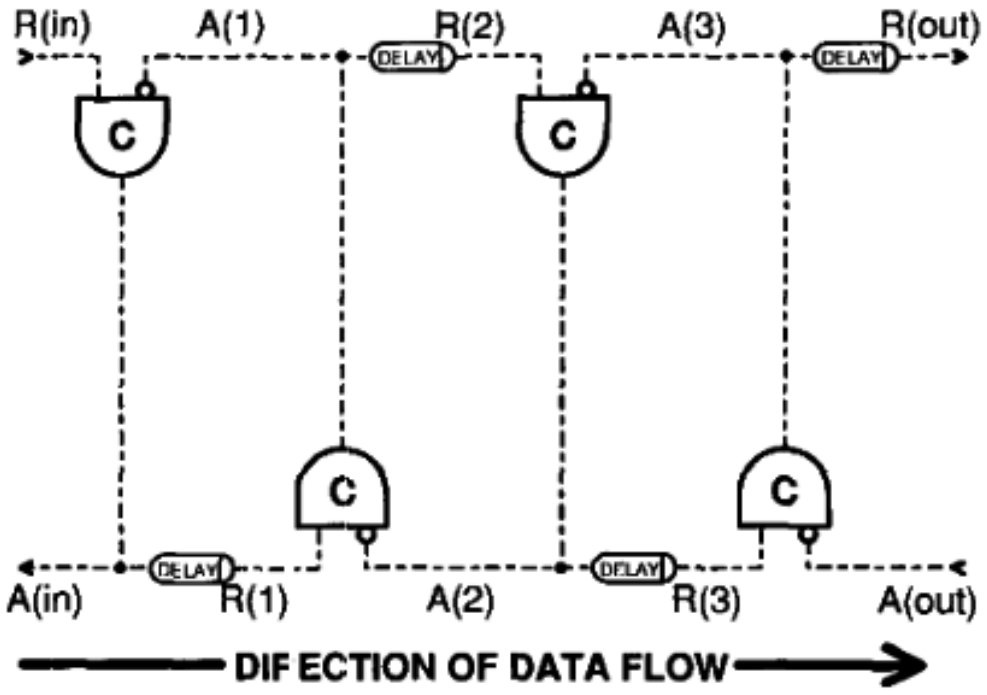


FIGURE 10. Control Circuit for a Micropipeline

With data paths omitted, the control circuit for a micropipeline is a string of Muller C-elements. In this figure one of four identical stages is shaded and alternate stages have been drawn upside down. At the input and output to each stage there are request, R(n), and acknowledge, A(n), signals. Inverters in the acknowledge paths are represented by "bubbles" at one input of each Muller C-element. The delays shown explicitly here may not be required for simple data paths. Notice that each loop in this circuit contains exactly one inversion, the bubble, and is therefore an oscillator. The Muller C-elements retard the oscillation in each loop to coordinate it with the actions of adjacent loops. In this and other figures, dotted wires carry event signals.

Data flow

`if <cond> then <body1> else <body2>` (Fig. 1a) et

`while <cond> do <body>` (Fig. 1b).

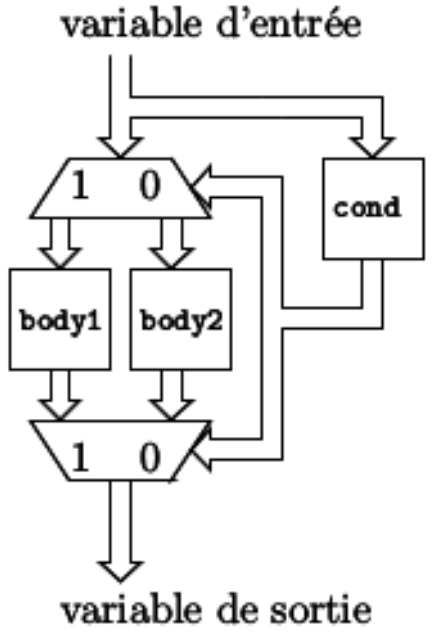


Fig. 1a : *Modèle d'architecture matérielle asynchrone pour le branchement* `if <cond> then <body1> else <body2>`.

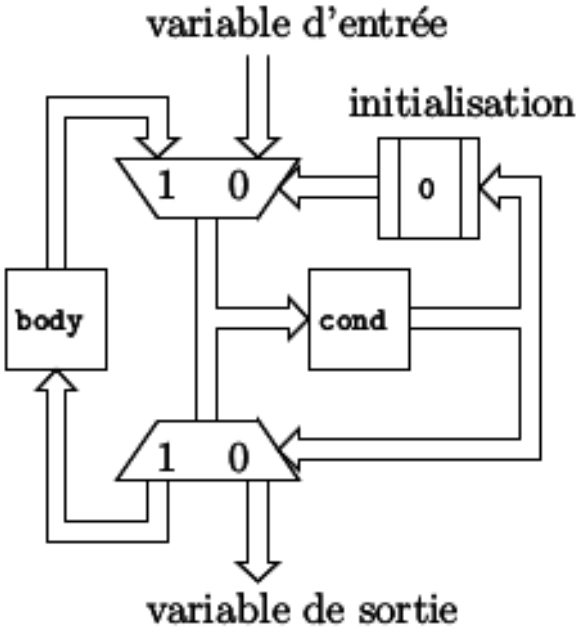


Fig. 1b : *Modèle d'architecture matérielle asynchrone pour le branchement* `while <cond> do <body>`.

Latch avec poignée de main

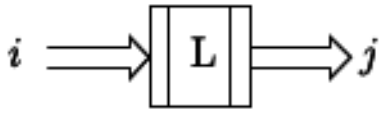


Fig. 2a : *Symbole du* LATCH *(partie « données »).*

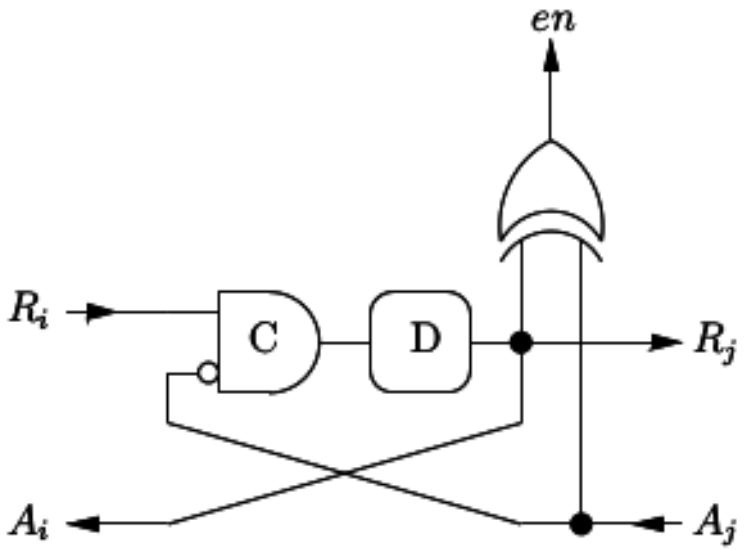


Fig. 2b : *Handshake pour le* LATCH *(partie « contrôle »). Le signal en sert à contrôler le latch.*

Element utile pour MUX / DEMUX

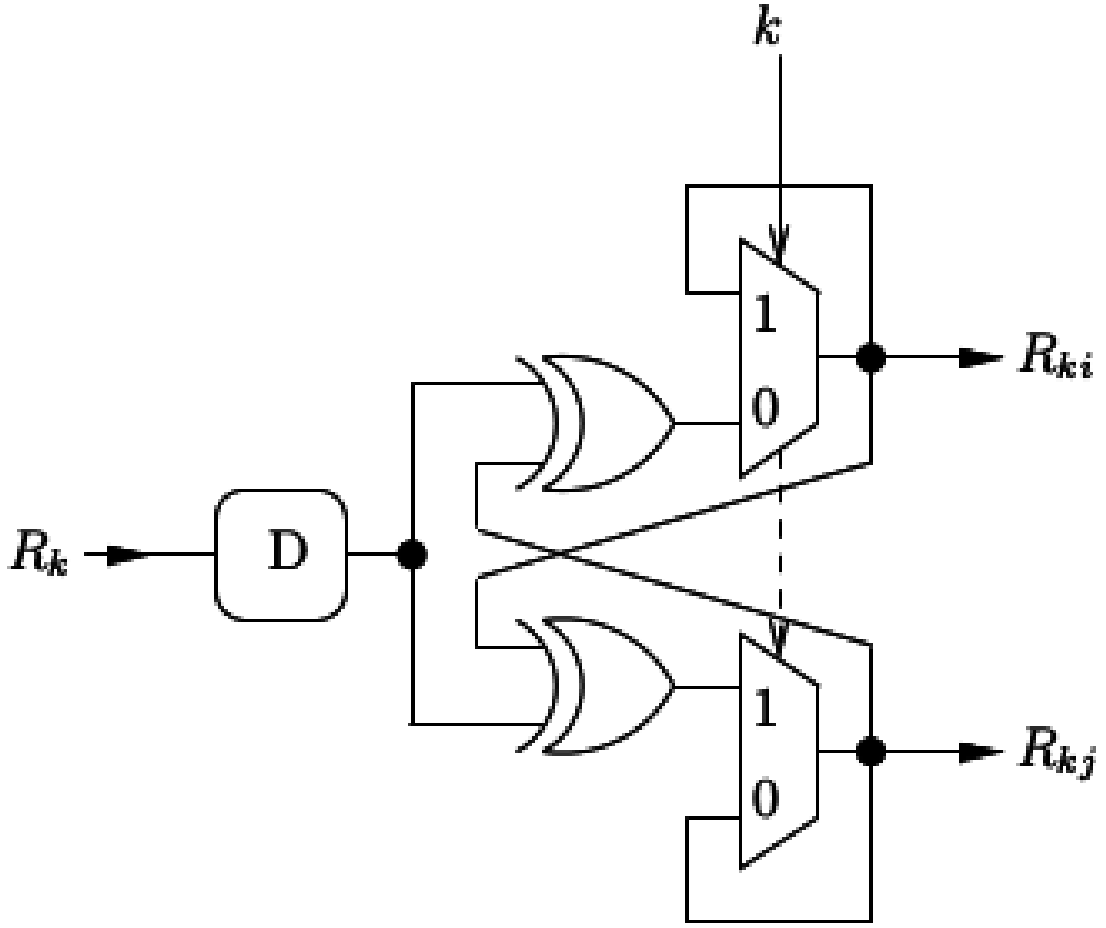


Fig. 3 : *Circuit de transformation de* $\{k, R_k\}$ *en* $\{R_{ki}, R_{kj}\}$.

MUX

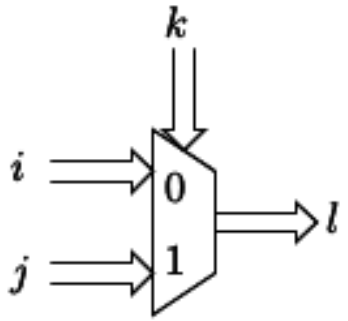


Fig. 4a : *Symbole du* MUX*22 (partie « données »).*

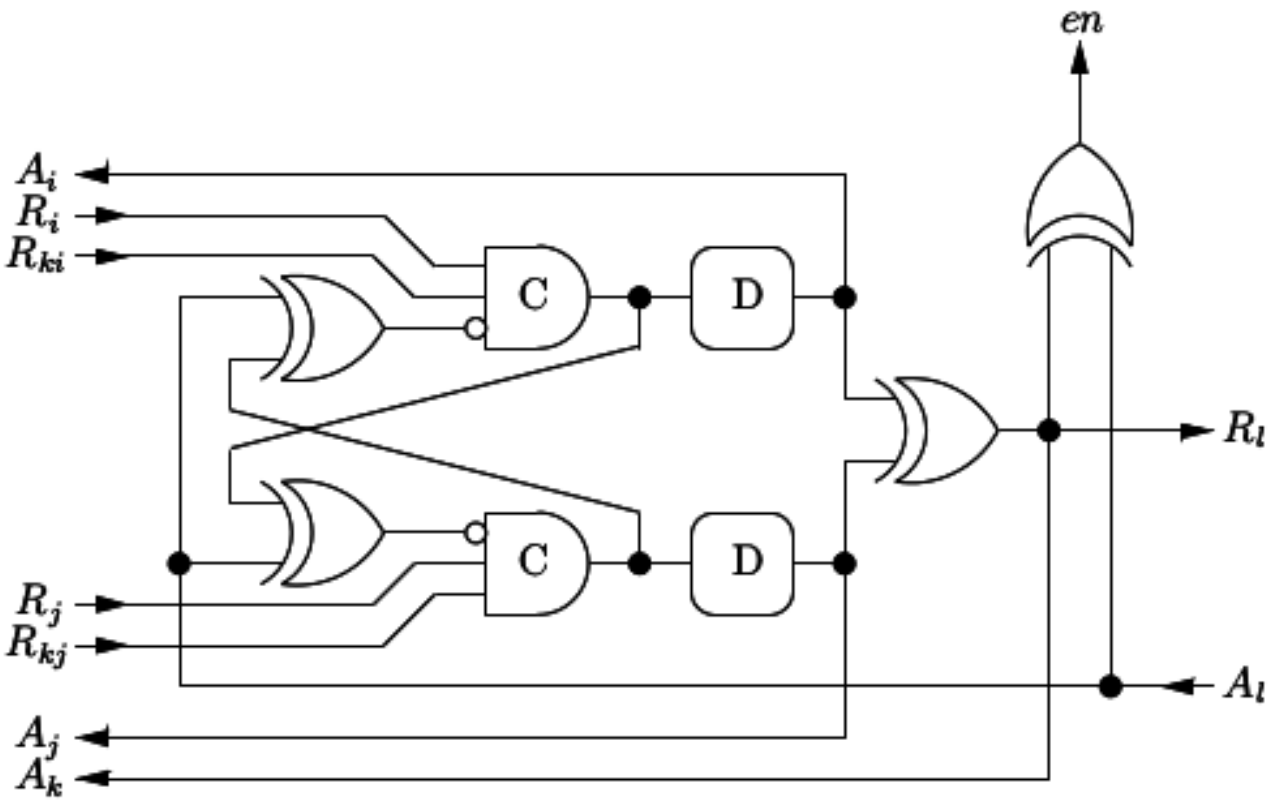
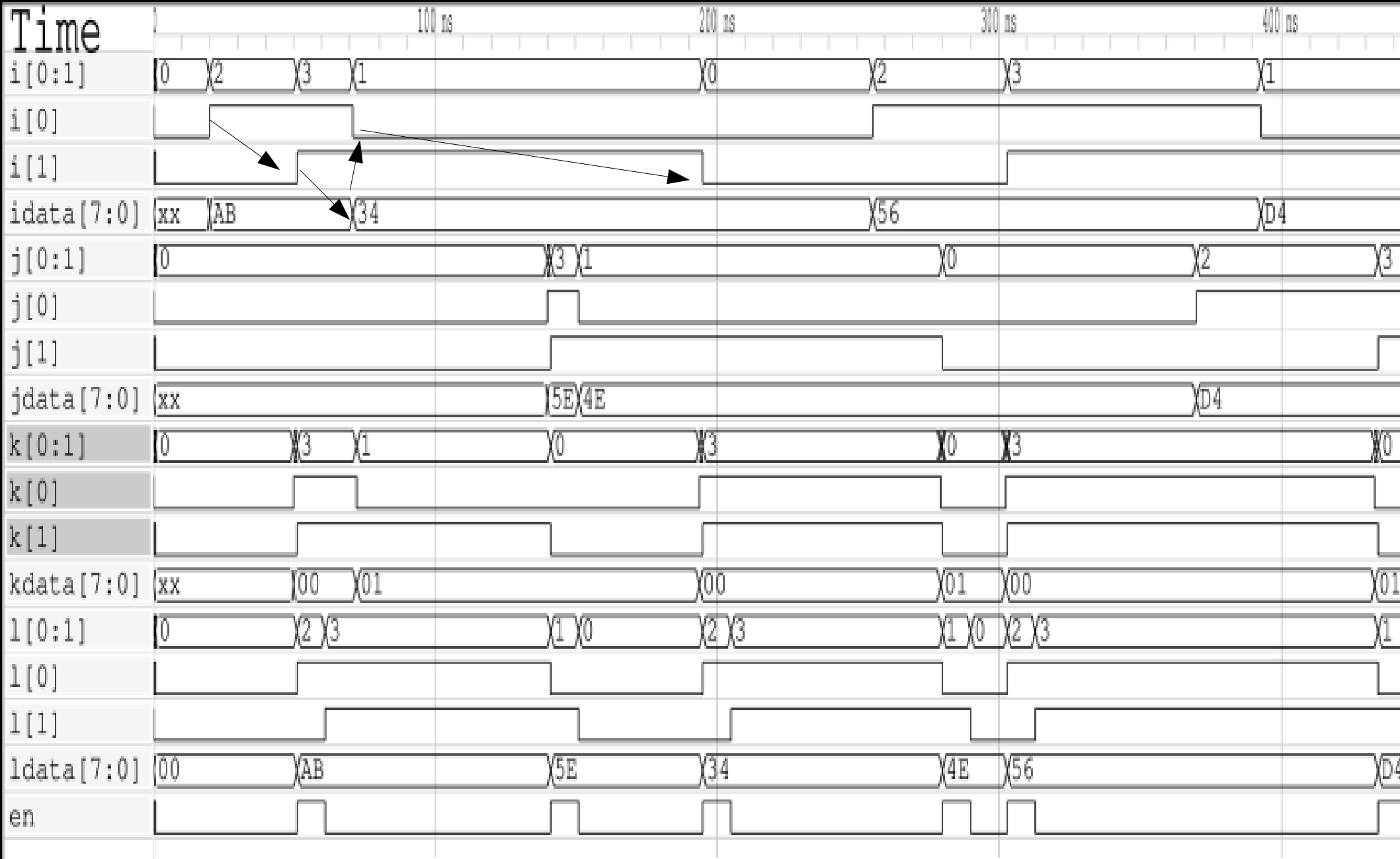


Fig. 4b : *Handshake pour le* MUX*22 (partie « contrôle »). Le signal en sert à contrôler le latch de sortie.*


Time
100 ns
200 ns
300 ns
400 ns
i[0:1]
i[0]
i[1]
idata[7:0]
j[0:1]
j[0]
j[1]
jdata[7:0]
k[0:1]
k[0]
k[1]
kdata[7:0]
l[0:1]
l[0]
l[1]
ldata[7:0]
en

I		J		K	
10.0	0xAB	130.0	0x5E	40.0	0x00
20.0	0x34	10.0	0x4E	21.0	0x01
60.0	0x56	90.0	0xD4	53.0	0x00
90.0	0xD4	60.0	0x56	84.0	0x01
10.0	0x4E	20.0	0x34	22.0	0x00
130.0	0x5E	10.0	0xAB	130.0	0x01

https://www.edaplayground.com/x/3fCs

DEMUX

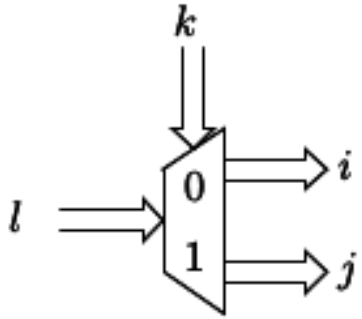


Fig. 5a : *Symbole du* DEMUX*22 (partie « données »).*

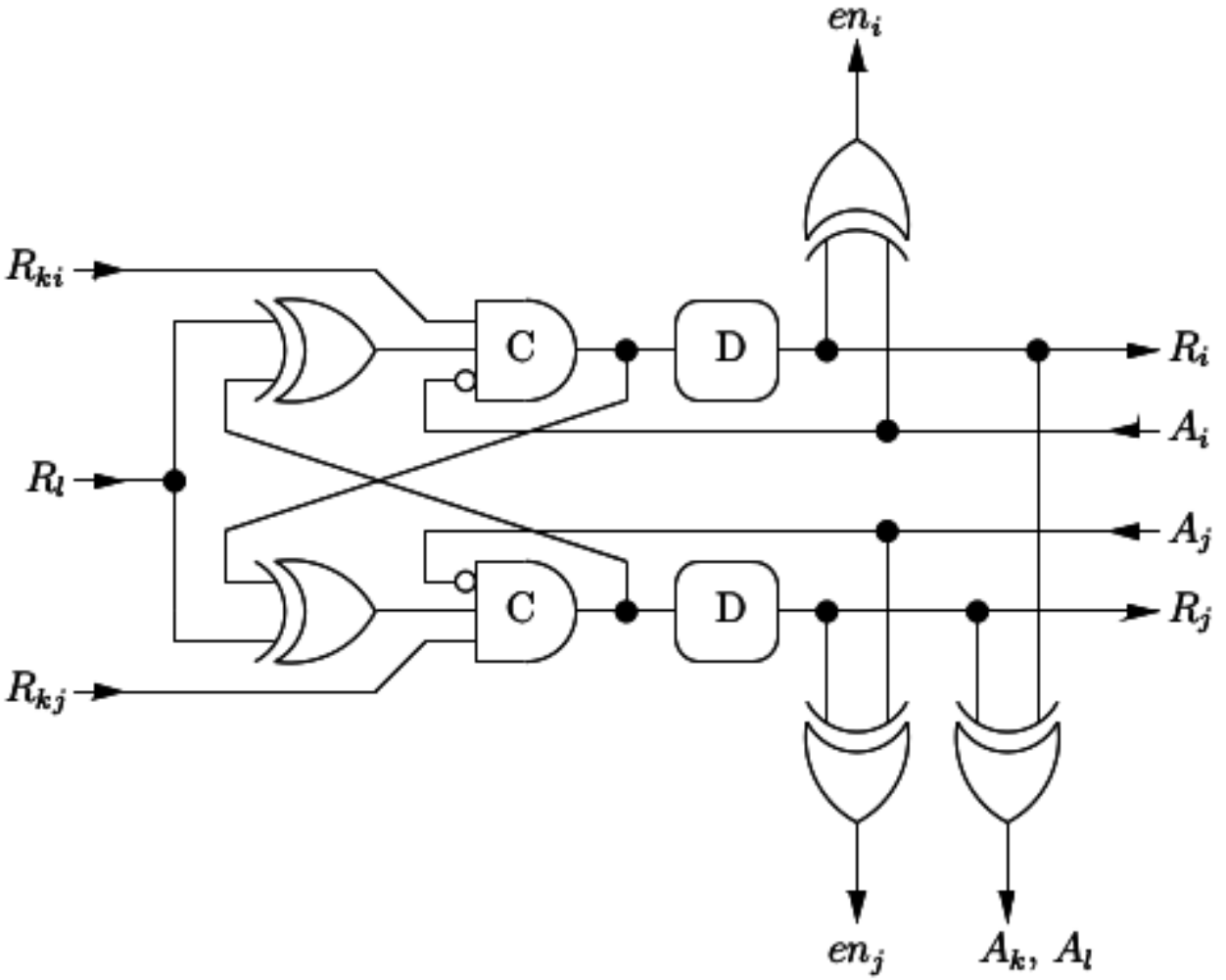


Fig. 5b : *Handshake pour le* DEMUX*22 (partie « contrôle »). Les signaux en_i et en_j (facultatifs) servent à contrôler les deux latch de sortie.*

Latch avec jeton initial

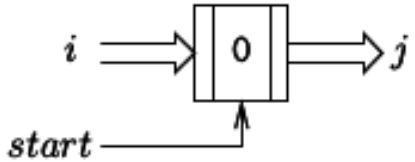


Fig. 6a : *Symbole du* LATCH_INIT_*0 qui émet initialement un 0 (partie « données »).*

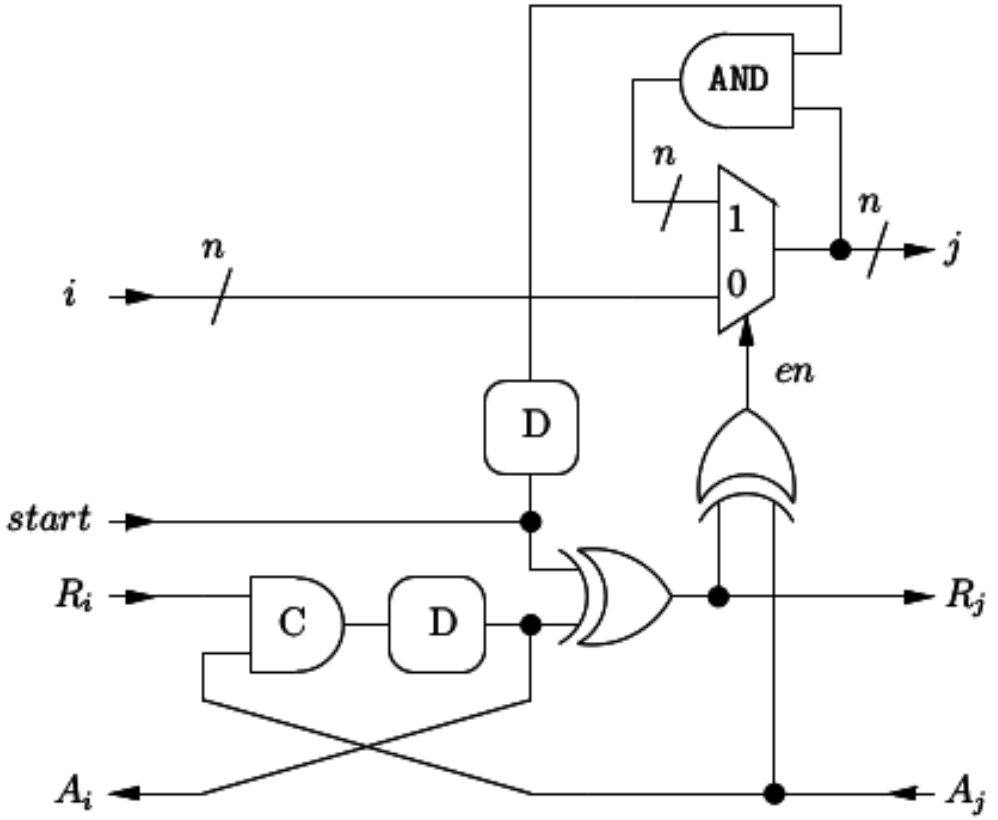
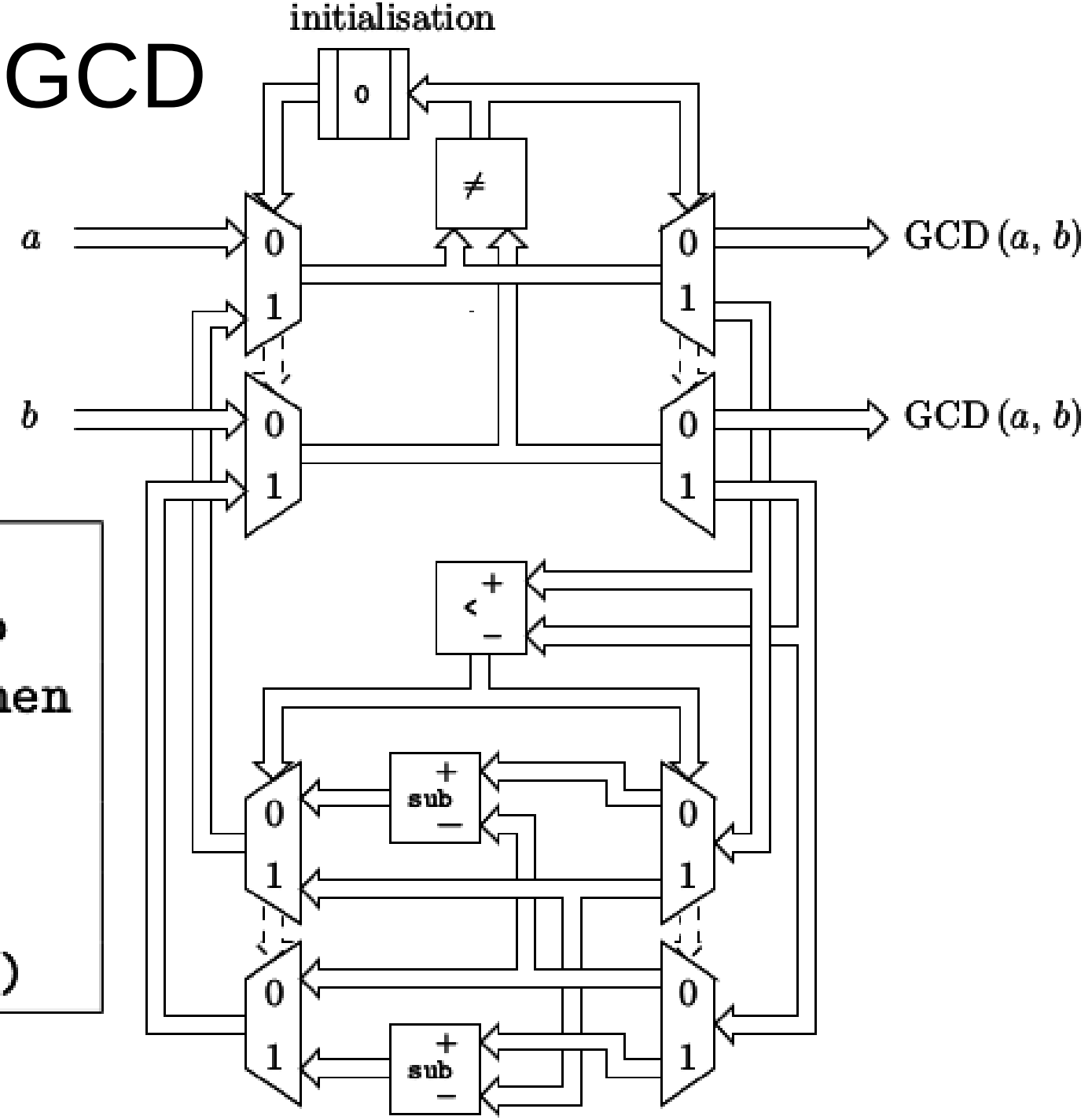


Fig. 6b : *Handshake pour le* LATCH_INIT_*0 (parties « données » et « contrôle »). Le signal start est actif sur transition montante.*

Algorithme GCD

```
Input : a, b > 0
while ( a ≠ b ) do
    if ( a < b ) then
        b ← b − a
    else
        a ← a − b
Output : GCD (a, b)
```



Validation par simulation

ACCELERATOR DESIGN WITH OPENCL

(ATHENS WEEK 19-24 MARCH, 2018)

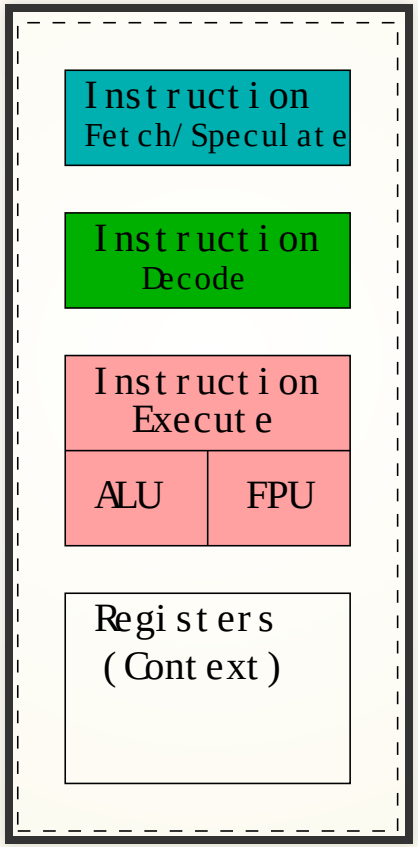
WHAT DO WE KNOW SO FAR ?

- There are three types of parallelism
  - Task Parallelism
  - Data Parallelism
  - Pipeline
- We saw the reasons for memory stalls and latency.
- The techniques to hide latency through Caching.
- The Virtual Memory.

WHAT DO WE KNOW SO FAR ?

- We saw the evolution of processors from
  - Uniprocessor to ...
  - Multicores with Simultaneous Multi-Threading.
- And we said hello to the world from our GPU (Mali-T628).

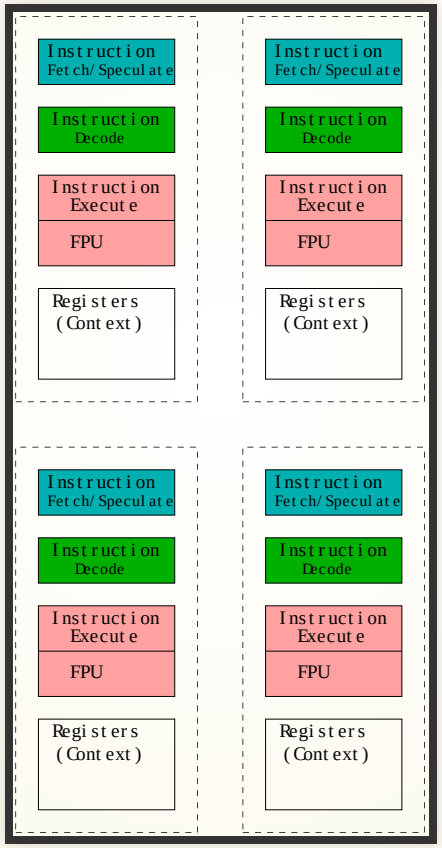
GPU ARCHITECTURE : UNIPROCESSOR



GPU ARCHITECTURE : EVOLUTION

- GPUs took a completely different path of evolution.
- Because they live in a embarrasingly data-parallel environment.
- The memory stalls/latency problems are still there.
- So are the solutions to hide them.

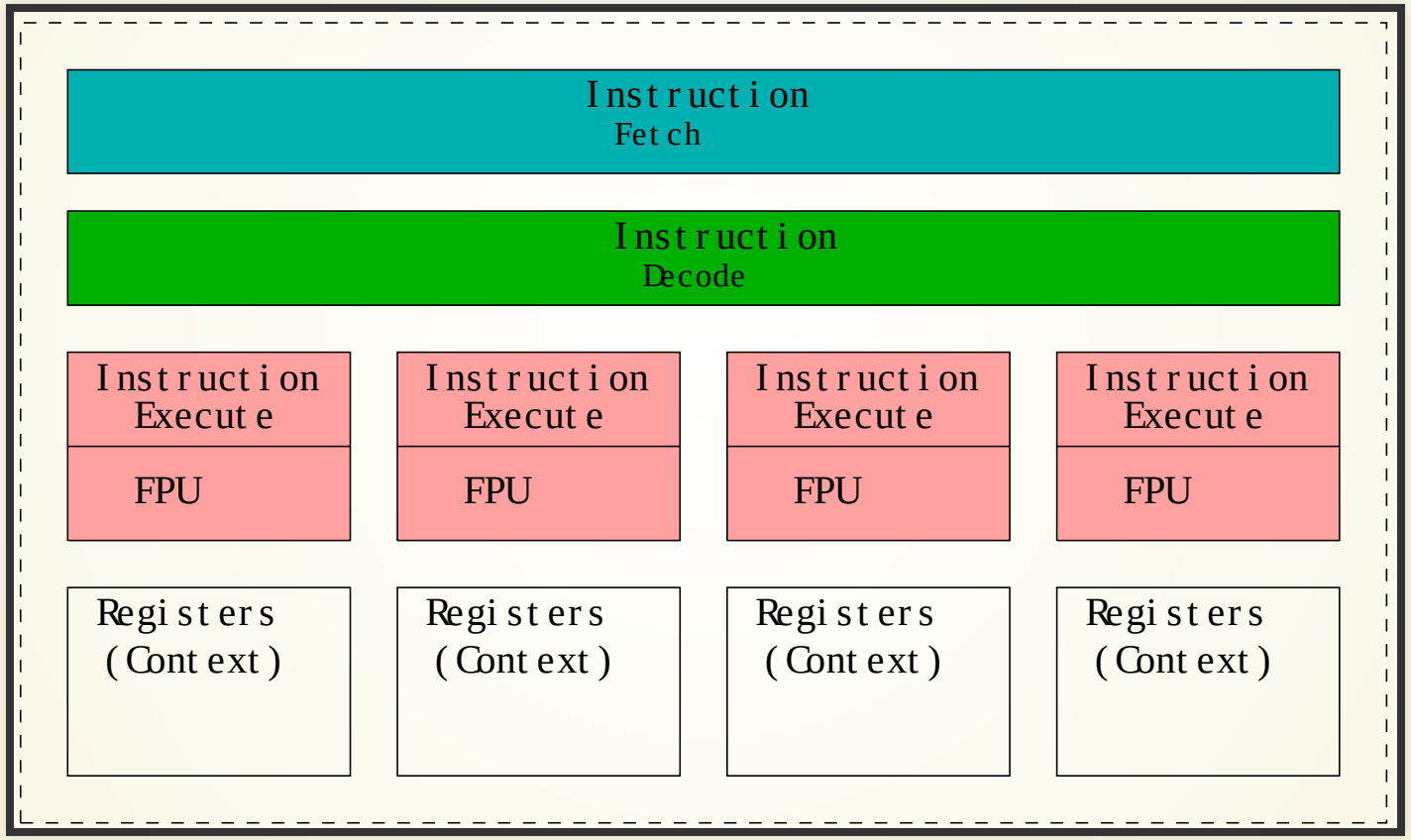
GPU ARCHITECTURE : MIMD



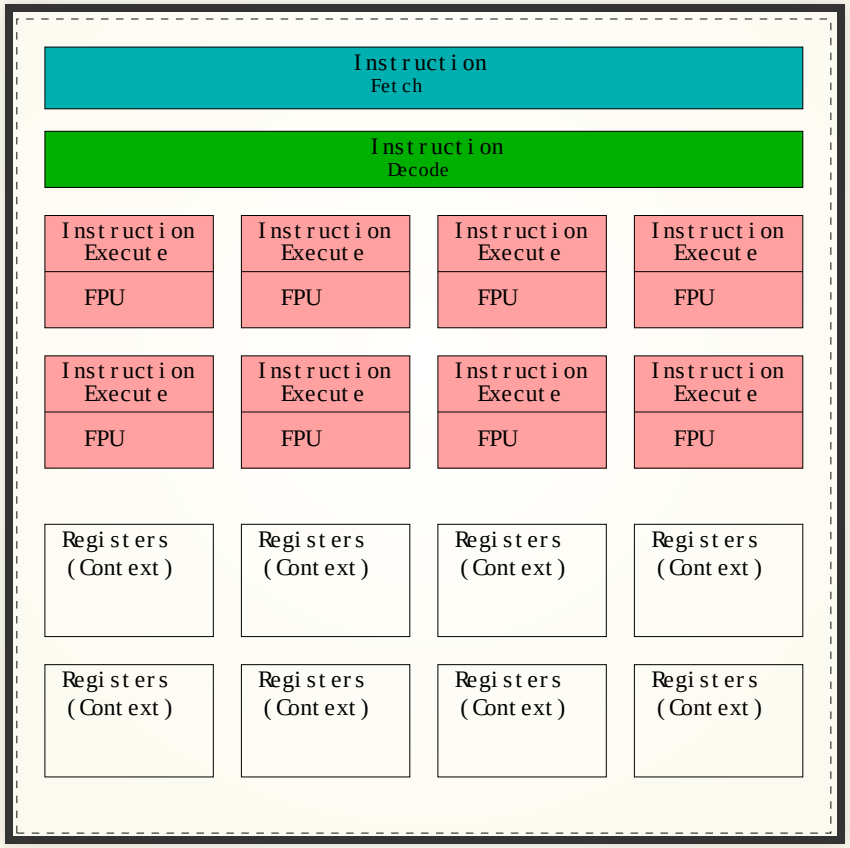
GPU Architecture : Evolution

- MIMD, but wait, we don't need the mutliple-instruction streams.
- let' get rid of them.

GPU ARCHITECTURE : SIMD



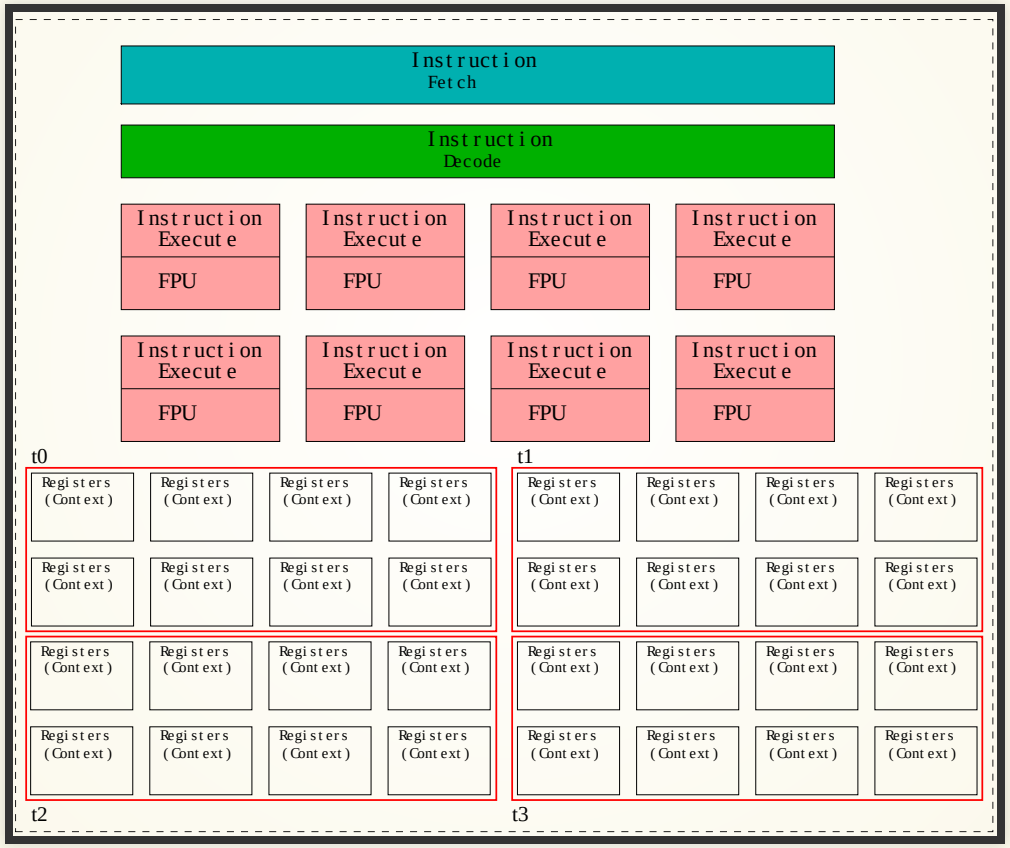
GPU ARCHITECTURE : MORE SIMD



GPU ARCHITECTURE : MORE SIMD

- Let's not forget our old friend Multi-Threading.
- Which helped us manage latency.

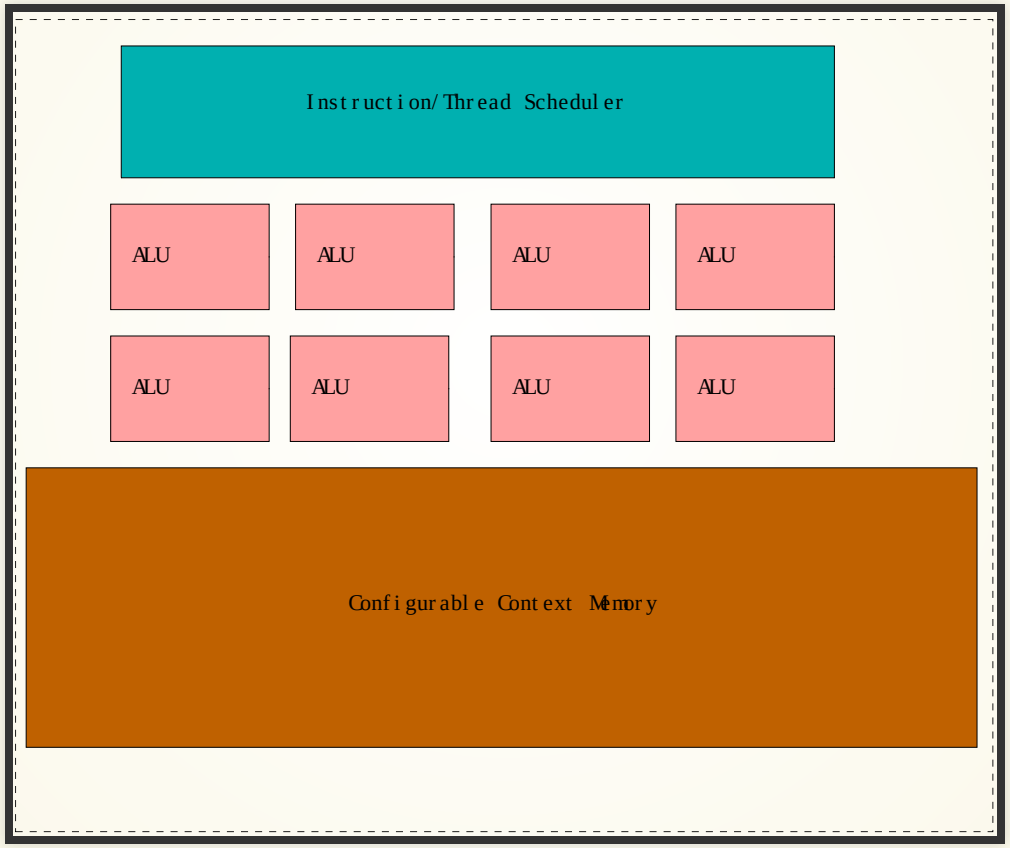
GPU ARCHITECTURE : SIMD WITH MULTI-THREADING.



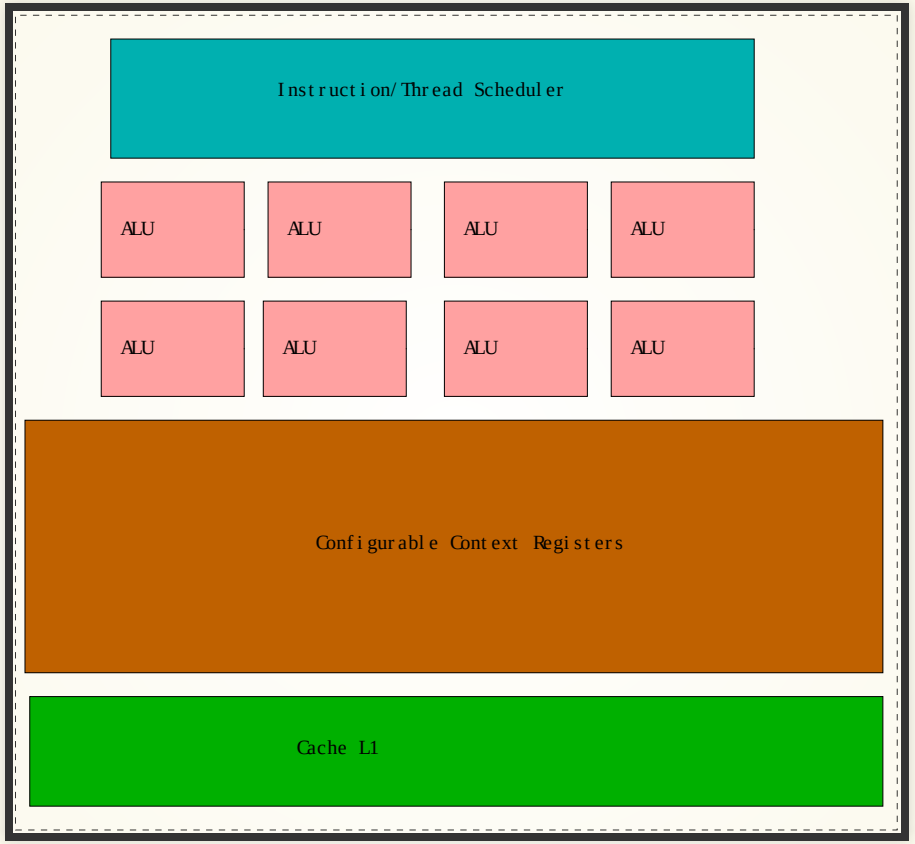
QUIZ

- What is the peak performance of this core in Gflops ?

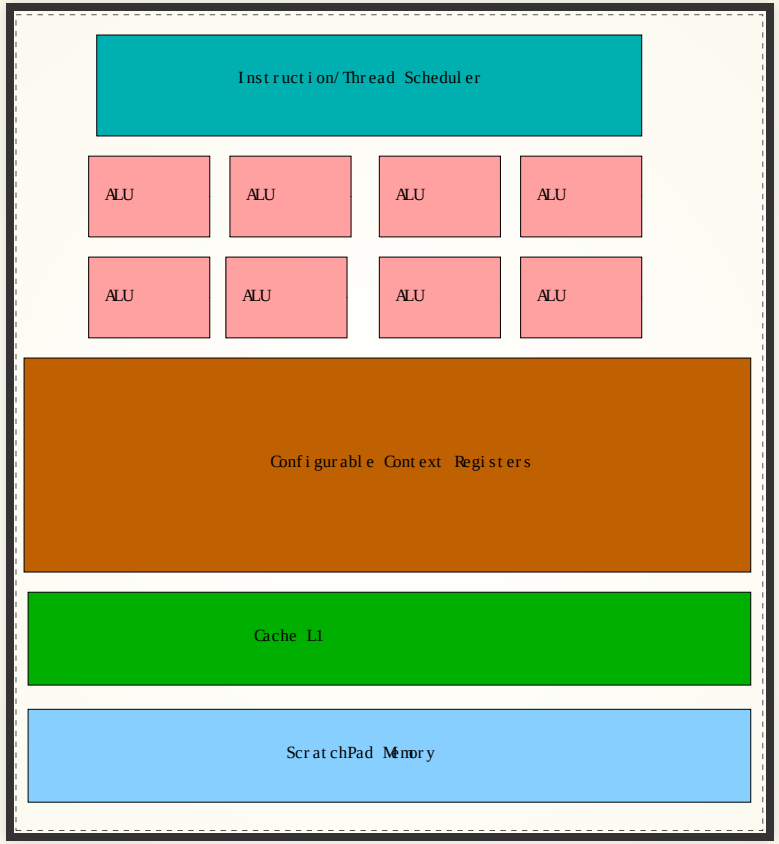
GPU ARCHITECTURE : REFINEMENTS



GPU ARCHITECTURE : REFINEMENTS

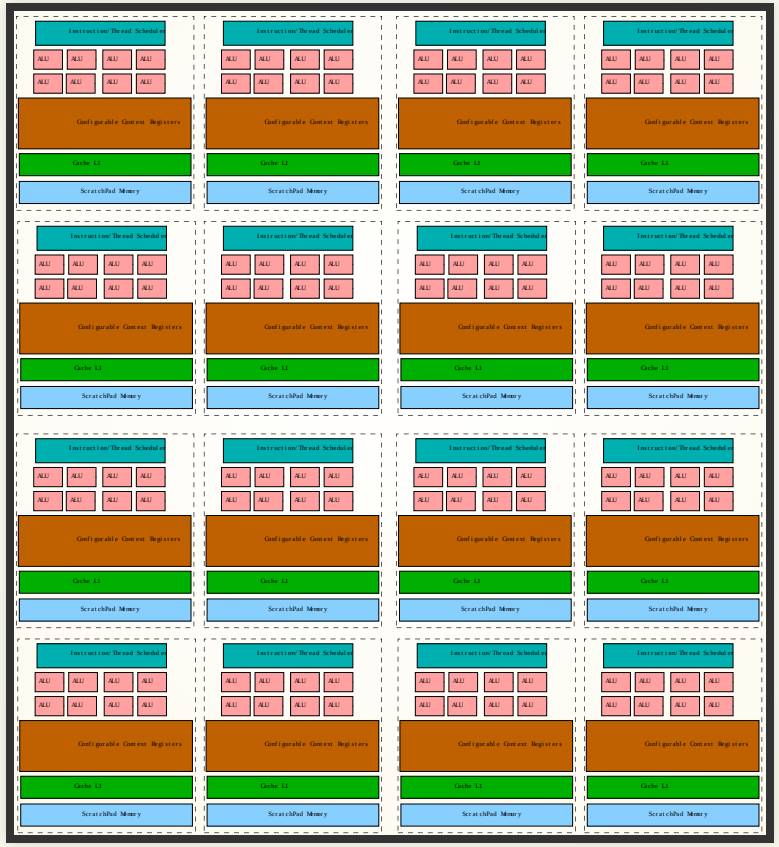


GPU ARCHITECTURE : REFINEMENTS



- Adding Scratchpad memory, so that threads can communicate locally.

GPU: MULTIPLE SHADER CORES



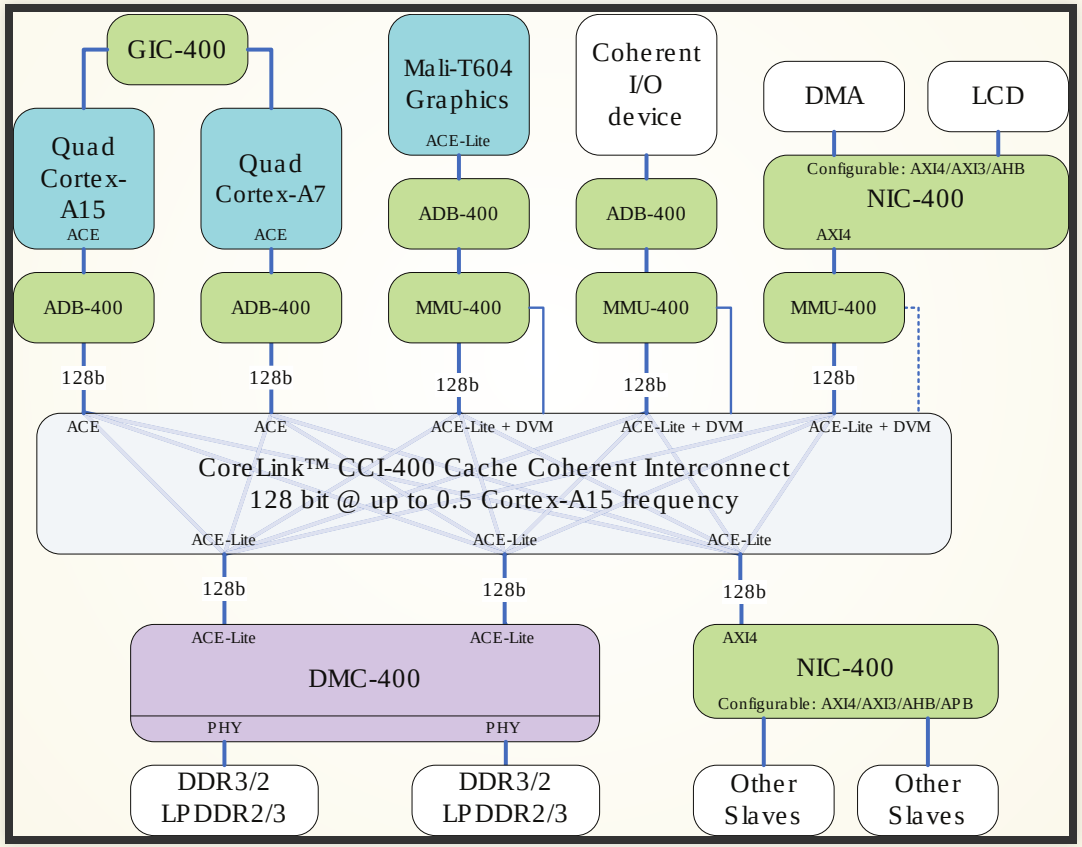
OUR GPU : MALI T628

- ARM MidGard family.
- Can be configures for 4-16 cores.
- configurable SIMD
  - 2x FP64, 4x FP32, 8x FP16, 2x int64, 4x int32, 8x int16, 16x int8
- Two L1 Caches/ Shader core 16KB
- L2 Cache can be configured for upto 64KB.
- Each core Rated at 17 Flops/cycle. (FP32)
- 64 byte Cache lines

SOURCE: MALI T628

- https://community.arm.com/graphics/b/blog/posts/the-mali-gpu-an-abstract-machine-part-3---the-midgard-shader-core
- https://community.arm.com/graphics/f/discussions/6557/mali-t628-gpu-activity-in-streamline

EXAMPLE HETEROGENEOUS SOCS



EXPRESSING PARALLEISM

- NDRangeKernel
- global_work_size() defines that total no. of elements.
- if each element is independent it is also the number of work_items.
- each work item can be associated with one thread.

EXPRESSING PARALLEISM

- the global work can be separated into groups.
- get_group_id() gives the id of the group.
- get_local_id() gives the id of the local work item within the group.

WORK ITEM RELATED FUNCTIONS:

- get_work_dim()
- get_global_size()
- get_global_id()
- get_local_size()
- get_local_id()
- get_num_groups()
- get_group_id()
- get_global_offset()

SYNCHRONIZATION FUNCTIONS: MEM FENCE

- mem_fence: all memory accesses preceding mem_fence must end before starting memory accesses following mem_fence.
- read_mem_fence : only for loads.
- write_mem_fence: only for stores.
  - arguments: CLK_LOCAL_MEM_FENCE: only load/stores to local memory.
  - arguments: CLK_GLOBAL_MEM_FENCE: only load/stores to global memory.

SYNCHRONIZATION FUNCTIONS: BARRIER

- All work-items in a work-group must execute this function before the work group can proceed.
- Barrier also issues a mem_fence either to CLK_LOCAL_MEM_FENCE or CLK_GLOBAL_MEM_FENCE.
- There is no way to synchronize work items in different work groups.

LAB WORK 1

- Vector addition with size N
- Calculate speedup with varying N.
- Measure Flops/s.
- Calculate the average of a vector.
- Calculate the average of a vector using workgroups.
- Measure speedup.

LAB WORK 2

- Write a Matrix multiplication routing with two matrices of size M x K, K x N.
- where M=K=N
- measure speed up
- use streamline to see various statistics about Cache/TLB miss.
- Measure Flops/S.

DEBUGGER: MGD

- in a405-xx.enst.fr (desktop) clone the git depot.
- source init.sh > /dev/null
- module load mali/4.4
- mgd
  - in odroid
  - source init_odroid.sh
  - mgddaemon
  - make debug

PERFORMANCE MONITOR: STREAMLINE

- run start_gator.sh in tpt39/
  - cd tpt39; ./start_gator.sh&
- in a405-XX.enst.fr
  - $ source init.sh
  - $ module load mali/4.4
  - $ streamline

Institut
Mines-Télécom

Beyond Bits: A Quaternary FPGA Architecture using Multi-V_t Multi-V_{dd} FDSOI Devices

Sumanta Chaudhuri
<sumanta.chaudhuri@telecom-paristech.fr>
16/05/2018

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

Bits, Trits or Quits ?

- What is the optimum radix for representing numbers ?
- Radix 10:
  - Alphabet (0..9), Width of Word $log_{10}(N)$
- Radix 1:
  - Alphabet (0), Width of Word N
- Radix 10000:
  - Alphabet (0..9999), Width of Word $\frac{log_{10}(N)}{4}$
- Radix r:
  - Alphabet (0..r-1), Width of Word $w = log_r(N)$

Bits, Trits or Quits ?

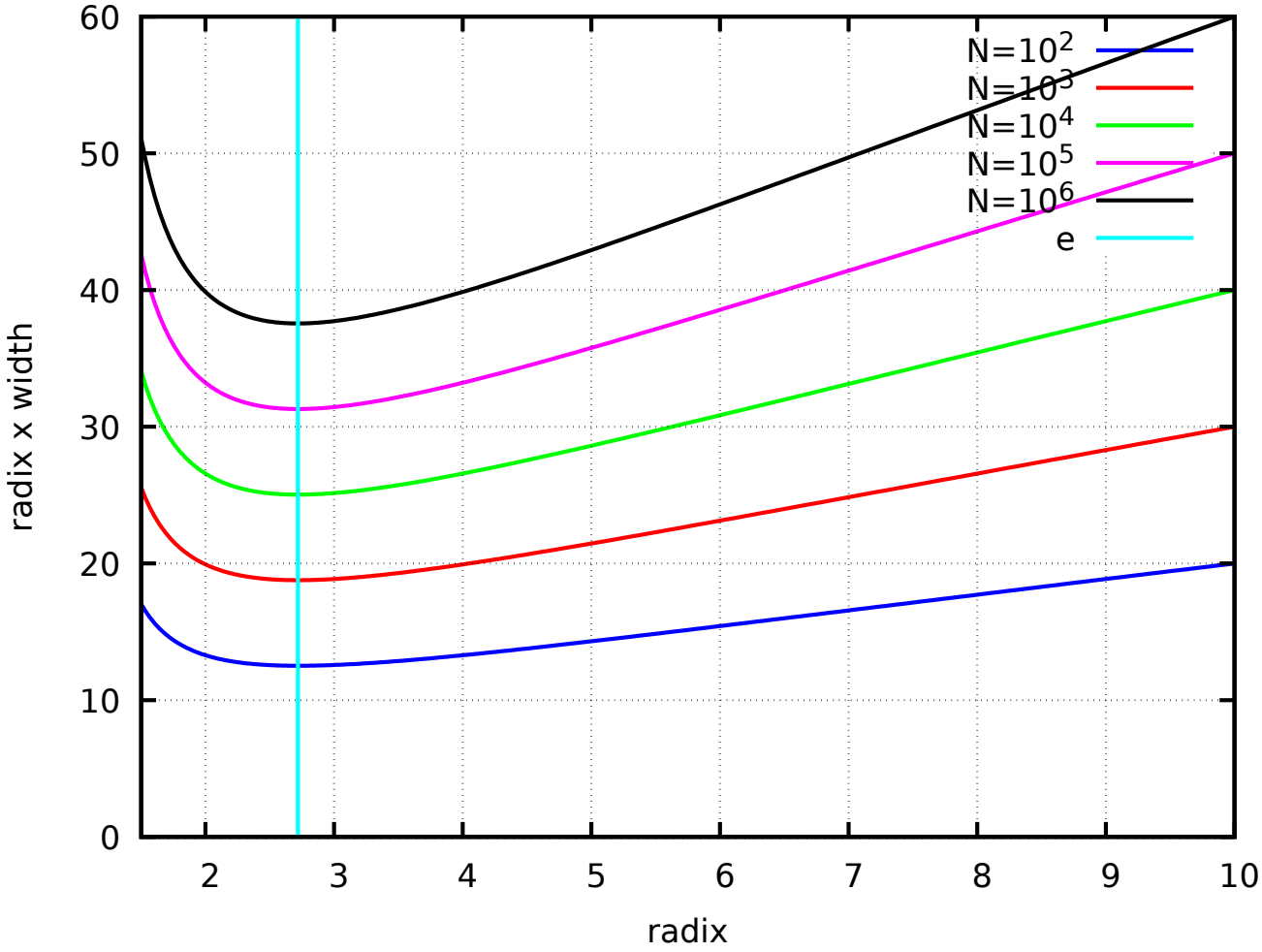


Figure: Most Economical radix is e (2.718), Binary and Quaternary are equivalent.

Compute or Communicate?

- Algorithms/Methods are either computation bound or communication bound.
- Communication bound methods
  - In case of processor: performance is determined by memory bandwidth
  - In case of circuits: performance is determined by interconnect.
- Multi-Valued Logic is efficient for communication. less wires.
- It is also near optimum for number representation.

Cost of Interconnect

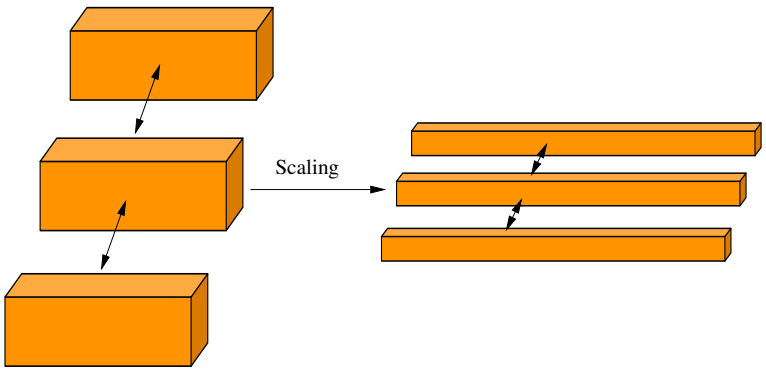


Figure: Interconnect resistance doesn't scale with technology.

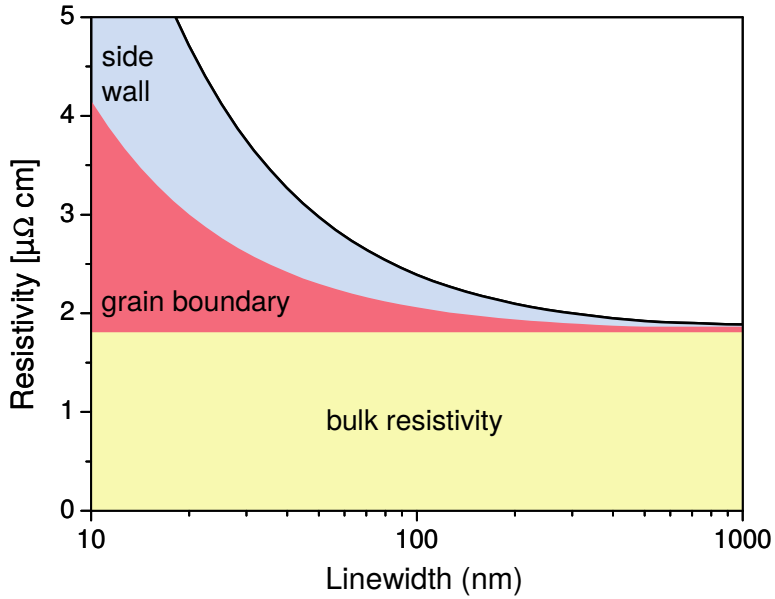


Figure INTC1 Cu Resistivity

Figure: surface scattering at lower dimensions. Taken from ITRS 2007

Interconnect Scaling

- Chips size gets bigger with every technology node.
- More Repeaters need to be inserted to compensate for RC delay.
- Rise in interconnect power consumption.

Interconnect: Possible Solutions

- Time Multiplexing ???
- Monolithic 3D ?
- Optical Interconnect ?
- Carbon Nanotube ?
- Multi-Valued Signalling ??

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

Multiple Valued Signalling

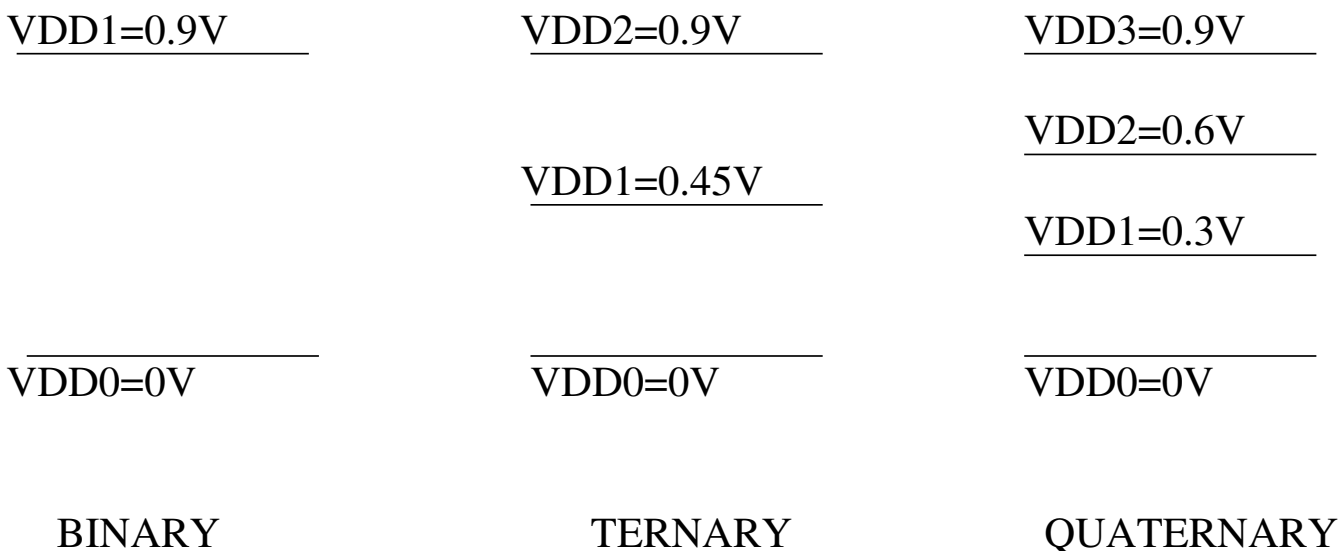


Figure: Multiple Valued Signalling

- Reduction of Routing Congestion.
- Saving Energy/Bit.
- More efficient arithmetic Implementation. (Optimum: ternary logic)

Multiple Valued Signalling: Energy Savings

Table: Energy for different transitions in a 4-valued signal, IN0(=0 $\times$ *vdd*),IN1(= $\frac{1}{3} \times VDD$), IN2(=$\frac{2}{3} \times VDD$), IN3(=*VDD*).

4-Valued transitions	
Transitions	Energy
t_{00},t_{11},t_{22},t_{33}	0
t_{01}, t_{12}, t_{23}, t_{32}, t_{21}, t_{10}	$C \times \frac{1}{9} \times vdd^2$
t_{02}, t_{13}, t_{20}, t_{31}	$C \times \frac{4}{9} \times vdd^2$
t_{03}, t_{30}	$C \times vdd^2$
Av. Energy/Tran	$0.27 \times Cvdd^2$

3-Valued transitions	
Transitions	Energy
t_{00},t_{11},t_{22}	0
t_{01}, t_{12}, t_{21}, t_{10}	$C \times \frac{1}{4} \times vdd^2$
t_{02}, t_{20}	$C \times \times vdd^2$
Av. Energy/Tran	$0.33 \times Cvdd^2$

2-Valued transitions	
transitions	Energy
t_{00},t_{11}	0
t_{01}, t_{10}	$C \times vdd^2$
Av. Energy/Tran	$0.5 \times Cvdd^2$

Multi-Valued Logic: Why ?

Why ?

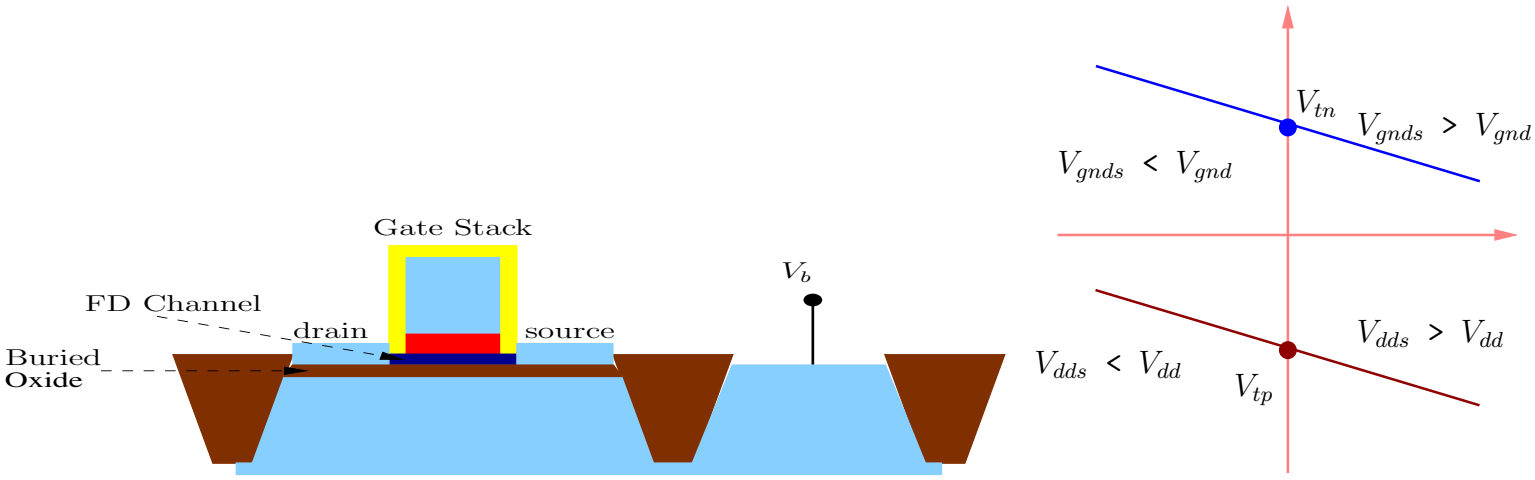
- Energy saving in the interconnect.
- Reduced routing congestion. 50% for quaternary, 33% for ternary.
- Area saving in the steering logic.

Why Not ?

- Increased Complexity of transmitter and receiver.
- Reduced noise margin.

Multi-Valued Logic: Why Now ?

- FDSOI permits the fine-tuning of V_t.



(a) FDSOI Transistor (b) Back-biasing

Figure: Brief Overview of FDSOI Technology.

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

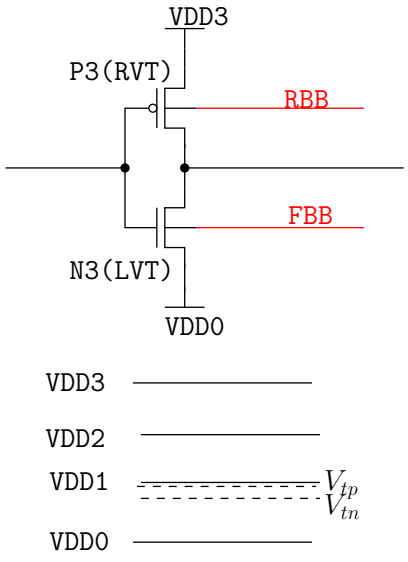
FDSOI: Advantages

- Field Depleted Silicon On Insulator.
- From 28nm we have only FDSOI or FINFET.
- Improved junction capacitance.
- Lower process variation, absence of RDF (Random Dopant Fluctuation)
- FDSOI permits the fine-tuning of V_t, electrically
  - Applying back-biasing to back plane.

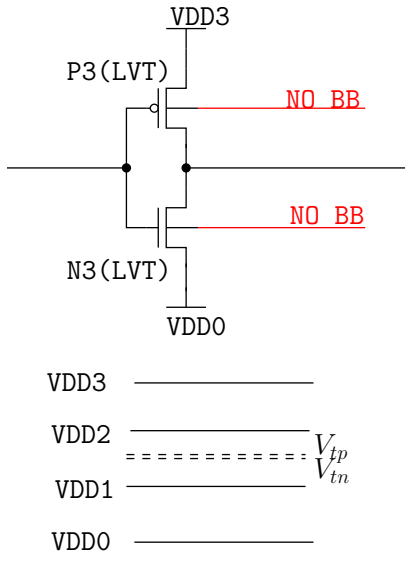
RVT	RBB upto -3V	FBB upto +300mv
LVT	FBB upto +3V	RBB upto -300mv

  - By increasing gate length. (Poly-Biasing)

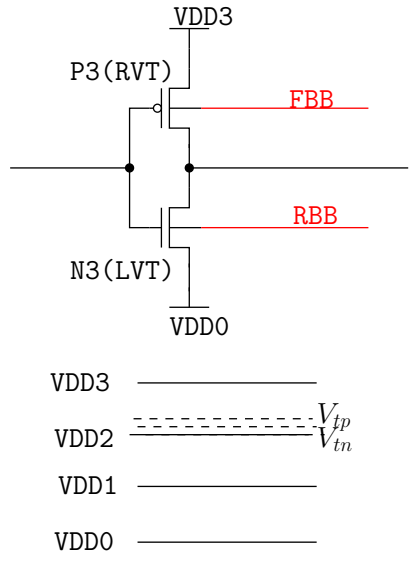
Basic Blocks: DLCs



DLC0



DLC1

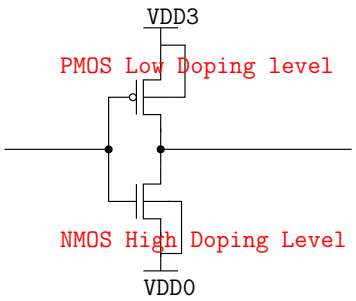


DLC2

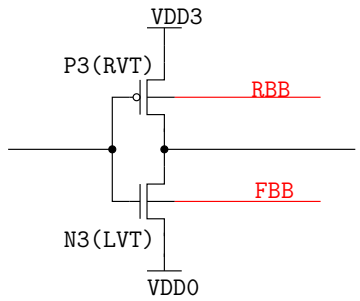
Table: Down-Literal Converters

Down-Literal Converters			
input	DLC0	DLC1	DLC2
0	3	3	3
1	0	3	3
2	0	0	3
3	0	0	0

DLCs: Comparison with Earlier work



DLC0: State of The Art



DLC0: FDSOI

- A basic block similar to inverter, required for multiplexers, and other circuits.
- Earlier implementation propose different dopings of the PMOS and NMOS.
- With FDSOI the V_t can be varied electrically.

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

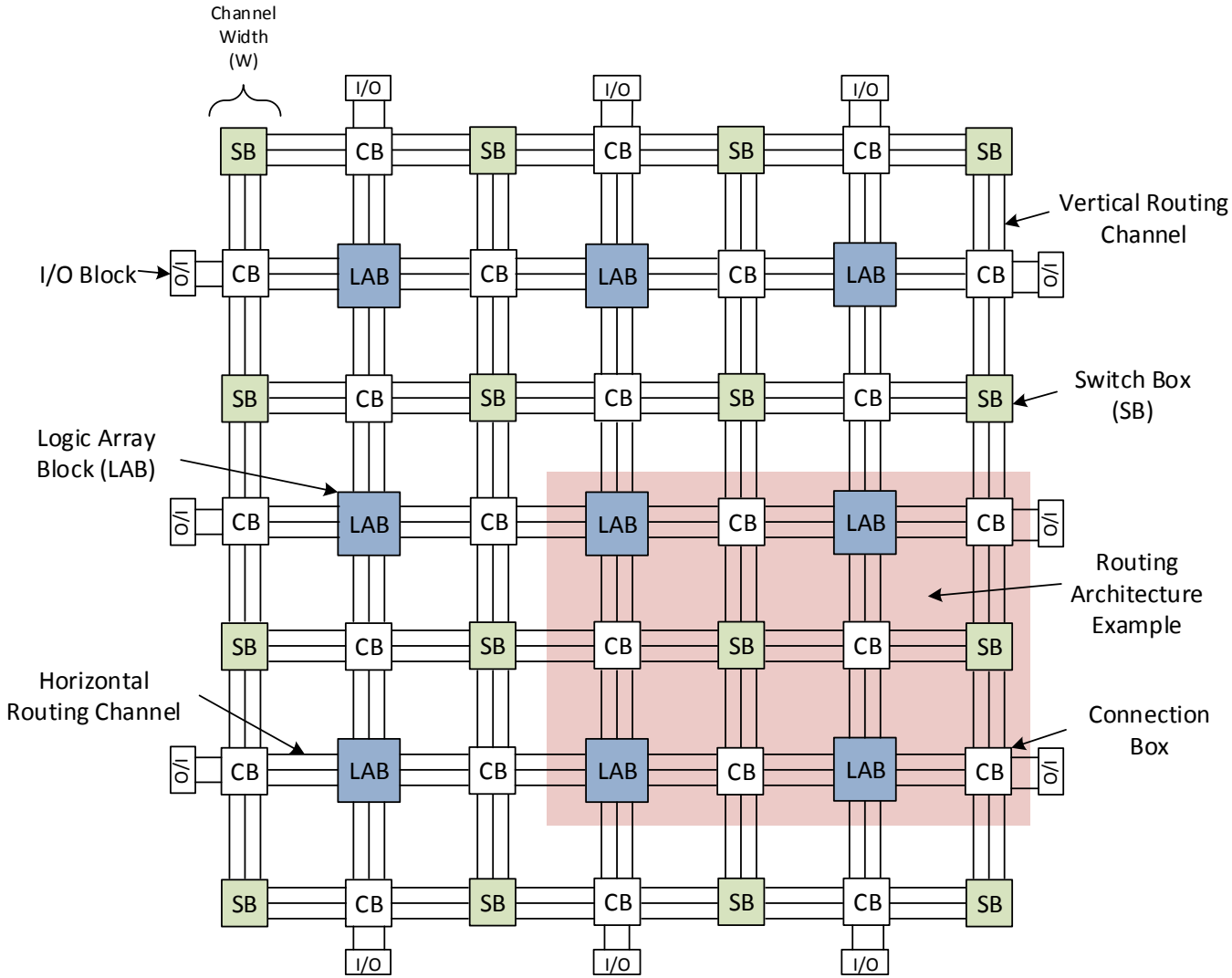
Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

FPGAs Have A Interconnect Problem.



FPGAs: The Interconnect Overhead

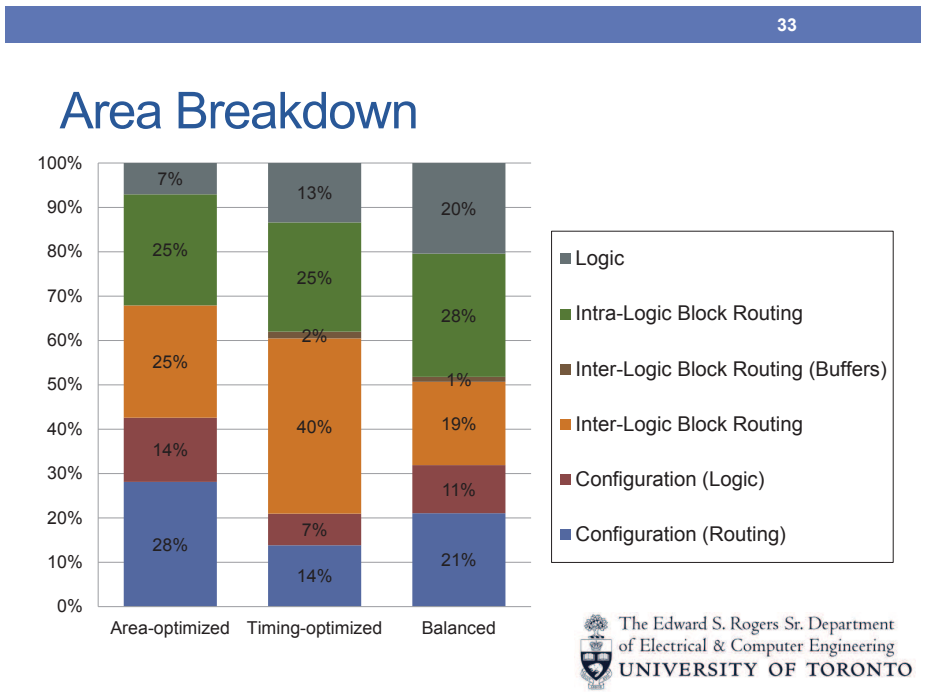


Figure: Area Overhead (A study published in FPL 2016 [4]).

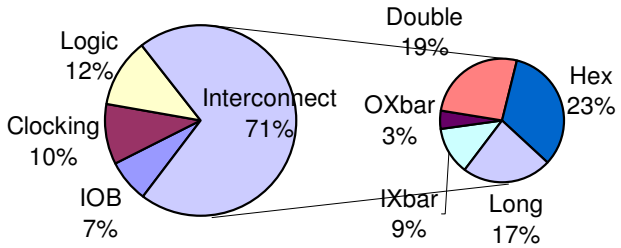
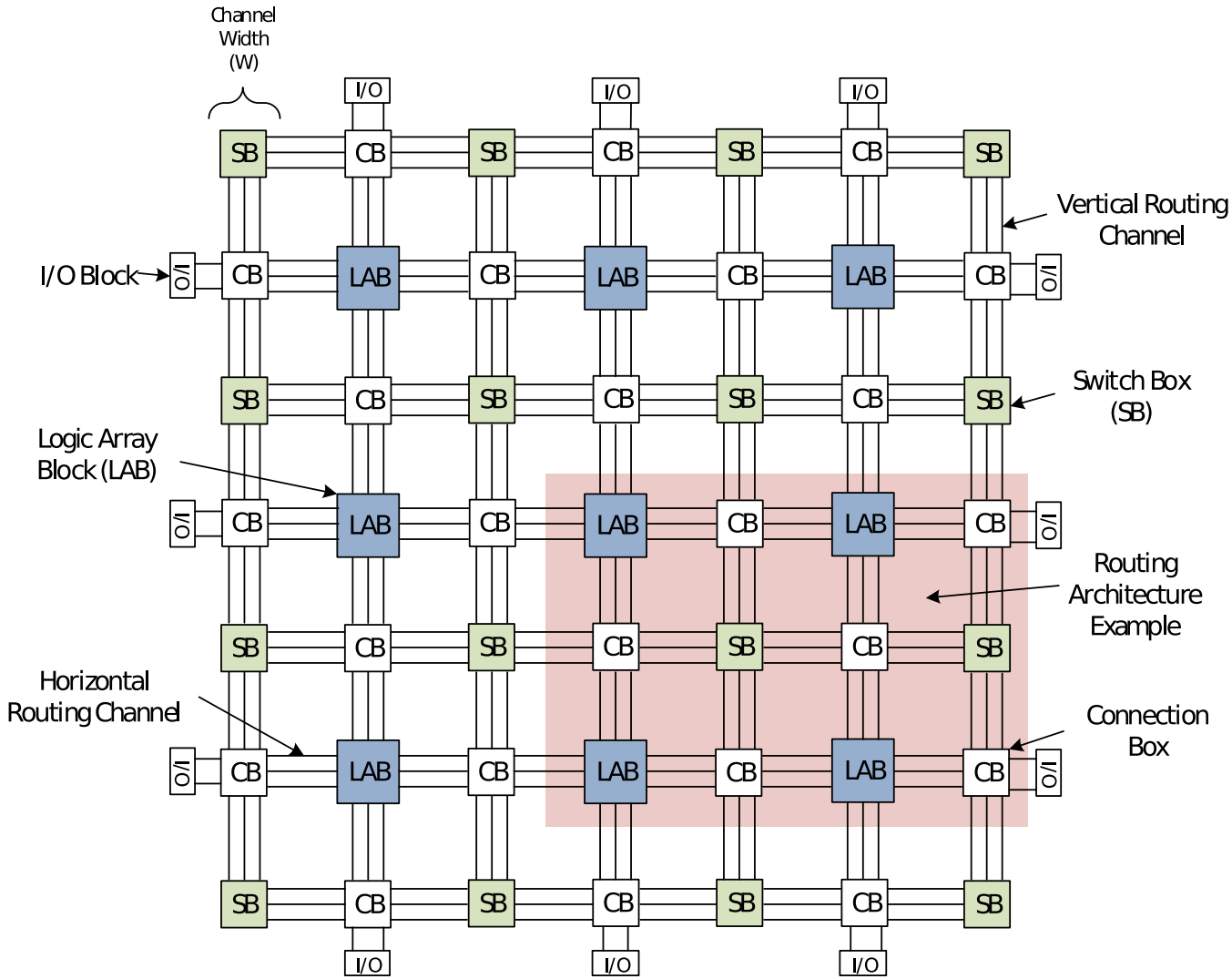
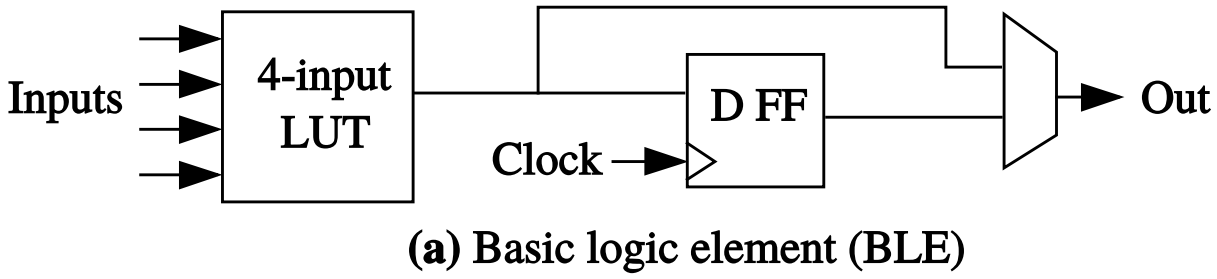


Figure: Power consumption in FPGAS. Taken from [3]

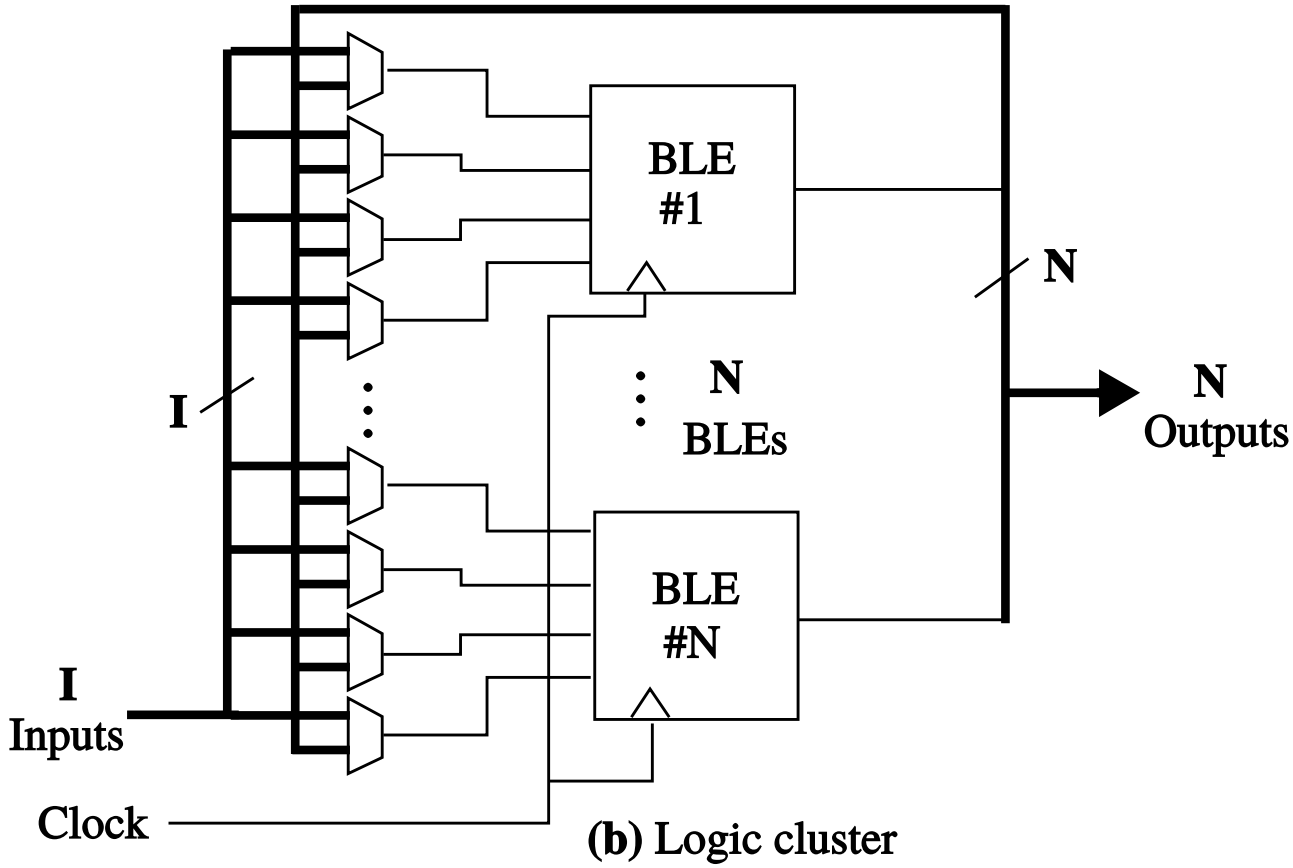
FPGA: Architetcure



FPGA: Logic Architetcure



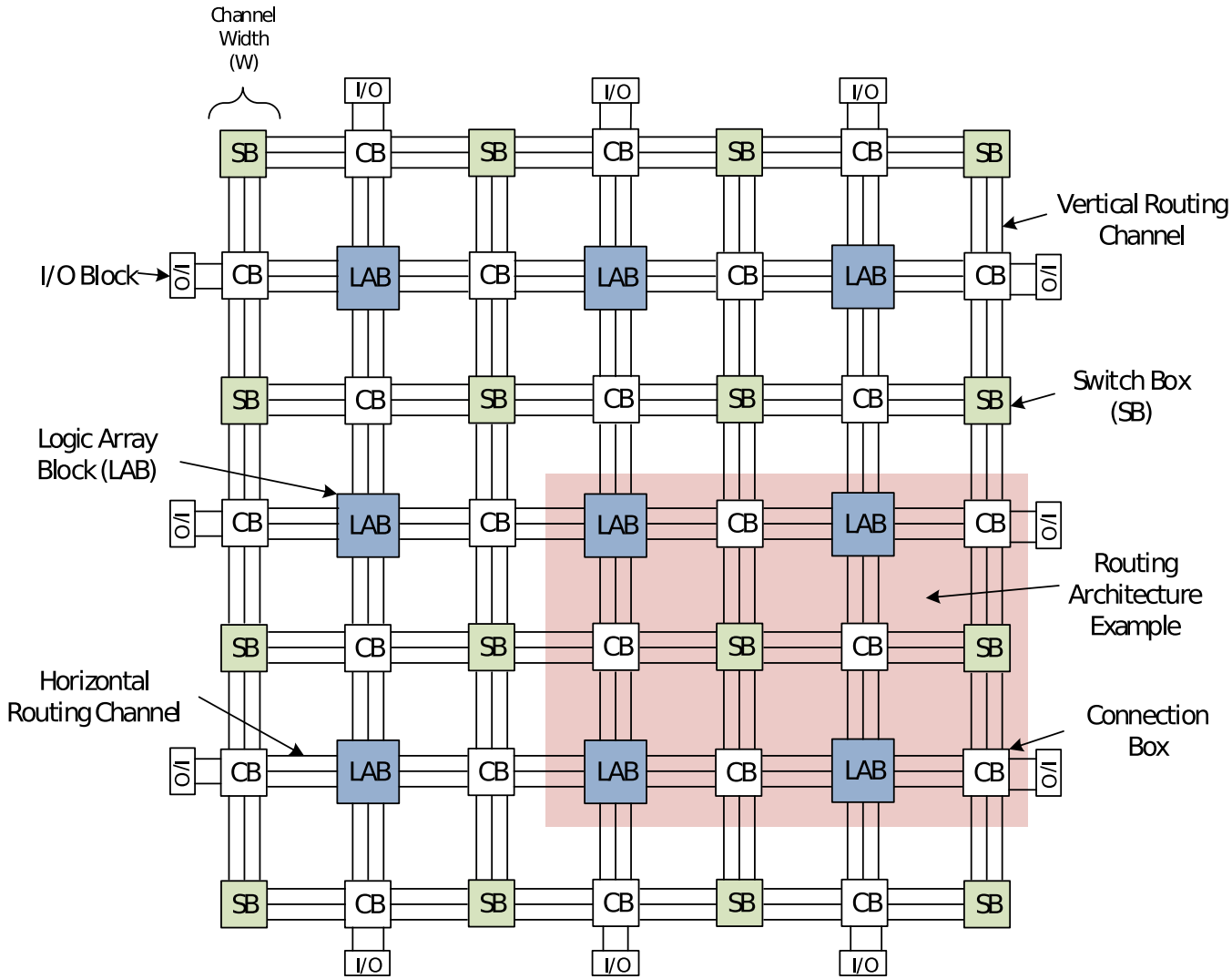
(a) Basic logic element (BLE)



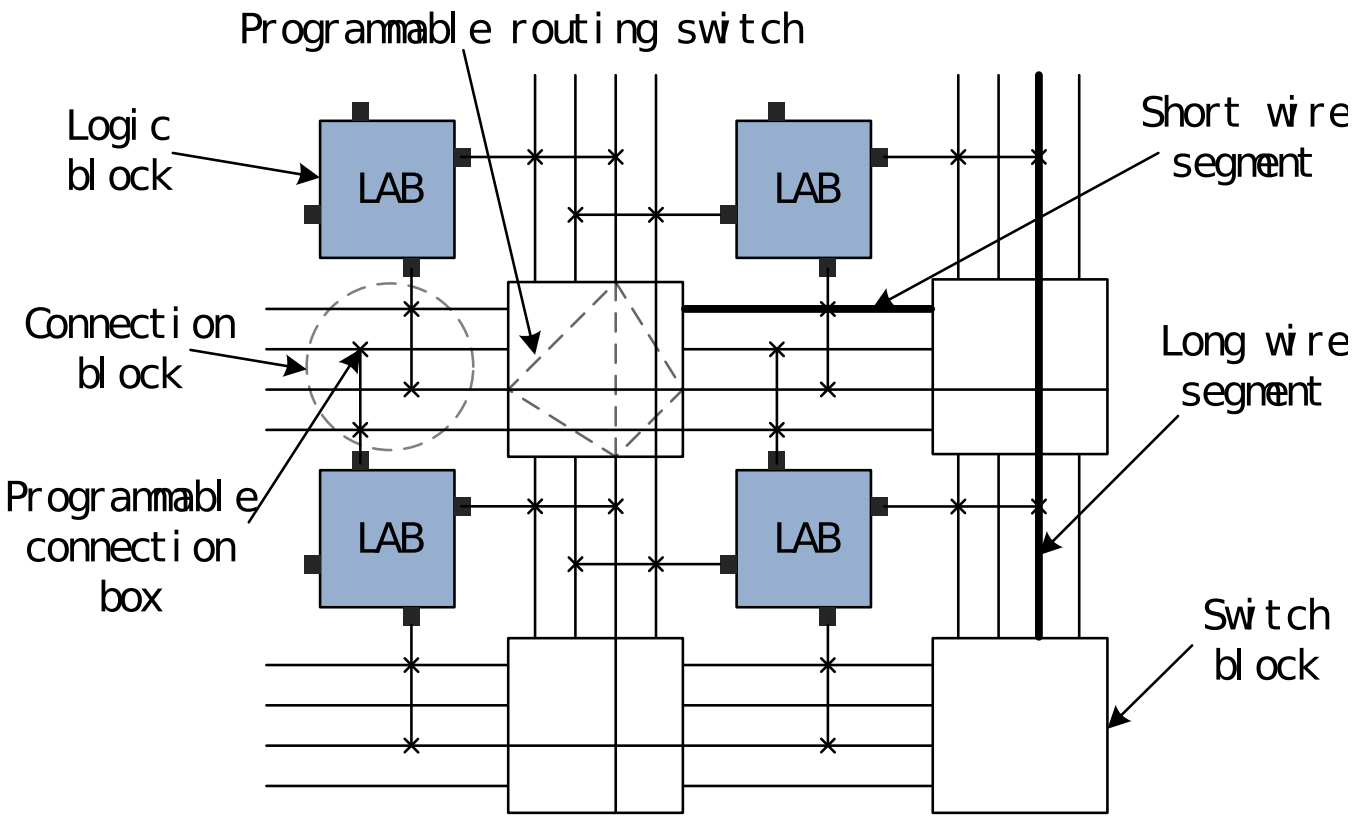
(b) Logic cluster

Fig. 1. Structure of basic logic element (BLE) and logic cluster.

FPGA: Architetcure



FPGA: Routing Architetcure



Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

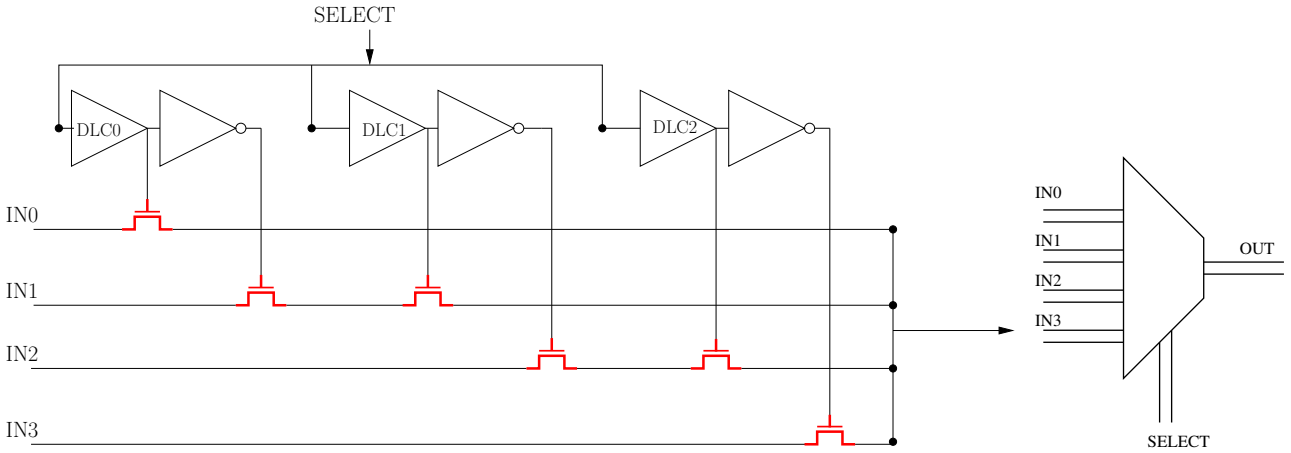
Experiments

Questions ?

FPGA: Primitives

- Lookup Tables
- Flip-Flops
- Switch Boxes
-
  - multiplexers
  - Buffers/ Repeaters
- Connection Boxes (basically Multiplexers)
-
  - multiplexers
  - Buffers/ Repeaters
- I/O Elements

Primitives: QMUX

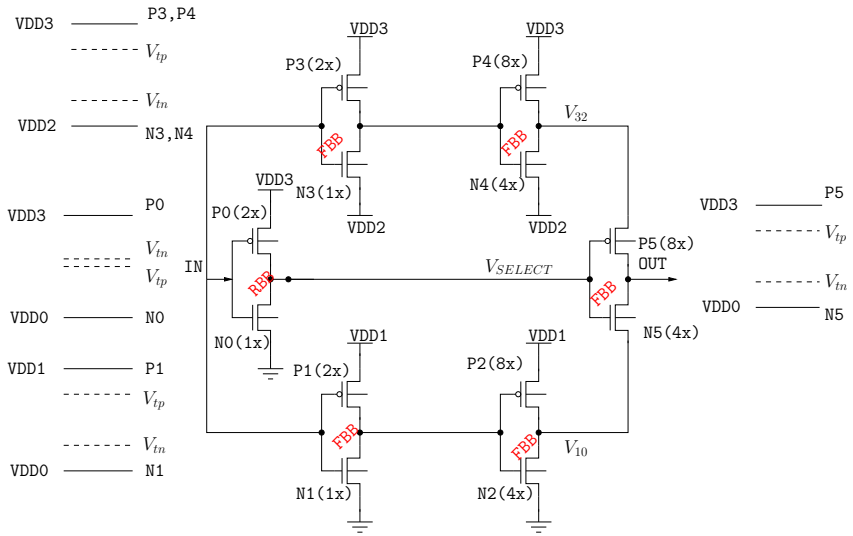


(a) 4-valued Multiplexers with DLCs (b) eq. binary

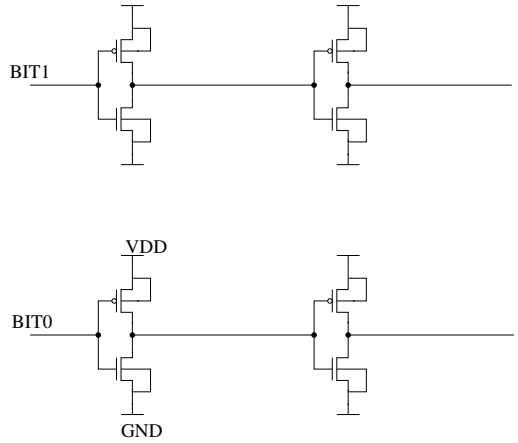
Table: Operation of the 4-valued Multiplexer

Mux Operation						
i/p	DLC0	$\overline{DLC0}$	DLC1	$\overline{DLC1}$	DLC2	$\overline{DLC2}$
0	3	0	3	0	0	3
1	0	3	3	0	3	0
2	0	3	0	3	3	0
3	0	3	0	3	3	0

Primitives: QBUFS



(c)



(d) eq. binary

IN	N0	P0	N1	P1	N2	P2	N3	P3	N4	P4	N5	P5
0	OFF	ON	OFF	ON	ON	OFF	OFF	ON	ON	OFF	ON	OFF
1	OFF	ON	ON	OFF	OFF	ON	OFF	ON	ON	OFF	ON	OFF
2	ON	OFF	ON	OFF	OFF	ON	OFF	ON	ON	OFF	OFF	ON
3	ON	OFF	ON	OFF	OFF	ON	ON	OFF	OFF	ON	OFF	ON

Primitives: Repeater Waveforms

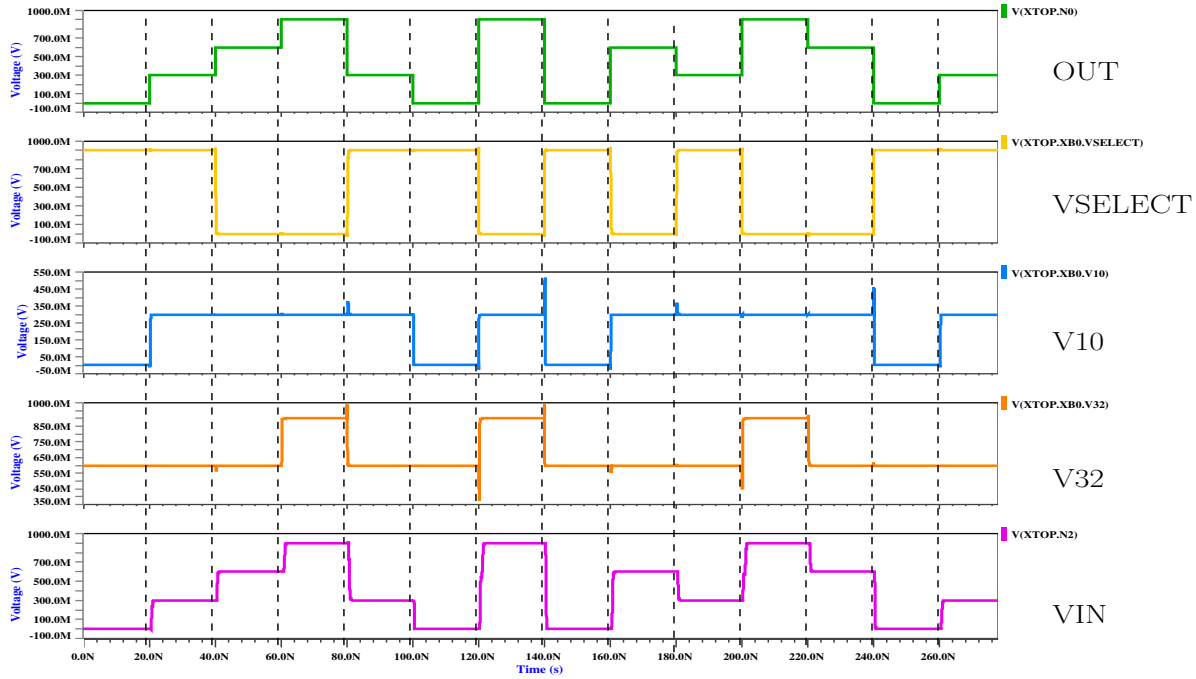


Figure: SPICE (ELDO) Simulation results for the Quaternary Repeater. Waveforms for V_{32}, V_{10} & V_{SELECT} illustrates the operation of the repeater.

Primitives: QLUTs

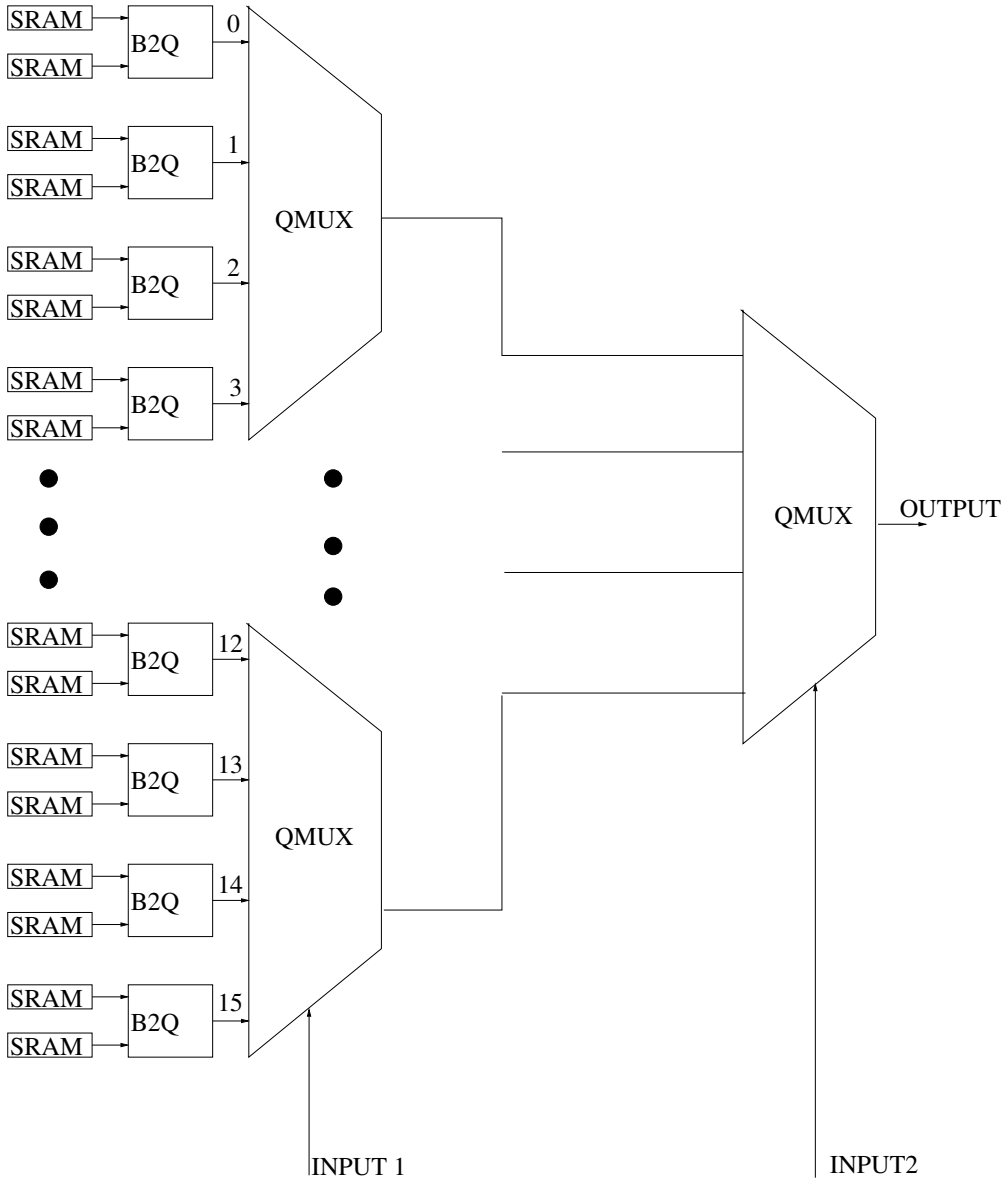
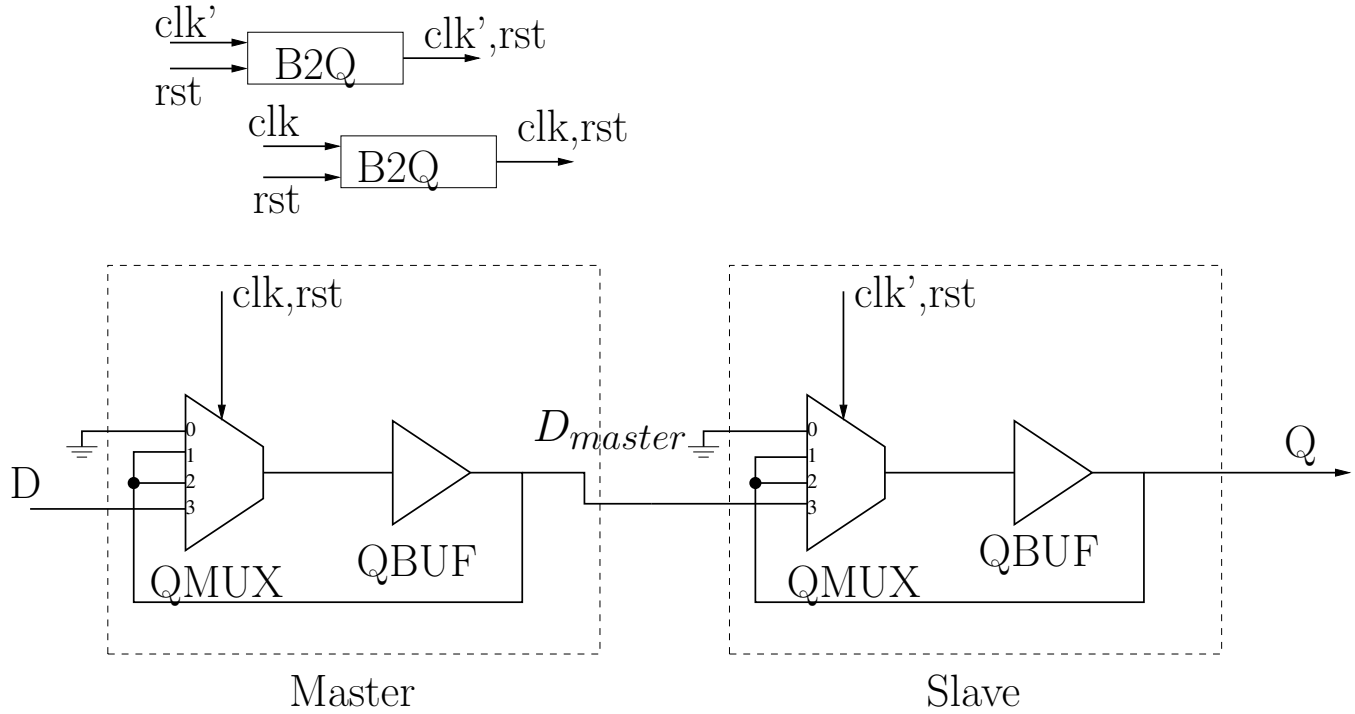


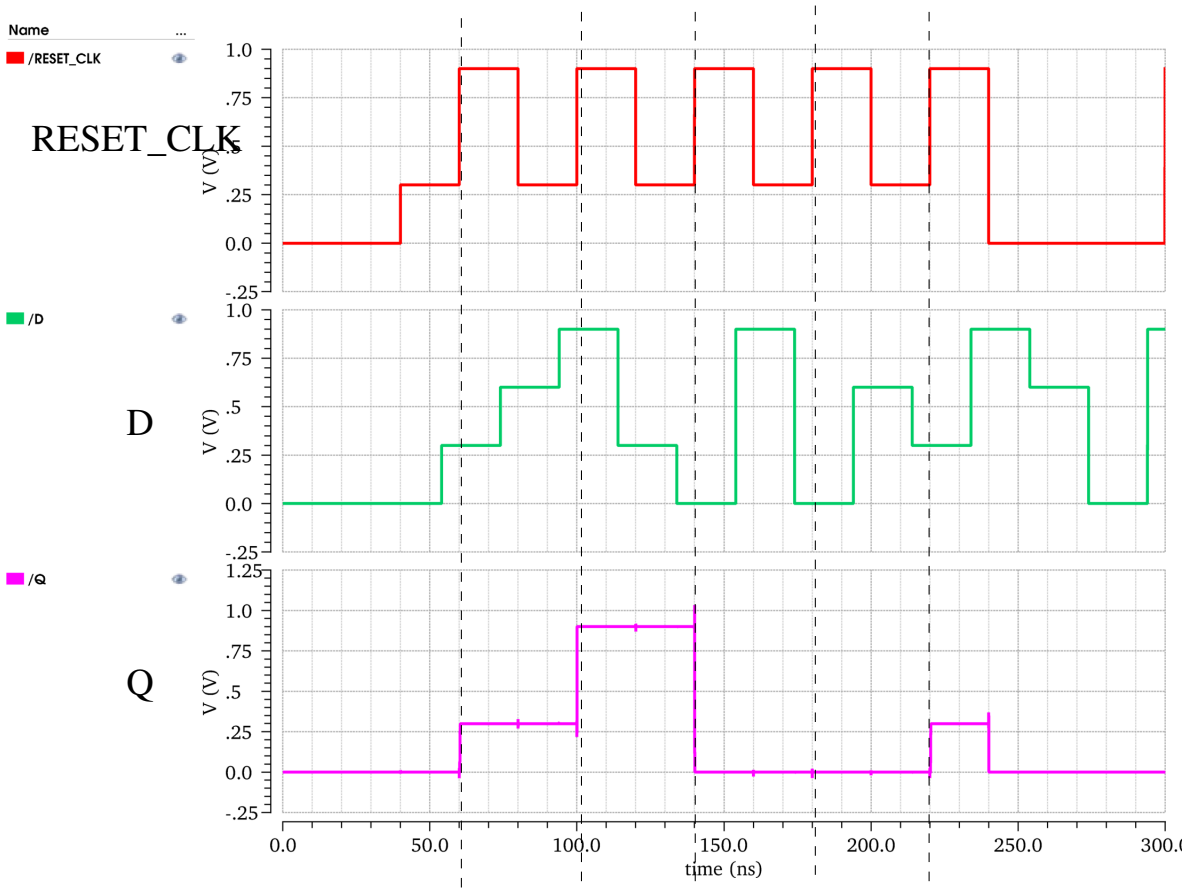
Figure: 2-input Quaternary Look-Up Tables (QLUT), with 16

Primitives: QFFs



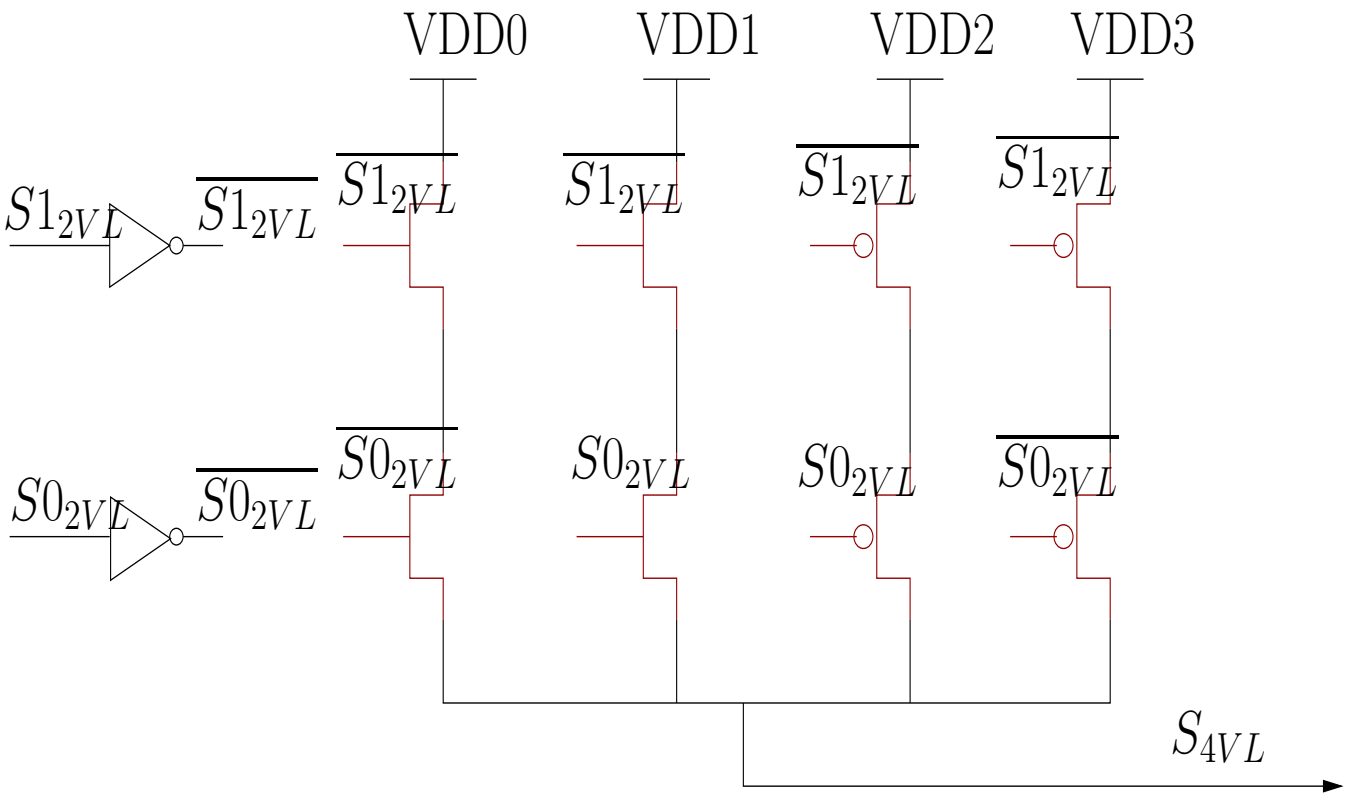
(a) Quaternary Flip-Flop with two master and slave latches, each comprising of a quaternary repeater and multiplexer.

Primitives:QFFs



(b) Waveforms for the FF operation, Reset and clock is combined into one quaternary signal where level ’0’ is reset, and clk oscillates between level ’1’ and ’3’

Primitives:B2Q



Primitives:Q2B

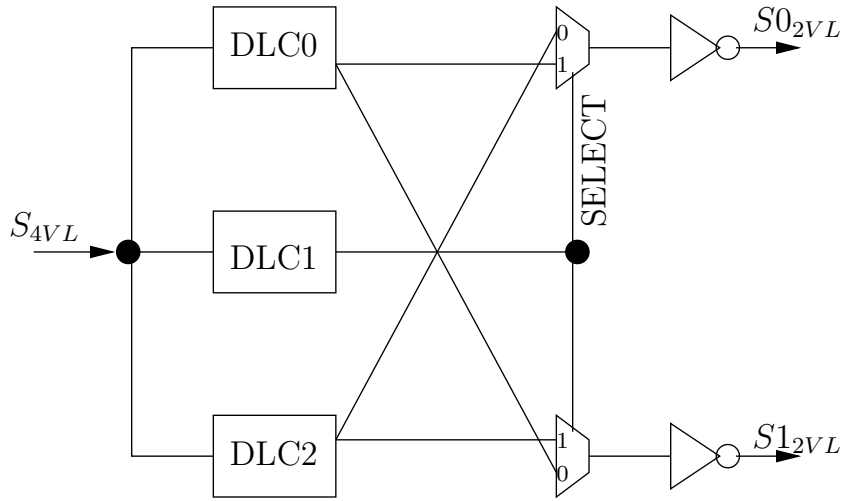


Table: Operation of 4-to-2 translator (Q2B),
0(=0 $\times$ *VDD*),1(= $\frac{1}{3} \times$ *VDD*), 2(=$\frac{2}{3} \times$ *VDD*), 3(=*VDD*).

S_{4VL}	DLC0	DLC1	DLC2	SELECT	S0	S1
VDD0	VDD3	VDD3	VDD3	1(VDD3)	1(VDD3)	1(VDD3)
VDD1	VDD0	VDD3	VDD3	1(VDD3)	0(VDD0)	1(VDD3)
VDD2	VDD0	VDD0	VDD3	0(VDD0)	1(VDD3)	0(VDD0)
VDD3	VDD0	VDD0	VDD0	0(VDD0)	0(VDD0)	0(VDD0)

Overall View: Quaternary FPGA

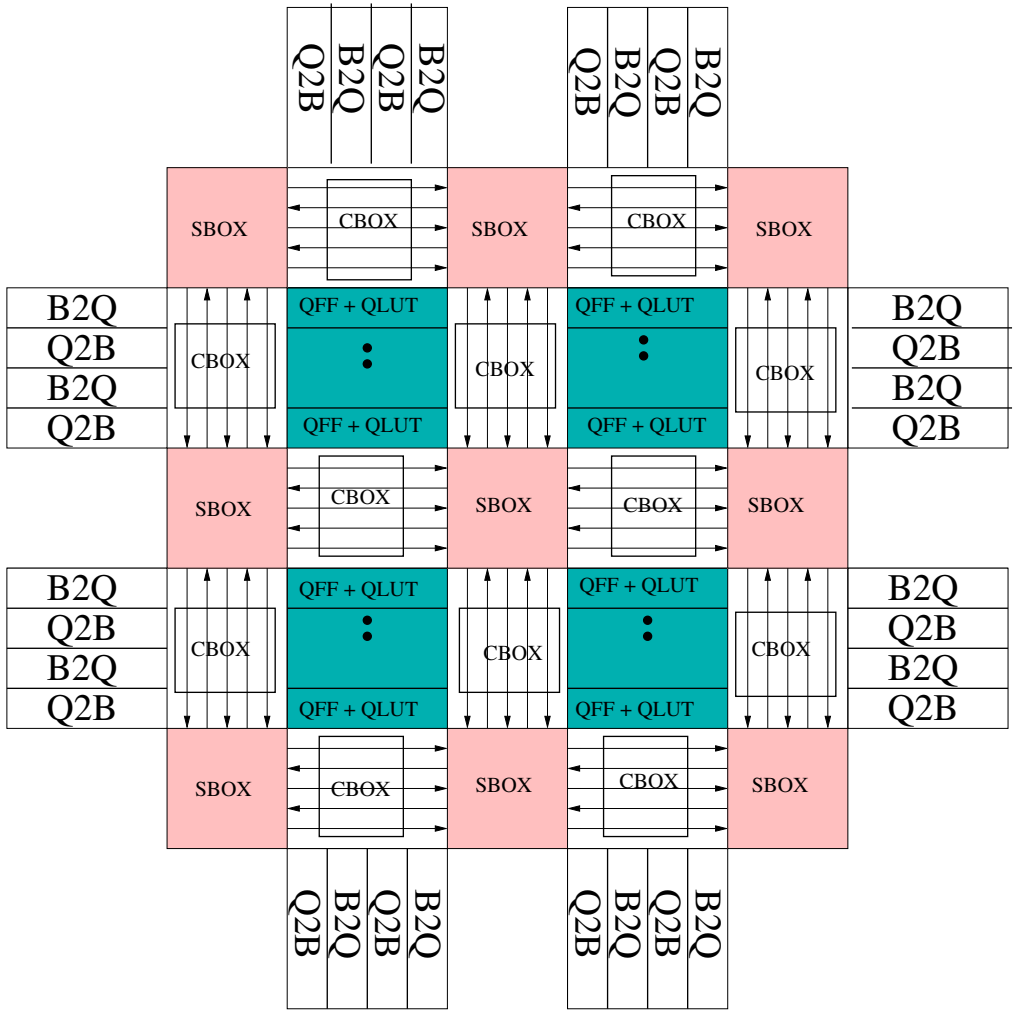


Figure: Overall QFPGA Architecture.

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

Modelling Tool: VPR

- Versatile Place & Route: A well known FPGA modelling and place/route tool.
- VTR: Verilog to Routing
-
  - includes Synthesis and Packing of binary digital circuits.

CLB: Binary Vs. Quaternary

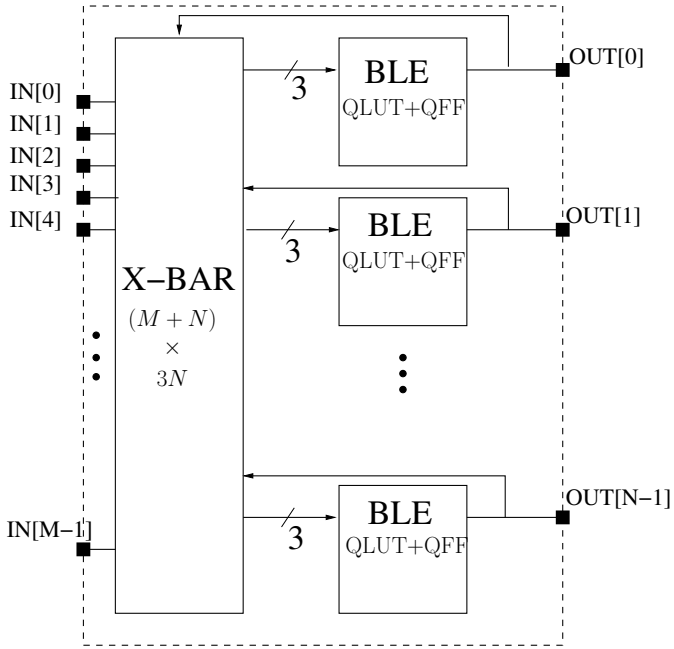


Figure: Configurable Logic Block with M inputs, and N BLEs and the input X-bar

CLB: Binary Vs. Quaternary

Table: VTR Architectural Parameters Used in the Experiments

Configurable Logic Block				
Parameter	Binary		Quaternary	
	Value	Transistor Count	Value	Transistor Count
LUT Input Size	6/(2x5)	548	3	1950
No. Of BLEs	10	5480	10	19500
CLB Outputs	10	-	10	-
CLB Inputs	40	-	30	-
CLB Output Feedback	10	-	10	-
Input XBAR	50x60	6240	30x40	4200
Total	-	11720	-	23700

Routing: Binary Vs. Quaternary

- The size of Switch boxes and Connection Boxes are determined by VPR.
- i.e often bigger the benchmark circuit: requires more routing resources.
- We model the timing and capcitances of binary and quaternary switches.

Routing: Area Model Parameters

Table: Area Model used in the Experiments.

	Binary	Quaternary
ipin_mux_trans_size	x	x
grid_logic_tile_area	11720x	23700x
mux_trans_size	x	x
buf_size	x	3x

Routing: Timing Model Parameters

Table: VTR Timing Model Parameters

Configurable Logic Block			
Block	Parameter	Binary	Quaternary
Input Crossbar	IO Delay Constant	τ ns	τ ns
Mux	IO Delay Constant	τ ns	τ ns
LUT6	Delay Matrix	τ ns	2τ ns
FF	T_Setup	-	-
FF	T_CLOCK_TO_Q	-	-
Routing Resources			
Connection Box	C_ipin_cblock	C pf	C pf
Connection Box	T_ipin_cblock	τ ns	2τ ns
SwitchBox/Switchlist	Cin	C pf	$3C$ pf
SwitchBox/Switchlist	Cout	C pf	C pf
SwitchBox/Switchlist	Tdel	τ ns	2τ ns

Experiments: Benchmarks

- One of the major bottleneck for multiple-valued logic is absence of synthesis tools.
- We handcrafted arithmetic benchmarks.
- Structural Benchmarks like *Ripple Carry Adders* and *Array Multipliers*
- It is also possible to use random benchmarks.

Experiments: Binary Vs. Quaternary

Bench-marks	Binary FPGA						
	BLEs	Size	Chan	Area			Timing
				Routing	Logic	Total	
Adder32	64	2x2	68	42773.5	46880	8.97e4	11.4712
Adder64	128	3x3	130	138054	105480	2.43e5	22.7411
Mult32	2048	11x11	50	5.40e5	1.42e6	1.95e6	53.716
Mult64	8192	22x22	58	1.89e6	5.17e6	7.06e6	110.327
Mult128	32768	41x41	80	9.40e6	1.97e7	2.92e7	303.546

Bench-marks	Quaternary FPGA								
	BLEs	Size	Chan	Area				Timing	
				Routing	Logic	Total	%Diff.		% Diff
Adder32	32	2x2	44	27286	94800	1.22e5	**+36%**	11.49	**0.0%**
Adder64	64	3x3	60	71336.4	213300	2.84e5	**+16.9%**	22.4717	**-0.02%**
Mult32	512	8x8	38	2.16e5	1.52e6	1.73e6	**-12%**	48.75	**-10.0%**
Mult64	2048	15x15	42	6.83e5	5.33e6	6.01e6	**-15%**	103.521	**-6.6%**
Mult128	8192	29x29	50	2.87e6	1.99e7	2.28e7	**-22%**	240.726	**-21%**

Summary

Why?

- Quternary is near optimum in terms of number representation.
- Reduces interconnect overhead by a factor of 2.
- Good for both computation and communication.

Summary of Experiments:

- 15% reduction in transistor area.
- 10% reduction in critical path delay.
- 2x reduction in wire routing area.

Future Work

- Development of a MVL synthesis tool.
- Prototype.

References

R. Cunha, H. Boudinov, and L. Carro.
Quaternary look-up tables using voltage-mode cmos logic design.
In *Multiple-Valued Logic, 2007. ISMVL 2007. 37th International Symposium on*, pages 56–56, May 2007.

B. Pelloux-Prayer, A. Valentian, B. Giraud, Y. Thonnart, J. P. Noel, P. Flatresse, and E. Beigné.
Fine grain multi-vt co-integration methodology in utbb fd-soi technology.
pages 168–173, Oct 2013.

Li Shang, Alireza S. Kaviani, and Kusuma Bathala.
Dynamic power consumption in virtex™-ii fpga family.

In *Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-programmable Gate Arrays*, FPGA

Plan

Motivation

Multiple Valued Logic

FDSOI: Key Features

Architecture

Primitives for Multi-Valued Logic

Experiments

Questions ?

Questions ?