인공지능 프로세서 AB9 NPU
- 개요와 프로그래밍 -

한국전자통신연구원 지능형반도체연구본부
AI Processor Research Department
Electronics and Telecommunications Research Institute

2021. 1. 13.

ETRI
한국전자통신연구원

AB9

AB9 NPU Processor 개요

ETRI
미래사회를 만들어가는
국가 지능화 종합 연구기관

AB9 (Artificial Brain 9)

AI NPU Processor developed by ETRI

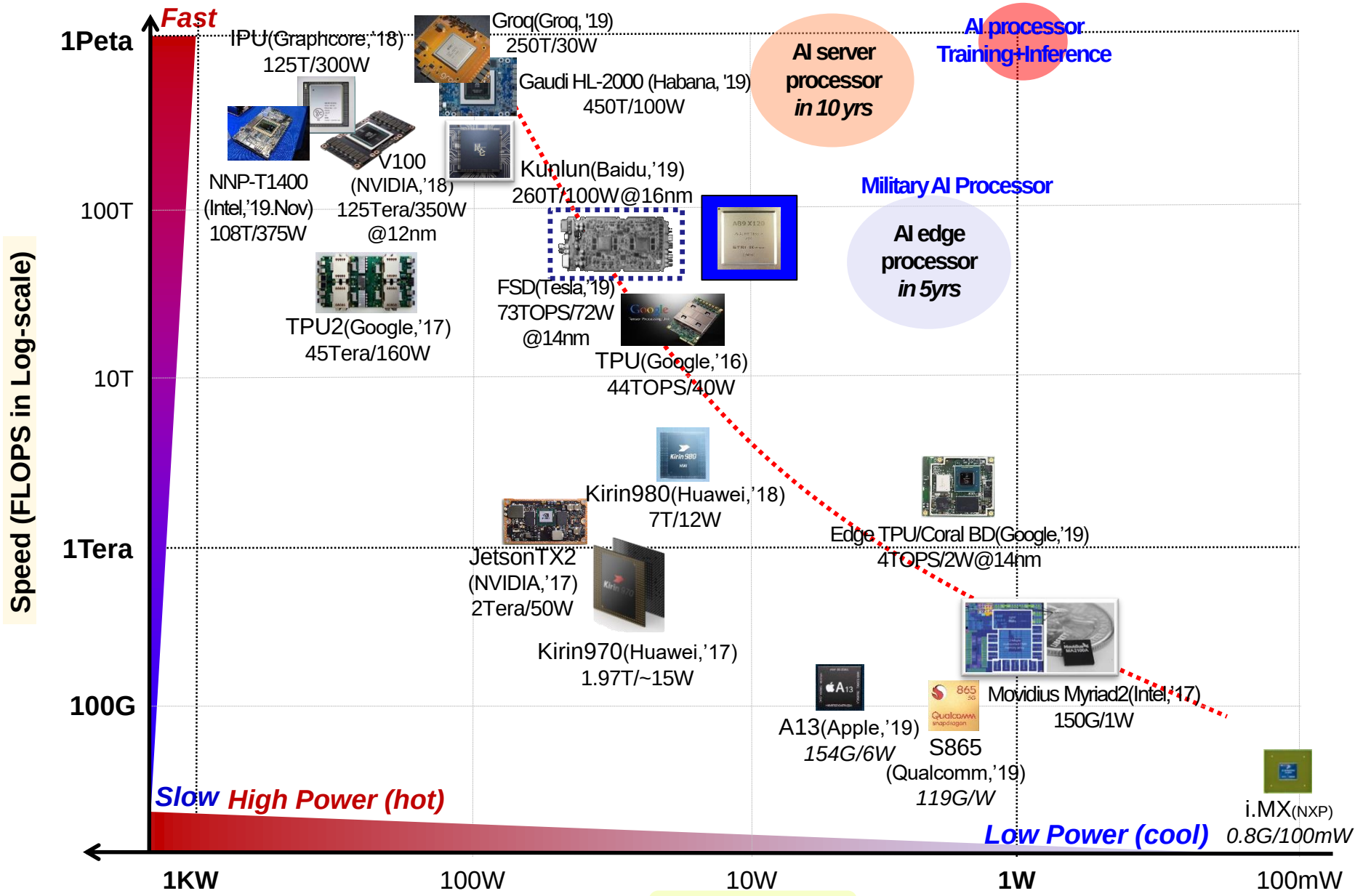
Designed during 2017.4~2020.2

The 1st K-NPU chip

Working, Fast 40TFLOPS, Lowest Power 15W

Full AI-SW support for AI chip in action

AI NPU Processors on the Market



NPU's Target

Super-Performance

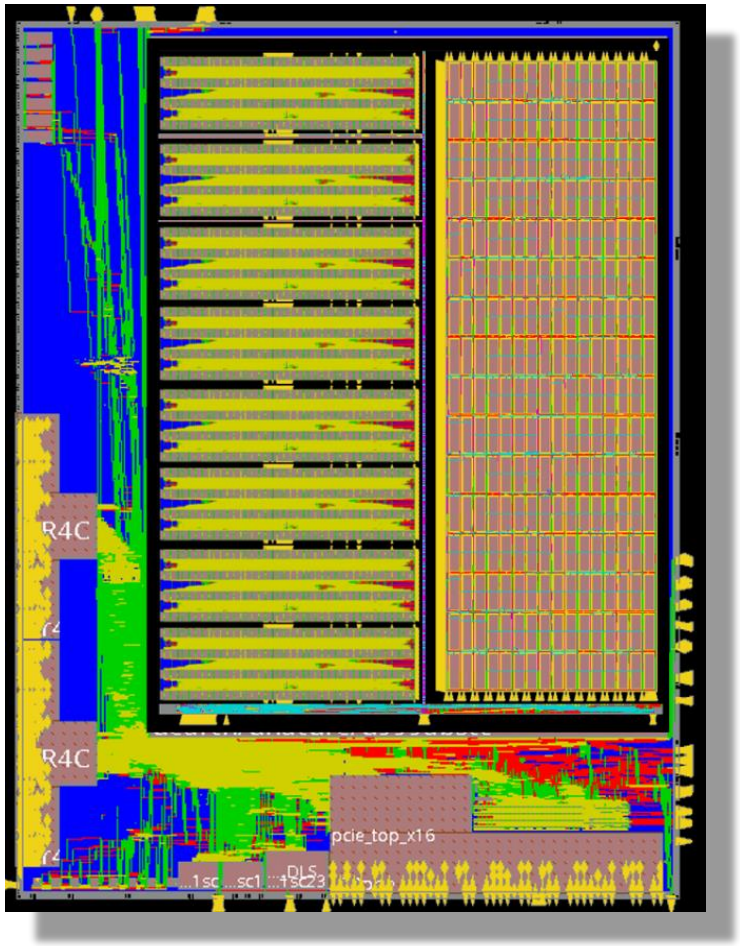
Power-efficiency

Others are

Too hot! ~ 300Watt

But Slow in terms of efficiencyiy

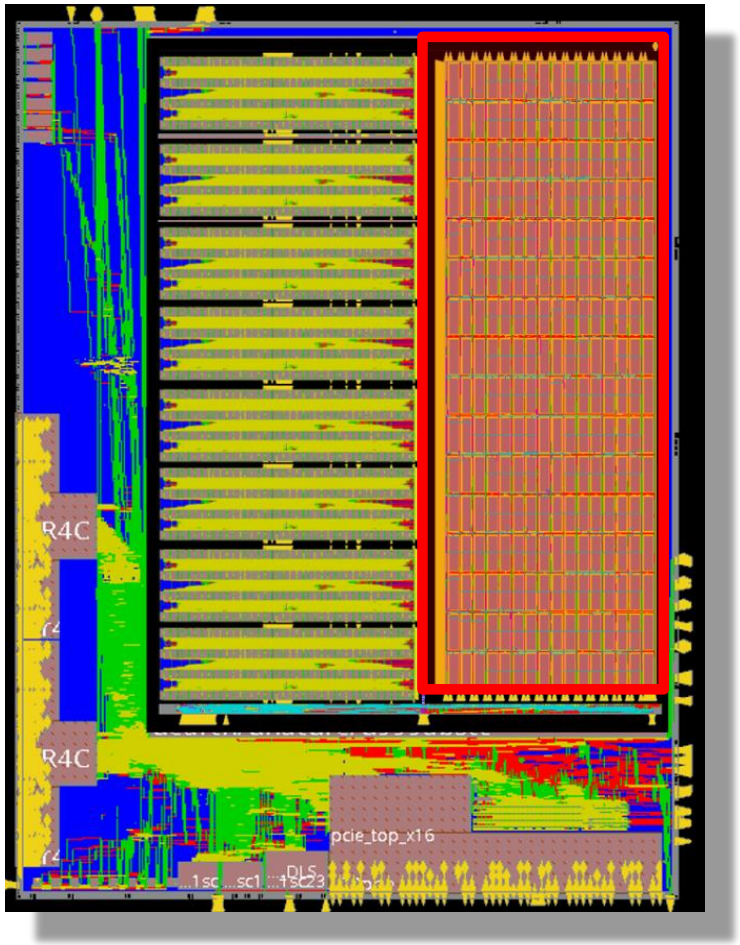
AB9 *from* ETRI



AB9

Performance: 40TFLOPS, 15W
Die Area: 17mm x 23mm, 391mm^2
Size: 1Billion Transistors@28nm
Core frequency: 1.25GHz
32,768 arithmetic units
Floating-point 16-bit

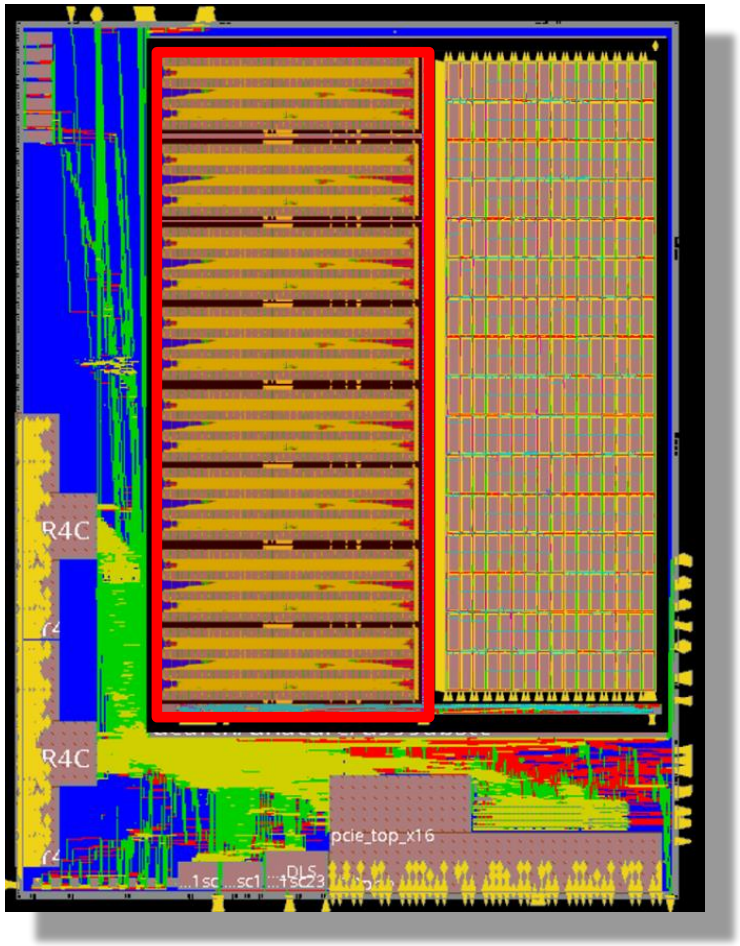
AB9 *from* ETRI



AB9

Artificial Brain NPU

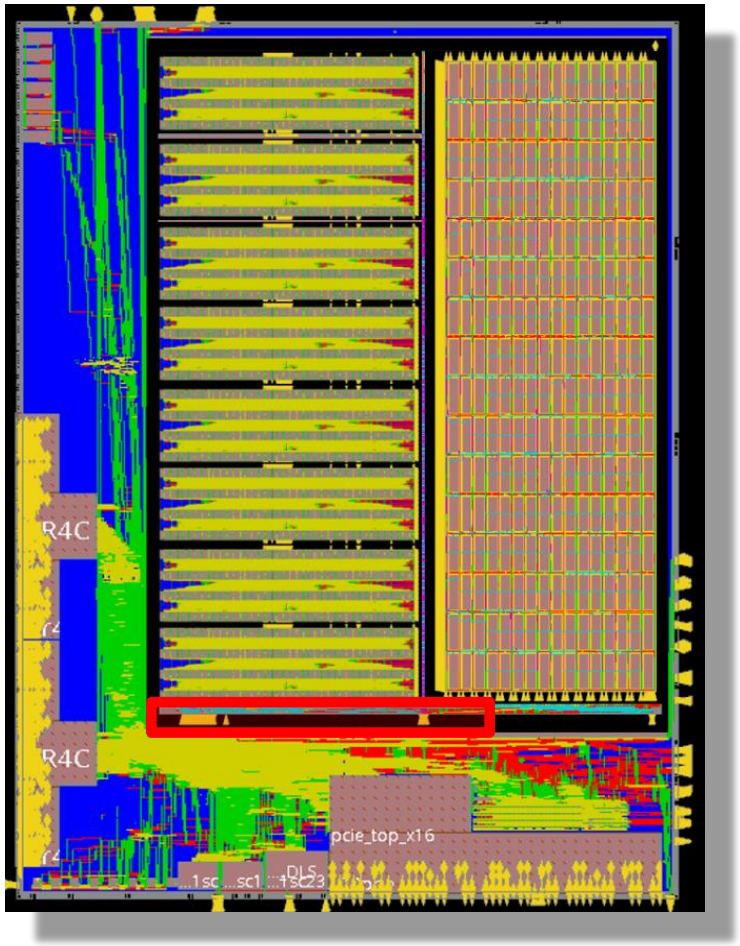
AB9 *from* ETRI



AB9

NPU Cache 40MB

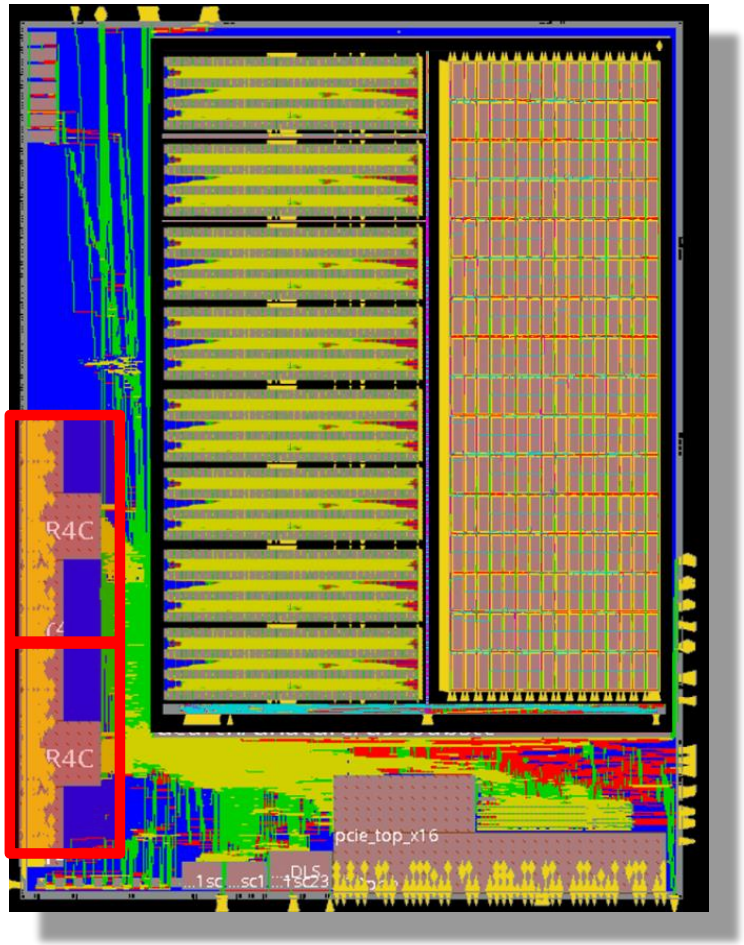
AB9 *from* ETRI



AB9

NPU ISA Core

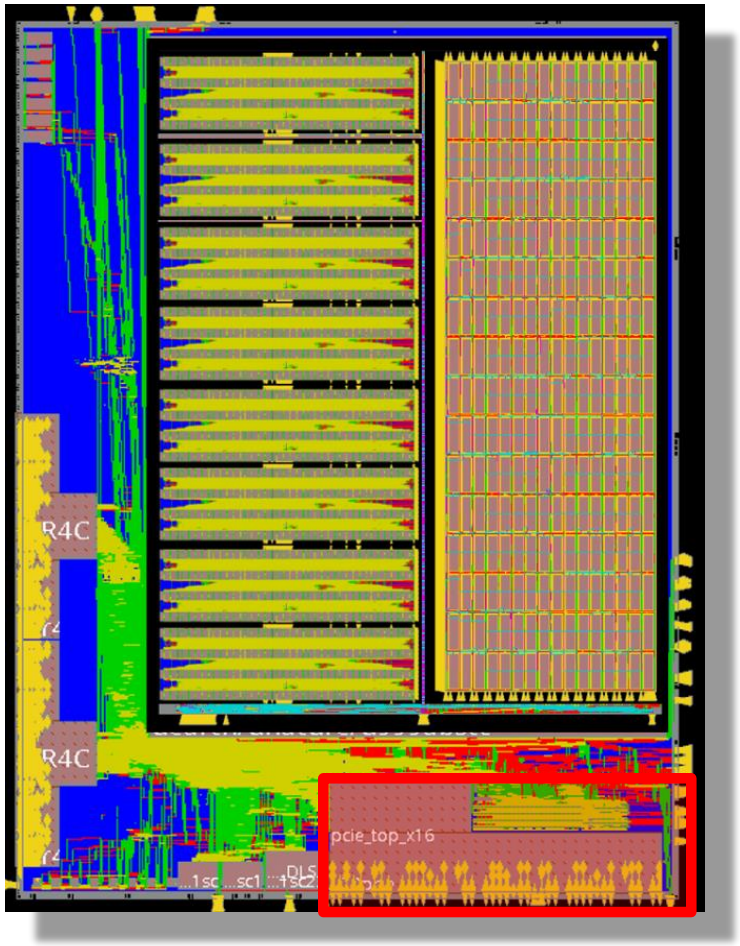
AB9 *from* ETRI



AB9

Dual-Channel
LPDDR4 2666 (1333MHz)

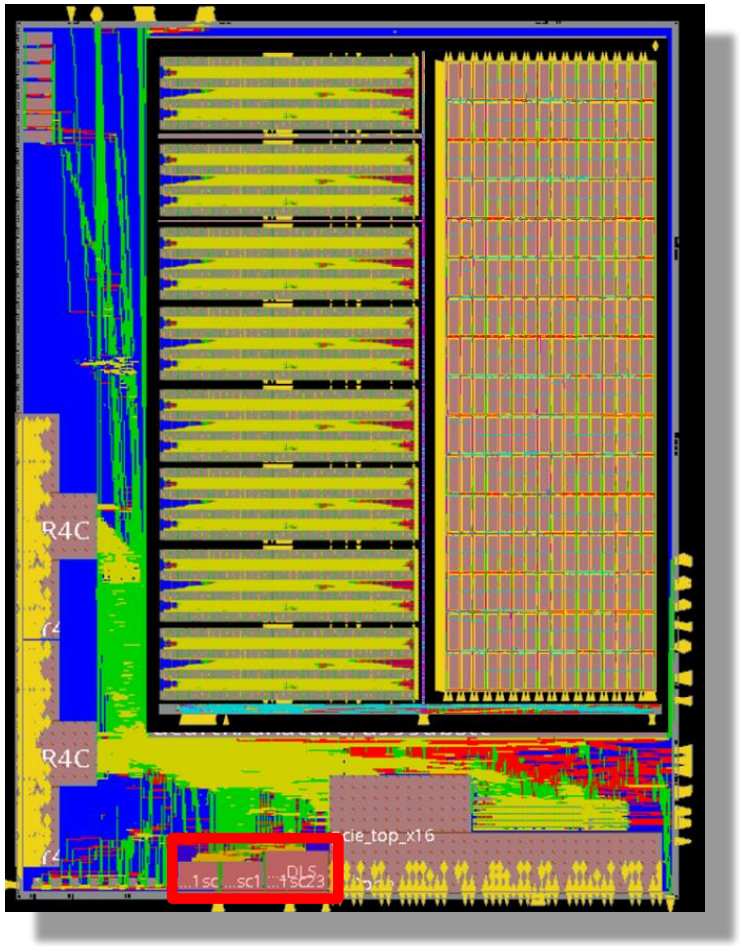
AB9 *from* ETRI



AB9

PCI Express 16 lanes

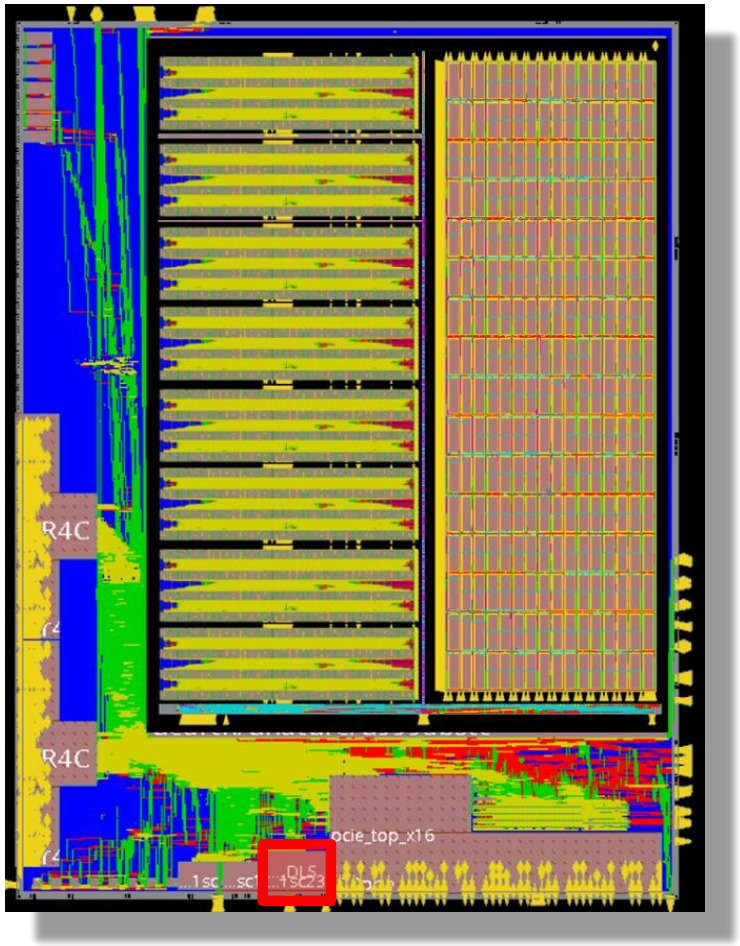
AB9 *from* ETRI



AB9

Quad 32b Cores

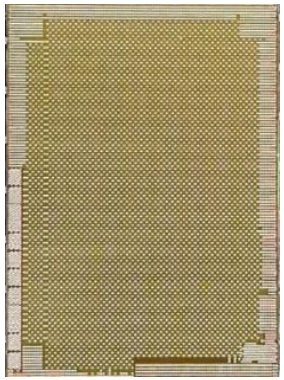
AB9 *from* ETRI



AB9

ISO 26262 Dynamic LockStep Core with ETRI-proprietary Functional Safety technology

AB9 *from* ETRI

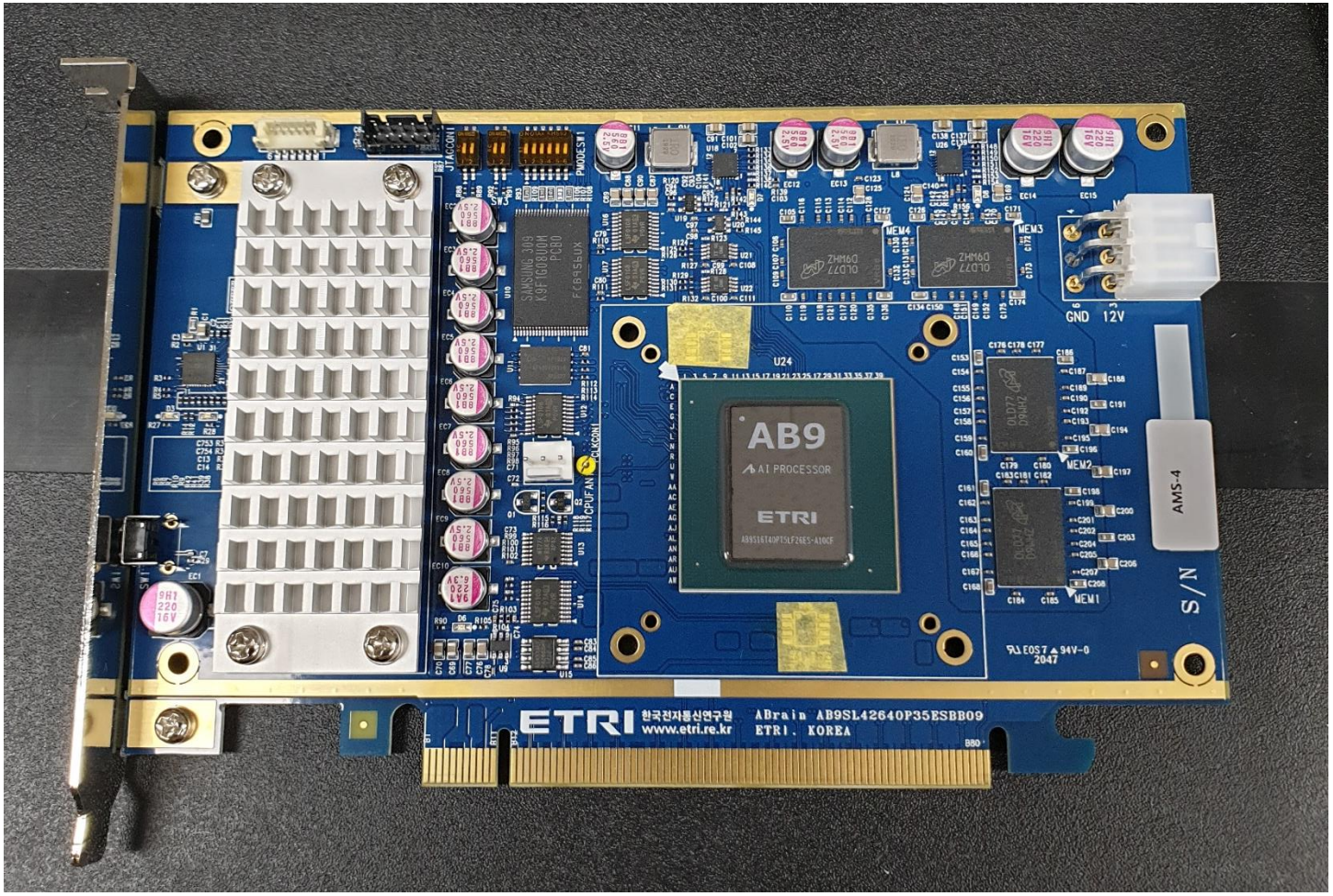


AB9 die

AB9 die on Package

AB9 Processor

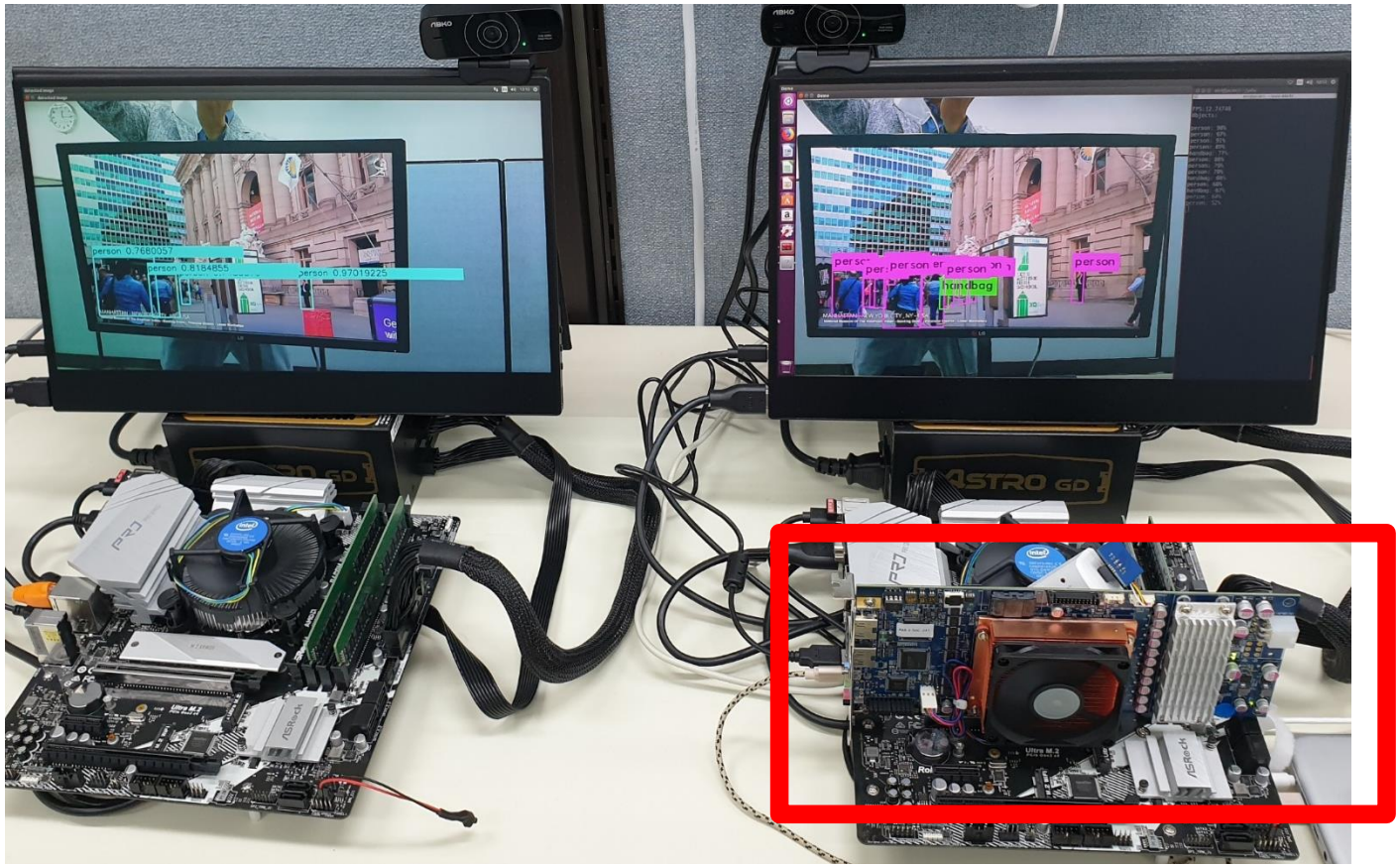
Abrain-S

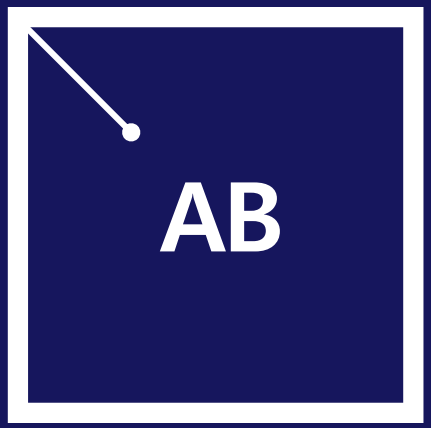


AB9 Single Chip Board

40TFLOPS, 15W

ABrain-S in Systems





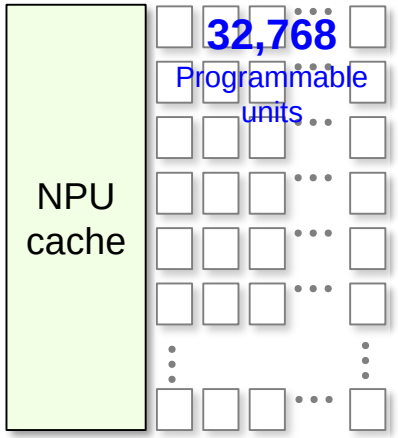
AB9 Internals

ETRI
미래사회를 만들어가는
국가 지능화 종합 연구기관

AB Technology

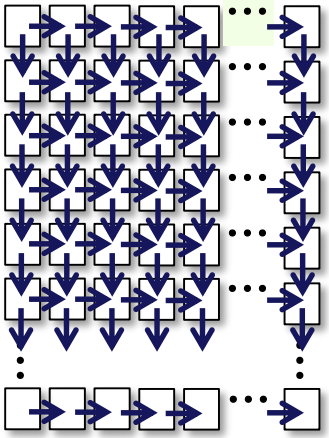
Artificial Brain Chip Technology from ETRI

ABNPU Architecture



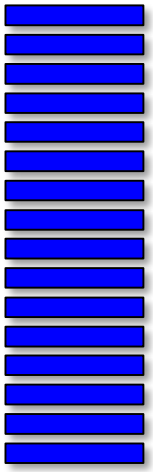
Super-Thread

16,384 threads



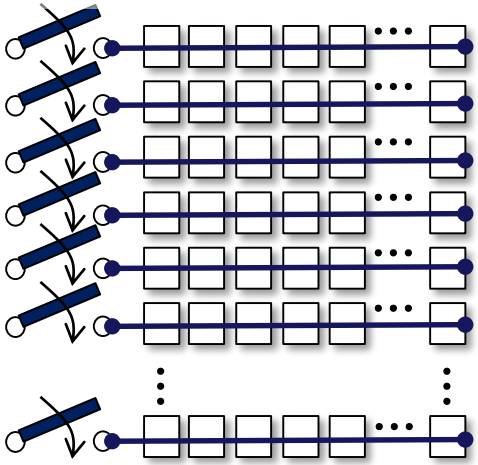
NPU Cache

40MB

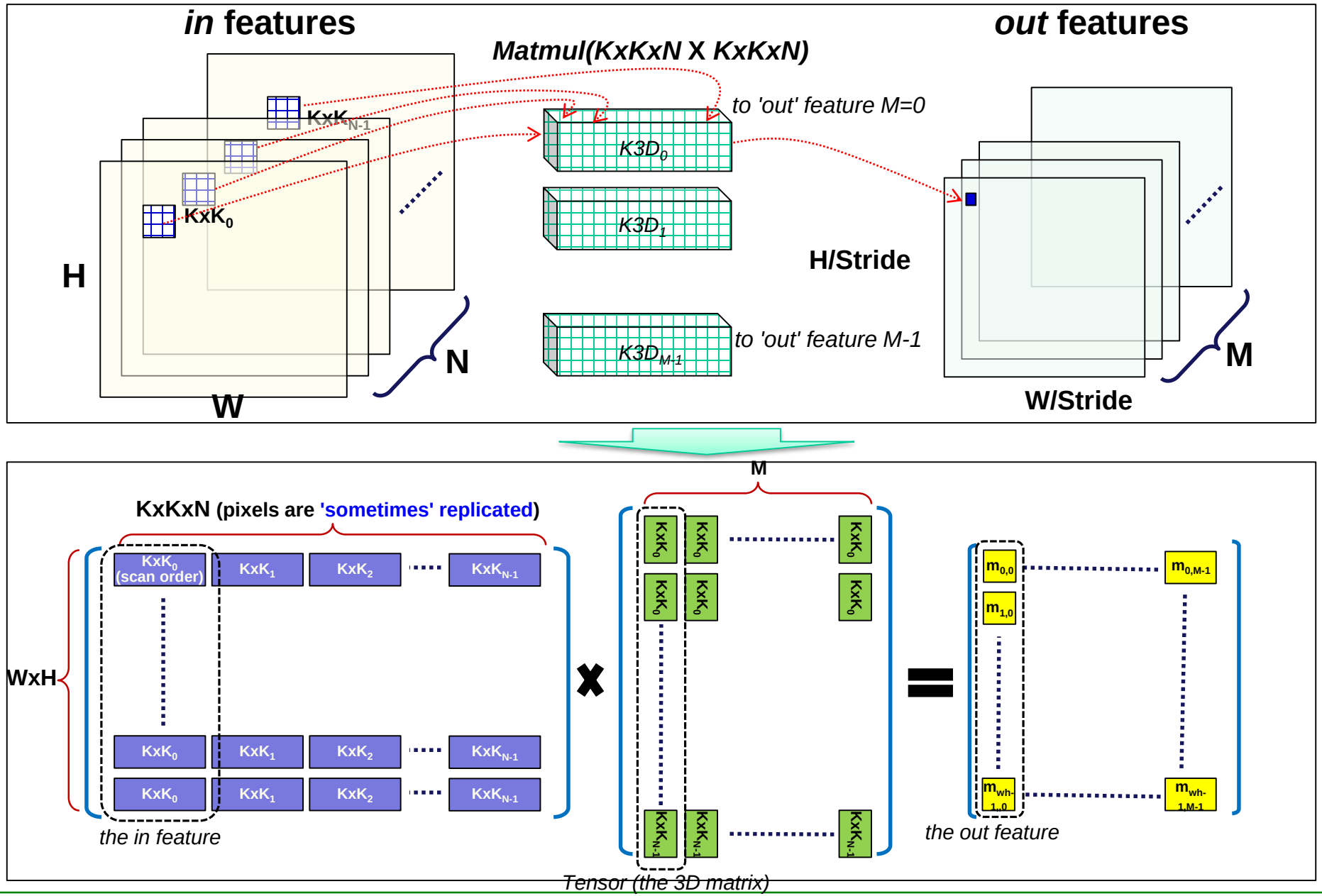


Power Planning plus NN Compiler

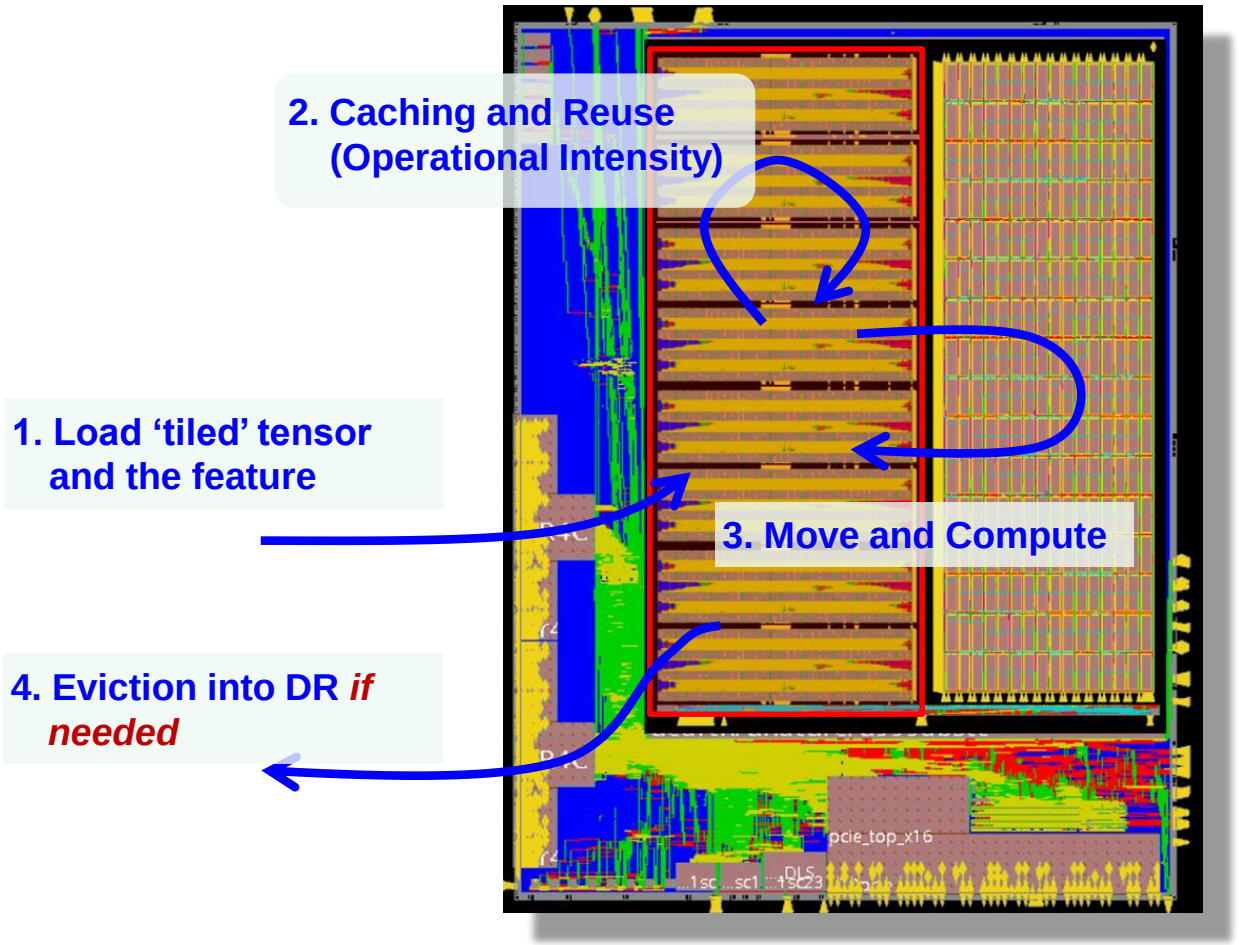
Micro-Power Gating

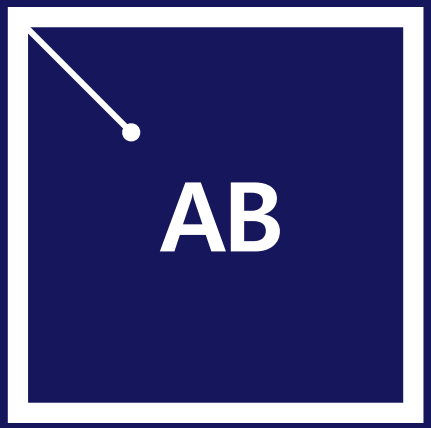


Actual Operation in the Chip



AB9 Operation Flow Simplified



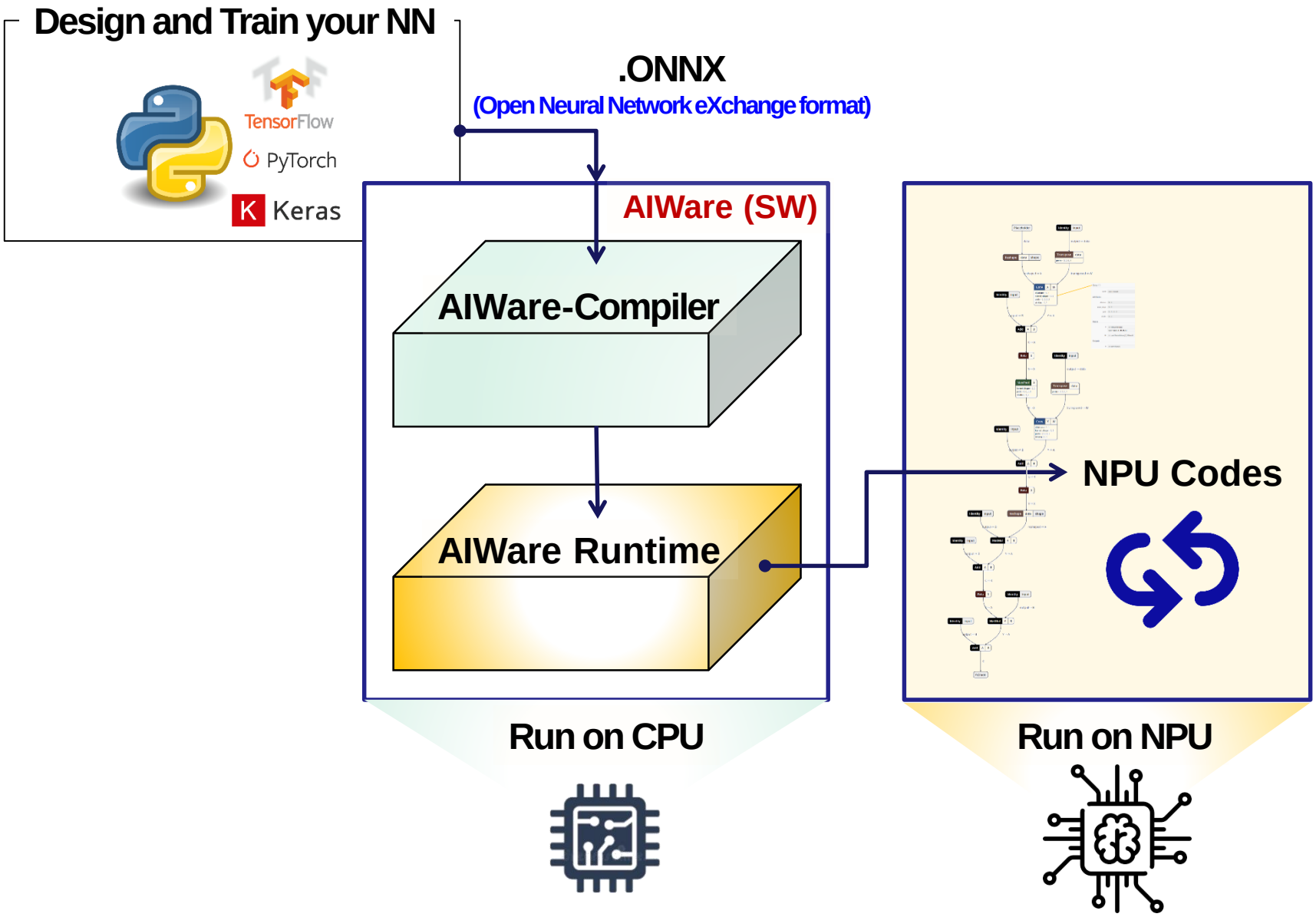


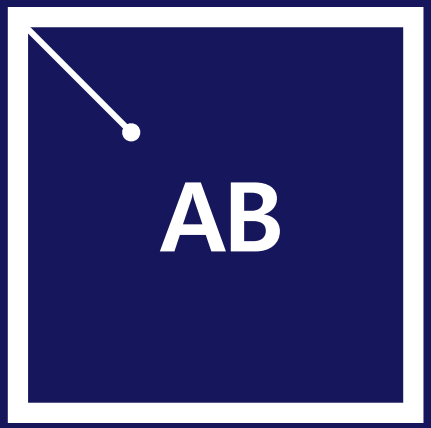
AB9 Programming

- Concept -

ETRI
미래사회를 만들어가는
국가 지능화 종합 연구기관

AB9 Programming Concept



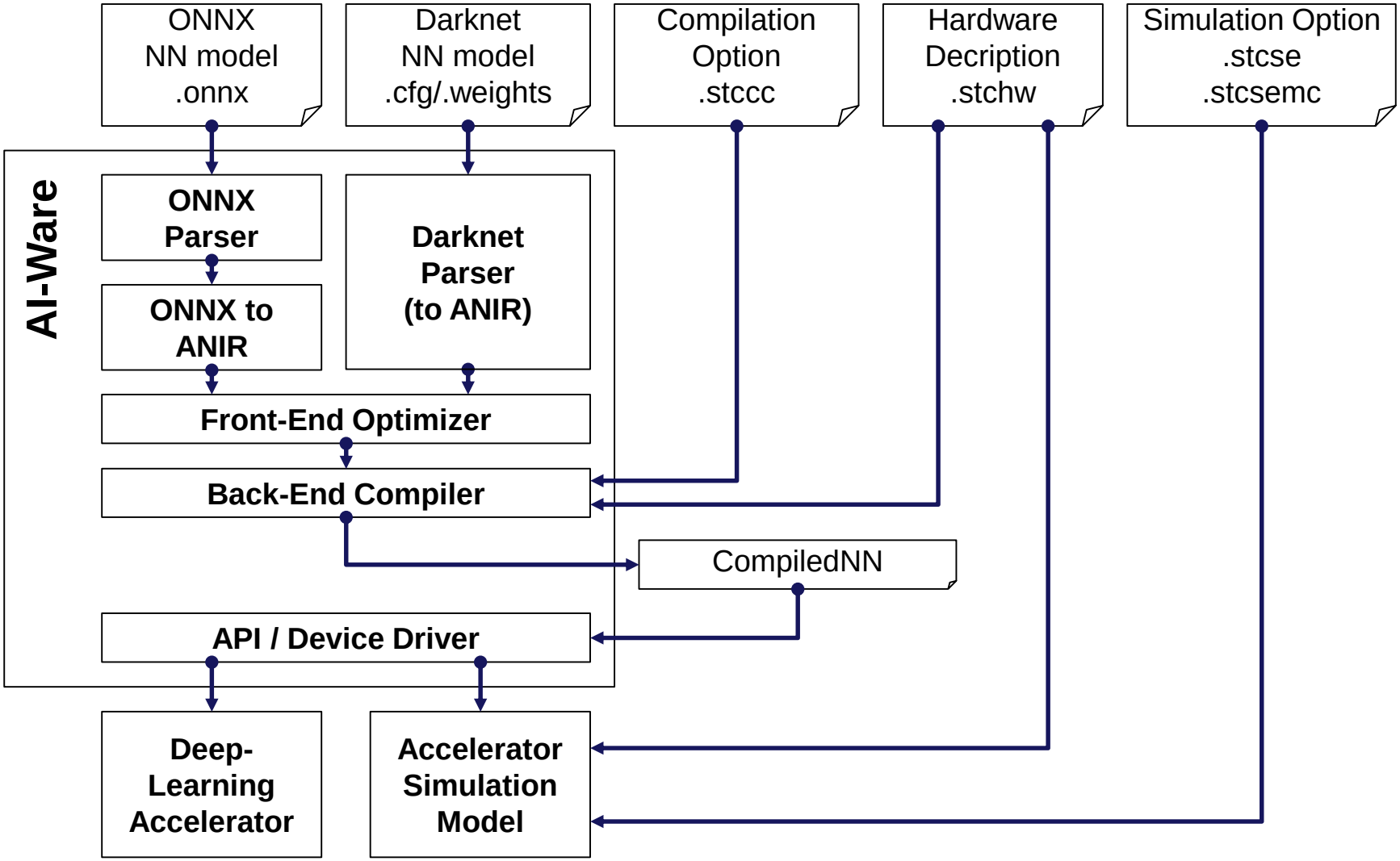


AB9 Programming

- AB9 AIWare API -

ETRI
미래사회를 만들어가는
국가 지능화 종합 연구기관

Architecture



Usage (Darknet NN Model)

```
STCCompiledNN * compiledNN = stcCompileDarknet
        (”yolov3.cfg”, ”yolov3.weights”, ”stcs32t.stchw”, ”bh128_bh1_ddr1.stccc”);


// stcStoreCompiledNN       (compiledNN, ”yolov3_compiledNN.h”);
// stcStoreCompiledNNbinary (compiledNN, ”yolov3_compiledNN.bin”);

// STCCompiledNN * compiledNN = stcLoadCompiledNN (”yolov3_compiledNN.h”);
// STCCompiledNN * compiledNN = stcLoadCompiledNNbinary (”yolov3_compiledNN.bin”);


STCHandle * handle = stcInitialize(0);
// STCHandle * handle = stcInitialize(”stcs32t.stchwmc”);  // for simulation model


stcConfigureStc(handle, 0, compiledNN);


float * inputData = (float *)malloc(sizeof(float) * 416 * 416 * 3);
stcSetInput(handle, 0, (void*)inputData, sizeof(float) * 416 * 416 * 3);


float * outputData = (float *)malloc(sizeof(float) * (13*13 + 26*26 + 52*52) * 255);
stcSetOutput(handle, 0, (void*)outputData, sizeof(float) * (13*13 + 26*26 + 52*52) * 255);


stcForwardNetwork(handle, 0);


stcClose(&handle);


stcReleaseCompiledNN(&compiledNN);
```

Usage (ONNX NN Model)

```
STCCompiledNN * compiledNN = stcCompileONNX
        (”yolov3-original.onnx”,”stcs32t.stchw”, ”bh128_bh1_ddr1.stccc”);


// stcStoreCompiledNN       (compiledNN, ”yolov3_compiledNN.h”);
// stcStoreCompiledNNbinary (compiledNN, ”yolov3_compiledNN.bin”);

// STCCompiledNN * compiledNN = stcLoadCompiledNN (”yolov3_compiledNN.h”);
// STCCompiledNN * compiledNN = stcLoadCompiledNNbinary (”yolov3_compiledNN.bin”);


STCHandle * handle = stcInitialize(0);
// STCHandle * handle = stcInitialize(”stcs32t.stchwmc”);  // for simulation model


stcConfigureStc(handle, 0, compiledNN);


float * inputData = (float *)malloc(sizeof(float) * 416 * 416 * 3);
stcSetInput(handle, 0, (void*)inputData, sizeof(float) * 416 * 416 * 3);


float * outputData = (float *)malloc(sizeof(float) * (13*13 + 26*26 + 52*52) * 255);
stcSetOutput(handle, 0, (void*)outputData, sizeof(float) * (13*13 + 26*26 + 52*52) * 255);


stcForwardNetwork(handle, 0);


stcClose(&handle);


stcReleaseCompiledNN(&compiledNN);
```

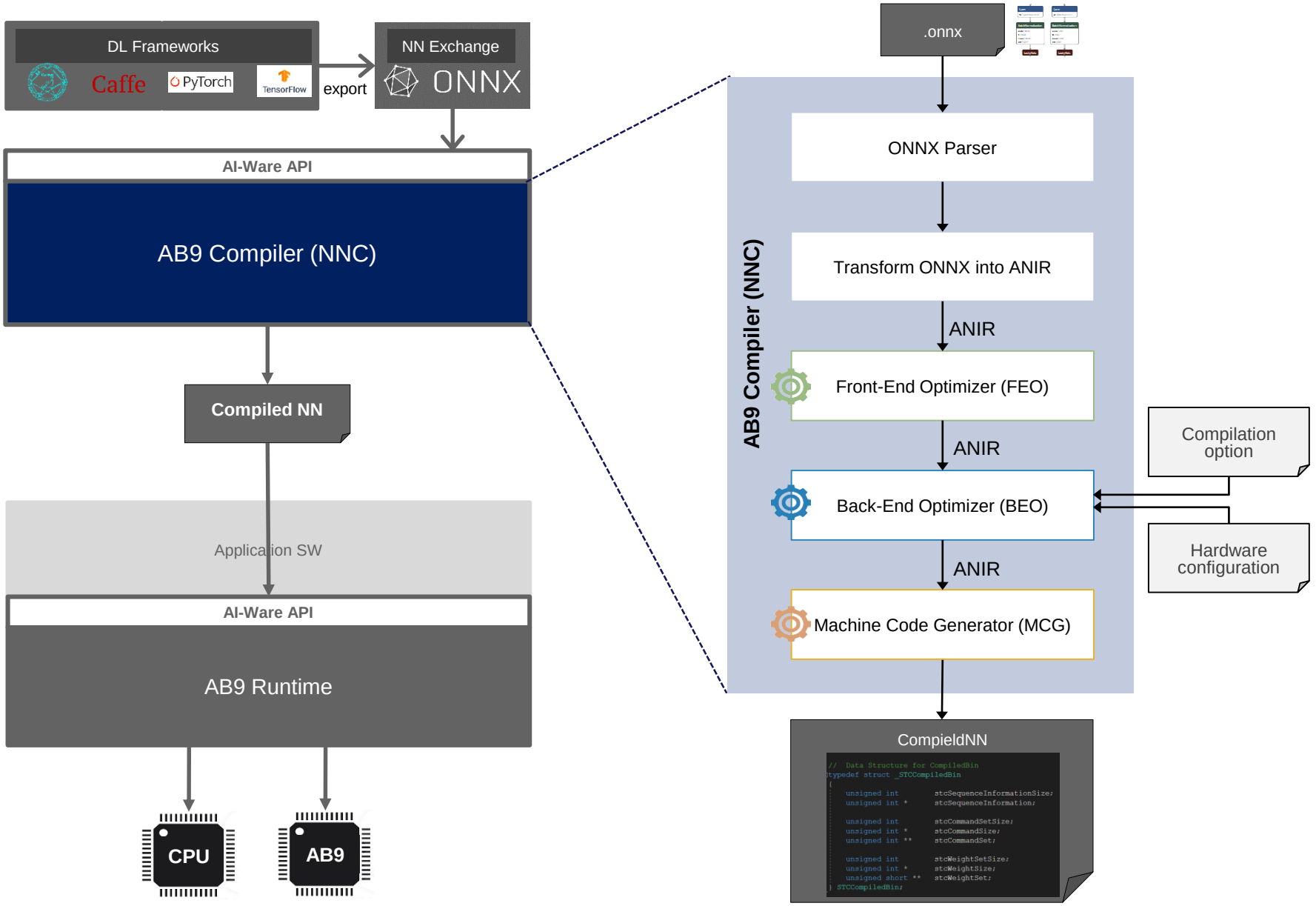
AB

AB9 Programming

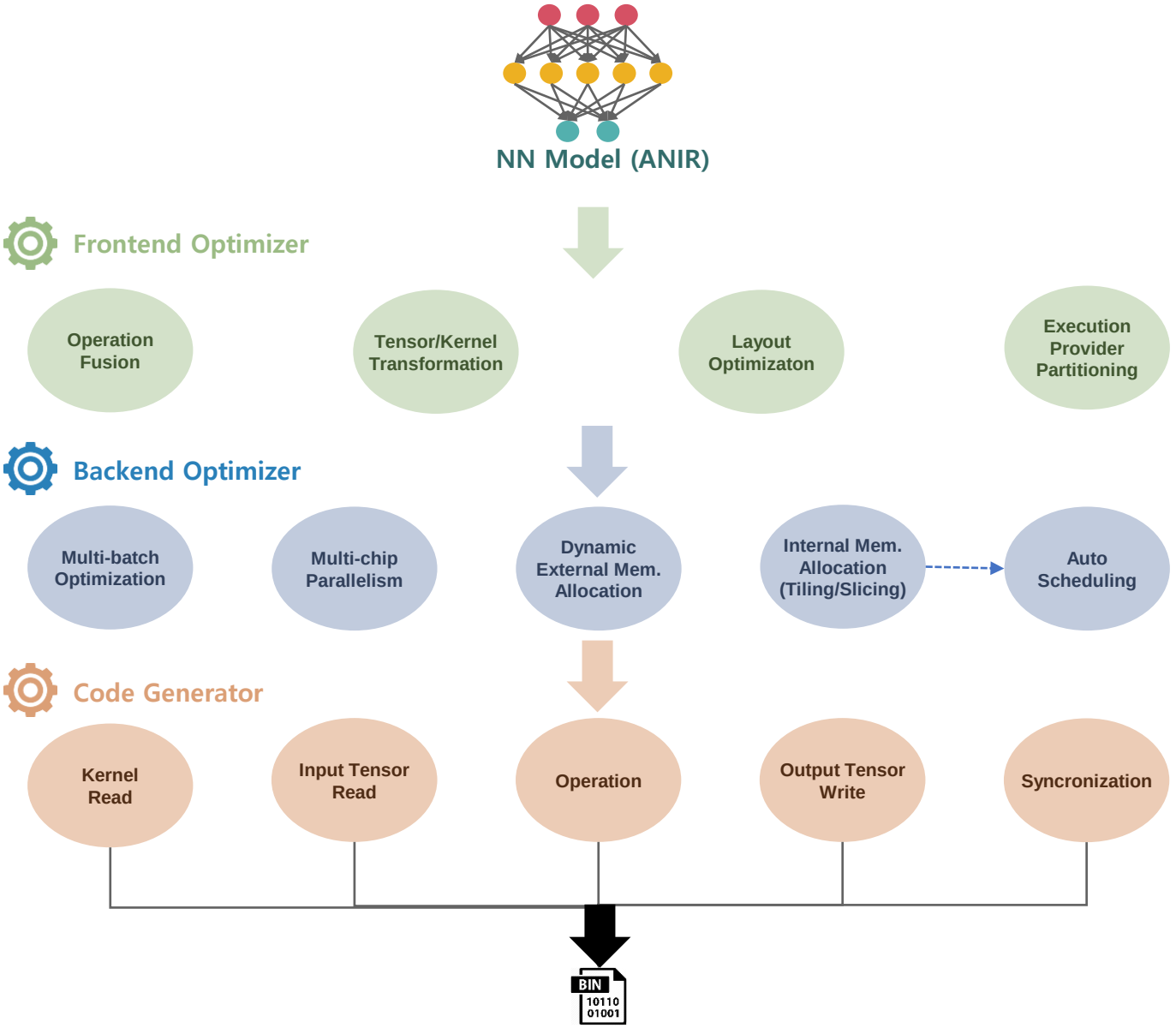
- AB9 NNC (NN Compiler) Internals -

ETRI
미래사회를 만들어가는
국가 지능화 종합 연구기관

AB9 NNC

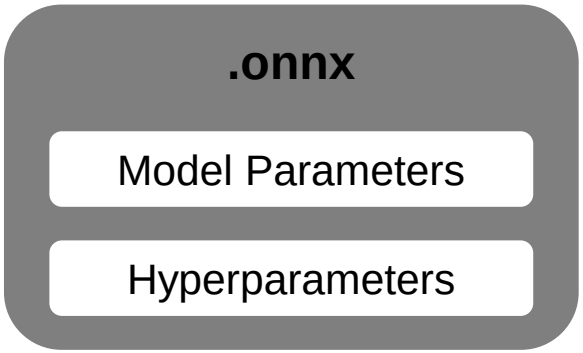

DL Frameworks
Caffe
PyTorch
TensorFlow
export
NN Exchange
ONNX
AI-Ware API
AB9 Compiler (NNC)
Compiled NN
Application SW
AI-Ware API
AB9 Runtime
CPU
AB9
.onnx
AB9 Compiler (NNC)
ONNX Parser
Transform ONNX into ANIR
ANIR
Front-End Optimizer (FEO)
ANIR
Back-End Optimizer (BEO)
ANIR
Machine Code Generator (MCG)
Compilation option
Hardware configuration
CompieldNN

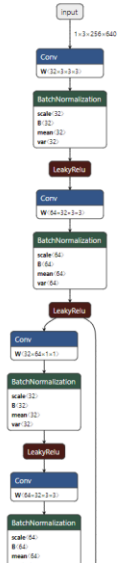
AB9 NNC Optimizers

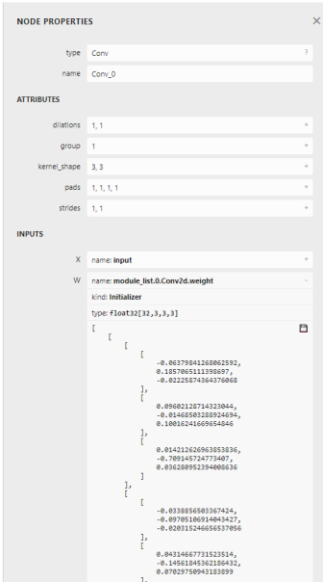


Interface between AI-Ware and NNC

❖ NNC Input: onnx model







❖ NNC Output: Executable binary & medadata

```
// Data Structure for CompiledNN
struct _STCCompiledNN
{

    unsigned int        stcSequnceInformationSize;
    unsigned int *      stcSequnceInformation;

    unsigned int        stcCommandSetSize;
    unsigned int *      stcCommandSize;
    unsigned int **     stcCommandSet;

    unsigned int        stcWeightSetSize;
    unsigned int *      stcWeightSize;
    unsigned short ** stcWeightSet;

} STCCompiledNN;
```

- NN Graph information
- AB9 Command information
- Weight infomation

AB9 NNC Execution

- **Prepare input files**
  - **Onnx model (.onnx)**
  - **AB9 Hardware Configuration (.stchw):** Provided by ETRI
  - **Compile Option (.stccc):** Provided by ETRI

- **AI-Ware API to execute NNC**

```
STCCompiledNN *compiledNN = stcCompileONNX("yolov3.onnx","ab9_config.stchw", "compile_option.stccc");
```

- **NNC standalone execution command**

1. **Execute NNC**
   - '-c' option: input files
   - '-d' option: output directory name
   - Output file: CompiledNN.h (Readable, C header format)

```
./stccc -c ./yolov3.onnx -c ./ab9_config.stchw -c ./compile_option.stccc -d ./compiledNN
```

2. **Convert NNC 'output file (compiledNN.h)' to 'binary files (*.bin)'**

```
./convertCompiledNNtoBM ./compiledNN/CompiledNN.h
          \./compiledNN/CompiledNN_Size.bin
          \./compiledNN/CompiledNN_Sequence.bin
          \./compiledNN/CompiledNN_Command.bin
          \./compiledNN/CompiledNN_Weight.bin
```

Print AB9 NNC Execution

```
elissa@hmkim:~/aldebaran/abts/abstc/stccc$ ./stccc -c ./cfg/semantic_segmentation.onnx -c ./cfg/stcs32t.stchw -c ./cfg/compile_option.stccc -d ./compiledNN/
/*****************************************************************************
*
* Copyright AI Processor Research Section, 2006-2021, All rights reserved.
* AI SoC Research Department, Artificial Intelligence Research Devision
* Electronics and Telecommunications Research Institute (ETRI)
*
* THESE DOCUMENTS CONTAIN CONFIDENTIAL INFORMATION AND KNOWLEDGE
* WHICH IS THE PROPERTY OF ETRI. NO PART OF THIS PUBLICATION IS
* TO BE USED FOR ANY OTHER PURPOSE, AND THESE ARE NOT TO BE
* REPRODUCED, COPIED, DISCLOSED, TRANSMITTED, STORED IN A RETRIEVAL
* SYSTEM OR TRANSLATED INTO ANY OTHER HUMAN OR COMPUTER LANGUAGE,
* IN ANY FORM, BY ANY MEANS, IN WHOLE OR IN PART, WITHOUT THE
* COMPLETE PRIOR WRITTEN PERMISSION OF ETRI.
*
*****************************************************************************/

========== STC COMMAND COMPILER (STCCC ver. 6.0) ===========

@ I: Start command compilation...
@ I: Finished to parse './cfg/stcs32t.stchw'
@ I: Finished to parse './cfg/compile_option.stccc'
[0/500] LAYER_INPUT : input.1 : 640 x 240 x 3
[1/500] LAYER_CONVPOOL
    INPUT X : input.1
    INPUT W : endecoder.layers_enc1_1.0.conv.weight
        (k_M) : 32
        (k_C) : 3
    OUTPUT(NAME) : 117
    ATTRIBUTES dilations : 1, 1
    ATTRIBUTES group : 1
    ATTRIBUTES kernel_shape : 3, 3
    ATTRIBUTES pads : 1, 1
    ATTRIBUTES strides : 2, 2
[2/500] LAYER_BATCHNORM
    INPUT X : 117
    INPUT SCALE : endecoder.layers_enc1_1.0.bn.weight
    INPUT B : endecoder.layers_enc1_1.0.bn.bias
    INPUT MEAN : endecoder.layers_enc1_1.0.bn.running_mean
    INPUT VAR : endecoder.layers_enc1_1.0.bn.running_var
    OUTPUT(NAME) : 118
[3/500] LAYER_RELU
    INPUT X : 118
    OUTPUT(NAME) : 119
    ACT_TYPE : Relu
    •
    •
    •

release modelProto done...
@ I: Number of Layers = 18
@ I: Decided Activation Tile of layer 10 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 7
@ I: Decided Activation Tile of layer 17 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 6
@ I: Decided Activation Tile of layer 24 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 0
@ I: Decided Activation Tile of layer 24 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 4
@ I: Decided Activation Tile of layer 24 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 2
@ I: Decided Activation Tile of layer 24 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 6
@ I: Decided Activation Tile of layer 24 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 1
@ I: Decided Activation Tile of layer 24 (numTileX = 3, numTileY = 1) , load wgt tile, pt_swticth 5
@ I: Decided Activation Tile of layer 24 (numTileX = 3, numTileY = 1) , load wgt tile, pt_swticth 3
@ I: Decided Activation Tile of layer 24 (numTileX = 3, numTileY = 1) , load wgt tile, pt_swticth 7
    •
    •
    •

    •
    •
    •
@ I: Decided Activation Tile of layer 52 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 7
@ I: Decided Activation Tile of layer 56 (numTileX = 2, numTileY = 1) , load wgt tile, pt_swticth 6
@ I: Set Power Gating 1
@ I: Set Power Gating 4
@ I: Set Power Gating 7
@ I: Set Power Gating 10
@ I: Set Power Gating 13
@ I: Set Power Gating 17
@ I: Set Power Gating 21
@ I: Set Power Gating 24
@ I: Set Power Gating 28
@ I: Set Power Gating 32
@ I: Set Power Gating 37
@ I: Set Power Gating 40
@ I: Set Power Gating 44
@ I: Set Power Gating 49
@ I: Set Power Gating 52
@ I: Set Power Gating 56
@ I: Set Power Gating 61
@ I: Calculate command size of layer 1
@ I: Calculate command size of layer 4
@ I: Calculate command size of layer 7
@ I: Calculate command size of layer 10
@ I: Calculate command size of layer 13
@ I: Calculate command size of layer 17
@ I: Calculate command size of layer 21
@ I: Calculate command size of layer 24
@ I: Calculate command size of layer 28
@ I: Calculate command size of layer 32
@ I: Calculate command size of layer 37
@ I: Calculate command size of layer 40
@ I: Calculate command size of layer 44
@ I: Calculate command size of layer 49
@ I: Calculate command size of layer 52
@ I: Calculate command size of layer 56
@ I: Calculate command size of layer 61
@ I: Write layer 1 command AUTO AIWARE
@ I: Write layer 4 command AUTO AIWARE
@ I: Write layer 7 command AUTO AIWARE
@ I: Write layer 10 command AUTO AIWARE
@ I: Write layer 13 command AUTO AIWARE
@ I: Write layer 17 command AUTO AIWARE
@ I: Write layer 21 command AUTO AIWARE
@ I: Write layer 24 command AUTO AIWARE
@ I: Write layer 28 command AUTO AIWARE
@ I: Write layer 32 command AUTO AIWARE
@ I: Write layer 37 command AUTO AIWARE
@ I: Write layer 40 command AUTO AIWARE
@ I: Write layer 44 command AUTO AIWARE
@ I: Write layer 49 command AUTO AIWARE
@ I: Write layer 52 command AUTO AIWARE
@ I: Write layer 56 command AUTO AIWARE
@ I: Write layer 61 command AUTO AIWARE
@ I: Write weights
@ I: Write layer 1 weight
@ I: Write layer 4 weight
@ I: Write layer 7 weight
@ I: Write layer 10 weight
@ I: Write layer 13 weight
@ I: Write layer 17 weight
@ I: Write layer 21 weight
@ I: Write layer 24 weight
@ I: Write layer 28 weight
@ I: Write layer 32 weight
@ I: Write layer 37 weight
@ I: Write layer 40 weight
@ I: Write layer 44 weight
@ I: Write layer 49 weight
@ I: Write layer 52 weight
@ I: Write layer 56 weight
@ I: Write layer 61 weight
@ I: Successfully finished compilation of stc command
```

Execution command

1. Read input files

2. Parsing ONNX transform into ANIR Frontend Optimizer

3. Backend Optimizer

4. Code Generation

AB9 HW Configuration file (.stchw)

- **Specify AB9 Hardware information including AB9 memory map**

NAME	stcs32t
ID	0x2182
SA_WIDTH	128
SA_HEIGHT	128
DCL_BLOCK_SIZE	0x40000
DCL_BANK_OFFSET	0x200000
DCL_SUBBLOCK_NUM	8
DCL_DIAGONAL_WRITE	YES
BASE_FC_SEQT	0xF0001000
BASE_FC_AG	0xF0003000
BASE_MM0W	0xF000A000
BASE_MM0R	0xF000B000
BASE_MM1	0xF000C000
BASE_MS0	0xF000E000
BASE_AU	0x01000000
BASE_DCL	0x00000000
MM_XADDR	0x00000000
MM_XGAP_H	0x00000004
MM_XGAP_C	0x00000008
MM_DADDR	0x0000000C
MM_DPADDR	0x00000010
MM_DGAP_H	0x00000014
MM_DGAP_B	0x00000018
MM_DGAP_C	0x0000001C
MM_DGAP_S	0x00000020
MM_LEN_W	0x00000024
MM_NUM_H	0x00000028
MM_NUM_BH	0x0000002C
MM_NUM_C	0x00000030
MM_PAD	0x00000034
MM_RUN	0x00000038

AG_II	0x00000000
AG_IJ	0x00000004
AG_IK	0x00000008
AG_IL	0x0000000C
AG_IM	0x00000010
AG_IN	0x00000014
AG_IO	0x00000018
AG_ISADDR	0x0000001C
AG_IPARAM1	0x00000020
AG_IPARAM2	0x00000024
AG_IPARAM3	0x00000028
AG_IPARAM4	0x0000002C
AG_IPARAM5	0x00000030
AG_OI	0x00000034
AG_OJ	0x00000038
AG_OK	0x0000003C
AG_OSADDR	0x00000040
AG_OPARAM1	0x00000044
AG_UI	0x00000048
AG_UJ	0x0000004C
AG_USADDR	0x00000050
AG_UPARAM1	0x00000054
NCSQT_BLK	0x00000000
NCSQT_SADDR	0x00001000
NCSQT_EADDR	0x00001004
NCSQT_CMDBLKRPT	0x00001008
SC_PRE_DELAY	1782
SC_ALL_DELAY	14812

NNC Compile Option file (.stccc)

CompileOption	
Batch_height	128
Batch_size	1
SeqInfo_address	0x90000000
STCcmd_address	0x98000000
Input_address0	0x100000000
Input_memsize	0x20000000
Output_address0	0x120000000
Output_memsize	0x20000000
Weight_address	0x210000000
AUtable_address	0x2F0000000
Hidden_address	0x140000000
Hidden_memsize	0x40000000
Input_datatype	HALF
Output_datatype	HALF
Weight_datatype	HALF
Input_endian	LITTLE
Output_endian	LITTLE
Weight_endian	LITTLE
ABCLayer_endian	BIG
Intermediate_endian	BIG
DC_alloc_automation	YES
DDR_alloc_automation	YES
MM_Speed_mode	YES
Smart_powergating	YES
Powergating_Off	NO
AXI_4Kbnd_constraint	NO
Disable_fuse_upsample	NO
Disable_fuse_shortcut	NO
Disable_fuse_yolo	NO
Merge_sequences	NO
Enable_ptm	YES
Disable_SAIO	NO
Disable_MM0D2X_debug	NO
EndCompileOption	

Batch information

Data Address information in DDR Memory

Data Type & Endian information

NNC Automation Functions

stcCommand Example (CompiledNN.h)

```
// STCCMD for layer 0 with pooling (layer_0_convolutional)
unsigned int stcCommand1[] = {
        0x100FB000, // FC_CMD_CWC, MM0 X2D Tile 0 batch 0
        0x40000000, // MM0 XADDR (offset: 0)
        0x000001C6, // MM0 XGAP_H (454 byte)
        0x00019292, // MM0 XGAP_C (103058 byte)
        0x00000000, // MM0 DADDR
        0x00000000, // MM0 DPADDR
        0x000001C8, // MM0 DGAP_H (456 byte)
        0x00200000, // MM0 DGAP_B
        0x00001560, // MM0 DGAP_C (5472 byte)
        0x00000000, // MM0 DGAP_S (subblk 0)
        0x000001C8, // MM0 LEN_W (456 byte)
        0x000000E4, // MM0 NUM_H (228)
        0x0000000C, // MM0 NUM_BH (12)
        0x00000003, // MM0 NUM_C (3)
        0x00000000, // MM0 PAD (left 0 byte, up 0 line)
        0x00001111, // MM0 RUN (AXI 4KBND Const 0, padl 0, padu 0, pad 0, endian 1, xaddr_extend 1)
        0x100FC000, // FC_CMD_CWC, MM1 X2D Tile 0
        0x00000000, // MM1 XADDR (offset: 0)
        0x000002DA, // MM1 XGAP_H (730 byte)
        0x00000000, // MM1 XGAP_C (0 byte)
        0x00000000, // MM1 DADDR
        0x00000000, // MM1 DPADDR
        0x00000000, // MM1 DGAP_H (0 byte)
        0x00200000, // MM1 DGAP_B
        0x00000000, // MM1 DGAP_C (0 byte)
        0x00010000, // MM1 DGAP_S (subblk 2)
        0x000002DA, // MM1 LEN_W (730 byte)
        0x00000060, // MM1 NUM_H (96)
        0x00000001, // MM1 NUM_BH (1)
        0x00000001, // MM1 NUM_C (1)
        0x00000000, // MM1 PAD (left 0 byte, up 0 line)
        0x00001021, // MM1 RUN (AXI 4KBND Const 0, padl 0, padu 0, pad 0, endian 0, xaddr_extend 2)
        0x10163000, // FC_CMD_CWC, AG Registers
        0x82AC0002, // AGI I_LOOP (iteration 3, offset 2736)
        0x0000400A, // AGI J_LOOP (iteration 11, offset 1)
        0x0039000A, // AGI K_LOOP (iteration 11, offset 228)
        0x00010002, // AGI L_LOOP (iteration 3, offset 4)
        0x00E40002, // AGI M_LOOP (iteration 3, offset 912)
        0x00030012, // AGI N_LOOP (iteration 19, offset 12)
        0x02AC0000, // AGI O_LOOP (iteration 1, offset 2736)
        0x00000000, // AGI Start addr. (subblk 0)
        0x001200E4, // AGI PARAM1 (batch size 1, actnum 19, slice width 228)
        0x007518F5, // AGI PARAM2 (boundary 0x0, batch height 29, component 0x118f5)
        0x000C000B, // AGI PARAM3 (lastlinenum 12, slice height 12)
        0x00000000, // AGI PARAM4 (padding size top 0, bottom 0, left 0, right 0)
        0x04040C0C, // AGI PARAM5 (stride y 4, stride x 4, down sampling y 12, down sampling x 12)
        0x8004C05F, // AGO I_LOOP (iteration 96, offset 19)
        0x00004012, // AGO J_LOOP (iteration 19, offset 1)
        0x0004C000, // AGO K_LOOP (iteration 1, offset 19)
        0x00004000, // AGO Start addr. (subblk 1)
        0x0002FFFF, // AGO PARAM1 (aumode 2, y component 0x3, y for diagonal write 0x3fff)
        0x0000416C, // AGU I_LOOP (iteration 365, offset 1)
        0x000000AA, // AGU J_LOOP (iteration 171, offset 0)
        0x00008000, // AGU Start addr. (subblk 2)
        0x0000005F, // AGU PARAM1 (unused 0, actnum 96)
```

Load input

Load weights

Set Address Generators

⋮

⋮

```
0x00016010, // NC CMD: First MAC (1)
0x016A6200, // NC CMD: MAC (362)
0x00012080, // NC CMD: Addition (st z0) (1)
0x00010008, // NC CMD: No Operation (1)
0x00012A64, // NC CMD: leaky ReLU + Max pool (st z1) (1)
0x00016010, // NC CMD: First MAC (1)
0x016A6200, // NC CMD: MAC (362)
0x00012080, // NC CMD: Addition (st z0) (1)
0x00010008, // NC CMD: No Operation (1)
0x00012A64, // NC CMD: leaky ReLU + Max pool (st z1) (1)
0x00016010, // NC CMD: First MAC (1)
0x016A6200, // NC CMD: MAC (362)
0x00012080, // NC CMD: Addition (st z0) (1)
0x00010008, // NC CMD: No Operation (1)
0x00012A64, // NC CMD: leaky ReLU + Max pool (st z1) (1)
0x00016010, // NC CMD: First MAC (1)
0x016A6200, // NC CMD: MAC (362)
0x00012080, // NC CMD: Addition (st z0) (1)
0x00010008, // NC CMD: No Operation (1)
0x00012A66, // NC CMD: leaky ReLU + Max pool (st z1) + Zout1 (1)
0x00010008, // NC CMD: No Operation (1)
0x00BE000B, // NC CMD: Output drain (190)
0x10032000, // FC_CMD_CWC, NC Seq. Table info.
0x00000000, // Start addr.
0x000000B0, // End addr.
0x00000001, // number of block repeat
0x20010005, // FCMD_SPROBE_ALL0 ( mm1b0_running mm0rb0_running )
0x30000001, // FCMD_SNCSEQ
0x200100E0, // FCMD_SPROBE_ALL0 ( sain_running saout_running ncseqt_running )
0x100FA000, // FC_CMD_CWC, MM0 D2X Tile 0 batch 0
0x4004B7B6, // MM0 XADDR (offset: 0)
0x00000036, // MM0 XGAP_H (54 byte)
0x000005B2, // MM0 XGAP_C (1458 byte)
0x00000000, // MM0 DADDR
0x00000000, // MM0 DPADDR
0x00000026, // MM0 DGAP_H (38 byte)
0x00200000, // MM0 DGAP_B
0x00000026, // MM0 DGAP_C (38 byte)
0x00008000, // MM0 DGAP_S (subblk 1)
0x00000026, // MM0 LEN_W (38 byte)
0x00000013, // MM0 NUM_H (19)
0x00000001, // MM0 NUM_BH (1)
0x00000060, // MM0 NUM_C (96)
0x00000000, // MM0 PAD (left 0 byte, up 0 line)
0x00011113, // MM0 RUN (AXI 4KBND Const 0, speed mode 1, padl 0, padu 0, pad 0, endian 1, xaddr_extend 1)
0x20010002, // FCMD_SPROBE_ALL0 ( mm0wb0_running )
0xF0000000  // FCMD_TERM
};
```

Set NC commands

Set NC Sequence

Run SA

Store output

Termination Interrupt