Introduction to Information Science and Engineering

Step2. New principles of emerging power efficient computing platforms

http://archlab.naist.jp/Lectures/ARCH/ca04_2001_0502/ca042005e.pdf

Copyright © 2024 NAIST Y.Nakashima

Download the template and submit through UNIPA.

http://archlab.naist.jp/Lectures/ARCH/ca04_2001_0502/ca042005e.docx

in http://archlab.naist.jp/Lectures

Goal of today

Q7. Which is correct formula representing practical energy-efficiency of computers?
実用的省エネコンピュータの正しい評価式はどれ？
A. Watt x time、B. Watt x time^2 (square)、C. Watt x time^3 (cube)
A. Watt数 x 時間、B. Watt数 x時間の2乗、C. Watt数 x 時間の3乗

Q8. What is the accuracy of π as 32-bit data?
コンピュータが32bitで表現できる円周率の精度は?

Q9. Can a computer calculate 5 billion + 1 correctly?
コンピュータは,50億+1を正しく計算できるか?

Q10. CPU/GPU is best for AI and BC?
CPU/GPUはAIやBCに最適?

Sustainable Power Consumption

*Ref. LCS/JST https://www.jst.go.jp/lcs/pdf/fy2021-pp-01.pdf	Basic	AI	Total
*Consumed energy (global)			67,000TWh (2019)
Training a robot for the Rubik's-cube x 1000		2.8TWh	
*Consumed energy in servers (global)	90TWh	15TWh	105TWh (2018)
*Consumed energy in servers (est.) modest	190TWh	320TWh	510TWh (2030)
*Consumed energy in servers (est.) as is	450TWh	1,200TWh	1,700TWh (2030)
*Consumed energy in servers (est.) modest	3,400TWh	7,200TWh	11,000TWh (2050)
*Consumed energy in servers (est.) as is	53,000TWh	221,000TWh	274,000TWh (2050)
**Ref. AIST https://unit.aist.go.jp/rpd-envene/PV/ja/about_pv/e_source/esource_2.htm			Total
Current Solar power available for Humans			679TWh (2019)
? Realistic Solar power available in 2050		10TW (0.01%) *24*365	? 87,600TWh (2050)
**Max Solar power available for Humans		1,000TW (1%) *24*365	8,760,000TWh (20xx)
**Max Solar power on the surface of the Earth		100,000TW (100%)	

Metrics of computers

Smaller number is better.

Power (watt)	Delay (hour)	PDP	EDP	EDDP
100	2	**200**	400	800
200	1.2	240	**288**	346
300	1	300	300	**300**

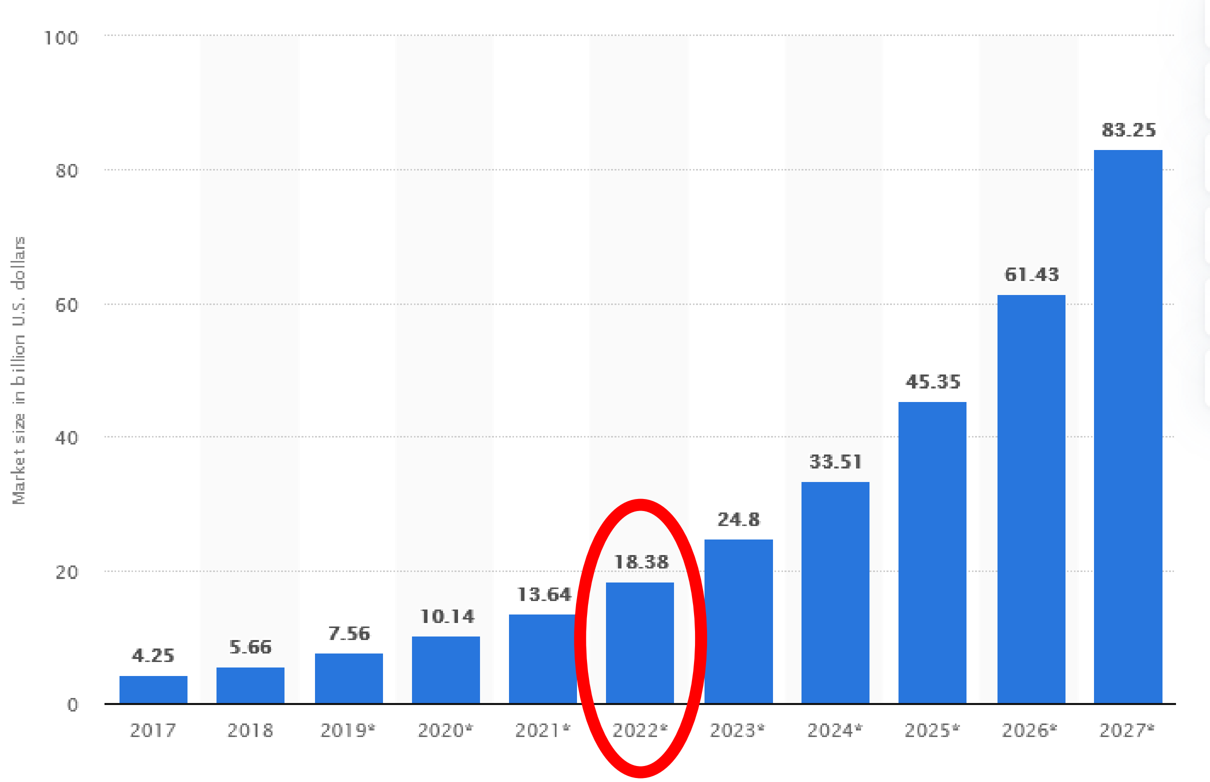
Price=¥1k/job	Price=¥2k/job	Price=¥4k/job
Income=¥12k Cost=¥10k	Income=¥24k Cost=¥10k	Income=¥48k Cost=¥10k
Income=¥20k Cost=¥20k	Income=¥40k Cost=¥20k	Income=¥80k Cost=¥20k
Income=¥24k Cost=¥30k	Income=¥48k Cost=¥30k	Income=¥96k Cost=¥30k

When cost is important, PDP is good.

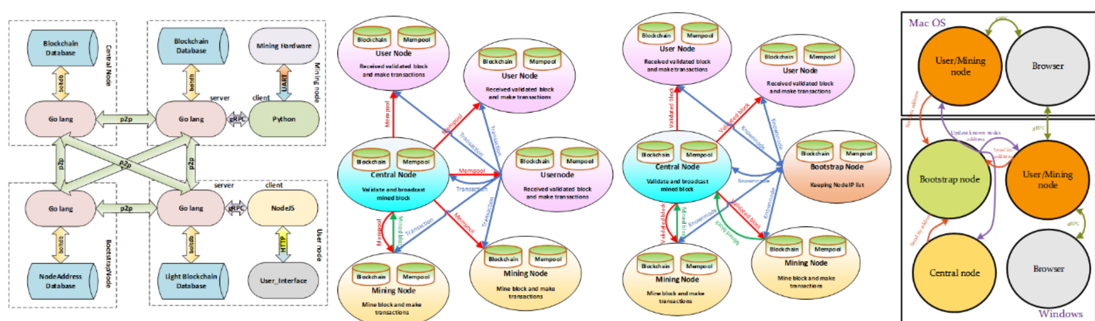
When speed is important, EDP or EDDP is good.

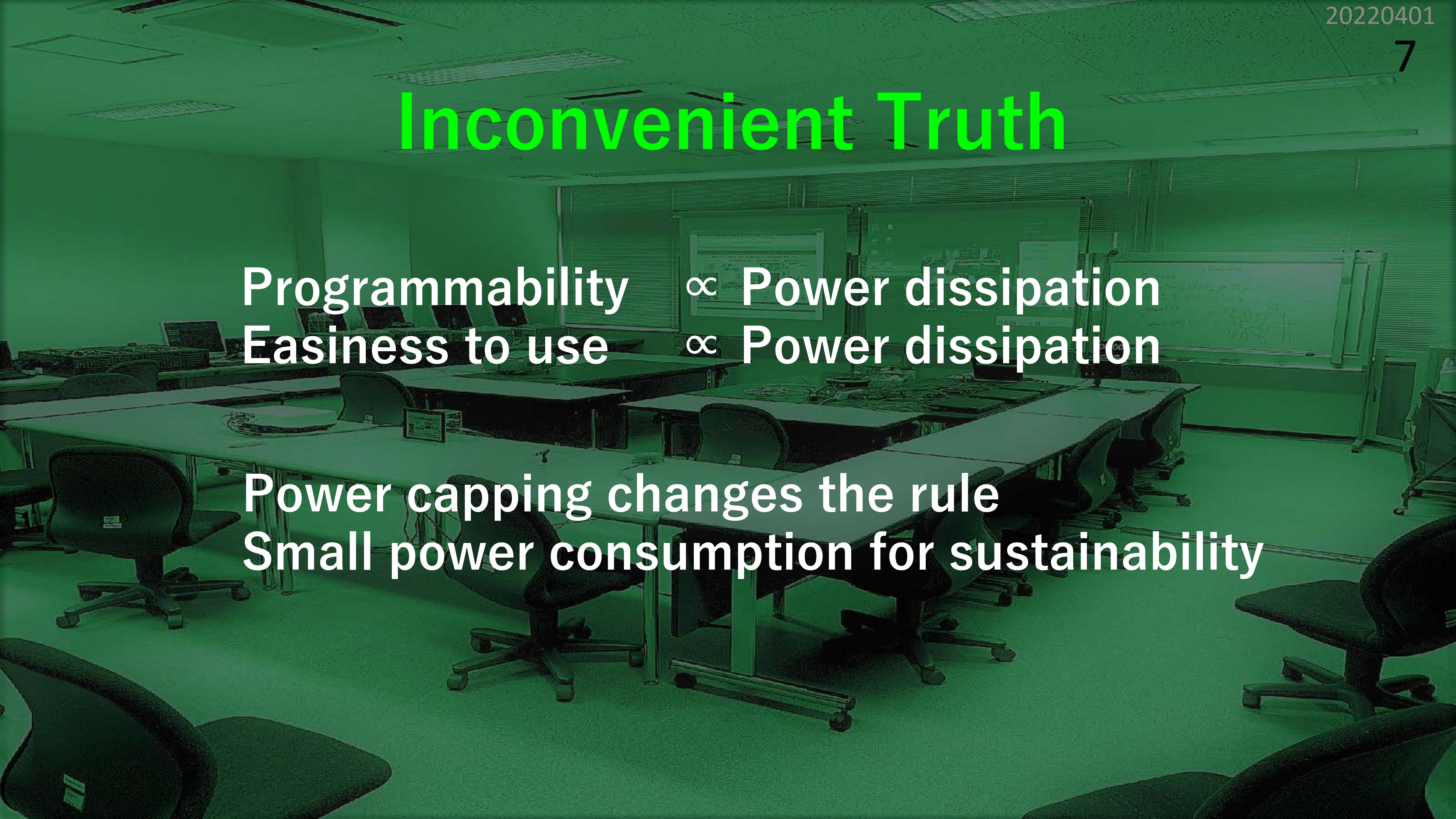
AI and BC are power eaters

This statistic shows the revenue of the global artificial Intelligence chip market from 2017 to 2027. In 2020, the global artificial Intelligence chip market revenue is forecast to grow to 10.14 billion U.S. dollars.

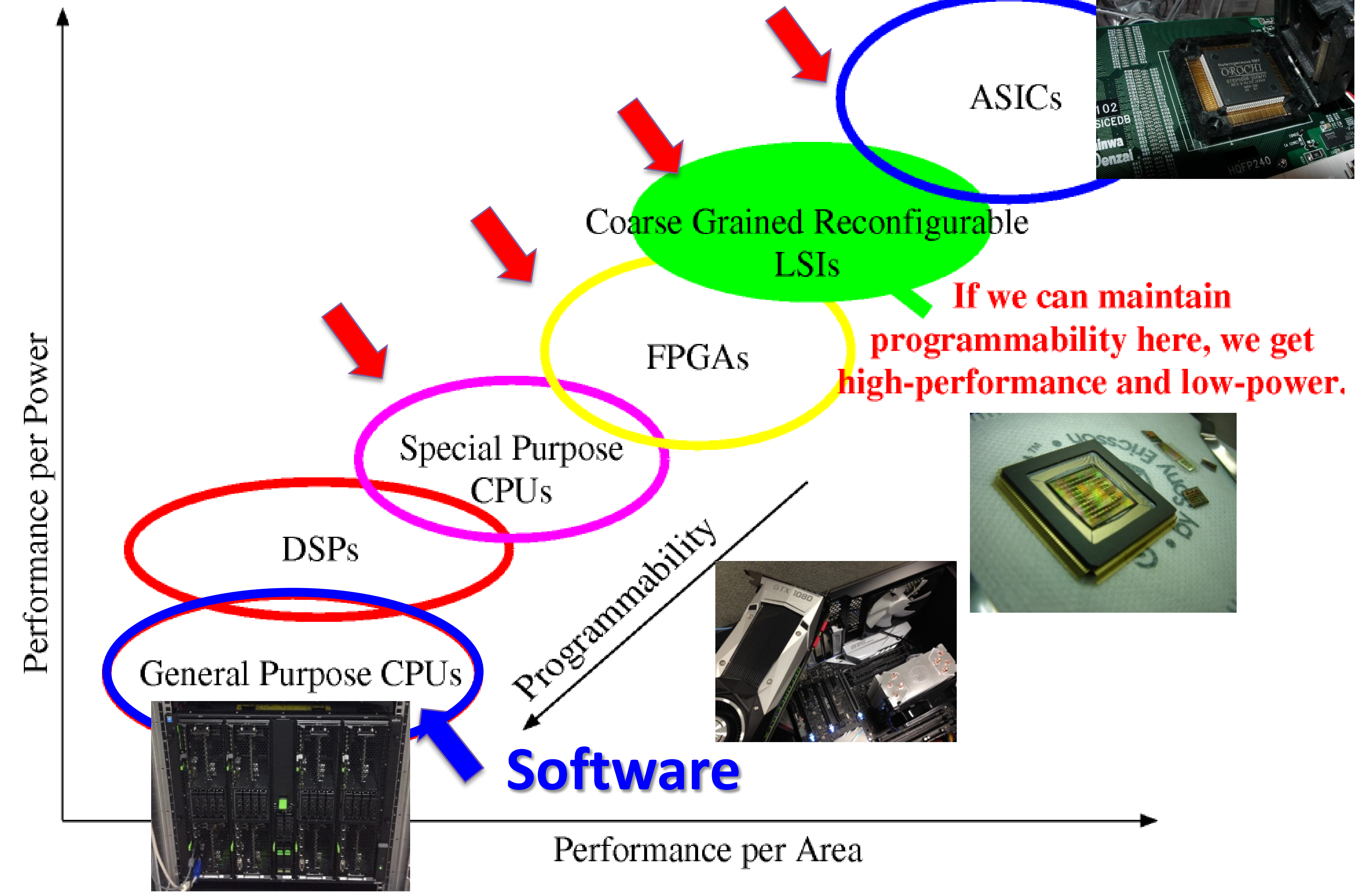


出典：Global artificial intelligence (AI) chip market revenue 2017–2027 ,Published by Thomas Alsop, Nov 30, 2020

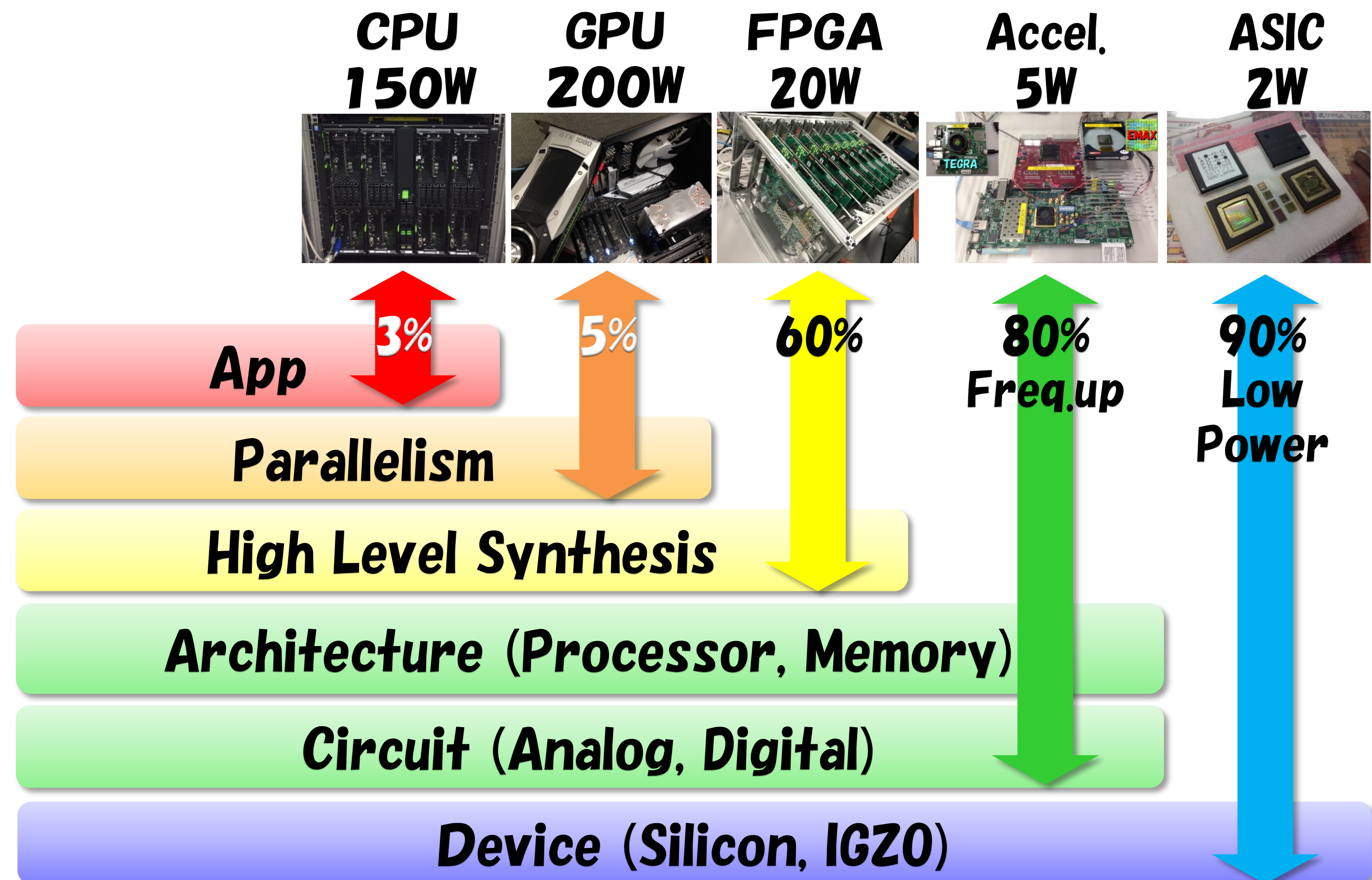



Inconvenient Truth
Programmability ∝ Power dissipation
Easiness to use ∝ Power dissipation
Power capping changes the rule
Small power consumption for sustainability

Choices in current computing platforms

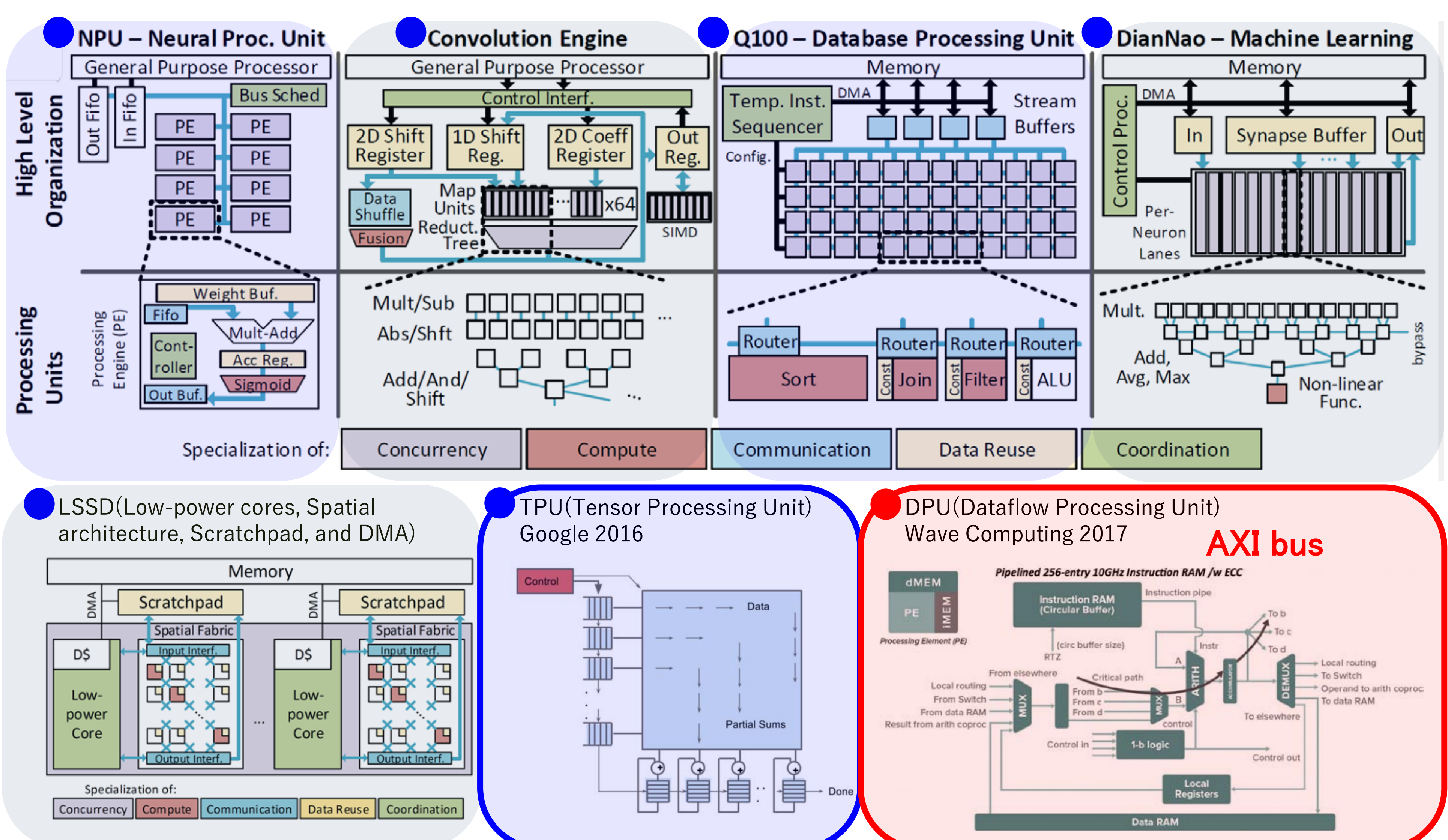


Skills for acceleration



Systolic type DSA(Domain Specific Architecture) is emerging

- 2nd generation: ALU + external memory … wide memory bus
- 3rd generation: Network of near-memory ALUs … narrow memory bus



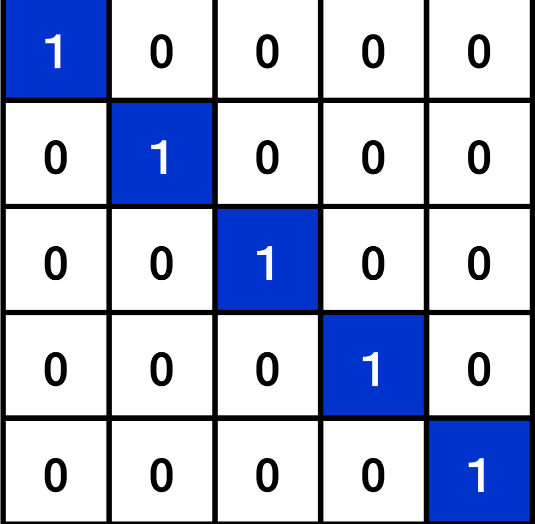
Keywords are Architecture and Moderate Hardware

Semiconductor stops speeding up. Hardware never rescue heavy software.

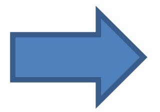
Architects connect both into a system.

Example: How to detect a 135° slit in a image

Input image



1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

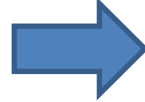


Multiply with weights

1	−1	−1	−1	−1
−1	1	−1	−1	−1
−1	−1	1	−1	−1
−1	−1	−1	1	−1
−1	−1	−1	−1	1

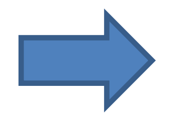
Input image

0	0	0	0	0
0	1	0	1	0
1	0	1	0	1
0	0	0	0	0
0	0	0	0	0



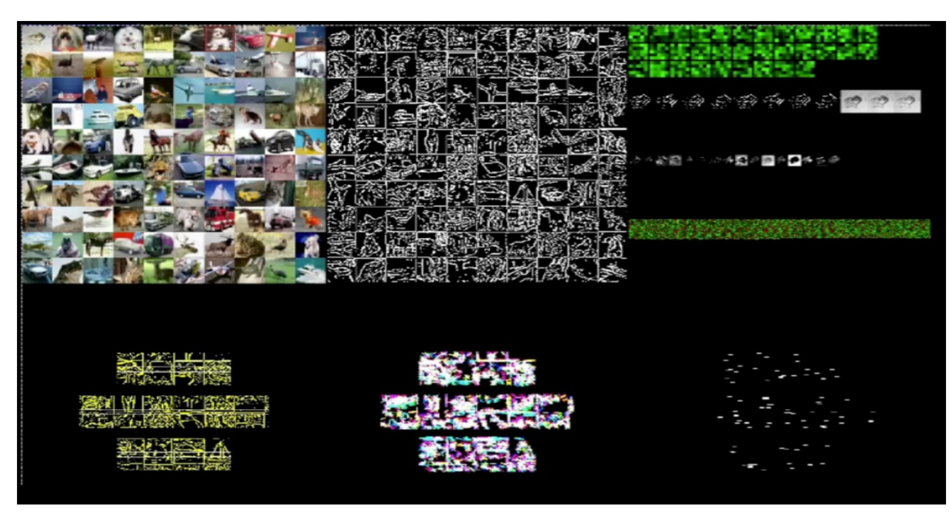
Multiply with weights

1	−1	−1	−1	−1
−1	1	−1	−1	−1
−1	−1	1	−1	−1
−1	−1	−1	1	−1
−1	−1	−1	−1	1

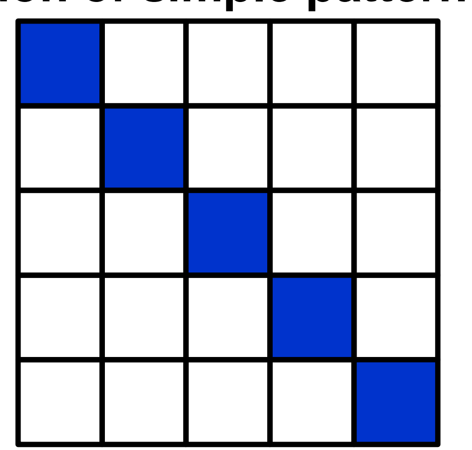


-1

Example: Many Implementations for Pattern Matching



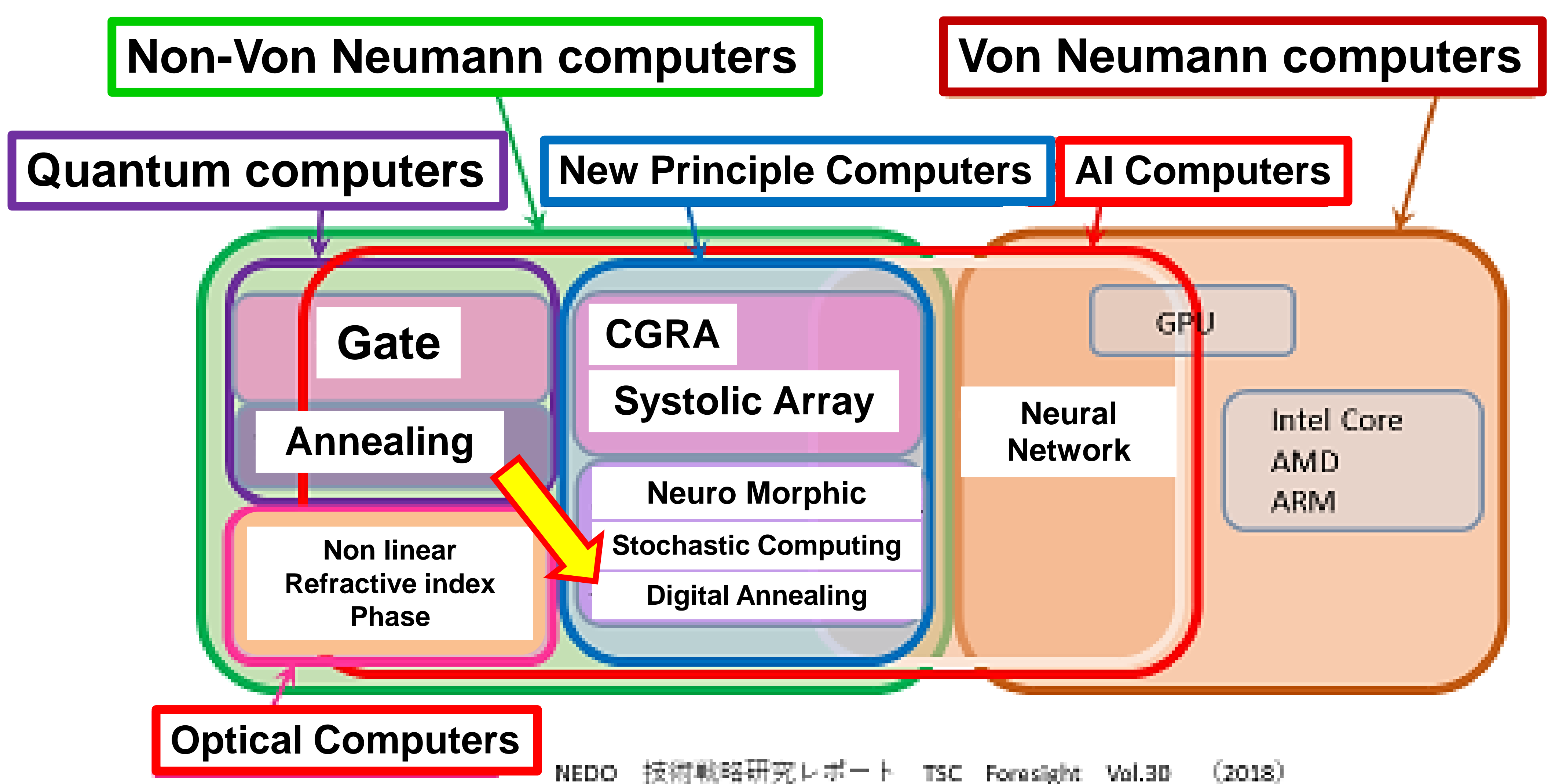
Combination of simple pattern matching



1. Special purpose circuit
2. Mem Capacitor
3. Stochastic Logic
4. Ternary CAM
5. CPU/GPU FPU Matrix Multiplication
6. CGRA FPU Convolution

#. Quantum gate, Quantum annealing, Optical Molecule, Bio, Spintronics

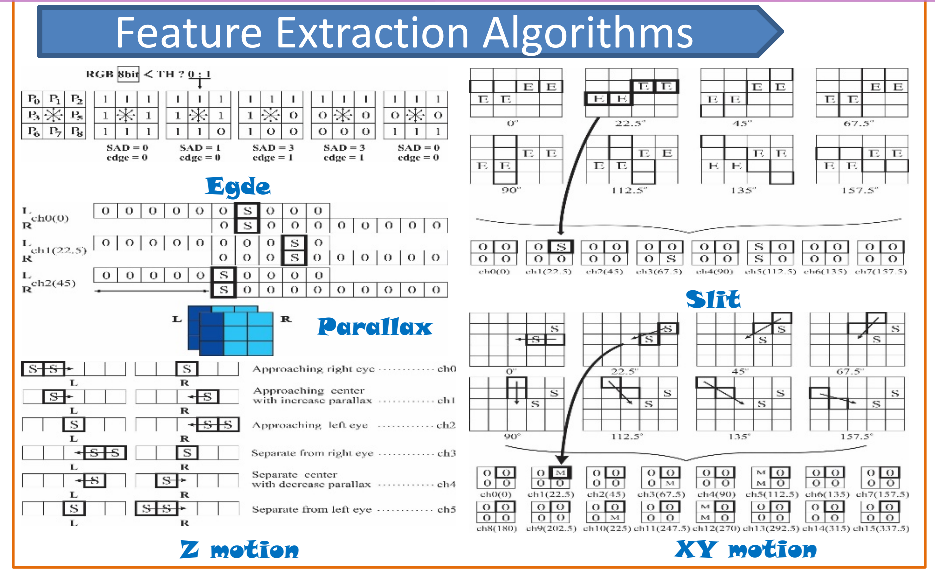
Next generation computing platforms

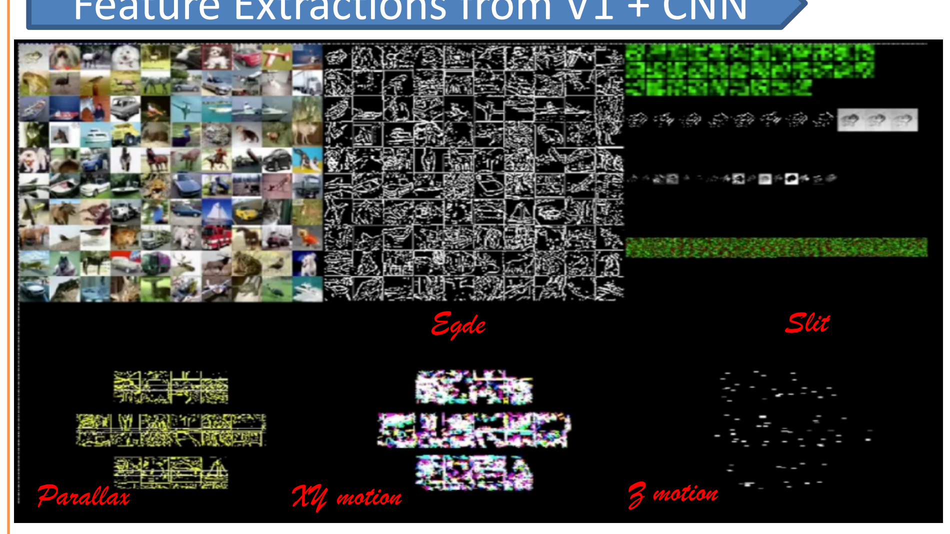


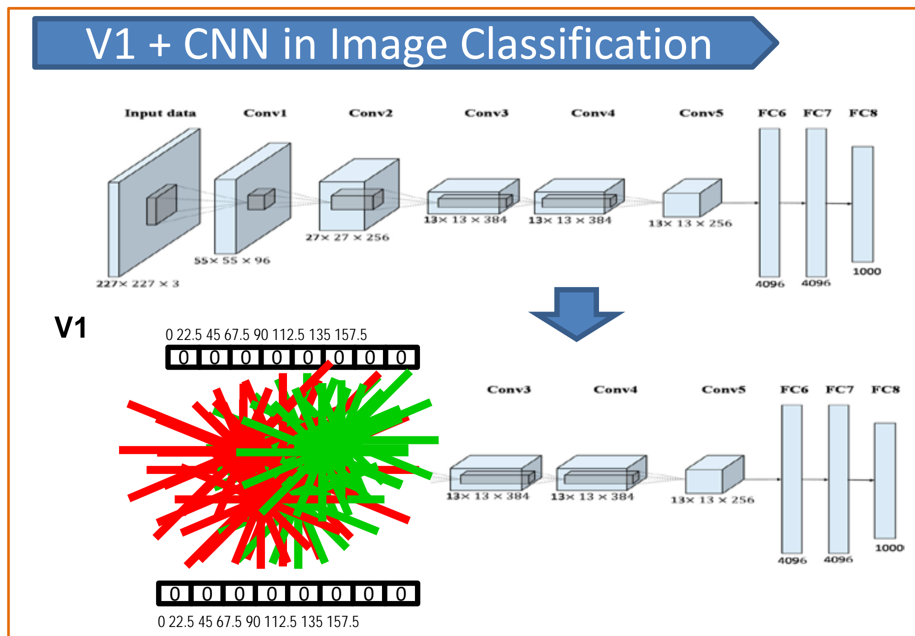
1. Special purpose circuit

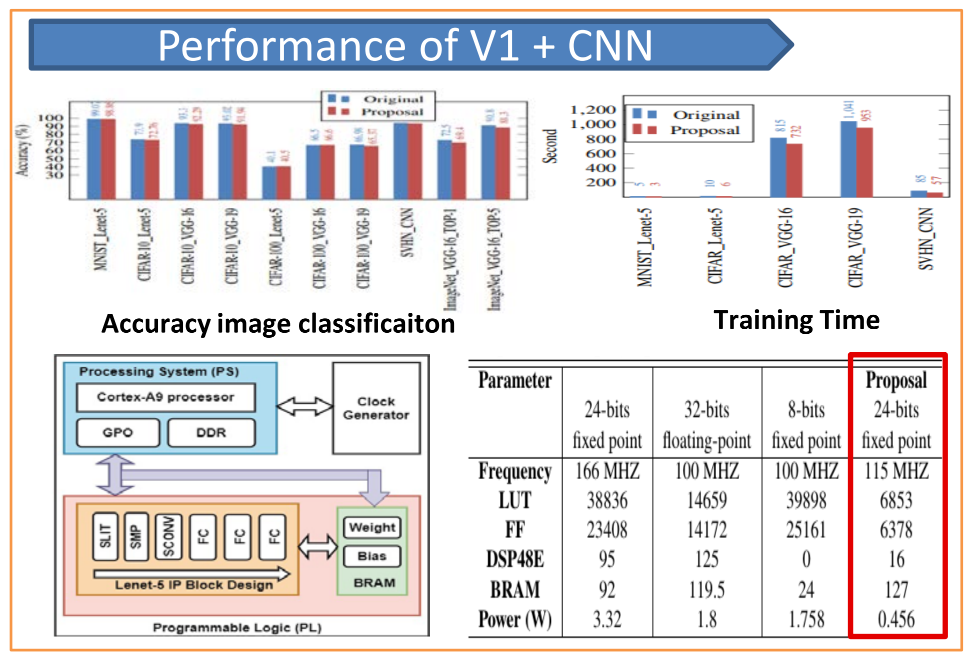
A hardware similar to Visual cortex (V1)

1. Primary visual cortex inspired feature extraction hardware





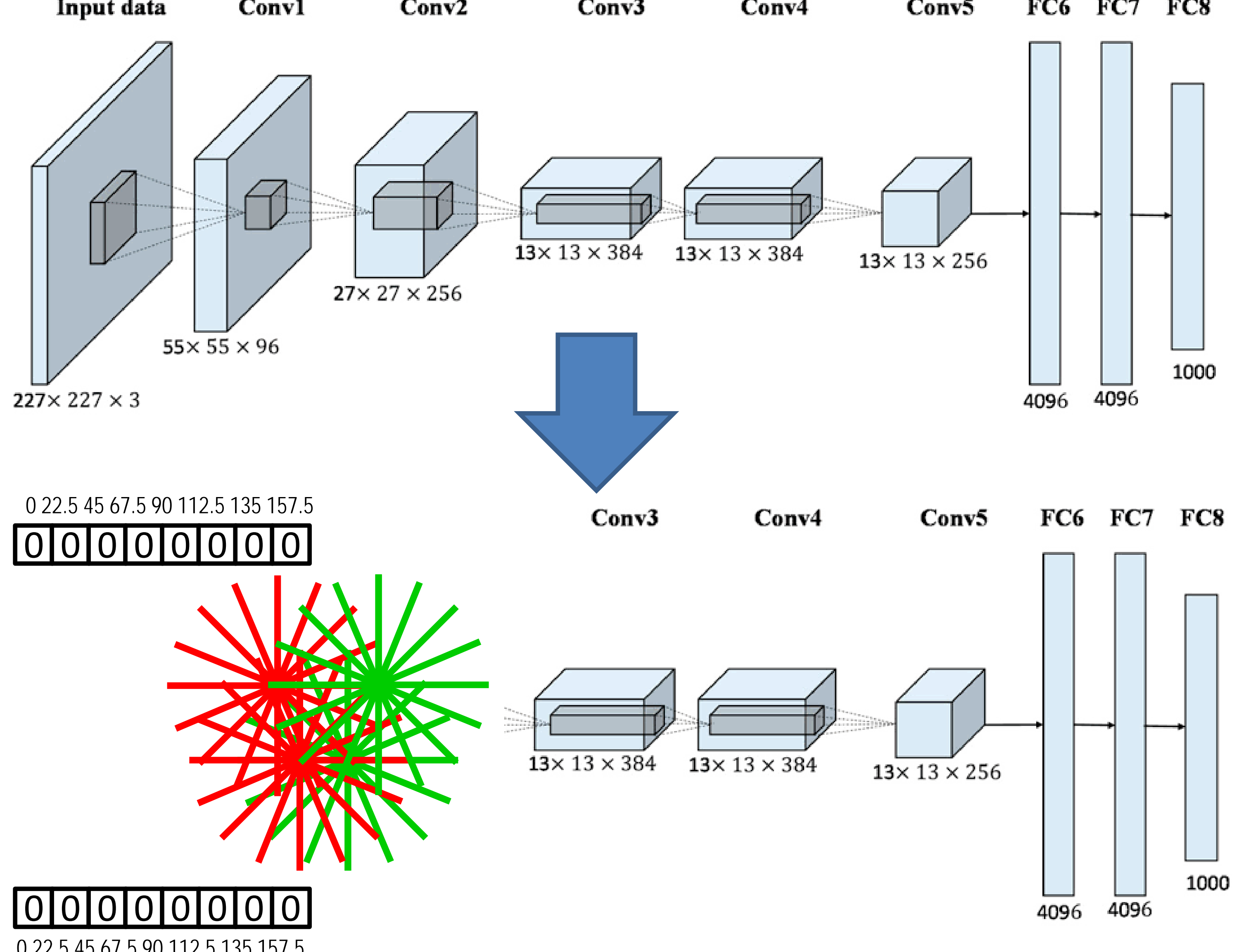




Parameter	24-bits fixed point	32-bits floating-point	8-bits fixed point	Proposal 24-bits fixed point
Frequency	166 MHZ	100 MHZ	100 MHZ	115 MHZ
LUT	38836	14659	39898	6853
FF	23408	14172	25161	6378
DSP48E	95	125	0	16
BRAM	92	119.5	24	127
Power (W)	3.32	1.8	1.758	0.456

【Best Paper Award】Tran Thi Diem, Mutsumi Kimura and Yasuhiko Nakashima: "Primary Visual Cortex Inspired Feature Extraction Hardware Model", SigTelCom2020, Aug. (2020)

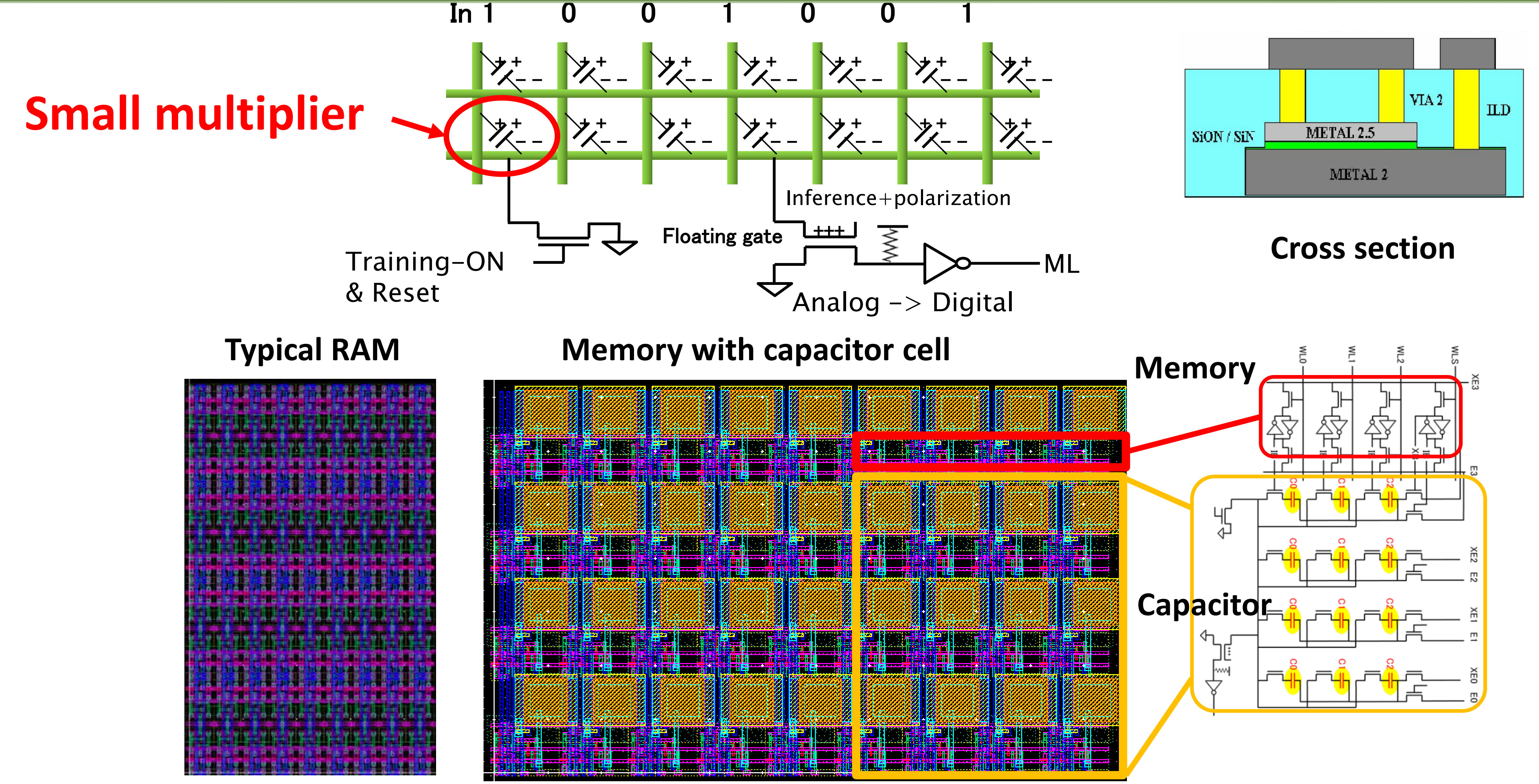
Special hardware for feature extraction can reduce the cost



2. Mem Capacitor

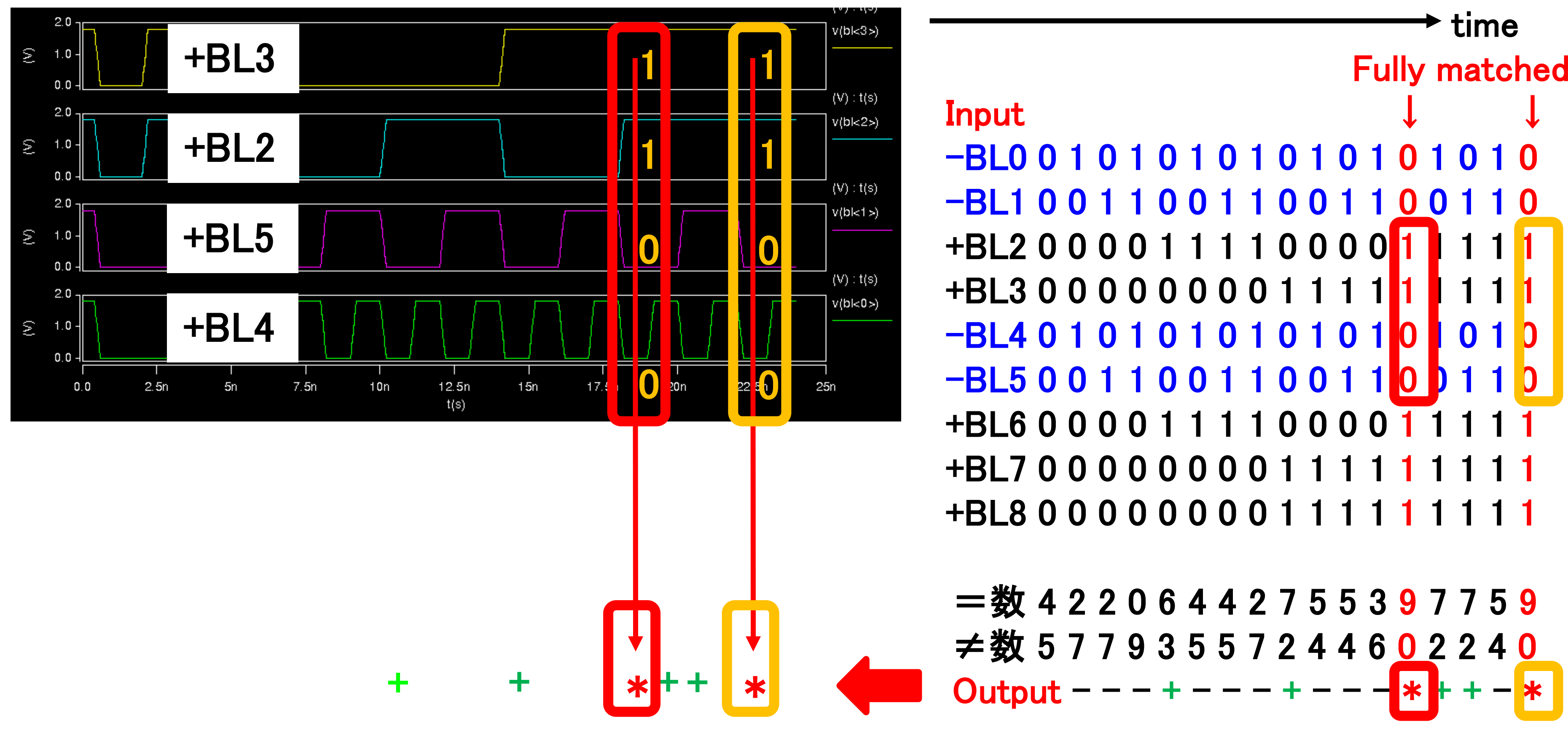
Huge number of small and simple component similar to real neuron and synapse

2. Multiply and add by Memcapacitor



Kimura Mutsumi, Ishisaki Yuma, Miyabe Yuta, Yoshida Homare, Ogawa Isato, Yokoyama Tomoharu, Haga Ken-ichi, Tokumitsu Eisuke, Nakashima Yasuhiko: "Neuromorphic System using Memcapacitors and Autonomous Local Learning", IEEE Transactions on Neural Networks and Learning Systems: Regular Papers, 10.1109/TNNLS.2021.3106566, Sep. (2021)

Tolerant pattern matching



Output is proportional to matched input

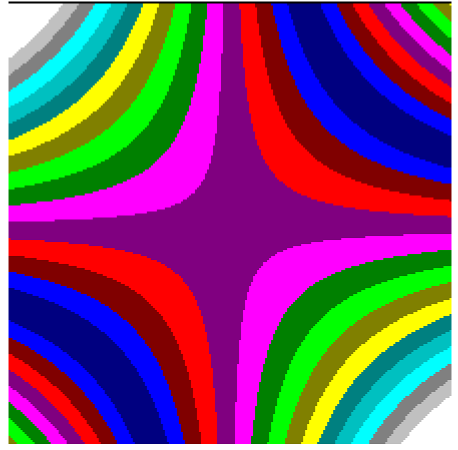
−: mismatch

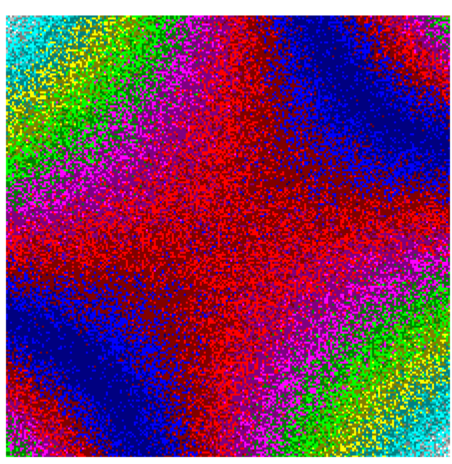
+: partial match

*: match

3. Stochastic Logic

Digital but ultra small multiplier with random numbers

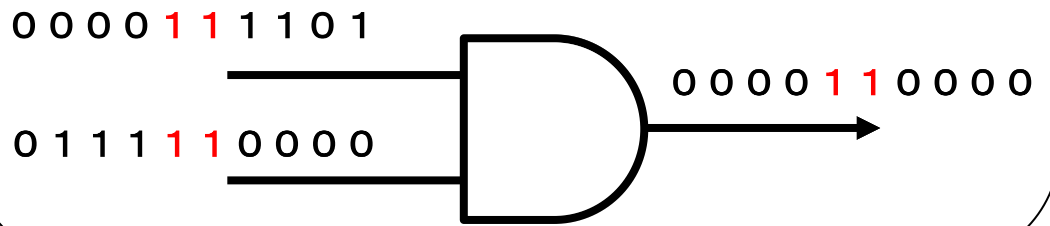




3. Computation with Random Number Generator

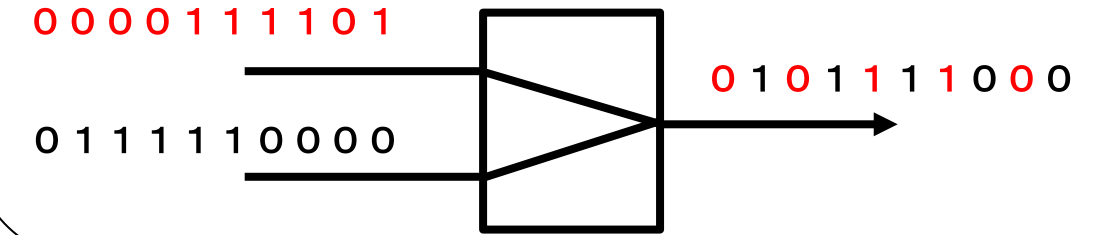
Multiplier by AND gate

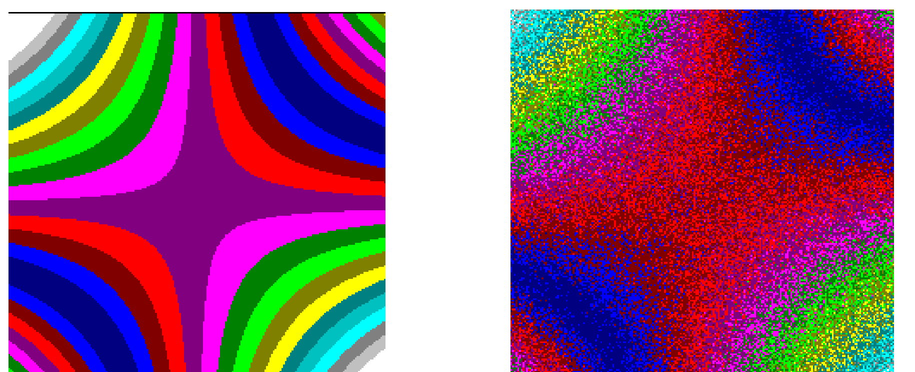
０．５０×０．５０≒０．２０



Scaled adder by random selector

０．５０＋０．５０≒０．５０＊２



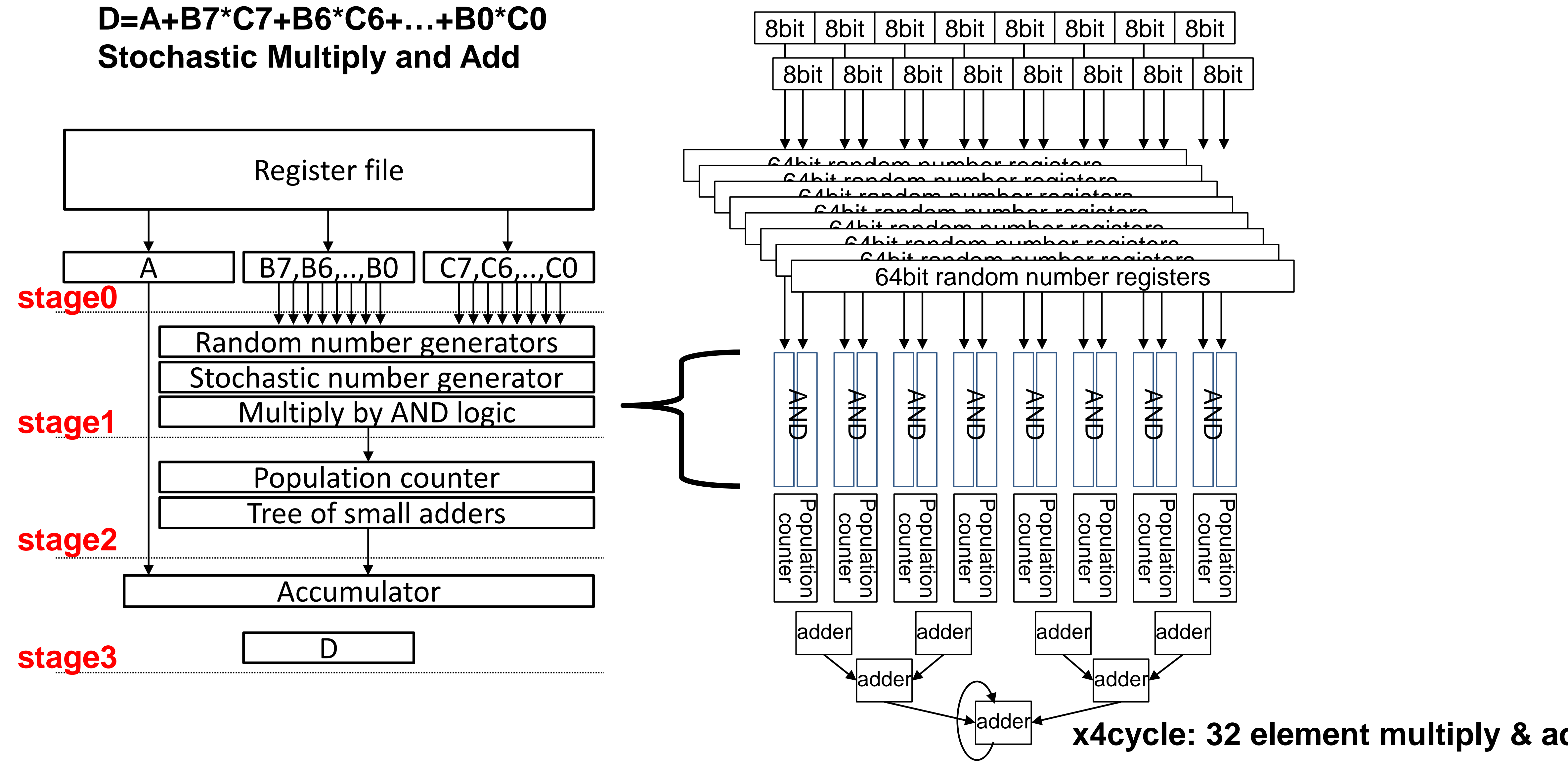


Adder by population counter

０．５０＋０．５０＝１．００

Tati Erlina, Yan Chen, Renyuan Zhang and Yasuhiko Nakashima: "An Efficient Time-based Stochastic Computing Circuitry Employing Neuron-MOS", GLSVLSI2019, pp.51-56, May. (2019)

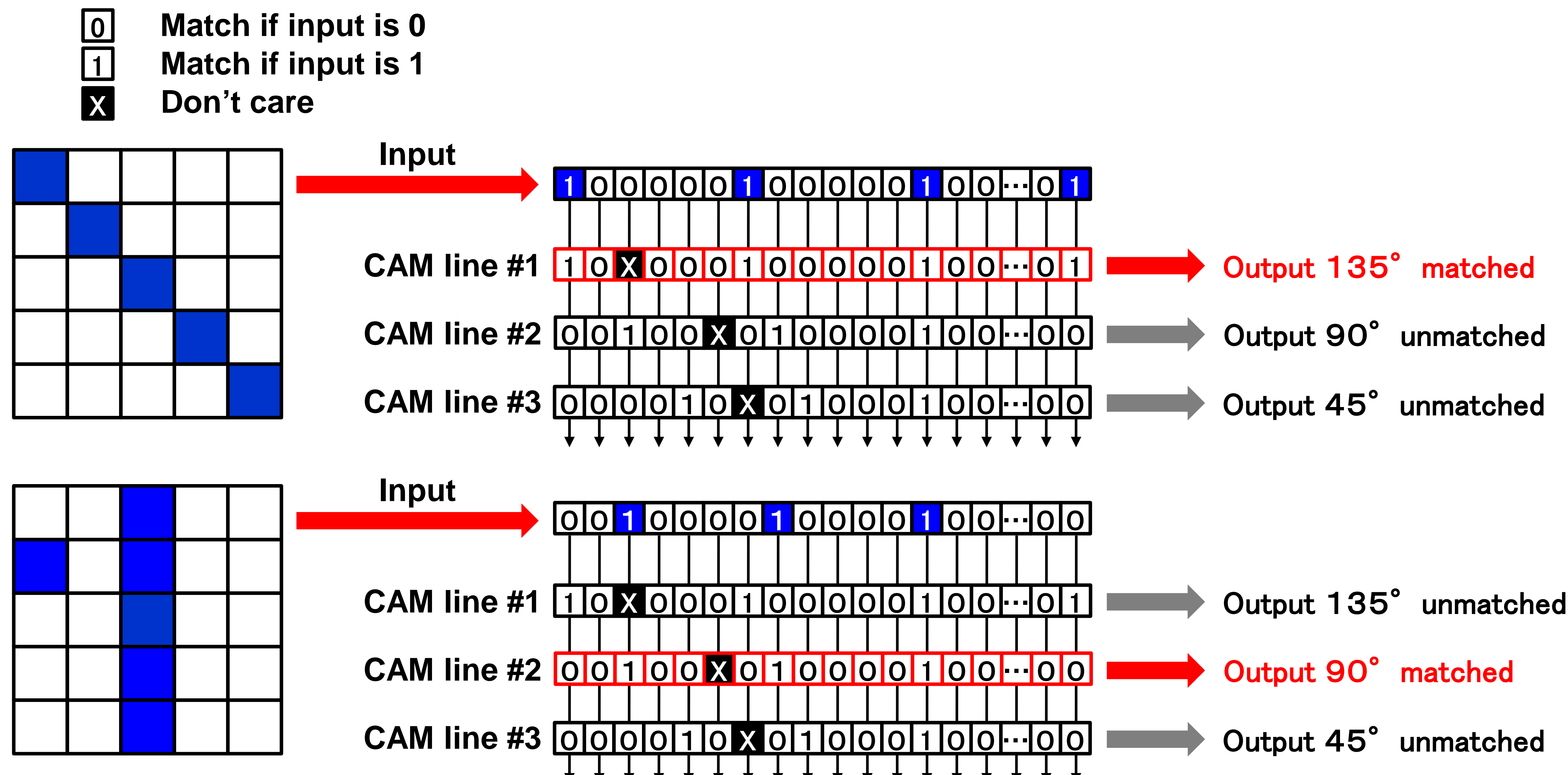
Small and high-speed stochastic FMA is in each unit



4. Ternary CAM

Compare with huge number of predefined patterns

4. Ternary CAM … Content Addressable Memory



RAM and Ternary CAM (Content Addressable Memory)

RAM cell

CAM cell

16bit CAM

CAM block

16bit RAM

5. CPU/GPU FPU Matrix Mult.

CPU is the only platform for programmers who can write only C language.

Goal of today

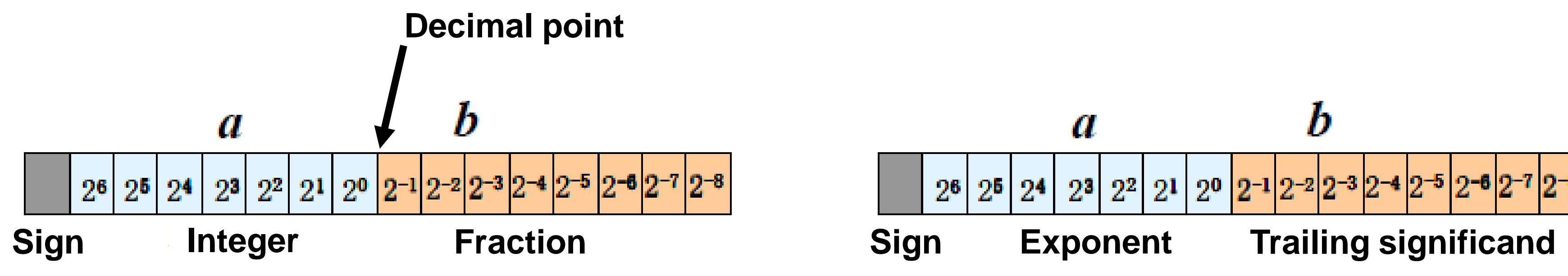
Q7. Which is correct formula representing practical energy-efficiency of computers?
実用的省エネコンピュータの正しい評価式はどれ？
A. Watt x time、B. Watt x time^2 (square)、C. Watt x time^3 (cube)
A. Watt数 x 時間、B. Watt数 x時間の2乗、C. Watt数 x 時間の3乗

Q8. What is the accuracy of π as 32-bit data?
コンピュータが32bitで表現できる円周率の精度は?

Q9. Can a computer calculate 5 billion + 1 correctly?
コンピュータは,50億+1を正しく計算できるか?

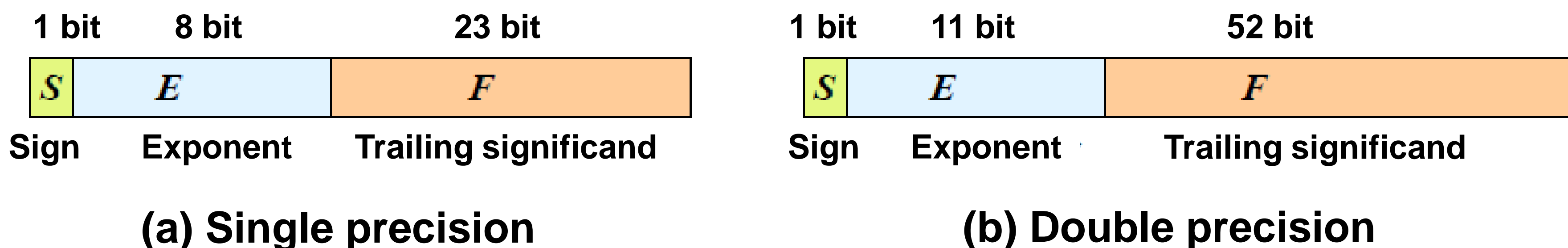
Q10. CPU/GPU is best for AI and BC?
CPU/GPUはAIやBCに最適?

Fixed-point number vs Floating-point number



(a) Fixed-point format: a + b

(b) Floating-point format: b x 2^a



(a) Single precision

(b) Double precision

Single/double precision floating-point format

Single precision "float"

Sign	Exponent	Trailing significand
1 bit	8 bit	23 bit

```
00111111100000000000000000000000
```

Double precision "double"

Sign	Exponent	Trailing significand
1 bit	11 bit	52 bit

```
0011111111110000000000000000000000000000...0
```

- **Subnormal number** … exp=0 and significand≠0
  - (-1)^S * 2^(-126) * 0.significand
  - (-1)^S * 2^(-1022) * 0.significand
- **Normal number** … 0<exp<255 (-1)^S * 2^(exp-127) * 1.significand
  - … 0<exp<2047 (-1)^S * 2^(exp-1023) * 1.significand
- **Signed zero** … exp=0 and significand=0
- **Signed infinite** … exp=255/2047 and significand=0
- **Not a number** … exp=255/2047 and significand ≠0
  - ex. result of 0/0

Accuracy of single/double precision (π)

PI = 3.14159265358979323846

- **Nearest single precision is 3.141592**

```
0/1 00000000 00000000000000000000000··· +/-0
0/1 01111100 00000000000000000000000··· +/-0.125
0/1 01111101 00000000000000000000000··· +/-0.25
0/1 01111110 00000000000000000000000··· +/-0.5
0/1 01111111 00000000000000000000000··· +/-1
0/1 10000000 00000000000000000000000··· +/-2
0/1 10000000 10000000000000000000000··· +/-3
0/1 10000000 10010010000111111011010··· +/-3.14159250
0/1 10000000 10010010000111111011011··· +/-3.14159274
0/1 10000000 10010010000111111011100··· +/-3.14159297
0/1 10000000 11111111111111111111111··· +/-3.99999976
0/1 10000001 00000000000000000000000··· +/-4
0/1 10000010 00000000000000000000000··· +/-8
0/1 11111111 00000000000000000000000··· +/-∞
0/1 11111111 xxxxxxxxxxxxxxxxxxxxxxx     ··· Not a number
```

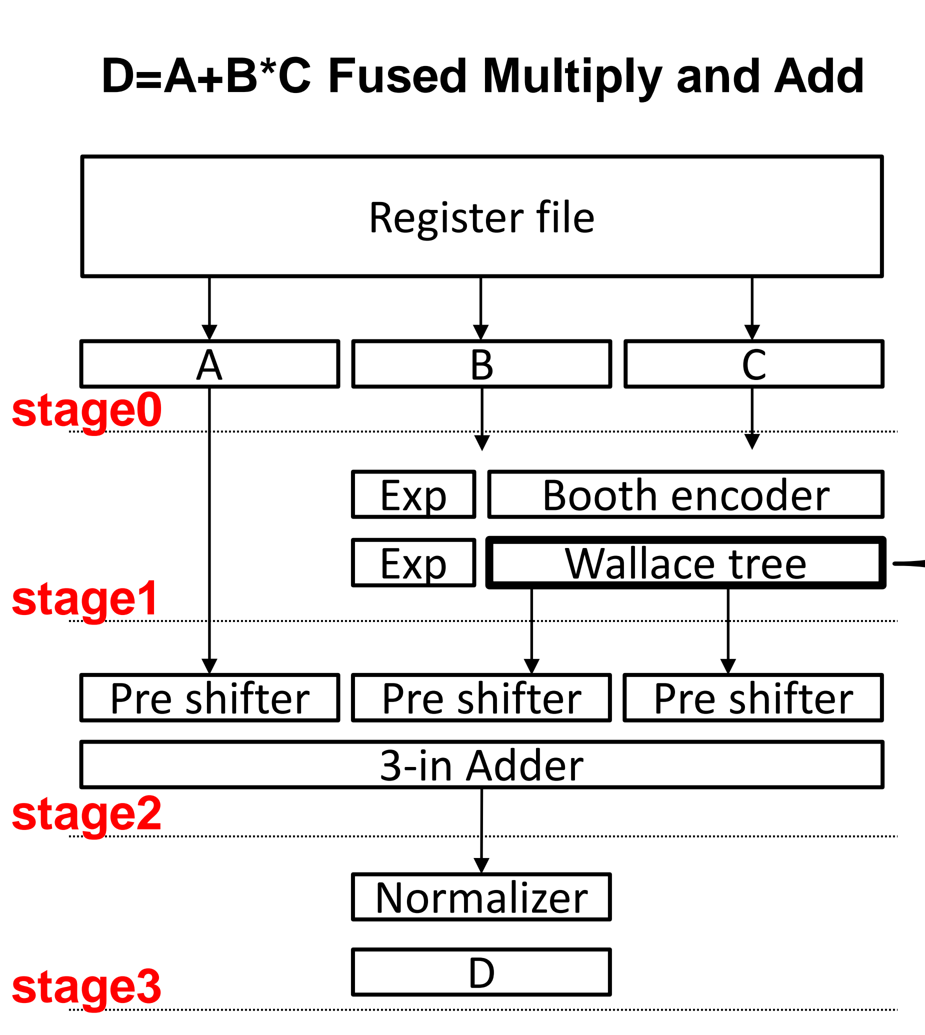
PI = 3.14159265358979323846

- **Nearest double precision is 3.141592653589793**

```
0/1 00000000000 00000000000000000000000...0000 ··· +/-0
0/1 01111111100 00000000000000000000000...0000 ··· +/-0.125
0/1 01111111101 00000000000000000000000...0000 ··· +/-0.25
0/1 01111111110 00000000000000000000000...0000 ··· +/-0.5
0/1 01111111111 00000000000000000000000...0000 ··· +/-1
0/1 10000000000 00000000000000000000000...0000 ··· +/-2
0/1 10000000000 10000000000000000000000...0000 ··· +/-3
0/1 10000000000 10010010000111111011010...0111 ··· +/-3.1415926535897967
0/1 10000000000 10010010000111111011010...1000 ··· +/-3.1415926535897911
0/1 10000000000 10010010000111111011010...1001 ··· +/-3.1415926535897956
0/1 10000000000 11111111111111111111111...1111 ··· +/-3.9999999999999955
0/1 10000000001 00000000000000000000000...0000 ··· +/-4
0/1 10000000010 00000000000000000000000...0000 ··· +/-8
0/1 11111111111 00000000000000000000000...0000 ··· +/-∞
0/1 11111111111 xxxxxxxxxxxxxxxxxxxxxxx...xxxx ··· Not a number
```

- **Rounding error**

Pipelined floating point unit




```
/*   --stage-1 (13in)------------------------------------------------------------------------------------------------------*/
/*   pp[ 0]                                                            ~s  s  s 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   pp[ 1]                                                           1 ~s 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2     s */
/*   pp[ 2]                                                     1 ~s 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4     s       */
/*                                                              |  | HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA FA HA HA *//* HA,24FA,HA,FA,2HA */
/*   S1[0]                                                     30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C1[0]                                                        29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1    */
/*                                                                                                                                                        */
/*   pp[ 3]                                               1 ~s 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6     s             */
/*   pp[ 4]                                         1 ~s 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8     s                   */
/*   pp[ 5]                                   1 ~s 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10     s                         */
/*                                            |  | HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA FA HA HA                   *//* 2HA,23FA,HA,FA,2HA */
/*   S1[1]                                   36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4             */
/*   C1[1]                                      35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7                      */
/*                                                                                                                                                        */
/*   pp[ 6]                             1 ~s 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12     s                               */
/*   pp[ 7]                       1 ~s 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14     s                                     */
/*   pp[ 8]                 1 ~s 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16     s                                           */
/*                          |  | HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA FA HA HA                                     *//* 2HA,23FA,HA,FA,2HA */
/*   S1[2]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10                               */
/*   C1[2]                    41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13                                        */
/*                                                                                                                                                        */
/*   pp[ 9]           1 ~s 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18     s                                                 */
/*   pp[10]     1 ~s 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20     s                                                       */
/*   pp[11] ~s 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22     s                                                             */
/*           | HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA FA HA HA                                                       *//* 2HA,23FA,HA,FA,2HA */
/*   S1[3]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16                                                 */
/*   C1[3]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19                                                          */
/*                                                                                                                                                        */
/*   pp[12] 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24     s                                                                   */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
/*   --stage-2 (9in)-------------------------------------------------------------------------------------------------------*/
/*   S1[0]                                                     30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C1[0]                                                        29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  | */
/*   S1[1]                                   36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5 4*        | */
/*                                            |  |  |  |  |  | HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA  | *//* HA,26FA,3HA */
/*   S2[0]                                   36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C2[0]                                                  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2       */
/*                                                                                                                                                        */
/*   C1[1]                                      35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7                      */
/*   S1[2]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10*  |  |  |                      */
/*   C1[2]                  | 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13           |  |  |                      */
/*                          | HA HA HA HA HA HA FA FA FA FA FA FA fA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA  |  |  |                      *//* 6HA,22FA,3HA */
/*   S2[1]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7                      */
/*   C2[1]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11                                  */
/*                                                                                                                                                        */
/*   S1[3]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16*                                                */
/*   C1[3]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19  |  |  |                                                 */
/*   pp[12] 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24    s*           |  |  |                                                 */
/*          FA FA FA FA FA fA FA FA FA FA FA FA FA FA FA FA FA fA FA FA FA FA FA HA FA HA HA HA  |  |  |                                                 *//* 22FA,HA,FA,3HA */
/*   S2[2]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16                                                 */
/*   C2[2]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20                                                             */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
/*   --stage-3 (6in)-------------------------------------------------------------------------------------------------------*/
/*   S2[0]                                   36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C2[0]                                                  31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  |  | */
/*   S2[1]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7                 |  | */
/*                          |  |  |  |  |  | HA HA HA HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA HA HA  |  | *//* 5HA,25FA,5HA */
/*   S3[0]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C3[0]                                37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3          */
/*                                                                                                                                                        */
/*   C2[1]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11                                  */
/*   S2[2]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16  |  |  |  |  |                                  */
/*   C2[2]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20              |  |  |  |  |                                  */
/*          HA HA HA HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA HA  |  |  |  |  |                                  *//* 5HA,23FA,4HA */
/*   S3[1]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11                                  */
/*   C3[1]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17                                                    */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
/*   --stage-4 (4in)-------------------------------------------------------------------------------------------------------*/
/*   S3[0]                 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C3[0]                                37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  |  |  | */
/*   S3[1]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11                         |  |  | */
/*           |  |  |  |  | HA HA HA HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA HA HA HA HA HA  |  |  | *//* 5HA,27FA,8HA */
/*   S4     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C4                 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4             */
/*                                                                                                                                                        */
/*   C3[1]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17                                                    */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
/*   --stage-5 (3in)-------------------------------------------------------------------------------------------------------*/
/*   S4     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C4                 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  |  |  |  | */
/*   C3[1]  47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17                                      |  |  |  | */
/*          HA HA HA HA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA HA HA HA HA HA HA HA HA HA HA  |  |  |  | *//* 4HA,27FA,13HA */
/*   S5     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C5     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5                */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
/*   --stage-6 (2in+fadd) シフト調整後---------------------------------------------------------------------------------------*/
/*   S5     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C5     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  |  |  |  |  | */
/*   AD     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24                                                           |  |  |  |  | */
/*          FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA FA HA HA HA HA HA HA HA HA HA HA HA HA HA HA HA HA HA HA HA  |  |  |  |  | *//* 24FA,19HA */
/*   S6     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0 */
/*   C6     47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6                   */
/*   ------------------------------------------------------------------------------------------------------------------------------------------------------*/
```

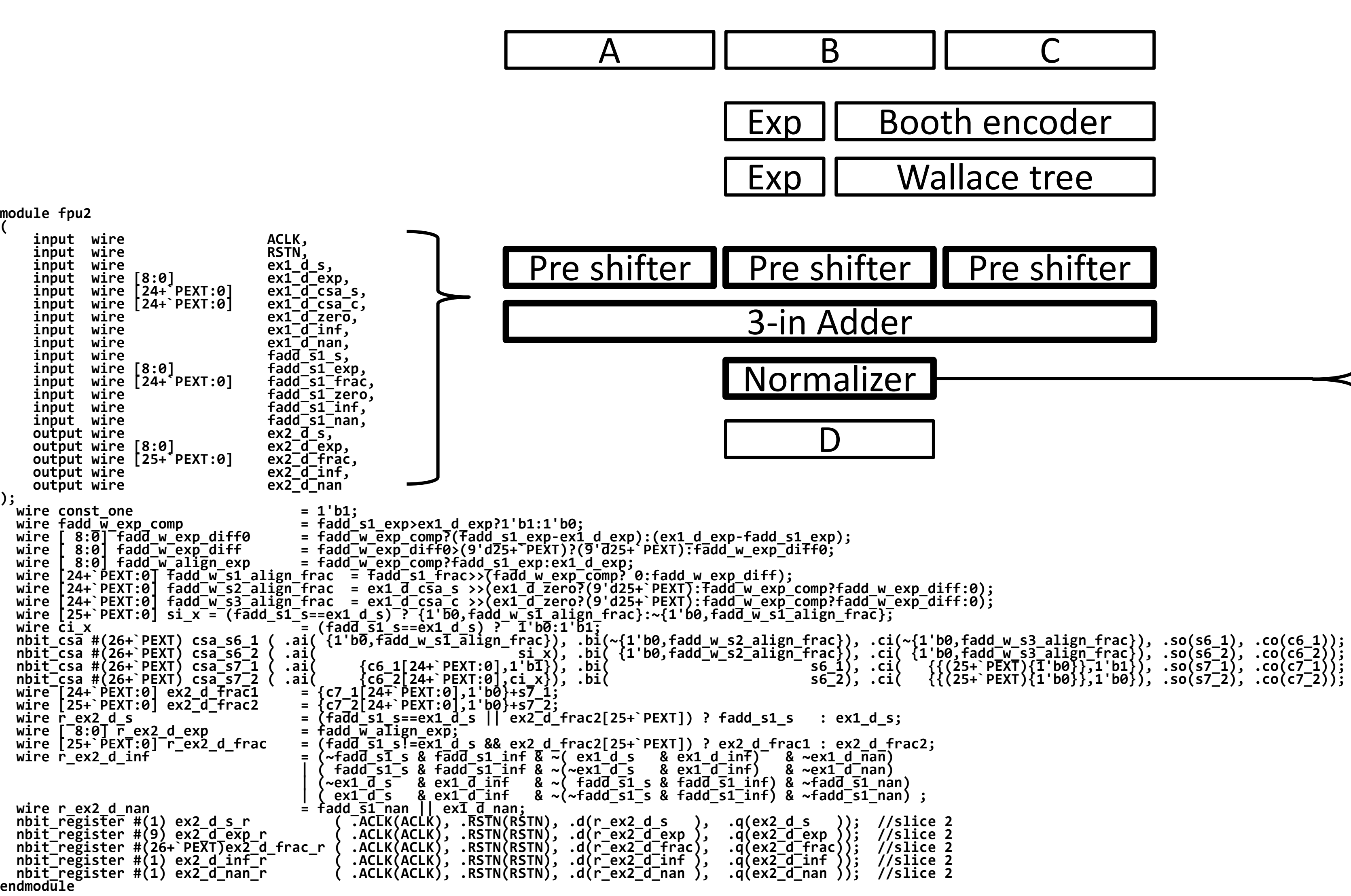
FPU 1st stage

A	B	C
	Exp	Booth encoder
	Exp	Wallace tree

```
module fpu1
(
  input  wire                ACLK,
  input  wire                RSTN,
  input  wire [1:0]          op,
  input  wire [31:0]         ex1,
  input  wire [31:0]         ex2,
  input  wire [31:0]         ex3,
  output wire                ex1_d_s,
  output wire [8:0]          ex1_d_exp,
  output wire [24+`PEXT:0]   ex1_d_csa_s,
  output wire [24+`PEXT:0]   ex1_d_csa_c,
  output wire                ex1_d_zero,
  output wire                ex1_d_inf,
  output wire                ex1_d_nan,
  output wire                fadd_s1_s,
  output wire [8:0]          fadd_s1_exp,
  output wire [24+`PEXT:0]   fadd_s1_frac,
  output wire                fadd_s1_zero,
  output wire                fadd_s1_inf,
  output wire                fadd_s1_nan
);
  wire                const_one       = 1'b1;
  wire                s1_s            = (op==2'd3)? 1'b0:ex1[31];
  wire [7:0]          s1_exp          = (op==2'd3)? 8'd0:ex1[30:23];
  wire [23:0]         s1_frac         = (op==2'd3)? 24'd0:(~(|s1_exp))?{1'b0,ex1[22:0]}:{1'b1,ex1[22:0]};
  wire                s1_zero         = (op==2'd3)? 1'b1: (~(|s1_exp)) & (~(|ex1[22:0]));
  wire                s1_inf          = (op==2'd3)? 1'b0: (  &s1_exp ) & (~(|ex1[22:0]));
  wire                s1_nan          = (op==2'd3)? 1'b0: (  &s1_exp ) & (  |ex1[22:0] );
  wire                s2_s            = (op==2'd1)?~ex2[31]:ex2[31];
  wire [7:0]          s2_exp          = ex2[30:23];
  wire [23:0]         s2_frac         = (~(|s2_exp))?{1'b0,ex2[22:0]}:{1'b1,ex2[22:0]};
  wire                s2_zero         = (~(|s2_exp)) & (~(|ex2[22:0]));
  wire                s2_inf          = (  &s2_exp ) & (~(|ex2[22:0]));
  wire                s2_nan          = (  &s2_exp ) & (  |ex2[22:0] );
  wire                s3_s            = (op==2'd2)? 1'b0   :ex3[31];
  wire [7:0]          s3_exp          = (op==2'd2)? 8'd127 :ex3[30:23];
  wire [23:0]         s3_frac         = (op==2'd2)? 24'h80_0000:(~(|s3_exp))?{1'b0,ex3[22:0]}:{1'b1,ex3[22:0]};
  wire                s3_zero         = (op==2'd2)? 1'b0       :(~(|s3_exp)) & (~(|ex3[22:0]));
  wire                s3_inf          = (op==2'd2)? 1'b0       :(  &s3_exp ) & (~(|ex3[22:0]));
  wire                s3_nan          = (op==2'd2)? 1'b0       :(  &s3_exp ) & (  |ex3[22:0] );
  wire [47:0]         booth_s;
  wire [47:0]         booth_c;
  bit24_booth_wallace bw_impl         (.ai(s2_frac), .bi(s3_frac), .so(booth_s), .co(booth_c));
  wire                r_ex1_d_s       = s2_s ^ s3_s;
  wire [24+`PEXT:0]   r_ex1_d_csa_s   = booth_s[47:23-`PEXT]; /* sum */
  wire [24+`PEXT:0]   r_ex1_d_csa_c   = booth_c[47:23-`PEXT]; /* carry */
  wire                r_ex1_d_zero    = (s2_zero && !s3_inf && !s3_nan) || (s3_zero && !s2_inf && !s2_nan);
  wire [8:0]          r_ex1_d_exp     = {1'b0,s2_exp} + {1'b0,s3_exp} < 9'd127 ? 9'd0 :
                                        {1'b0,s2_exp} + {1'b0,s3_exp} - 9'd127;
  wire                r_ex1_d_inf     = (s2_inf && !s3_zero && !s3_nan) || (s3_inf && !s2_zero && !s2_nan);
  wire                r_ex1_d_nan     = s2_nan || s3_nan || (s2_inf && s3_zero) || (s3_inf && s2_zero);
  nbit_register #(1)  ex1_d_s_r          (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_s     ),  .q(ex1_d_s     ));  //slice 1
  nbit_register #(9)  ex1_d_exp_r        (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_exp   ),  .q(ex1_d_exp   ));  //slice 1
  nbit_register #(25+`PEXT) ex1_d_csa_s_r (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_csa_s),  .q(ex1_d_csa_s ));  //slice 1
  nbit_register #(25+`PEXT) ex1_d_csa_c_r (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_csa_c),  .q(ex1_d_csa_c ));  //slice 1
  nbit_register #(1)  ex1_d_zero_r       (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_zero  ),  .q(ex1_d_zero  ));  //slice 1
  nbit_register #(1)  ex1_d_inf_r        (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_inf   ),  .q(ex1_d_inf   ));  //slice 1
  nbit_register #(1)  ex1_d_nan_r        (.ACLK(ACLK), .RSTN(RSTN), .d(r_ex1_d_nan   ),  .q(ex1_d_nan   ));  //slice 1
  wire                r_fadd_s1_s     = s1_s;
  wire [8:0]          r_fadd_s1_exp   = (8'd0<s1_exp&&s1_exp<8'd255)?{1'b0,(s1_exp-8'd1)}:{1'b0,s1_exp};
  wire [24+`PEXT:0]   r_fadd_s1_frac  = (8'd0<s1_exp&&s1_exp<8'd255)?{s1_frac,{(`PEXT+1){1'b0}}}:{1'b0,s1_frac,{(`PEXT){1'b0}}};
  wire                r_fadd_s1_zero  = s1_zero;
  wire                r_fadd_s1_inf   = s1_inf;
  wire                r_fadd_s1_nan   = s1_nan;
  nbit_register #(1)  fadd_s1_s_r        (.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_s    ),  .q(fadd_s1_s   ));  //slice 1
  nbit_register #(9)  fadd_s1_exp_r      (.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_exp  ),  .q(fadd_s1_exp ));  //slice 1
  nbit_register #(25+`PEXT) fadd_s1_frac_r(.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_frac ),  .q(fadd_s1_frac));  //slice 1
  nbit_register #(1)  fadd_s1_zero_r     (.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_zero ),  .q(fadd_s1_zero));  //slice 1
  nbit_register #(1)  fadd_s1_inf_r      (.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_inf  ),  .q(fadd_s1_inf ));  //slice 1
  nbit_register #(1)  fadd_s1_nan_r      (.ACLK(ACLK), .RSTN(RSTN), .d(r_fadd_s1_nan  ),  .q(fadd_s1_nan ));  //slice 1
endmodule
```

```
module bit24_booth_wallace
(
    input  wire [23:0]   ai,     /* multiplicand */
    input  wire [23:0]   bi,     /* multiplier   */
    output wire [47:0]   so,     /* Partial product */
    output wire [47:0]   co      /* Partial product */
);
// synopsys template

    wire [24:0] sign_ext_ai    = {1'b0,ai};
    wire [24:0] not_ai         = ~sign_ext_ai;
    wire [24:0] shift_ai       = {ai,1'b0};
    wire [24:0] shift_not_ai   = ~shift_ai;

    // Depth:(24+3-1)/2=13
    //      first                            | middle ~ last
    //      28 27 26 25 24 23  ~  2  1  0 |28 27 26 25 24 23 ~  2  1  0
    //WIDTH:N0 ~s  s  s 24 23  ~  2  1  0 | 1 ~s 24 23 22 21 ~  0 N0  S
    wire [28:0]     pp[0:12];
    wire [11:0]     pp_neg;

    genvar gen_idx;
    generate
        for (gen_idx=0; gen_idx < 13; gen_idx=gen_idx+1) begin: IDX
            if(gen_idx==0) begin
                assign  pp[gen_idx] =       (bi[gen_idx+1:0] == 2'b00) ? 29'h0800_0000:
                                            (bi[gen_idx+1:0] == 2'b01) ? {4'b0100,  sign_ext_ai}:
                                            (bi[gen_idx+1:0] == 2'b10) ? {4'b0011, shift_not_ai}:
                                                                         {4'b0011,       not_ai} ;

                assign  pp_neg[gen_idx] =   bi[gen_idx+1];

            end
            else if(gen_idx==12) begin
                assign  pp[gen_idx] =       (bi[gen_idx*2-1] == 1'b0) ? {2'b11, {(25){1'b0}}, 1'b0, pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2-1] == 1'b1) ? {2'b11,  sign_ext_ai, 1'b0, pp_neg[gen_idx-1]}:
                                                                        {2'b11, {(25){1'b0}}, 1'b0, pp_neg[gen_idx-1]} ;
            end
            else begin
                assign  pp[gen_idx] =       (bi[gen_idx*2+1:gen_idx*2-1] == 3'b000) ? {2'b11, {(25){1'b0}}, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b001) ? {2'b11,  sign_ext_ai, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b010) ? {2'b11,  sign_ext_ai, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b011) ? {2'b11,     shift_ai, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b100) ? {2'b10, shift_not_ai, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b101) ? {2'b10,       not_ai, 1'b0,pp_neg[gen_idx-1]}:
                                            (bi[gen_idx*2+1:gen_idx*2-1] == 3'b110) ? {2'b10,       not_ai, 1'b0,pp_neg[gen_idx-1]}:
                                                                                      {2'b10, {(25){1'b1}}, 1'b0,pp_neg[gen_idx-1]} ;

                assign  pp_neg[gen_idx] =   bi[gen_idx*2+1];
            end
        end
    endgenerate

    /*   0st   */
    wire [30:0] s0_1st;
    wire [30:0] c0_1st;
    wire [32:0] s1_1st;
    wire [32:0] c1_1st;
    wire [32:0] s2_1st;
    wire [32:0] c2_1st;
    wire [31:0] s3_1st;
    wire [31:0] c3_1st;
    nbit_csa #(31) csa0_0st( .ai({pp[ 2],2'h0}          ), .bi ({2'h0,pp[ 1]}        ), .ci ({2'h0,pp[ 0]}          ), .so (s0_1st  ), .co (c0_1st  ));
    nbit_csa #(33) csa1_0st( .ai({pp[ 5],4'h0}          ), .bi ({2'h0,pp[ 4], 2'h0}), .ci ({4'h0,pp[ 3]}          ), .so (s1_1st  ), .co (c1_1st  ));
    nbit_csa #(33) csa2_0st( .ai({pp[ 8],4'h0}          ), .bi ({2'h0,pp[ 7], 2'h0}), .ci ({4'h0,pp[ 6]}          ), .so (s2_1st  ), .co (c2_1st  ));
    nbit_csa #(32) csa3_0st( .ai({pp[11][27:0],4'h0}    ), .bi ({1'h0,pp[10], 2'h0}), .ci ({3'h0,pp[ 9]}          ), .so (s3_1st  ), .co (c3_1st  ));

    /*   1st   */
    // pp12
    wire [35:0] s0_2st;
    wire [35:0] c0_2st;
    wire [38:0] s1_2st;
    wire [38:0] c1_2st;
    wire [31:0] s2_2st;
    wire [31:0] c2_2st;
    nbit_csa #(36) csa0_1st( .ai({s1_1st,3'h0}          ), .bi ({5'h00,c0_1st}       ),  .ci ( {6'h00,s0_1st[30:1]} ), .so (s0_2st  ), .co (c0_2st  ));
    nbit_csa #(39) csa1_1st( .ai({c2_1st,6'h00}         ), .bi ({1'h0,s2_1st,5'h00} ),  .ci ( {6'h00,c1_1st}       ), .so (s1_2st  ), .co (c1_2st  ));
    nbit_csa #(32) csa2_1st( .ai({pp[12][25:0],6'h00}   ), .bi ({c3_1st[30:0],1'h0} ),  .ci ( s3_1st               ), .so (s2_2st  ), .co (c2_2st  ));

    /*   2st   */
    wire [41:0] s0_3st;
    wire [41:0] c0_3st;
    wire [41:0] s1_3st;
    wire [41:0] c1_3st;
    nbit_csa #(42) csa0_2st( .ai({s1_2st,3'h0}          ), .bi ({6'h00,c0_2st}       ), .ci ({7'h00,s0_2st[35:1]}    ), .so (s0_3st  ), .co (c0_3st  ));
    nbit_csa #(42) csa1_2st( .ai({c2_2st[30:0],11'h000} ), .bi ({s2_2st,10'h000}     ), .ci ({3'h0,c1_2st}           ), .so (s1_3st  ), .co (c1_3st  ));

    /*   3st   */
    wire [44:0] s0_4st;
    wire [44:0] c0_4st;
    nbit_csa #(45) csa0_3st( .ai({s1_3st,3'h0}          ), .bi ({3'h0,c0_3st}        ), .ci ({4'h0,s0_3st[41:1]}     ), .so (s0_4st  ), .co (c0_4st  ));

    /*   4st   */
    // c1_3st;
    wire [43:0] s0_5st;
    wire [43:0] c0_5st;
    nbit_csa #(44) csa0_4st( .ai({c1_3st[40:0],3'h0}    ), .bi (c0_4st[43:0]         ), .ci (s0_4st[44:1]            ), .so (s0_5st  ), .co (c0_5st  ));

    assign so = {s0_5st,s0_4st[0],s0_3st[0],s0_2st[0],s0_1st[0]};
    assign co = {c0_5st[42:0],5'h0};

endmodule
```

FPU 2nd and 3rd stage




```
module fpu3
(
  input  wire                                ex2_d_s,
  input  wire [ 8:0]                         ex2_d_exp,
  input  wire [25+`PEXT:0]                   ex2_d_frac, //「」「」「」
  input  wire                                ex2_d_inf,
  input  wire                                ex2_d_nan,
  output wire [31:0]                         f
);
  function [31:0] normalizer;
    input                ex1_d_s;
    input [8:0]          ex1_d_exp;
    input [25+`PEXT:0]   ex1_d_frac; //「」「」「」
    input                ex1_d_inf;
    input                ex1_d_nan;

    reg [5:0]            ex2_w_lzc; //「」「」「」
    reg                  ex2_d_s;
    reg [7:0]            ex2_d_exp;
    reg [22:0]           ex2_d_frac;

    begin
      /* normalize */
      ex2_w_lzc = (ex1_d_frac[`PEXT+25]) ? 6'd62:
                                          (ex1_d_frac[`PEXT+24]) ? 6'd63:
              (ex1_d_frac[`PEXT+23]) ? 6'd0: (ex1_d_frac[`PEXT+22]) ? 6'd1:
              (ex1_d_frac[`PEXT+21]) ? 6'd2: (ex1_d_frac[`PEXT+20]) ? 6'd3:
              (ex1_d_frac[`PEXT+19]) ? 6'd4: (ex1_d_frac[`PEXT+18]) ? 6'd5:
              (ex1_d_frac[`PEXT+17]) ? 6'd6: (ex1_d_frac[`PEXT+16]) ? 6'd7:
              (ex1_d_frac[`PEXT+15]) ? 6'd8: (ex1_d_frac[`PEXT+14]) ? 6'd9:
              (ex1_d_frac[`PEXT+13]) ? 6'd10:(ex1_d_frac[`PEXT+12]) ? 6'd11:
              (ex1_d_frac[`PEXT+11]) ? 6'd12:(ex1_d_frac[`PEXT+10]) ? 6'd13:
              (ex1_d_frac[`PEXT+ 9]) ? 6'd14:(ex1_d_frac[`PEXT+ 8]) ? 6'd15:
              (ex1_d_frac[`PEXT+ 7]) ? 6'd16:(ex1_d_frac[`PEXT+ 6]) ? 6'd17:
              (ex1_d_frac[`PEXT+ 5]) ? 6'd18:(ex1_d_frac[`PEXT+ 4]) ? 6'd19:
              (ex1_d_frac[`PEXT+ 3]) ? 6'd20:(ex1_d_frac[`PEXT+ 2]) ? 6'd21:
              (ex1_d_frac[`PEXT+ 1]) ? 6'd22:(ex1_d_frac[`PEXT+ 0]) ? 6'd23:
              (ex1_d_frac[`PEXT- 1]) ?(6'd23+ 1):(6'd24+`PEXT);

      if (ex1_d_nan) begin
        ex2_d_s    = 1'b1;
        ex2_d_frac = 23'h400000;
        ex2_d_exp  = 8'hff;
      end
      else if (ex1_d_inf) begin
        ex2_d_s    = ex1_d_s;
        ex2_d_frac = 23'h000000;
        ex2_d_exp  = 8'hff;
      end
      else if (ex2_w_lzc == 6'd62) begin //「」「」「」
        if (ex1_d_exp >= 253) begin
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = 23'h000000;
          ex2_d_exp  = 8'hff;
        end
        else begin
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = ex1_d_frac>>(2+`PEXT); //「」「」「」
          ex2_d_exp  = ex1_d_exp[7:0] + 8'h2;
        end
      end
      else if (ex2_w_lzc == 6'd63) begin //「」「」「」
        if (ex1_d_exp >= 254) begin
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = 23'h000000;
          ex2_d_exp  = 8'hff;
        end
        else begin
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = ex1_d_frac>>(1+`PEXT); //「」「」「」
          ex2_d_exp  = ex1_d_exp[7:0] + 8'h1;
        end
      end
      else if (ex2_w_lzc <= (6'd23+`PEXT)) begin //「」「」「」
        if (ex1_d_exp >= ex2_w_lzc + 255) begin
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = 23'h000000;
          ex2_d_exp  = 8'hff;
        end
        else if (ex1_d_exp <= ex2_w_lzc) begin /* subnormal num */
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = (ex1_d_frac<<ex1_d_exp)>>`PEXT; //「」「」「」
          ex2_d_exp  = 8'h00;
        end
        else begin /* normalized num */
          ex2_d_s    = ex1_d_s;
          ex2_d_frac = (ex1_d_frac<<ex2_w_lzc)>>`PEXT; //「」「」「」
          ex2_d_exp  = ex1_d_exp - {2'd0,ex2_w_lzc};   //「」「」「」
        end
      end
      else begin /* zero */
        ex2_d_s    = 1'b0;
        ex2_d_frac = 23'h000000;
        ex2_d_exp  = 8'h00;
      end
      normalizer = {ex2_d_s, ex2_d_exp, ex2_d_frac};
    end
  endfunction
  assign f = normalizer( ex2_d_s, ex2_d_exp, ex2_d_frac, ex2_d_inf, ex2_d_nan);
endmodule
```

Goal of today

Q7. Which is correct formula representing practical energy-efficiency of computers?
実用的省エネコンピュータの正しい評価式はどれ？
A. Watt x time、B. Watt x time^2 (square)、C. Watt x time^3 (cube)
A. Watt数 x 時間、B. Watt数 x時間の2乗、C. Watt数 x 時間の3乗

Q8. What is the accuracy of π as 32-bit data?
コンピュータが32bitで表現できる円周率の精度は?

Q9. Can a computer calculate 5 billion + 1 correctly?
コンピュータは,50億+1を正しく計算できるか?

Q10. CPU/GPU is best for AI and BC?
CPU/GPUはAIやBCに最適?

Representable range and Accuracy

- C has no framework to detect integer-overflow
  - INT32_MAX + 1 ⇒ INT32_MIN with no report
  - Σint32 will not represent the population of the world

- Float can represent larger than 10billion, but …
  - int … sign:1bit, integer:31bits
    - 2147479552 + 1 ⇒ 2147479553
  - double … sign:1bit, exp:11bits, significand:52bits
    - 2147479552.0 + 1.0 ⇒ 2147479553.0
  - float … sign:1bit, exp:8bit, significand:23bits
    - 2147479552.0 + 1.0 ⇒ 2147479552.0 (incorrect)

Cancellation of significant digits

- **Loss of accuracy**
- **We know a formula to solve**

$Ax^2 + Bx + C = 0$

$$x1 = \frac{-B + \sqrt{(B^2 - 4AC)}}{2A} \qquad x2 = \frac{-B - \sqrt{(B^2 - 4AC)}}{2A}$$

- **In the case B≫4AC,**

$|B| - \sqrt{(B^2 - 4AC)}$ significantly loses important bits

- **A=1, B=-1000000, C=1**

|-1000000.0| = 1000000.000000000000000 (0x412e848000000000)
√(1000000000000.0 - 4.0)= 999999.999997999984771 (0x412e847fffffbce4)
difference = 0.000002000015229 (0x3ec0c70000000000)

only 24bits remains

Correct program

Avoid $|B| - \sqrt{(B^2 - 4AC)}$

$$x1 = \frac{|B| + \sqrt{(B^2 - 4AC)}}{2A} \quad \text{if } B < 0$$

$$x1 = -\frac{|B| + \sqrt{(B^2 - 4AC)}}{2A} \quad \text{if } B \geqq 0$$

$$x2 = \frac{C}{x1 * A}$$

Formative assessment

- Convert INT32_MAX to floating-point number.

hint:
INT32_MAX is 0x7FFFFFFF (2147483647)

Formative assessment

● **Convert INT32_MAX to floating-point number.**

hint: INT32_MAX is 0x7FFFFFFF (2147483647)

Nearest single precision number:

0x4F000000 (2147483648.0) error=1

1.00000000000000000000000 * 2^31 (158-127)

0x4EFFFFFF (2147483520.0) error=127

1.11111111111111111111111 * 2^30 (157-127)

100000000000000000000000-------- (2147..48)

\- 000000000000000000000001-------- (128)

0x4F000001 (2147483904.0) error=257

1.00000000000000000000001 * 2^31 (158-127)

100000000000000000000000 --------- (2147..48)

\+ 000000000000000000000001--------- (256)

Poor Skill to Write Programs

```
// Convolution
conv_forward(nnet->input,
             nnet->weight,
             nnet->kernel_size,
             nnet->output);

// C=alpha*A*B + beta*C
cblas_sgemm(CblasRowMajor, TransA, TransB,
          m, n, k, alpha, A, k, B, n, beta, C, n);
```

You may know multiply(*) and add(+) are included.
But what happens inside ?

Example: Typical Convolutional Neural Network

```
for (ic=0; ic<IC; ic++) { /* set input channel */
  ip0 = &in[ic*M*M]; /* top of input */
  for (row=1; row<M-1; row++) { /* image loop */
    for (col=1; col<M-1; col++) {
      for (oc=0; oc<OC; oc++) { /* set output channel */
        op = &out0[oc*M*M+row*M+col]; /* top of output */
        kp = &ker[(oc*IC+ic)*K*K];
        kidx = 0;
        for (y=-((K-1)/2); y<=(K-1)/2; y++) { /* kernel loop */
          for (x=-((K-1)/2); x<=(K-1)/2; x++) {
            if (ic == 0 && kidx == 0)
              *(float*)&*op  = *(float*)&ip0[(row+y)*M+col+x] * *(float*)&kp[kidx];
            else
              *(float*)&*op += *(float*)&ip0[(row+y)*M+col+x] * *(float*)&kp[kidx];
            kidx++;
} } } } } }
```

Address (col)
row
In-ch#0,1,2
In-ch#3,4,5
In-ch#6,7,8
In-ch#9,A,B
Out-ch
Chip#0
Chip#1
Chip#2
Chip#3
Chip#4
Chip#5

Goal of today

Q7. Which is correct formula representing practical energy-efficiency of computers?
実用的省エネコンピュータの正しい評価式はどれ？
A. Watt x time、B. Watt x time^2 (square)、C. Watt x time^3 (cube)
A. Watt数 x 時間、B. Watt数 x時間の2乗、C. Watt数 x 時間の3乗

Q8. What is the accuracy of π as 32-bit data?
コンピュータが32bitで表現できる円周率の精度は?

Q9. Can a computer calculate 5 billion + 1 correctly?
コンピュータは,50億+1を正しく計算できるか?

Q10. CPU/GPU is best for AI and BC?
CPU/GPUはAIやBCに最適?

6. CGRA FPU Convolution

ALU chaining and full use of narrow memory bandwidth

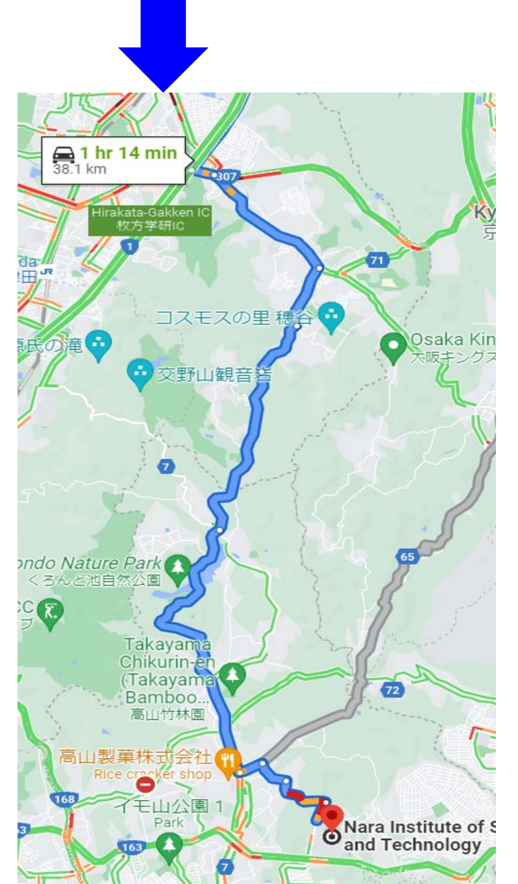
CGRA is similar to stream of water

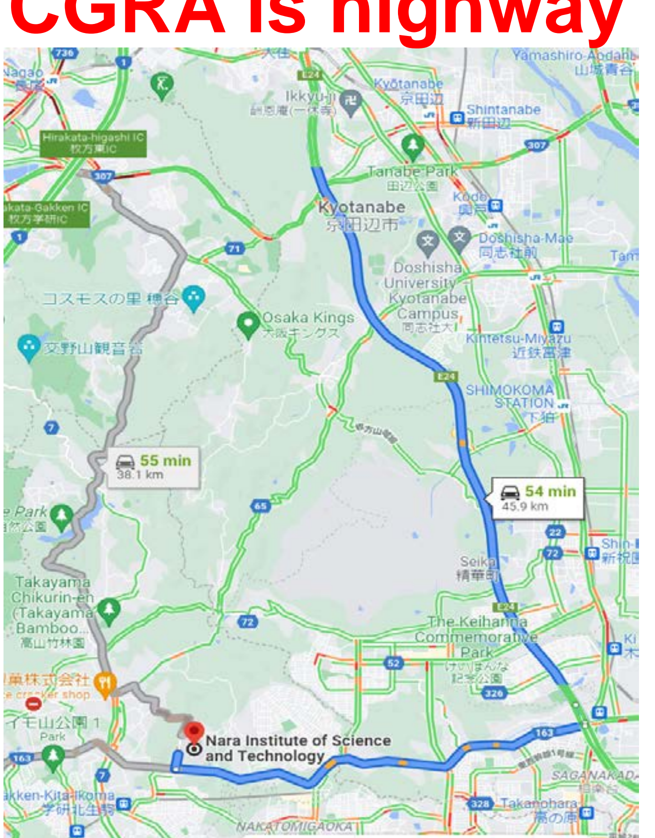
Coarse Grained Reconfigurable Array
Other name: Reconfigurable Data Path (RDP)

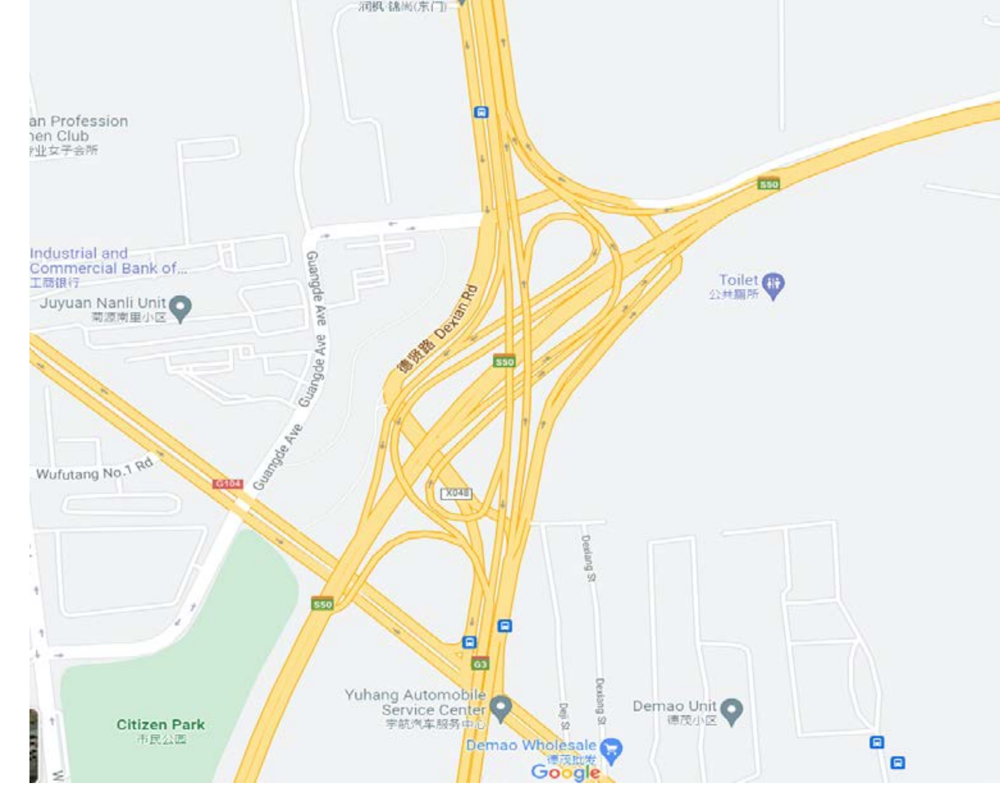
Von Neumann computers have many traffic signals

CGRA is highway

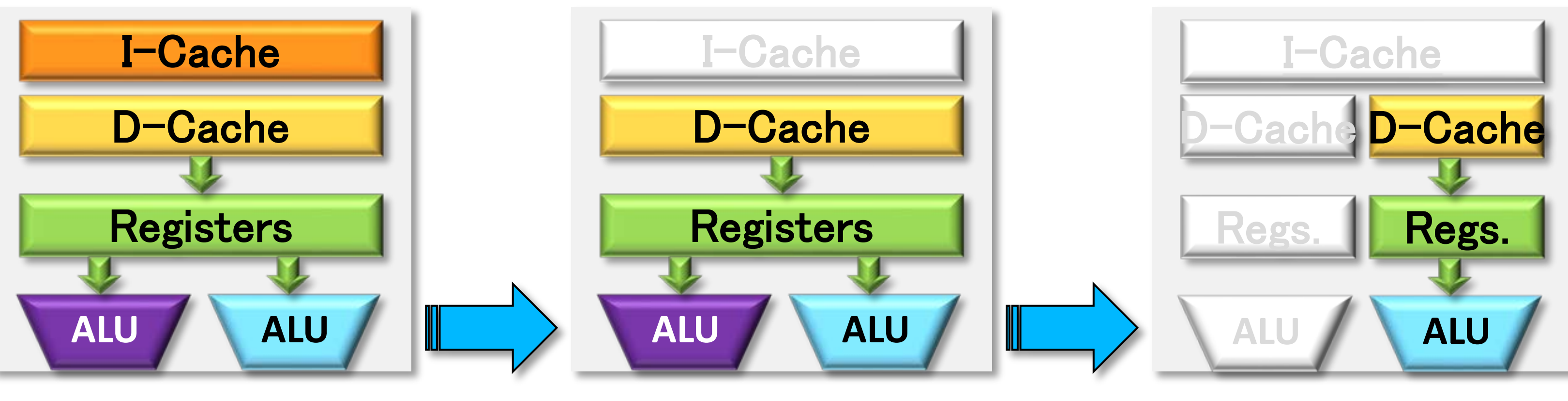
Non-stop execution

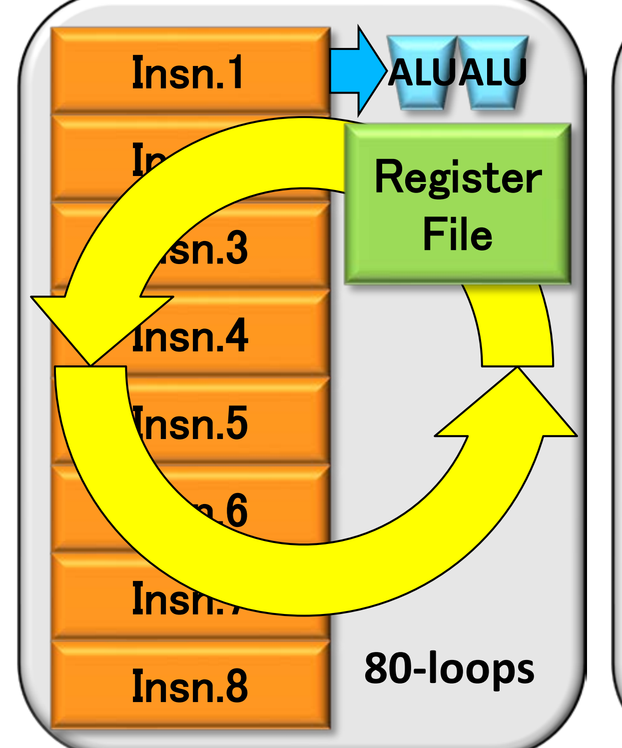


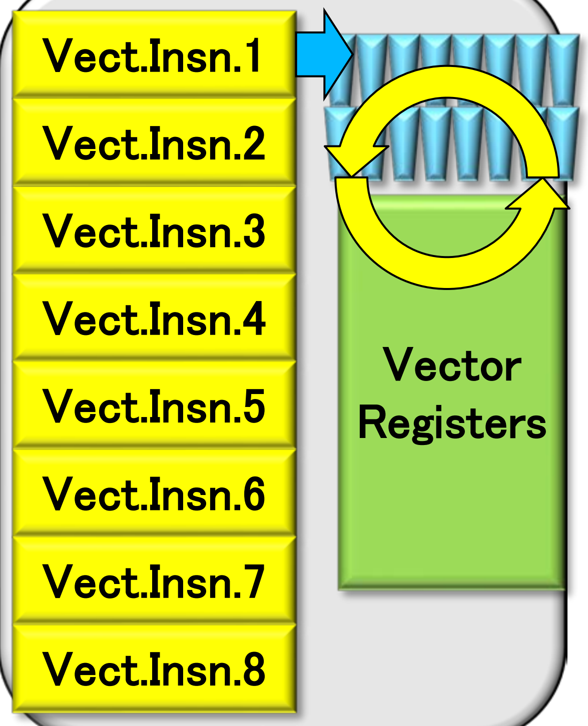


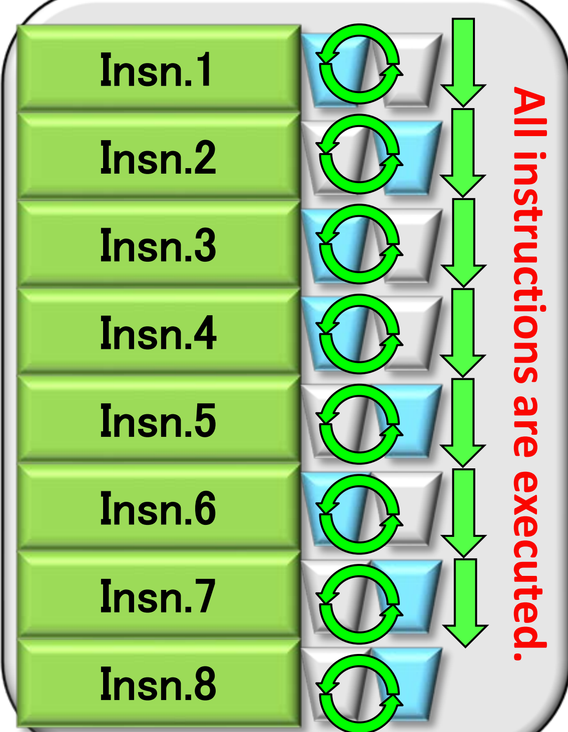


6. CGRA FPU Convolution









"GoogleのTPUにも使われたシストリックアレイアーキテクチャとDeep Learningについて", 富士通研究所技術講演会, Jul. (2017)

"Deep Learningに向けたApproximate Computingとシストリックアレイアーキテクチャ", 革新的コンピューティングの研究開発戦略検討会, CRDS/JST, Jul. (2017)

"Approximate Computingとシストリックアレイ", ジスクソフト技術講演会, Dec. (2017)

"99%メモリなアクセラレータIMAX(In Memory Accelerator eXtension)", CAE計算環境研究会@関西シスラボ 第8回シンポジウム, Mar. (2017)

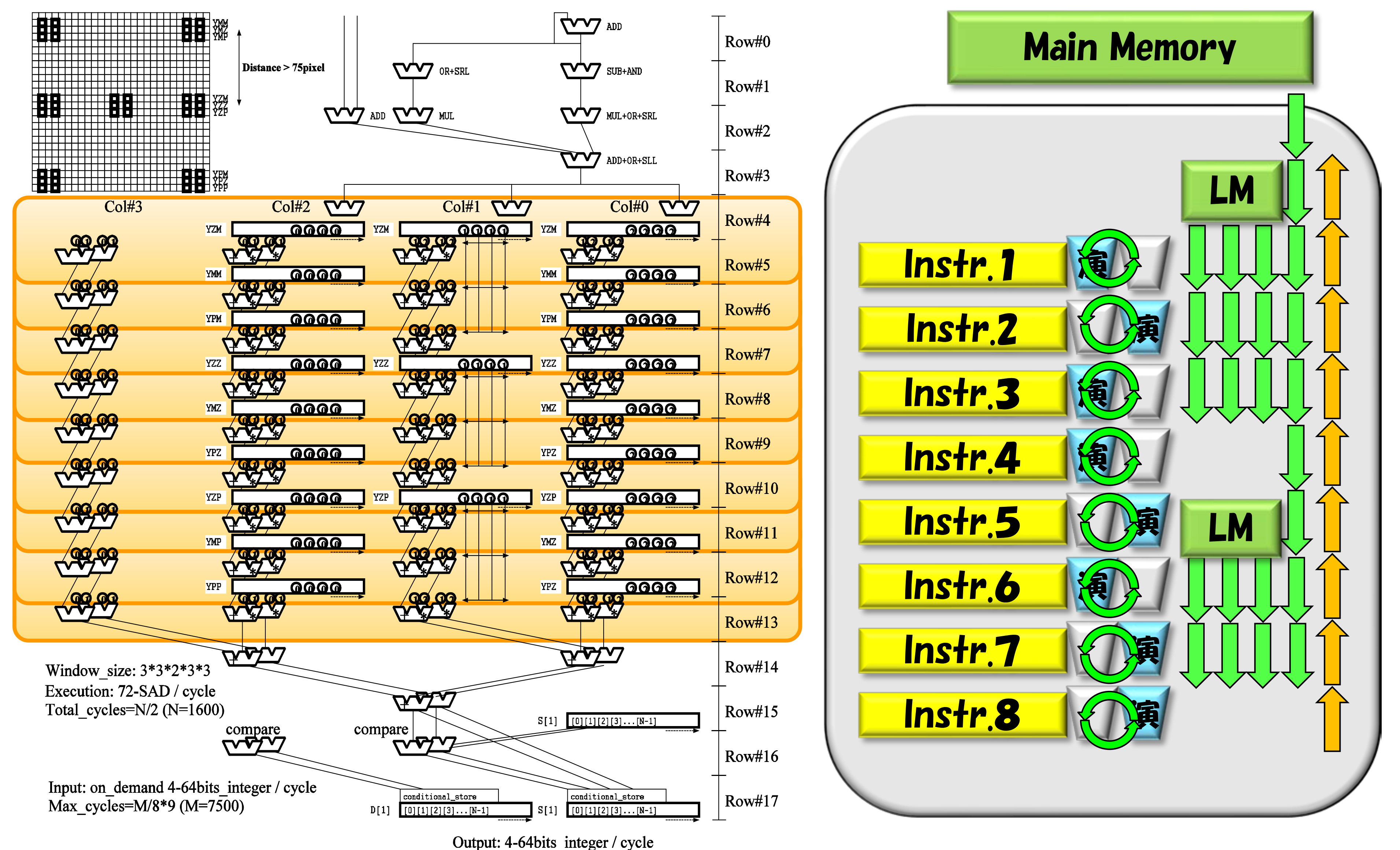
"Systolic Arrays as The Last Frontiers", Invited talk in IPB Seminar and UI seminar @ Indonesia, Jan. (2019)

"プログラマビリティを維持できる限界に向けて", SONY本社研究紹介, Mar. (2020)

"IMAX2: A CGRA with FPU+Multithreading+Chiplet", Panel: CGRA and their Opportunities as Application Accelerators, ASAP2021, invited panel, Jul. (2021)

"コンピュータ(データセンタ)の消費電力低減策意見交換会", LCS/JST, Jul. (2021)

Basic structure of CGRA



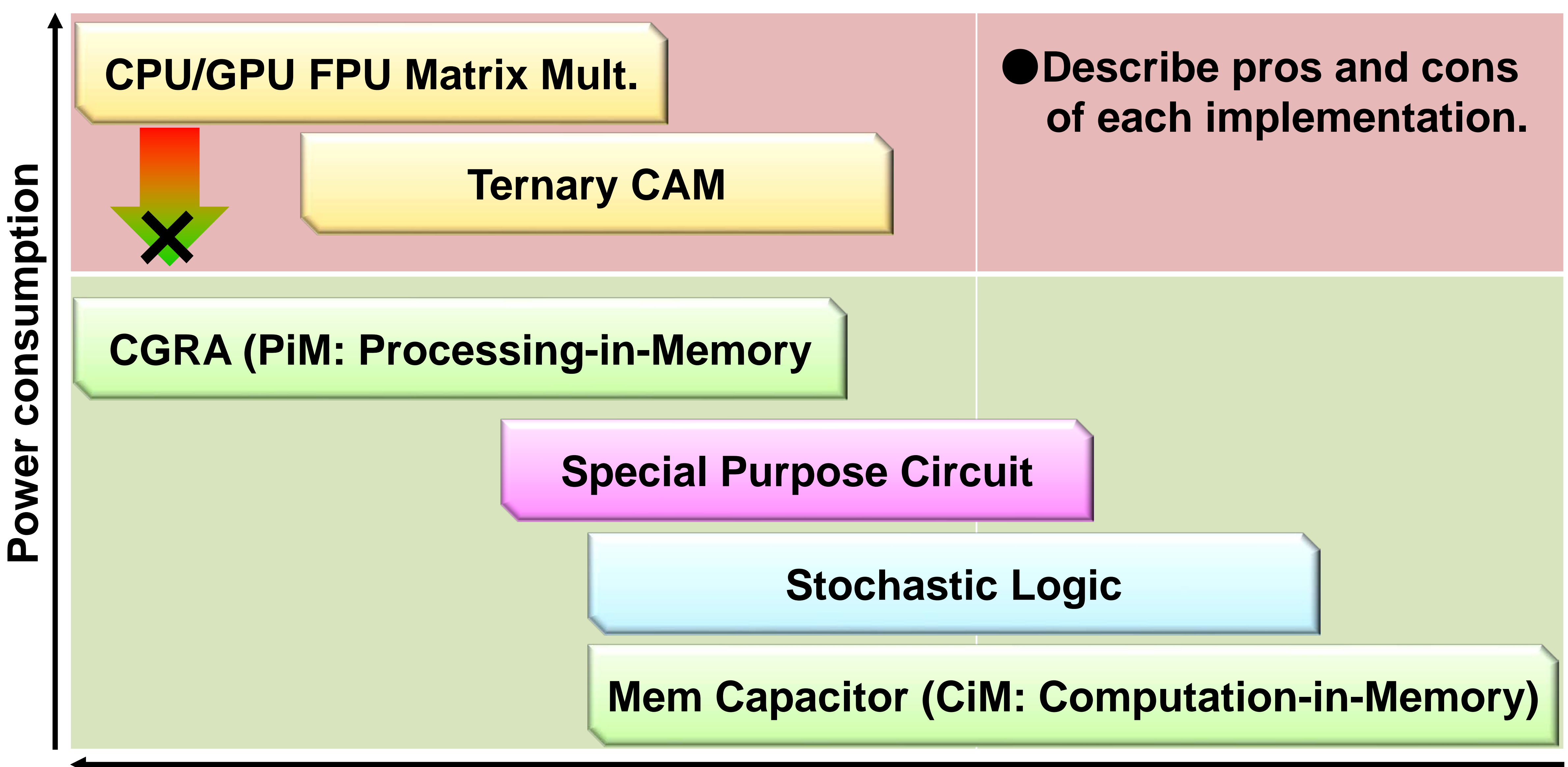
6. CGRA FPU Convolution … Prototype for 10240 operations

28nmLSI : 200x performance/area compared with GPGPU

6. CGRA FPU Convolution … Performance w/o transpose or unpacking

Kernel	CPU ARMv8 1.2GHz	GPU 256core JetsonTX2 1.3GHz DDR4 480Gbps 16nm 43.6mm^2	CGRA 64core*4 IMAX2 140MHz LPDDR5(16GB) 40Gbps 256Gbps/4266x32x2chip 2.2W [28nm想定 14.6mm^2 *4] 8nm想定 0.46mm^2 *4 (DeepScaleTool:58.4/31.818)	GPU 3584core GTX1080Ti 1.5GHz GDDR5 3872Gbps 16nm 471mm^2	GPU 10496core RTX3090 1.4GHz GDDR6X 7490Gbps 8nm 628mm^2	GPU 2048core Jetson AGX Orin 1.3GHz LPDDR5 1638Gbps 8nm
DDR bandwidth		12	1	97	187	41
Power		7.5W	ARM 0.6W + [31W] 0.6W+12W+2.2W(lpddr5) (DeepScaleTool:31/2.667)	250W	350W	60W
MM	3160msec	170 EDP=217K	16 [3msec] [EDP=284] EDP=133	12 EDP=36K	1.2 EDP=504	9.6 EDP=5530
CNN	2080msec	280 EDP=588K	23 [4msec] [EDP=505] EDP=237	18 EDP=81K	2.9 EDP=2943	10 EDP=6000
Lightfield	14500msec	1190 EDP=10.6M	754 [126msec] [EDP=501K] EDP=235K	43 EDP=462K	35 EDP=428K	158 EDP=1.5M
Sparse MM 3200^2	-	-	333+469 [134ms] [EDP=567K] EDP=266K	-	Cusparse使用 300 EDP=31.6M	Cusparse使用 324 EDP=6.30M
Sparse MM 4000^2	-	-	2378+734 [519ms] [EDP=8.51M] EDP=3987K	-	Cusparse使用 300 EDP=31.7M	Cusparse使用 330 EDP=6.53M

Formative assessment



Goal of today

Q7. Which is correct formula representing practical energy-efficiency of computers?
実用的省エネコンピュータの正しい評価式はどれ？
A. Watt x time、B. Watt x time^2 (square)、C. Watt x time^3 (cube)
A. Watt数 x 時間、B. Watt数 x時間の2乗、C. Watt数 x 時間の3乗

Q8. What is the accuracy of π as 32-bit data?
コンピュータが32bitで表現できる円周率の精度は?

Q9. Can a computer calculate 5 billion + 1 correctly?
コンピュータは,50億+1を正しく計算できるか?

Q10. CPU/GPU is best for AI and BC?
CPU/GPUはAIやBCに最適?

Download the template and submit through UNIPA.

http://archlab.naist.jp/Lectures/ARCH/ca04_2001_0502/ca042005e.docx

in http://archlab.naist.jp/Lectures

That’s all for today