**2019.3.14-2019.3.19（The 3st week）**

**Achievement:**

**Item1**

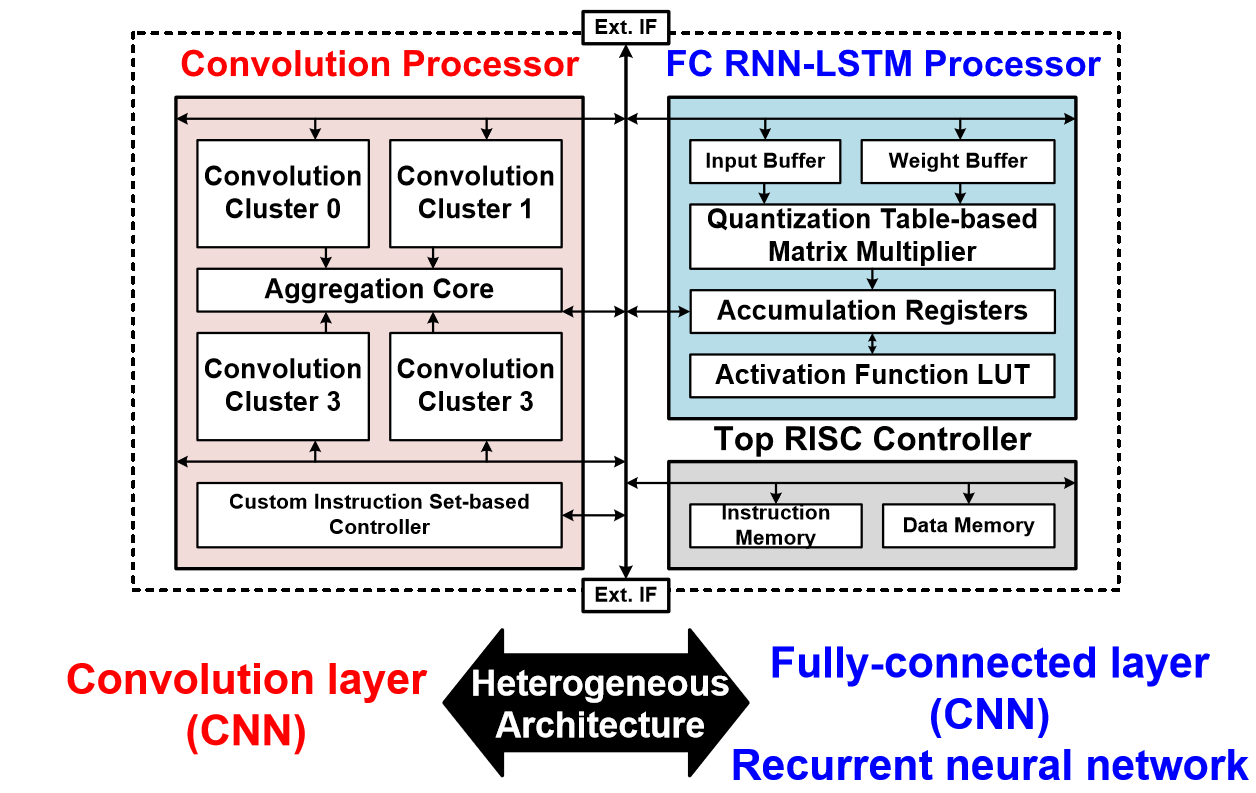
**Paper reading:**

**Title:** DNPU: An 8.1TOPS/W Reconfigurable CNN-RNN Processor for General-Purpose Deep Neural Networks（ISSCC 2017 14.2）

**Current situation:** The computational requirements in CNNs are quite different from those of RNNs. It’s not practical to accelerate FCLs and RLs with SoCs specialized for CLs , or accelerate CLs with FCL- and RL-dedicated SoCs. A highly reconfigurable CNN-RNN processor with high energy-efficiency is desirable to support general-purpose deep neural networks.

**Contributions：1)** the first CNN-RNN SoC with the highest energy efficiency **2)** a LUT-based reconfigurable multiplier optimized for the dynamic fixed-point with on-chip adaptation via overflow monitoring to exploit maximum efficiency from kernel reuse in the CP **3)** a quantization table (Q-table)-based matrix multiplication to reduce off-chip memory access and remove duplicated multiplications in the FRP.

**Overall architecture:**



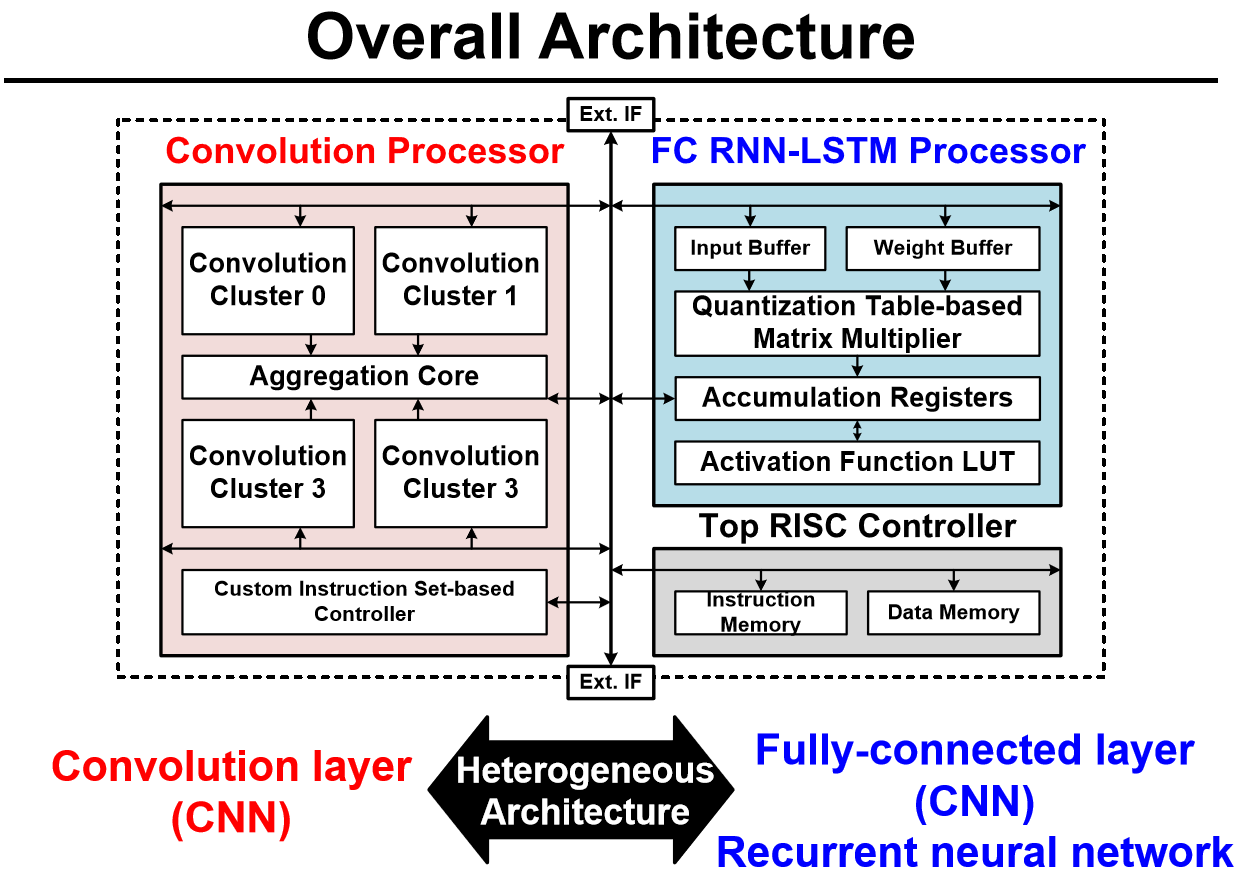
This chip consists of three parts: CP, FRP and a top RISC controller. The CP is composed of 4 convolution clusters and 1 aggregation core. Each convolution cluster performs convolution operation cores., and transfers the accumulation results to the accumulation core. One convolution core contains 4 PE groups with 48 PEs. The FRP performs matrix multiplication with the 128-enty Q-table, and 8 16b fixed-point multipliers are used to update the Q-table. The CP and FRP are able to process 4 different CLs and 8 RLs respectively, in parallel.

**Chip Specifications**:

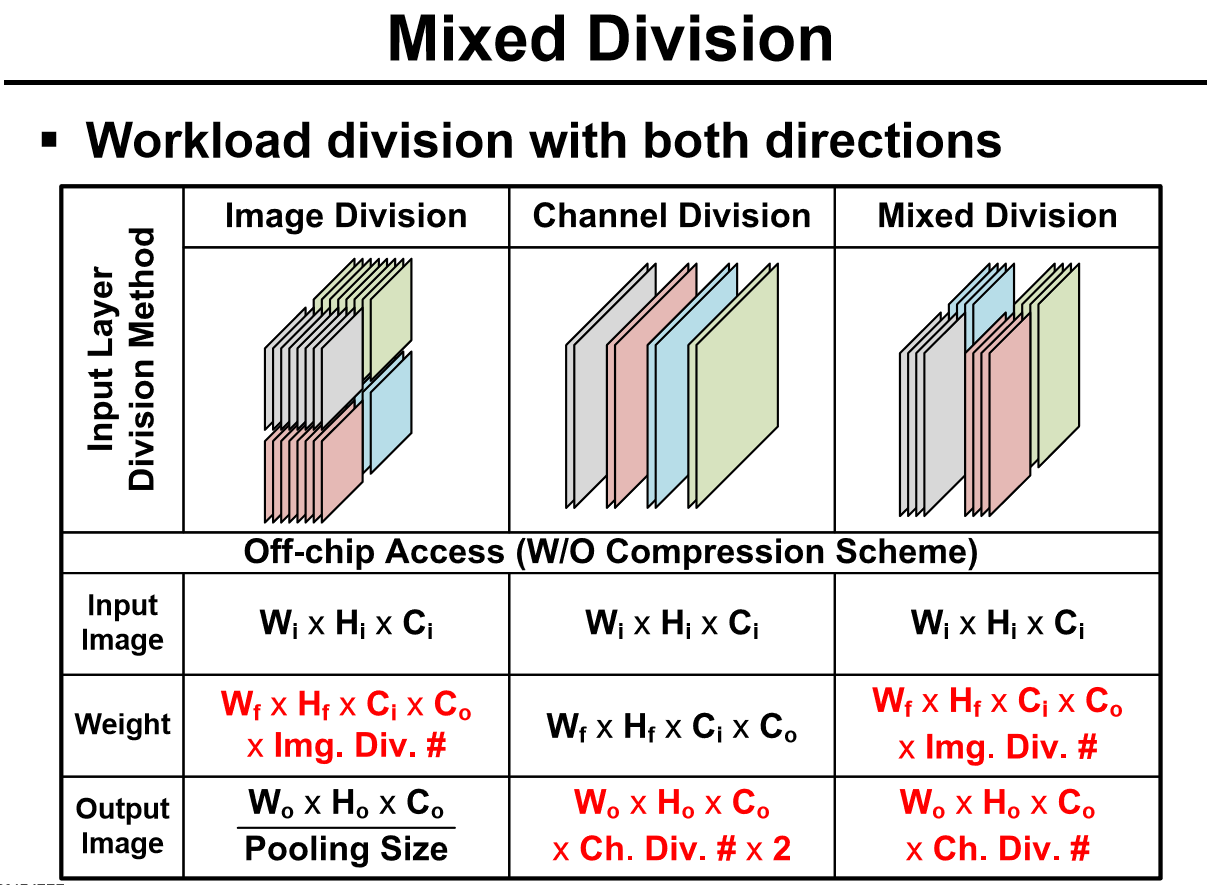
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Technology | 65nm 1P8M CMOS | | | | | | | |
| Chip size | 4\*4mm2 | | | | | | | |
| Gate count | \ | | | | | | | |
| On-chip Memory | CLP(280KB) FCRLP(10KB) | | | | | | | |
| Supply voltage | 0.77~1.1V | | | | | | | |
| Operating Frequency | CLP | | FCRLP | | NoC | | | ETC |
| ~200MHz | | ~200MHz | | ~400MHz | | | ~200MHz |
| Power Consumption |  | | CLP | FCRLP | | | ETC | Total |
| [50MHz@0.77V](mailto:50MHz@0.77V) | | 29mW | 2.6mW | | | 3mW | 34.6mW |
| 200MHz@1.1V | | 235mW | 21mW | | | 23mW | 279mW |
| Energy Efficiency |  | CLP Word Length:16 bit | | | | CLP Word Length:4 bit | | |
| 50MHz@0.77V | 2.1TOPS/W | | | | 8.1TOPS/W | | |
| 200MHz@1.1V | 1.0 TOPS/W | | | | 3.9TOPS/W | | |

**Summary**:

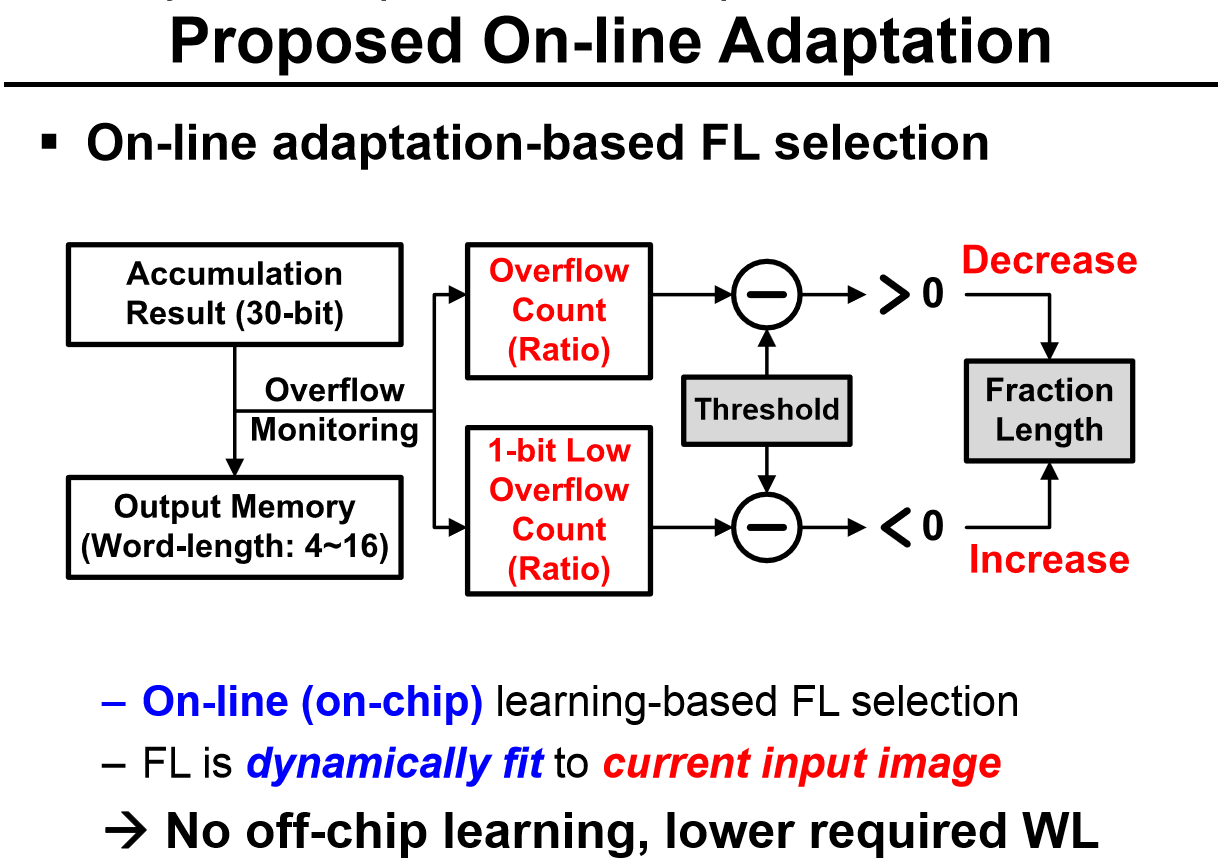
**Advantages**:1) the only one that supports both CNNs and RNNs,. Not only make use of massive data reuse in CNNs, but take the compute feature of RNNs into consideration.



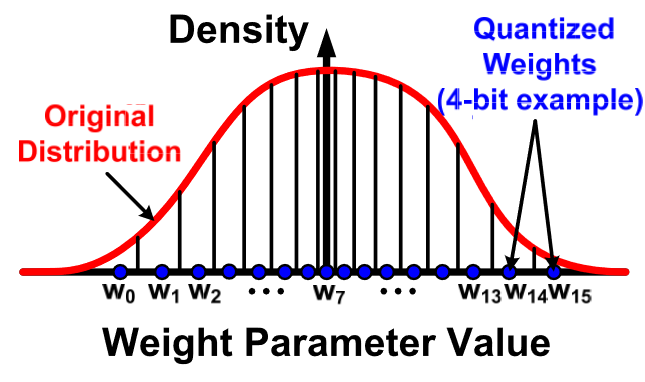
2)Propose input-layer mixed division method, we can calculate the best division method for each input-layer off-chip, thus save bandwidth and minimize off-chip memory access



3) Propose dynamic fixed-point with on-line adaption architecture to minimize the off-chip memory access of filter at a low accuracy loss. In many CNN layers, we don’t need to use 32b data for convolutions, instead, replace 32b data with few bit data like 4b, at the same time, lose only a few accuracy, that is to say, make a tradeoff between accuracy and data transfer pressure to save power consumption. Add on-line adaption architecture to further minimize data transfer.



4) Propose Quantization Table-based Multiplier to minimize off-chip accesses. In this example, use 16 quantized weights to replace initial weights, therefore avoid off-chip memory access.



**Shortcomings:**1) No details about on-chip adaptation with overflowing monitoring for accumulation

**Item2**

Finish the checkout (literal expression and graphs) of Part 2.5-2.6

**Item3**

Learn Asynchronous FIFO design:

**Plan next week:**

Item1: Paper reading.

Item2: Use Verilog to write some simple modules for practice.

Item3: Learn MIPs computer architecture.

**2019.3.7-2019.3.12（The 2nd week）**

**Achievement :**

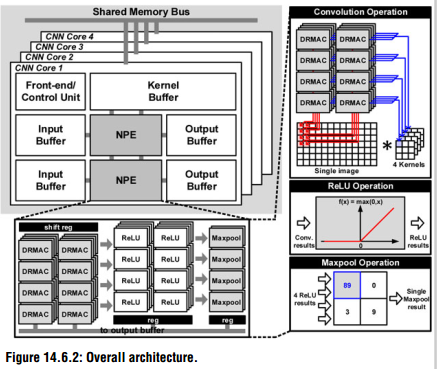
**Item1 Paper reading:**

**Title:** A 1.42TOPS/W Deep Convolutional Neural Network Recognition Processor for Intelligent IoE Systems（ISSCC 2016 14.6）

**Current situation:** It’s not practical to transmit data acquired by IoE(Internet-of-Everything) device to data center for process, doing in-situ processing of data can be more efficient, in which situation application-specific accelerator is of immediate use.

**Contributions：**1)a CNN-optimized neuron processing engine(NPE) 2) a dual-range multiply-accumulate(DRMAC) block for low-power convolution operations, 3) an on-chip memory architecture and a utilization scheme for reducing off-chip memory access, 4)kernel data compression for further reducing off-chip memory access

**Overall architecture:**



There are four homogeneous CNN cores in the chip，each core contains 2NPEs, 2 image buffers, 2 output buffers and a kernel buffer. The NPE consists of 32 MACs, 32 ReLU, 8 Maxpool blocks, used for multiply/accumulate, nonlinear and pooling process respectively, these blocks can operate in parallel to exploit data-level parallelism.

**Chip Specifications**:

|  |  |
| --- | --- |
| Technology | 65nm 1P8M CMOS |
| Chip size | 4\*4mm2 |
| Gate count | 3.2M Logic Gates |
| On-chip Memory | 36KB |
| Supply voltage | 1.2V(core)/3.3V(IO) |
| Clock rate | 125MHz |
| Peak throughput | 64GOPS |
| Arithmetic precision | 24b fixed-point |
| Targeted benchmark | MNIST,CIFAR-10,GTSRB,AlexNet |

**Summary**:

**Advantages**:1) ReLU and Maxpool blocks are turned off until convolution operations are finished to **save power** 2) 4 groups of 8 MACs share a input feature map to **reuse ifmap** and **save bandwidth** 3) dual range multiply-accumulate to **save power** 4) switch the roles of input buffer and output buffer to the opposite to **avoid off-chip memory access**, similar to this, Tanji3 use global buffer to store both ifmap and ofmap to avoid off-chip memory access 5) use off-chip principle component analysis to get basic kernels, only transmit basic kernels to chip and generate constant kernels on-chip to **save off-chip memory access**

**Shortcomings:**1) No mention about NPE use ratio.

**Item2 Tanji3 documentation:**

Finish the checkout (literal expression and graphs) of Part 2.4 Global buffer.

**Plan next week:**

Item1: Go on with the checkout of Tanji3 documentation with Part 2.5-2.6.

Item2: Paper reading.

Item3: Use Verilog to write some simple modules for practice.