FPGAの数値計算への応用の進展

会津大学/筑波大学計算科学研究センター
中里直人

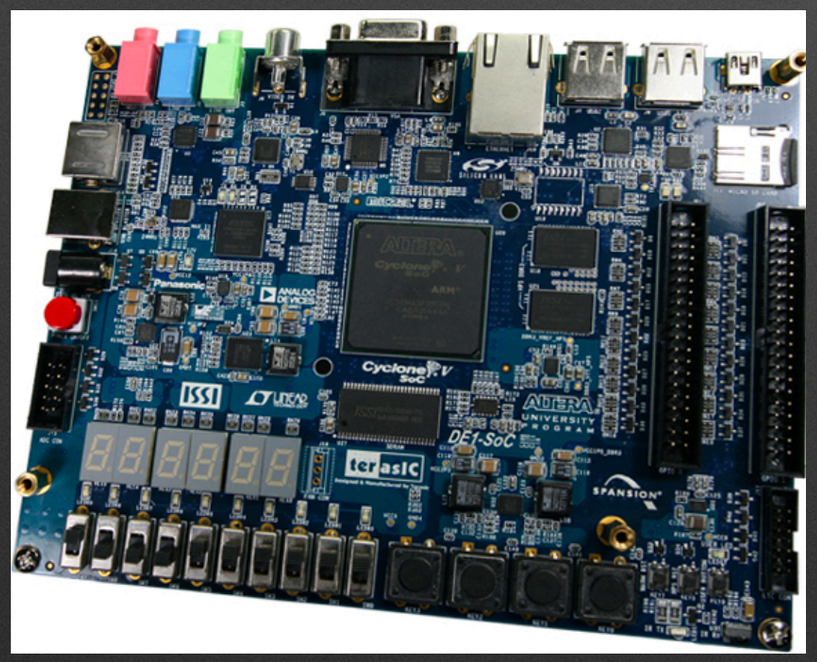
河野郁也(神戸大), 台坂博(一橋大), 湯浅富久子(KEK), 石川正(KEK)

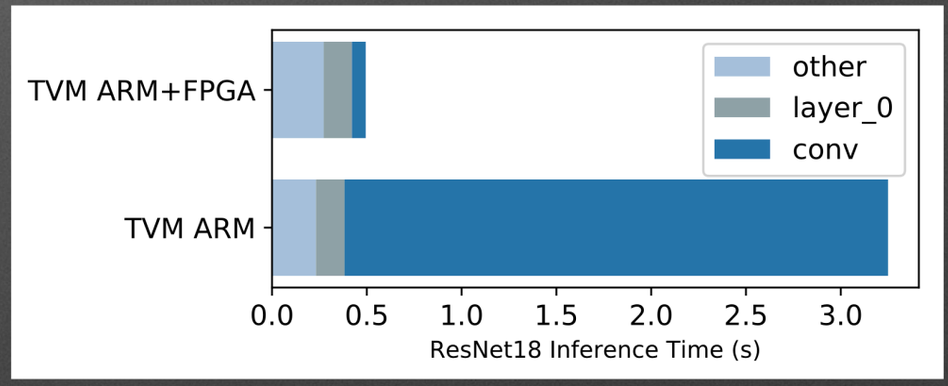
天体形成研究会 2018/11/02

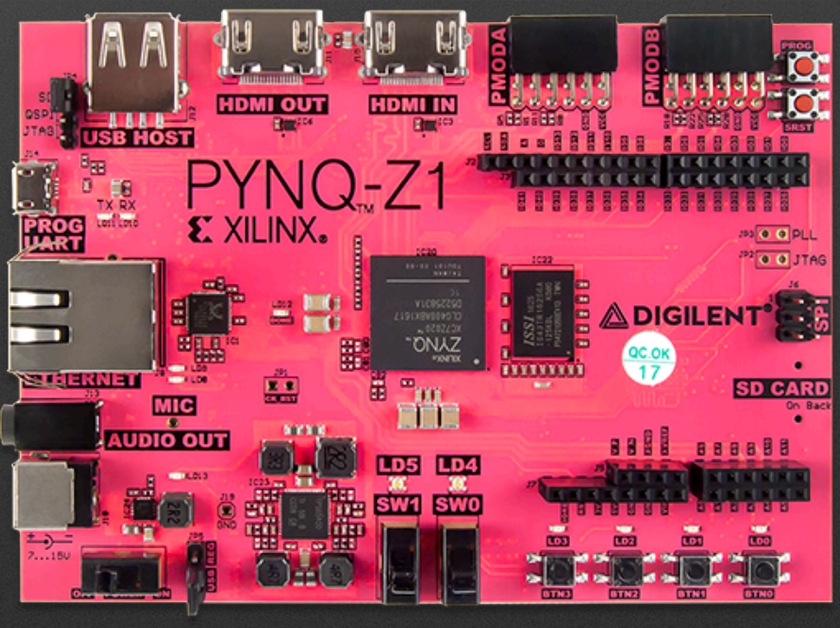
筑波大学計算科学研究センター

CPUとFPGAの統合化

- ***Processor and FPGA***
  - *Very useful for education of logic and FPGA*
  - *Prototyping of processors*
  - *Replace ARM SoC with FPGA SoC for efficiency*
  - *Real custom computing with CPU*







Recent FPGA has FP units

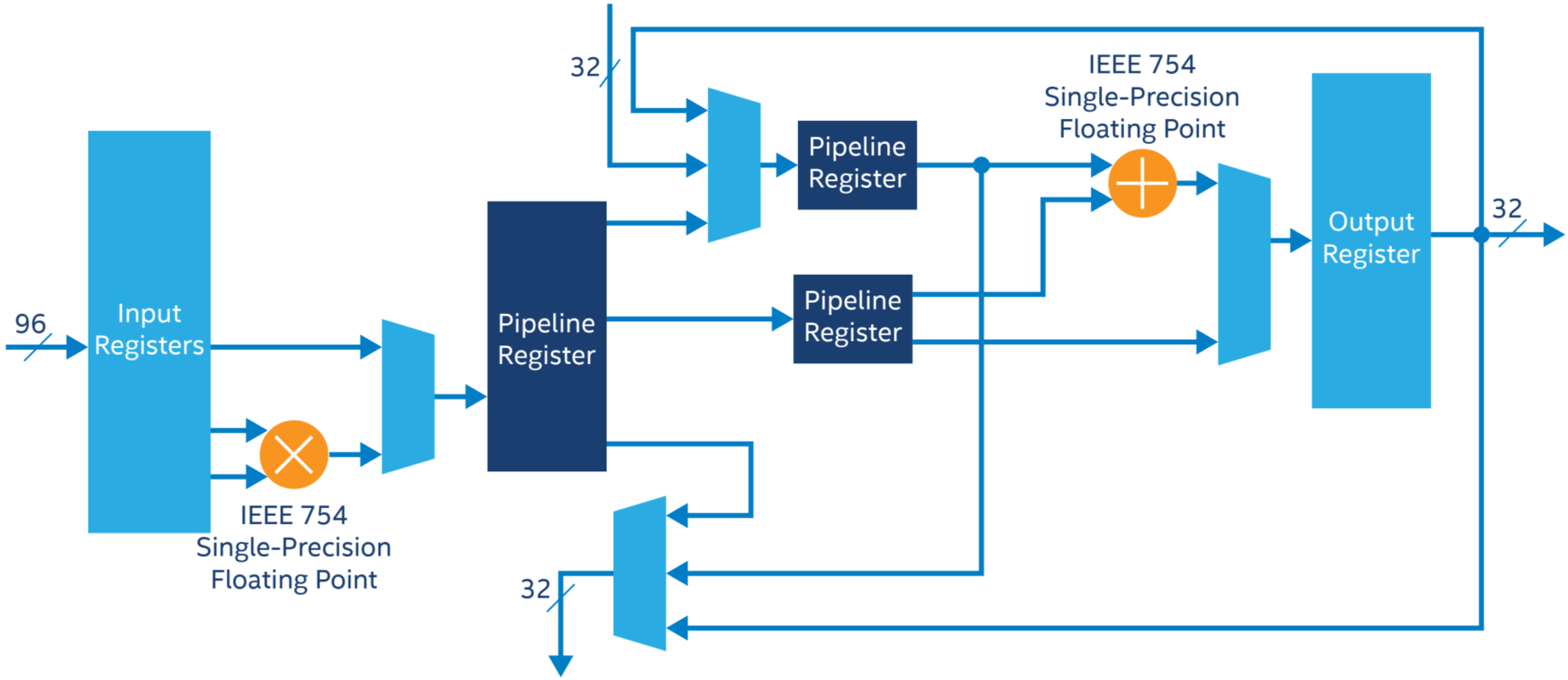


Figure 3. Floating-Point DSP Block Architecture in FPGAs

Arria10 ~ 1.5 TFLOPS in 32bit FP
Stratix10 ~ 10 TFLOPS in 32bit FP

Rise of FPGA in ML & HPC

- **"Cloud" has a node with FPGA**
  - **Amazon AWS F1**
  - **Alibaba Cloud**
  - **Microsoft Project BrainWave (ML)**

- **FPGA based HPC deployment**
  - **Cray CS500 in Parderborn Univ. (Germany)**
    - **Out of 272 nodes, 32 nodes has Stratix10**
  - **CCS will deploy an "FPGA+GPU" system**

CPU vs. FPGA in ML

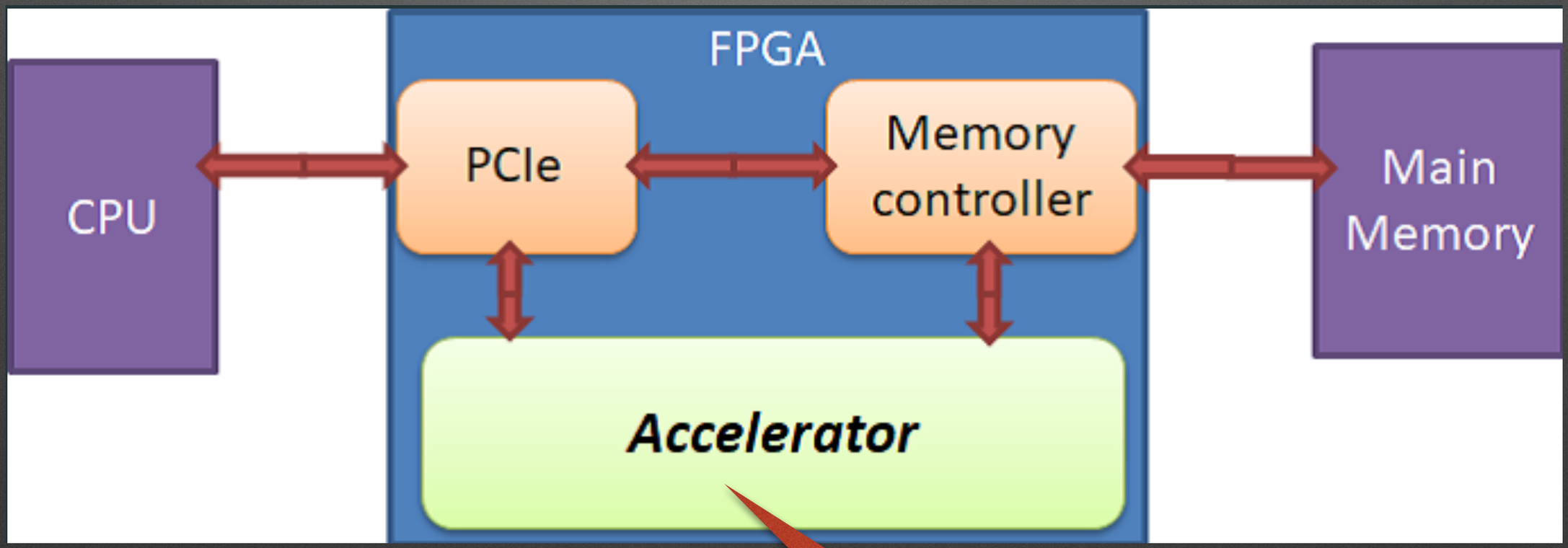
Table 1. Comparison of CPU-only vs. Brainwave-accelerated TP1 and DeepScan DNN models in Bing production.

Bing TP1			
	CPU-only	**Brainwave-accelerated**	**Improvement**
Model details	GRU 128x200 (x2) + W2Vec	LSTM 500x200 (x8) + W2Vec	Brainwave-accelerated model is > 10X larger and > 10X lower latency
End-to-end latency per Batch 1 request at 95%	9 ms	0.850 ms	
Bing DeepScan			
	CPU-only	**Brainwave-accelerated**	**Improvement**
Model details	1D CNN + W2Vec *(RNNs removed)*	1D CNN + W2Vec + GRU 500x500 (x4)	Brainwave-accelerated model is > 10X larger and 3X lower latency
End-to-end latency per Batch 1 request at 95%	15 ms	5 ms	

https://www.microsoft.com/en-us/research/publication/serving-dnns-real-time-datacenter-scale-project-brainwave/

Tsunami modeling in OpenCL (1)

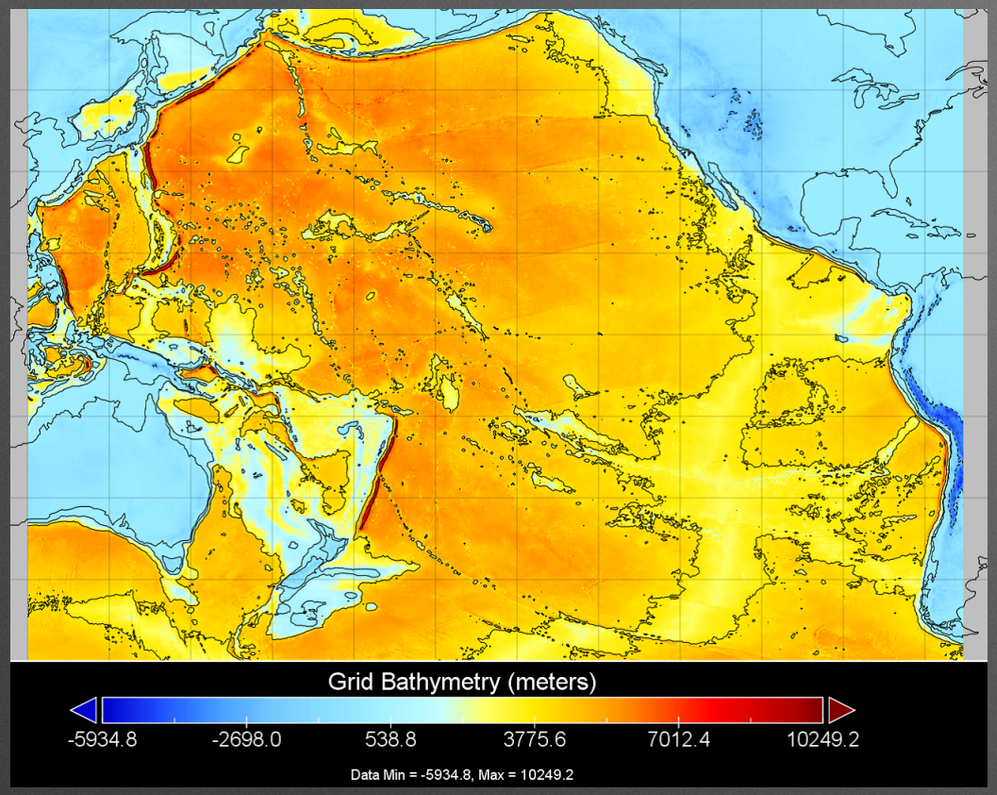
A most severe hurdle in FPGA applications is complex development …



Solution: OpenCL, Amazon HDK, Intel OPAE

We want to concentrate on this part !!

Tsunami modeling in OpenCL (2)



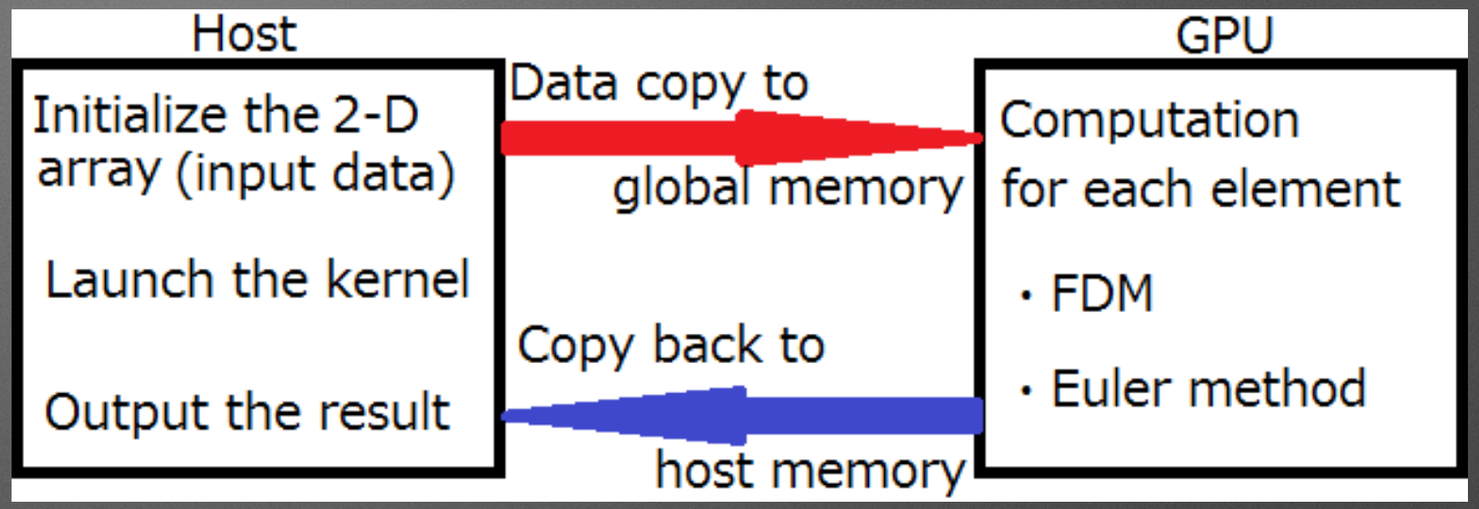


FIGURE 13. THE FLOW OF PARALLEL COMPUTATION ON GPU

Tsunami model

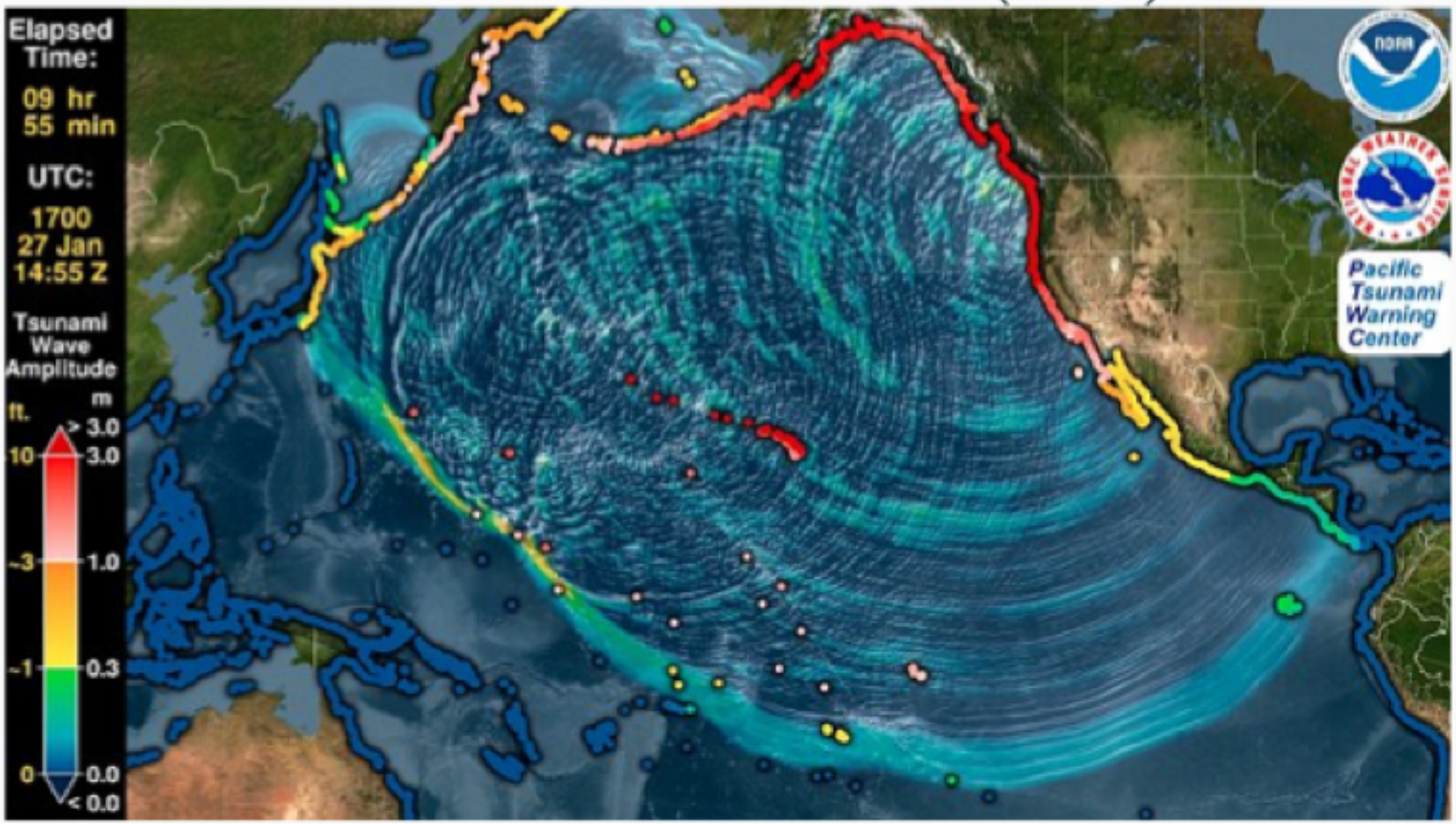
GPU acceleration

With OpenCL, we can replace GPU with FPGA with a standardized way

OpenCL kernel → FPGA configuration

Introduction: MOST (Method of Splitting Tsunami)

- The target program is **MOST (Method of Splitting Tsunami)**.
- V. Titov developed in early 1990s.
- Spatial decomposition (Splitting the computation into that for x- and y-coordinates)
- Euler method and Finite Difference Method (FDM)

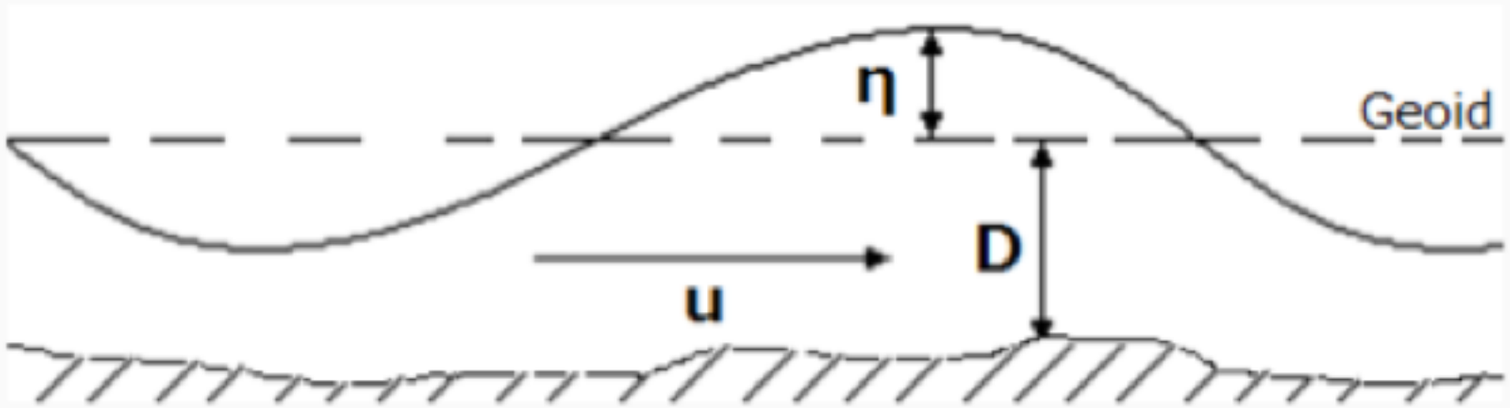


Shallow water equations for MOST algorithm

Partial differential equations (PDEs) for shallow water systems:

$$\begin{aligned} u_t + uu_x + vu_y + gH_x &= gD_x, \\ v_t + uv_x + vv_y + gH_y &= gD_y, \\ H_t + (uH)_x + (vH)_y &= 0. \end{aligned} \tag{1}$$

- u, v : wave velocity
- $H = H(x, y, t) = \eta(x, y, t) + D(x, y)$
- η : wave height, D : depth profile (bathymetry)
- g : gravitational acceleration



MOST (Method of Splitting Tsunami)

In the MOST algorithm, we use

- **Spatial decomposition** to Eq. (2) along each coordinate.
- Two auxiliary systems, $\mathbf{\Phi} = (u, 0, H)^T$ and $\mathbf{\Psi} = (0, v, H)^T$, which depend only on **one spatial variable** such as

$$\begin{cases} \dfrac{\partial \mathbf{\Phi}}{\partial t} + \mathbf{A}\dfrac{\partial \mathbf{\Phi}}{\partial x} = \mathbf{F}_1,\ 0 \le x \le X, & \text{(3a)} \\ \dfrac{\partial \mathbf{\Psi}}{\partial t} + \mathbf{B}\dfrac{\partial \mathbf{\Psi}}{\partial y} = \mathbf{F}_2,\ 0 \le y \le Y, & \text{(3b)} \end{cases}$$

where

$$\mathbf{F}_1 = \begin{pmatrix} gD_x \\ 0 \\ 0 \end{pmatrix}, \mathbf{F}_2 = \begin{pmatrix} 0 \\ gD_y \\ 0 \end{pmatrix}.$$

リーマン不変量に変換(行列Aを対角化)して離散化する

Performance benchmarking of OpenCL implementation on FPGA

For the performance evaluation of OpenCL implementation on FPGA, following 4 variations are presented;

- Code GPU (the baseline of FPGA implementation, same as Code Blocking)
- Code SR (Implementing shift registers)
- Code MC1 (Extension to compute multiple grids on one pipeline)
- Code MC2 (Further optimization for Code MC1)

OpenCL: implementing shift registers (Code SR)

In every clock cycle, a new data is shifted into the array.

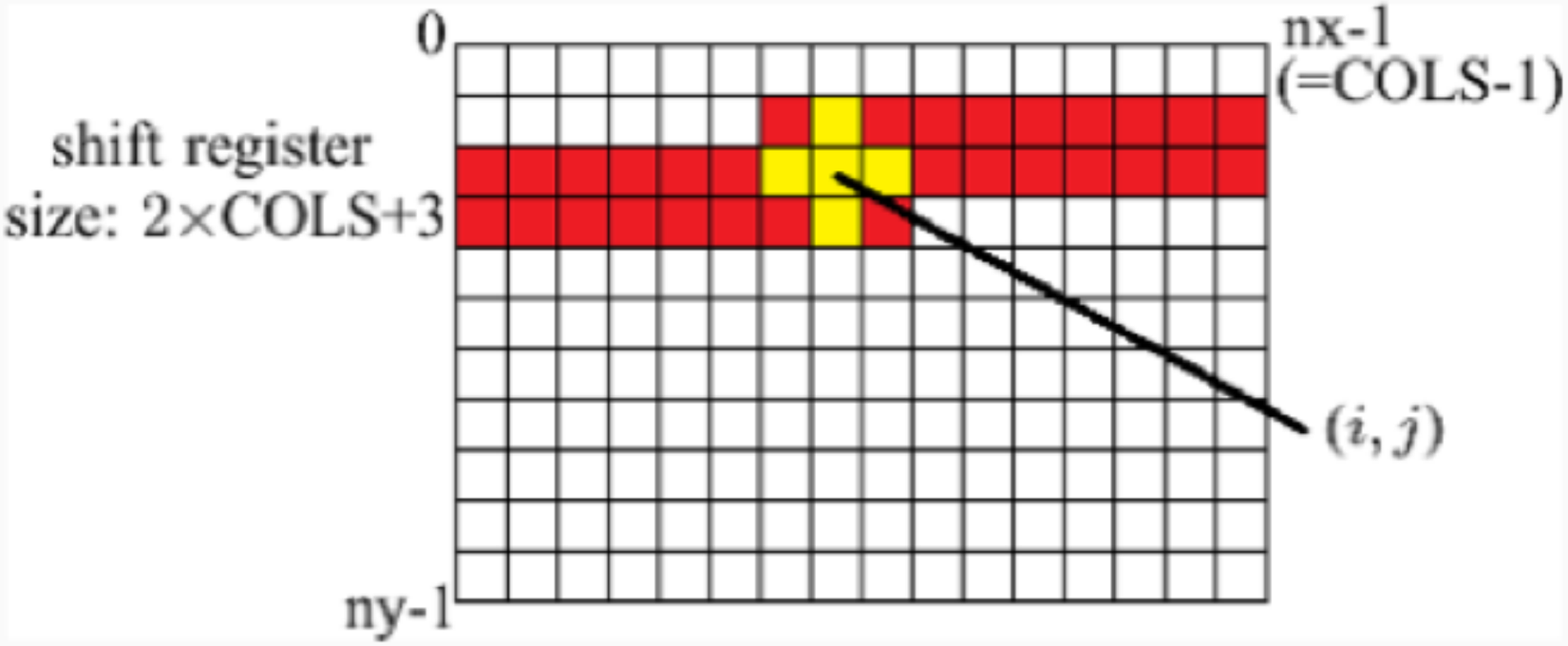


Figure 5: The data hold by shift register which requires for stencil computation when the quantities at (i, j) element are updated

- After inserted the sufficient number of data for updating the central element of the stencil, the computation is started.
- In this implementation, the **parallelism between each loop iteration** is extracted and **loop pipelining** is generated by the compiler.

OpenCL for FPGA: Benchmarking (Code GPU and SR)

The OpenCL kernel here for FPGA corresponds to with $N_{bsize} = 1$ on GPU.

	Device limit	Original kernel	Using shift register
Logic	427,200	115,274 (27%)	73,521 (17%)
I/O pins	992	161 (16%)	161 (16%)
DSP blocks	1518	345 (23%)	351 (23%)
Memory bits	55,562,240	3,552,586 (6%)	5,398,798 (10%)
RAM blocks	2,713	432 (16%)	502 (19%)

Table 3: Resource Usage of the design on DE5a-Net Arria 10 FPGA generated from OpenCL kernel

The computation time of the original kernel and the optimized kernel are;

- 2.5 hours (**can run** on FPGA, but too slow).
- 10.53 seconds (827 times faster).
  - Well **pipelined**, and exploited the **loop parallelizm** by the compiler
  - Successive iterations **launched every cycle** on FPGA

OpenCL for FPGA: Performance estimation of the FPGA (Code SR)

Adder	Multiplier	Divider	Sqrt	Madder	Others	Total
74	43	22	8	22	32	201

Table 4: The number of floating-point operators of FPGA design generated from optimized OpenCL kernel (**counted in outputted Verilog HDL file**)

- The option `-mad-enable` is used to extract multiply-add operations.
- The number in 6th column shows `fpext` or `fptrunc`.

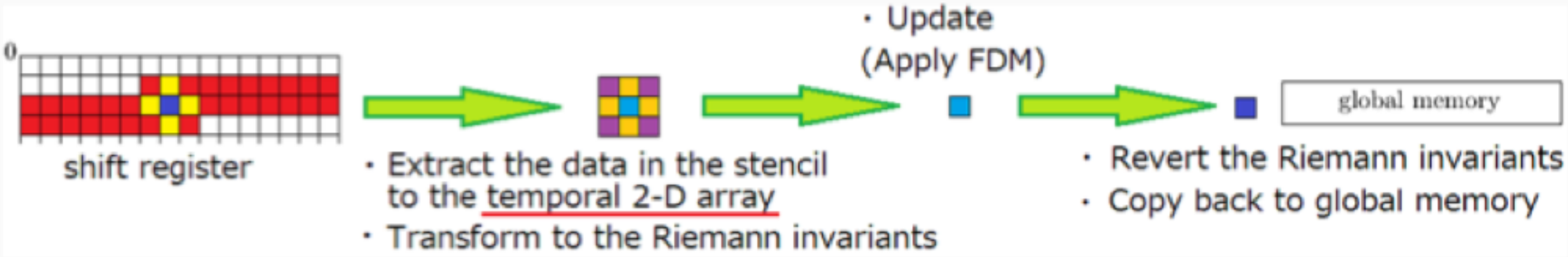
Evaluating the performance

- The clock frequency of the FPGA design is **248.63 MHz**.
- The peak performance is 248MHz $\times$ 200 opeartions $=$ **50 GFlops**.
- As the computation time is 10.53 seconds, 42.3 GFlops is the actual performance. This is **85% of the peak**.
  - Kernel occupancy and bandwidth efficiency of data transmission are kept 100% during the kernel computation.
  - **Storing** data in global memory causes **memory stall at most 8%**.

OpenCL for FPGA: Optimization of MC1 (Code MC2)

Current FPGA design of Code SR, MC1 are based on GPU kernels.

- Spatial blocking is applied for efficient memory access which depends on cache system.
- The data copy from the shift register to **other 2-D arrays** for the stencil computation makes the overhead.



OpenCL for FPGA: Optimization of MC1 (Code MC2)

In this optimization (Code MC2),

- The stencil computation is applied to the **data in shift registers**.
- The 1-D arrays which represents shift registers are stored as the format of **Array of Structure (AOS)**.
- The transformation to Riemann invariants are conducted before inserting the data into shift registers.
  - The shift register has the **data which has already transformed**.

OpenCL for FPGA: Comparison between Code MC1 and MC2

Analyse the performance difference between Code MC1 and MC2

- Here, number of operators is **extracted from the compilation log**.

	Adder	Multiplier	Madder	Total
Code MC1	465	181	32	1206
Code MC2	345	150	66	817

Table 7: The comparison for number of operators on DE5a-Net Arria 10 FPGA designed from Code MC1 and MC2 with $N_{buf} = 4$

The design by MC1 uses

- 1.5 times number of total operators, and
- less than half multiply-add operators

as much as that by MC2.

OpenCL for FPGA: FPGA design of MC2 by ver. 17.1 compiler

The performance of the design for $N_{buf} = 4$ is 153GFlops by using ver 17.1.

- **Increase of frequency** contributes the performance improvement.

$N_{buf} = 4$	ver. 16.0	ver. 17.1
Logic	168,093 (39%)	176,262 (41%)
DSP blocks	817 (54%)	756 (50%)
Memory bits	7,480,100 (13%)	6,738,632 (12%)
RAM blocks	589 (22%)	638 (24%)
Frequency (MHz)	199.80	**287.43**

Additionally,

- More multiply-add operations are extracted from the computation.

	Adder	Multiplier	Madder	Total DSP
ver. 16.0	345	150	66	817
ver. 17.1	207	146	160	767

- Two DIMMs (Twice memory bandwidth as ver. 16.0) are avaiable.

Summary: Tsunami modeling in OpenCL

Original OpenCL kernel

Resource usage of FPGA design by using GPU kernel

	Device limit	Original kernel
Logic	427,200	115,274 (27%)
DSP blocks	1518	345 (23%)
Memory bits	55,562,240	3,552,586 (6%)
RAM blocks	2,713	432 (16%)

2.5 hours / model

Optimal OpenCL kernel (4 units)

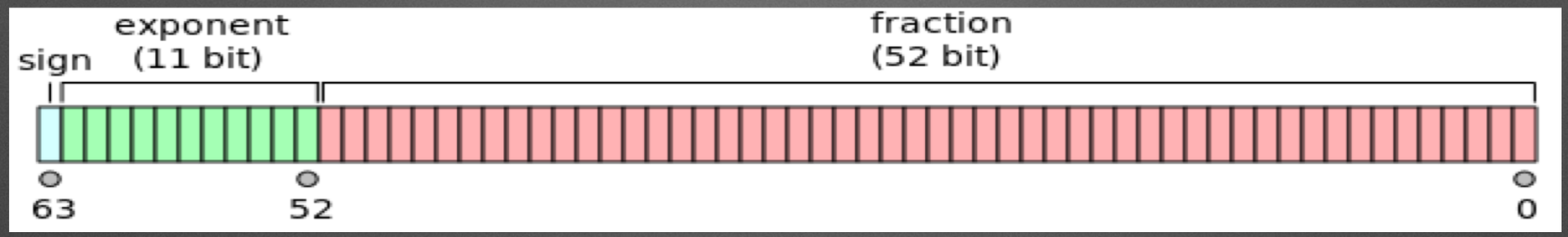
Logic	176,262 (41%)
DSP blocks	756 (50%)
Memory bits	6,738,632 (12%)
RAM blocks	638 (24%)

2.9 sec / model

Kono, Nakasato etal (2018) https://doi.org/10.1007/s11227-018-2315-8

Encoding for arithmetic operations

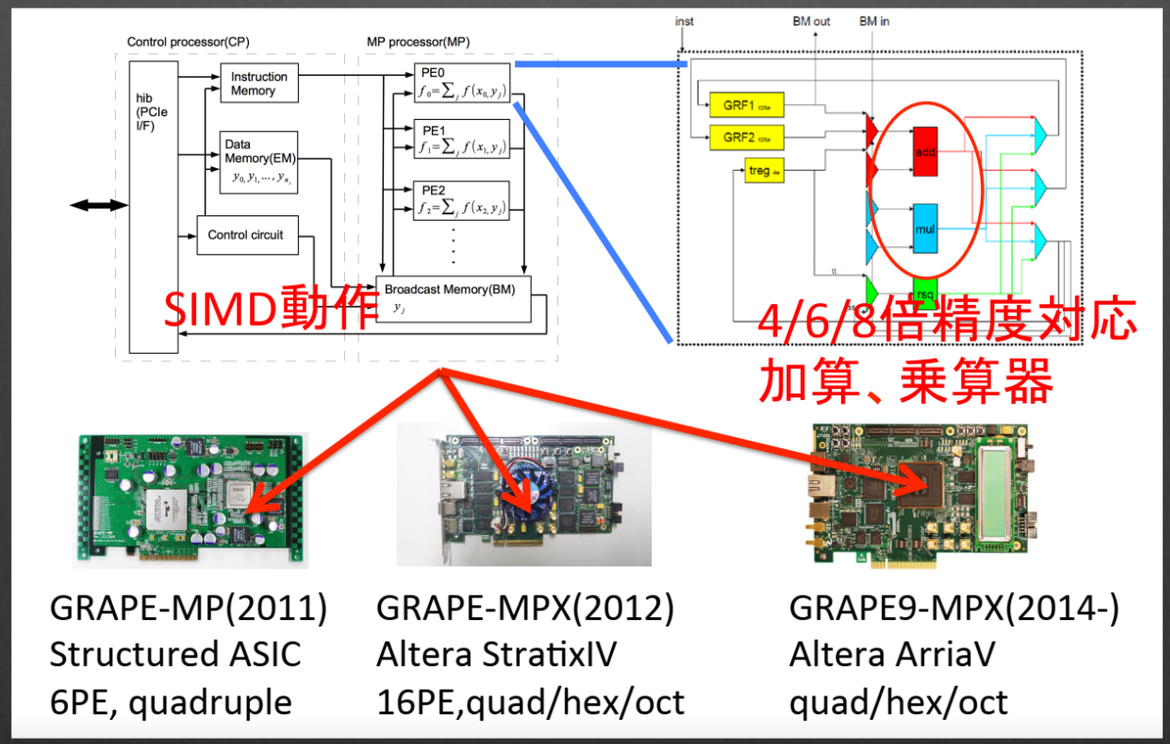
- *A standard for FP is IEEE 754*
  - *16, 32, 64 and 128 bits*



- *For AI processors*
  - *Various exotic encoding is be proposed*
  - *half precision (16 bit)*
  - *fp8, fp9, fp10 etc… (MS, Google, Baidu)*
  - *Flexpoint (Intel)*
  - *Int8, Int9 etc. Even Int1 (binary) or Int2 (ternary)*

直接多倍長精度計算

- GRAPE-MPをFPGAで実装(2011 - 2017)
  - GRAPE-MPと同じアーキテクチャを各種FPGAに実装
  - 演算精度の拡張：六倍(192bit), 八倍(256bit)精度
  - 指数部の拡張：11 (GMP) → 15(GMPX) → 19 (G9MPX)
  - 計算機システムとして大規模化



Daisaka, Nakasato, Ishikawa, Yuasa etal. papers

GRAPE9-MPX ラック@ KEK

- Altera ArriaV 64ボードで構成
  ホスト計算機8台に8 FPGA ボード
- ホスト計算機
  CPU: Intel Xeon 2687W v3 (x 2)
  (2台はXeon 2687w v4 x 2)
  MEM: 128GB(16GB x 8)
  MB: SuperMicro X10DRX
  (11 PCIeスロット)
- ホスト間は10G/1GbEで接続

まとめ

- 最新のFPGAはCPUと統合化されている
- MLでのFPGAの有効性。HPCでも
- FP32であれば非常に高速
- 特殊な演算フォーマットも得意
- DDR3/DDR4メモリが接続される
- OpenCLで「機能」だけを実装できる