The Intel FPGA for AI - Architecture and Applications

Bogdan Pasca, Martin Langhammer, Eriko Nurvitadhi, Sergey Gribok

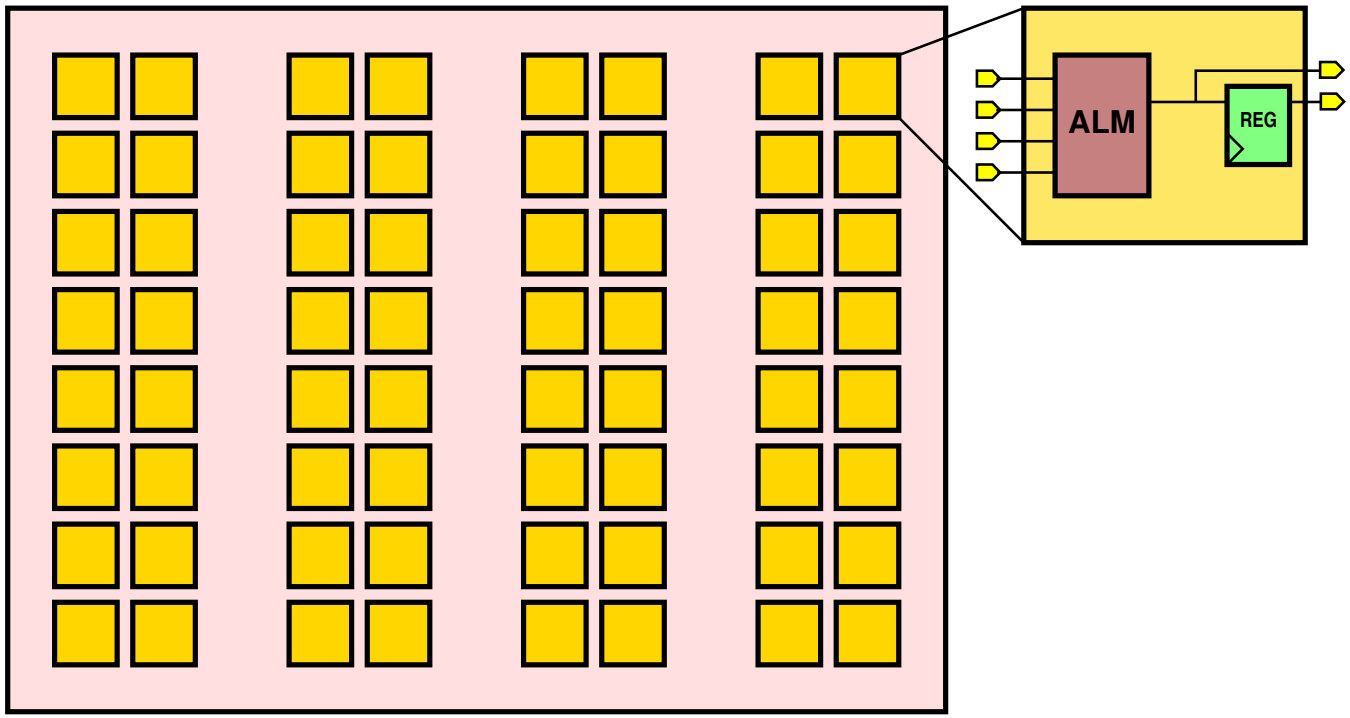
Intel Corporation

What's an FPGA again?

Field **P**rogrammable **G**ate **A**rray

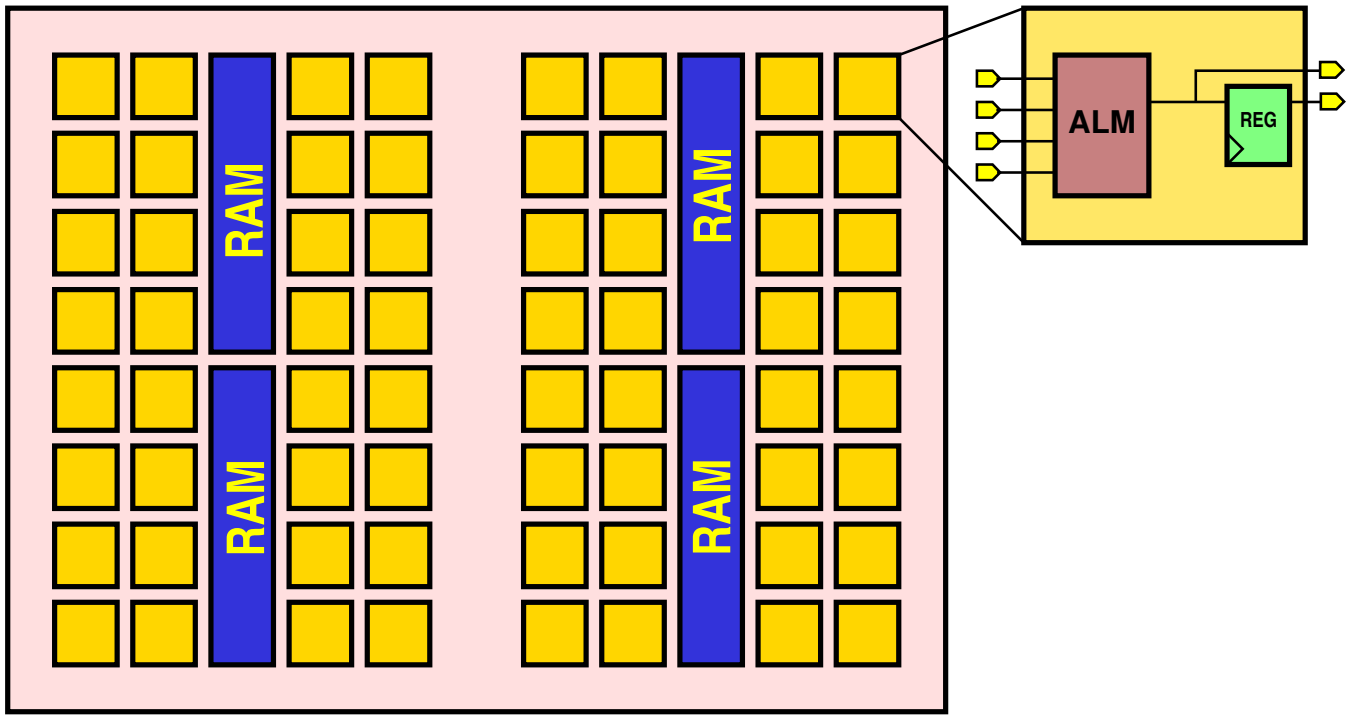
- integrated circuit
- has a regular architecture (hence **array**)
- logic elements can be programmed to perform various functions

Modern FPGA Architecture



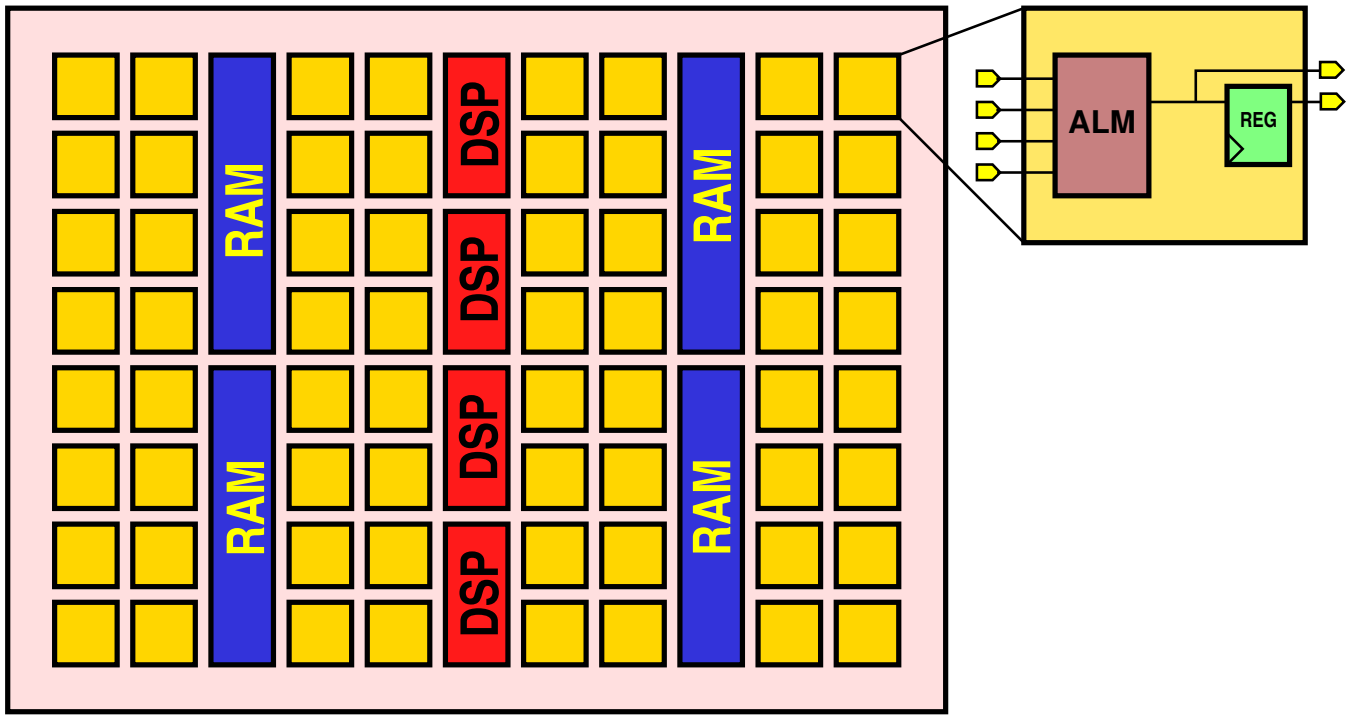
- a set of **configurable** logic elements

Modern FPGA Architecture



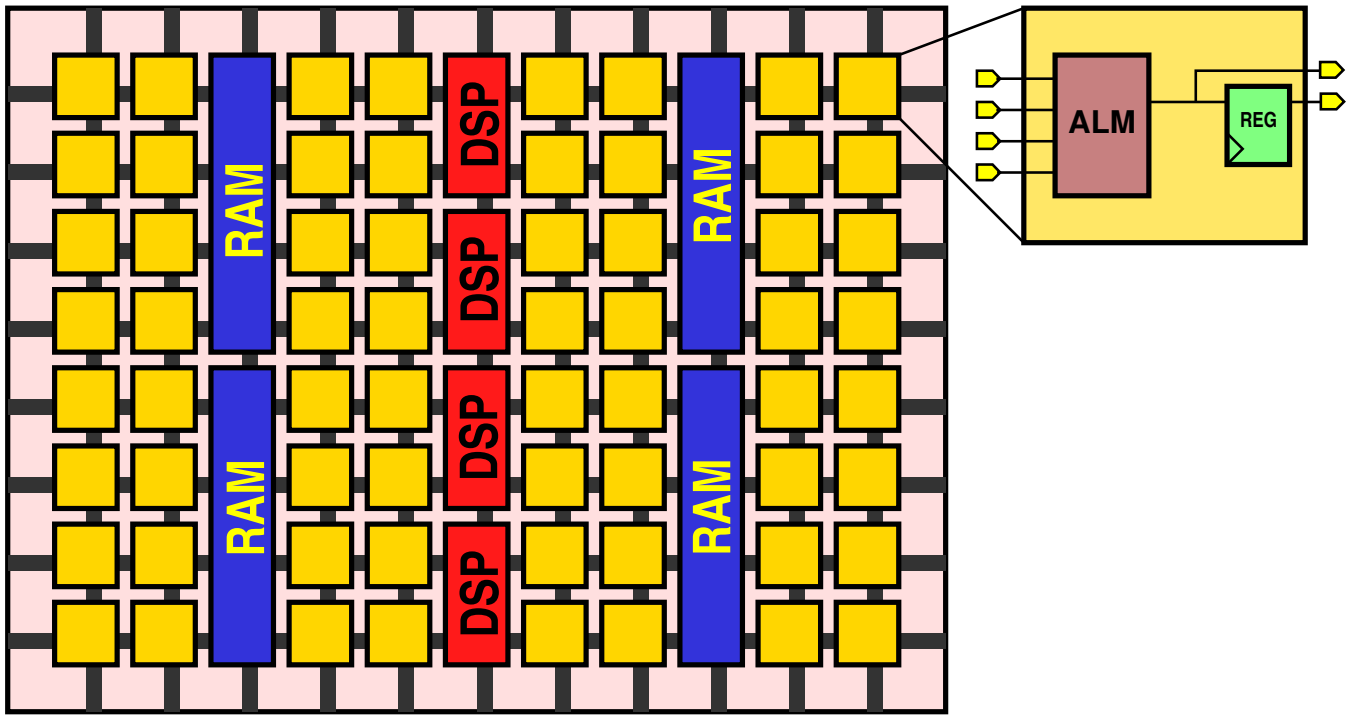
- a set of **configurable** logic elements
- on chip memory blocks

Modern FPGA Architecture



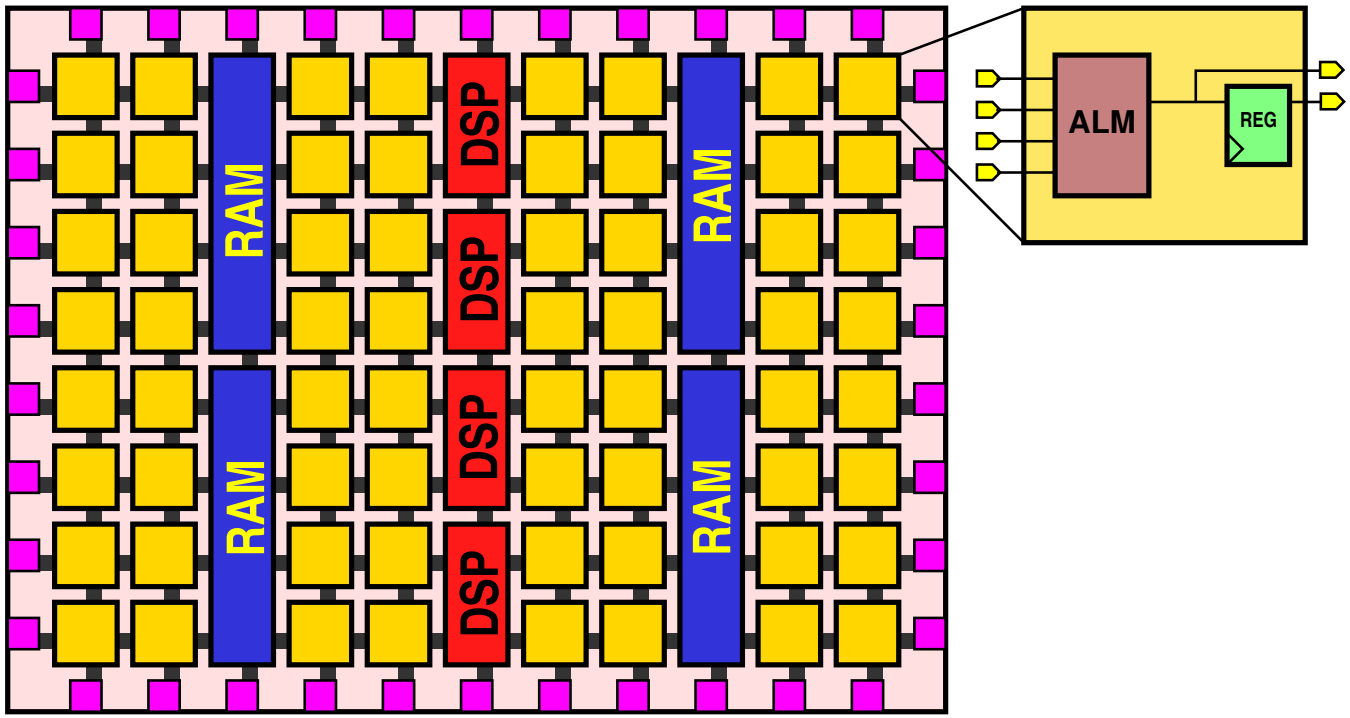
- a set of **configurable** logic elements
- on chip memory blocks
- digital signal processing (DSP) blocks (including multipliers)

Modern FPGA Architecture



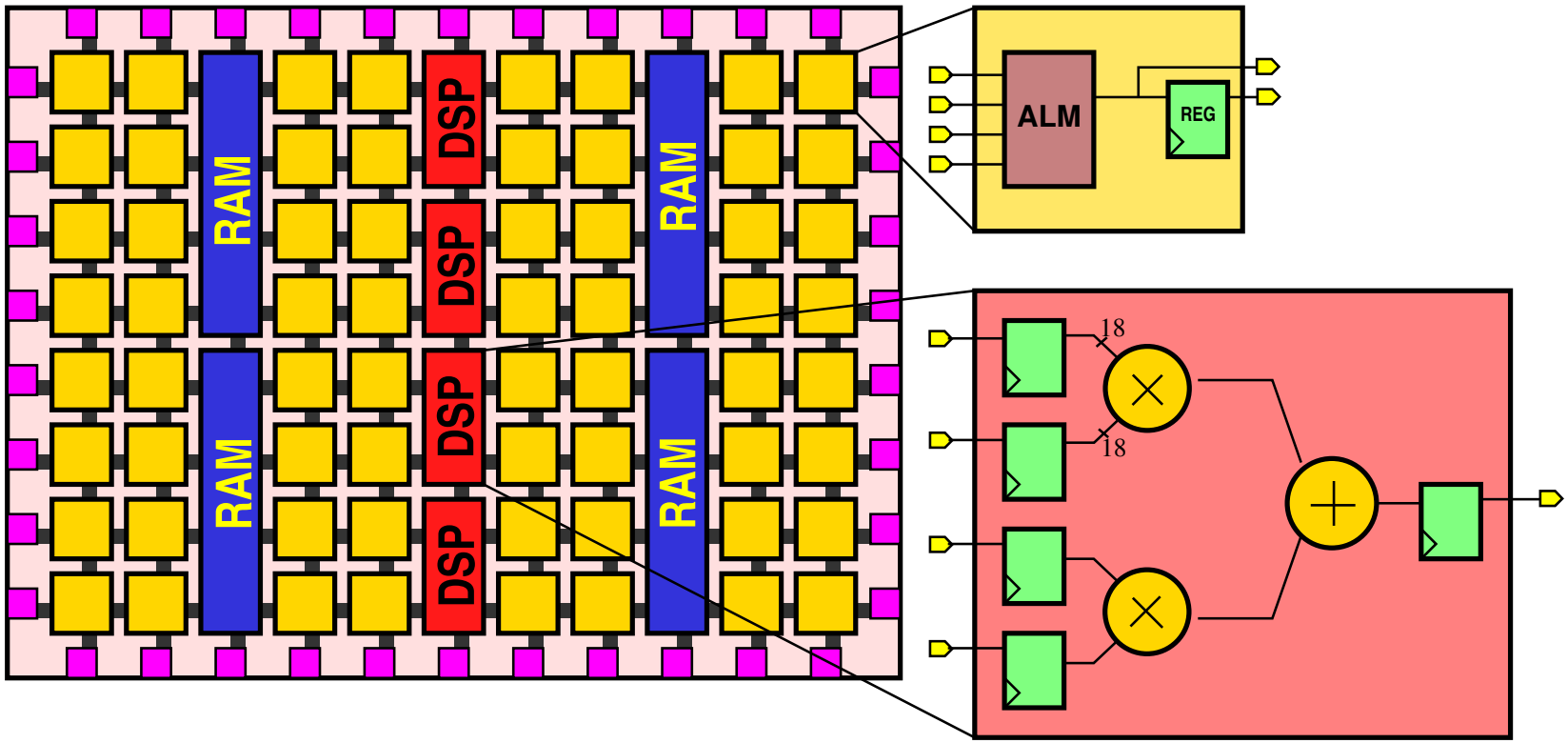
- a set of **configurable** logic elements
- on chip memory blocks
- digital signal processing (DSP) blocks (including multipliers)
- connected by a **configurable** wire network

Modern FPGA Architecture



- a set of **configurable** logic elements
- on chip memory blocks
- digital signal processing (DSP) blocks (including multipliers)
- connected by a **configurable** wire network
- all connected to outside world by I/O pins

Modern FPGA Architecture



- a set of **configurable** logic elements
- on chip memory blocks
- digital signal processing (DSP) blocks (including multipliers)
- connected by a **configurable** wire network
- all connected to outside world by I/O pins

Context

- AI compute requirements have outstripped FPGA arithmetic capabilities
- competitive neural network inference $\rightarrow$ INT8 or bfloat16
- new FPGA architectures match process-node evolution ($\approx$ 3 years)
- intercept market with competitive solution - modify an existing device.

Context

- AI compute requirements have outstripped FPGA arithmetic capabilities
- competitive neural network inference $\rightarrow$ INT8 or bfloat16
- new FPGA architectures match process-node evolution ($\approx$ 3 years)
- intercept market with competitive solution - modify an existing device.

Stratix 10 NX

- built on the production-qualified Stratix 10 MX platform
- using Intel 14nm FinFET process (yield & performance)
- enhance hard arithmetic capability
- swap general-purpose DSP Block with a new AI Tensor Block

A word on FP formats

Floating-point numbers represented in a floating-point format (wE, wF):

$$x = (-1)^s 2^e 1.Fx$$

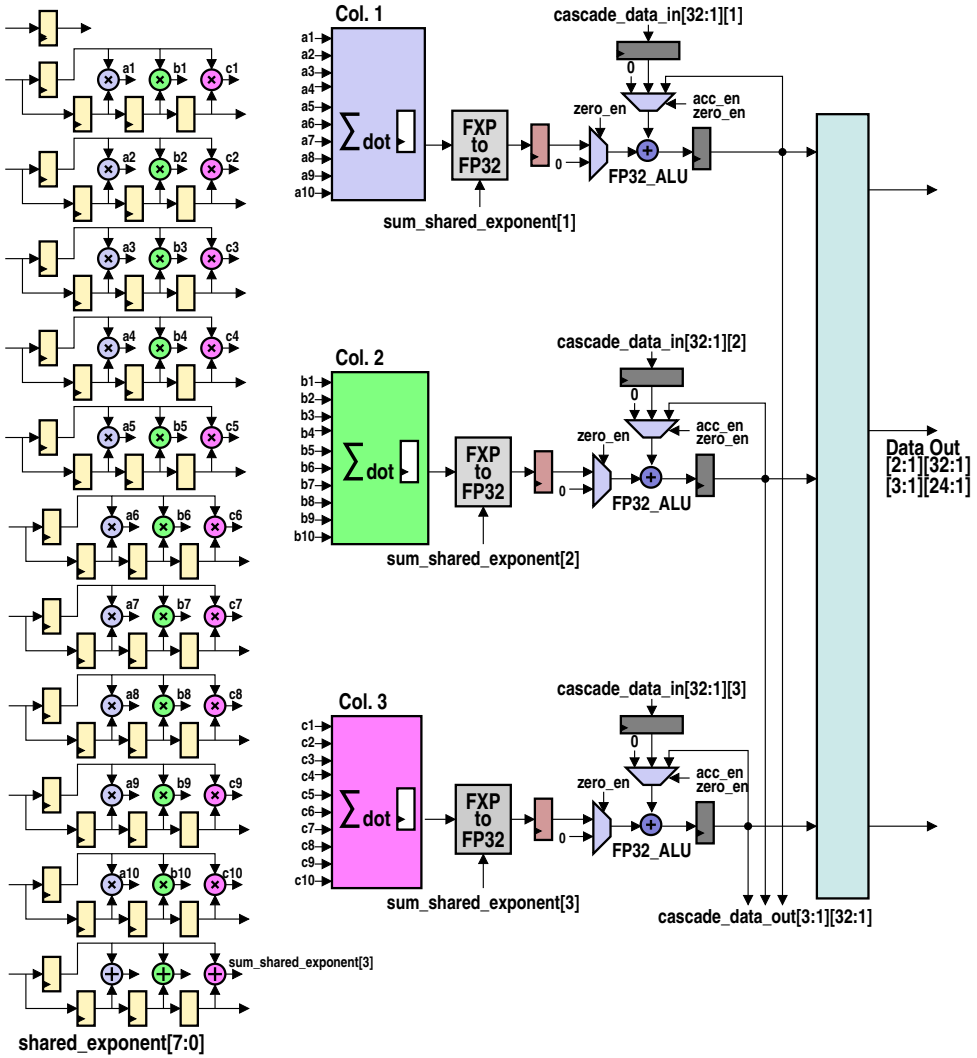
wE : exponent width (number of bits)
wF : fraction width

Format (wE, wF)	Name (IEEE-754)	Name (IEEE-754-2008)
(8,3)	*FP12*	*FP12*
(8,7)	*bfloat16*	*bfloat16*
(5,10)	half precision	binary16
(8,15)	*binary24*	*binary24*
(8,23)	single precision	binary32
(11,52)	double precision	binary64
(15,112)	quadruple precision	binary128

Block floating-point: one common exponent for a group of mantissas.

AI Tensor Block Architecture

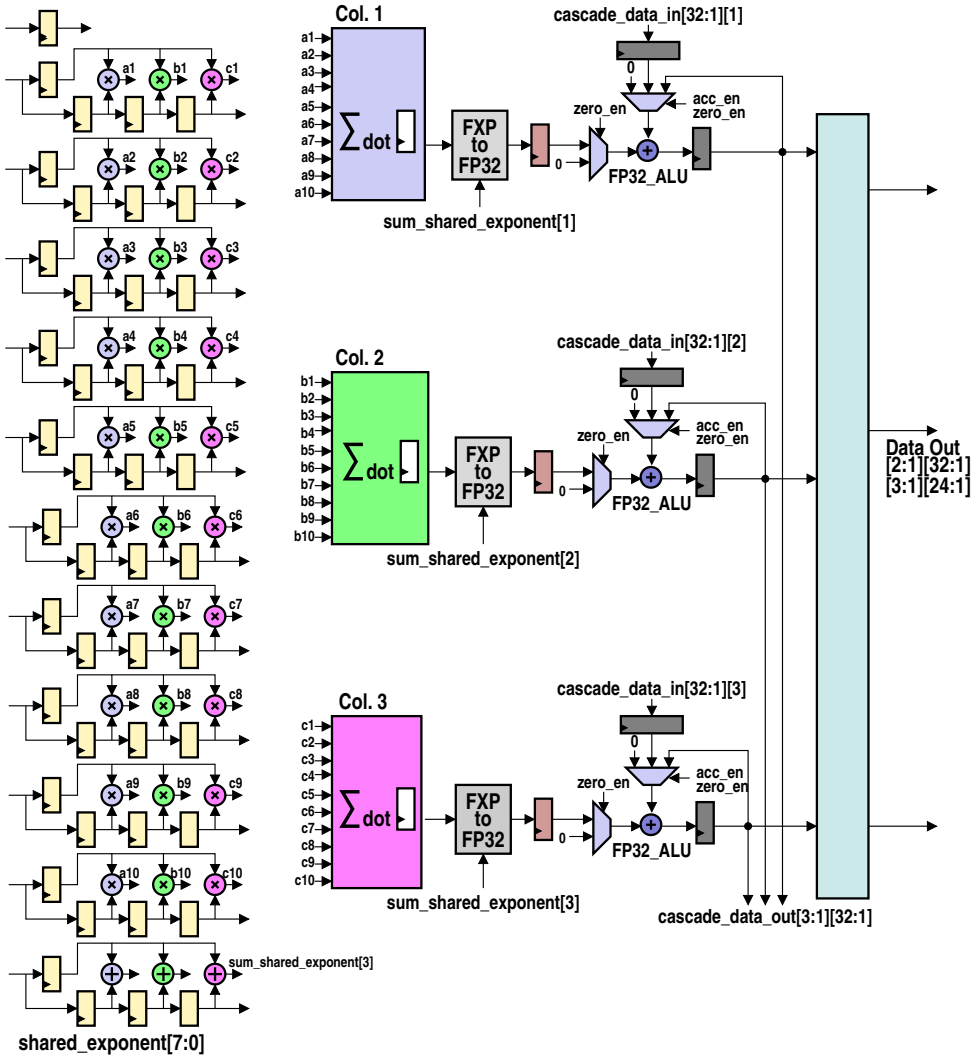
- **3 columns** of multipliers
  - **10 8-bit multipliers** or
  - **20 4-bit multipliers**
- **3 adder trees**
- **shared 8-bit exponent**
  - BlockFP12 or
  - BlockFP16
- accumulations
  - INT32 or
  - binary32.



Intel Corporation
INTEL PUBLIC June 9, 2022

AI Tensor Block Architecture

- **3 columns** of multipliers
  - **10 8-bit multipliers** or
  - **20 4-bit multipliers**
- **3 adder trees**
- **shared 8-bit exponent**
  - BlockFP12 or
  - BlockFP16
- accumulations
  - INT32 or
  - binary32.



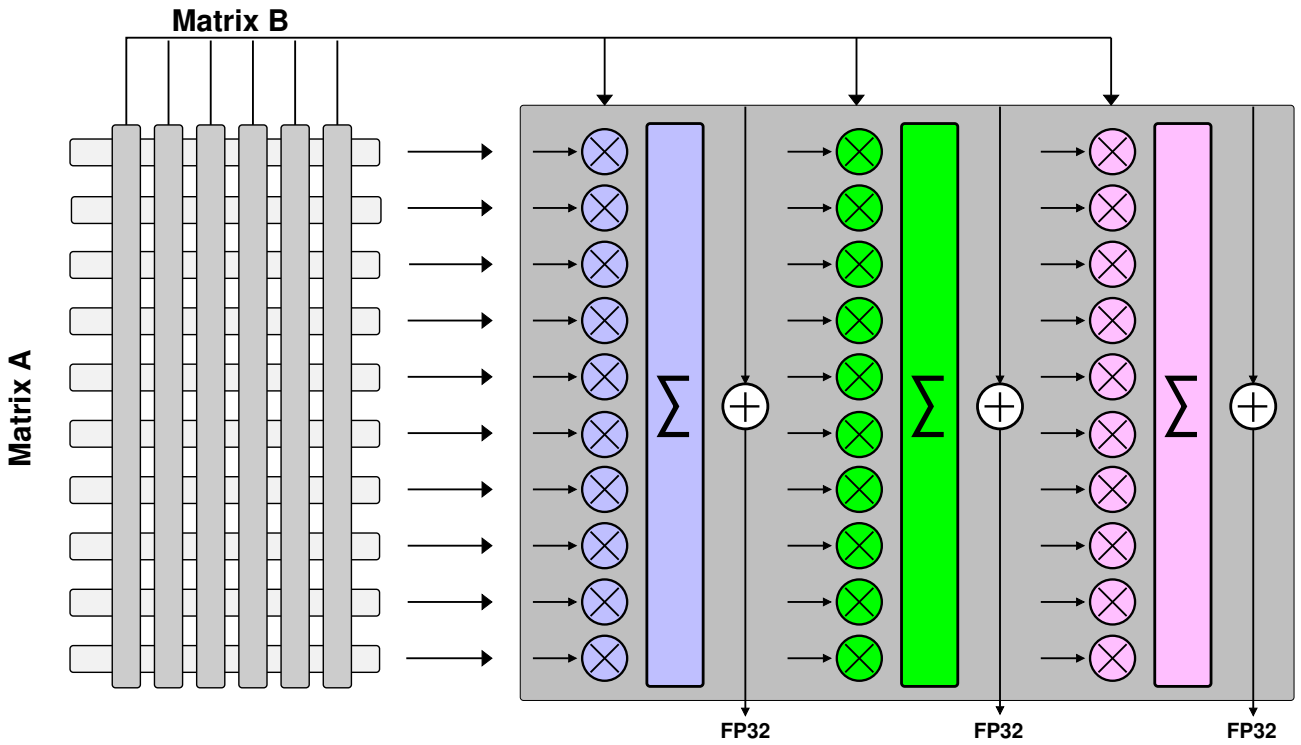
NX AI Tensor Block replaces a pair of 18-bit multipliers (DSP).

AI Tensor Block - Wire density

- Largest challenge - provide inputs to all the multipliers
  - 30 INT8 multipliers per block $\rightarrow$ 480 bits / cycle
- DSP Block interface: **104 data inputs** and **72 data outputs**.

AI Tensor Block - Wire density

- Largest challenge - provide inputs to all the multipliers
  - 30 INT8 multipliers per block $\rightarrow$ 480 bits / cycle
- DSP Block interface: **104 data inputs** and **72 data outputs**.
- AI Context - Convolutions
  - activations change often $\rightarrow$ 80-bit + 8-bit shared exponent
  - weights can be pre-loaded



Coefficient Loading

Parallel Load

- computation is stalled up to 3 cycles
- 3×10 coefficients provided via ”regular” input

Coefficient Loading

Parallel Load

- computation is stalled up to 3 cycles
- 3 $\times$ 10 coefficients provided via ”regular” input

Cascade Load

- up to C=36 AI Tensor blocks can be cascaded
- *first* block in chain only loads weights (88 bit)
- Cx3x2 cycles to load all coefficients

Coefficient Loading

Parallel Load

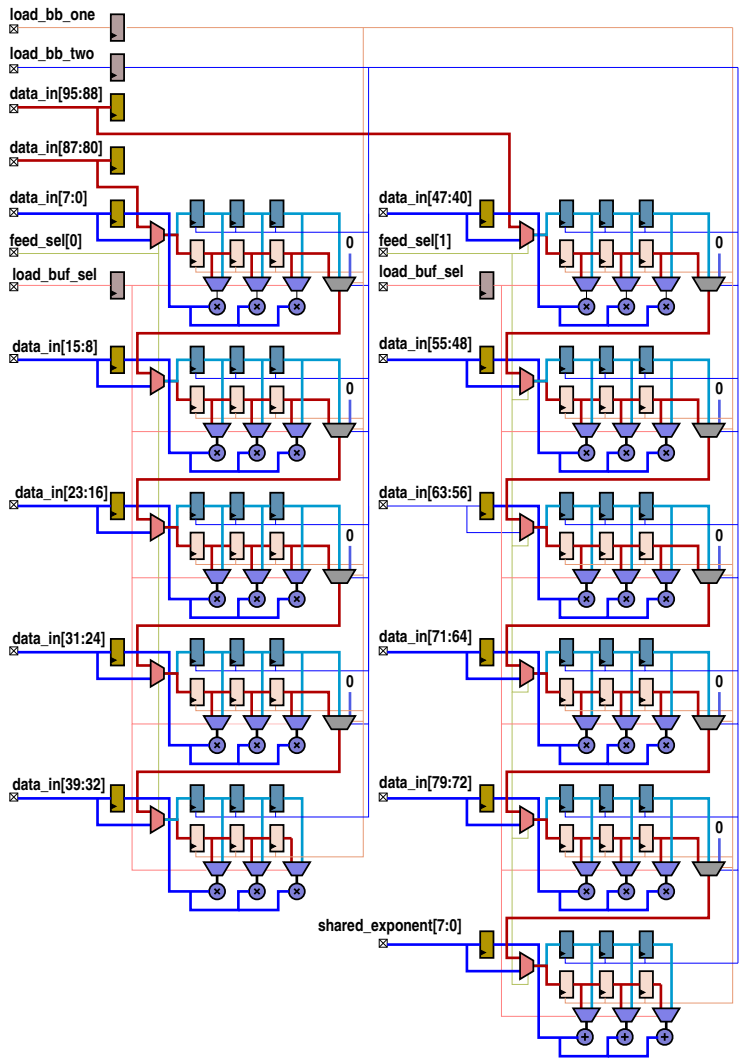
- computation is stalled up to 3 cycles
- 3 $\times$ 10 coefficients provided via ”regular” input

Cascade Load

- up to C=36 AI Tensor blocks can be cascaded
- *first* block in chain only loads weights (88 bit)
- Cx3x2 cycles to load all coefficients

Side Load

- two 8-bit ports are used to load the block
- 15 cycles required (18 cycles for shared exponent)



Modes

Tensor

- 30 INT8, 60 INT4 available
- only activations change every cycle
- 88 + 16 = 104 input wires consumed (side load).

Modes

Tensor

- 30 INT8, 60 INT4 available
- only activations change every cycle
- 88 + 16 = 104 input wires consumed (side load).

Vector

- multiplier inputs independently accessible
- one DOT6 (fixed-point) or DOT5 (FP) / AI Block
- 96 input wires are consumed

Modes

Tensor

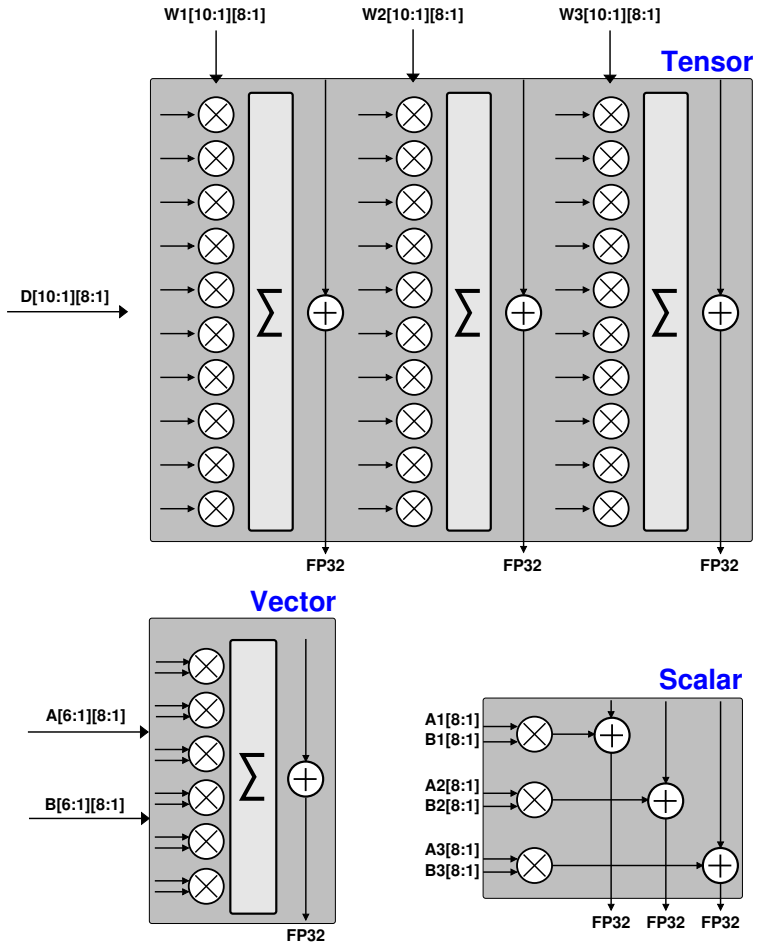
- 30 INT8, 60 INT4 available
- only activations change every cycle
- 88 + 16 = 104 input wires consumed (side load).

Vector

- multiplier inputs independently accessible
- one DOT6 (fixed-point) or DOT5 (FP) / AI Block
- 96 input wires are consumed

Scalar

- both inputs and outputs independently accessible
- 3 individual 8-bit multipliers / AI Block
- 3 independent bfloat16 Add/Sub / AI Block
- 2 independent bfloat24 Add/Sub / AI Block
- 1 independent binary32 Add/Sub / AI Block



Intel Corporation
INTEL PUBLIC June 9, 2022

Applications

Generating Larger Multipliers

Activation Functions

Generating Larger Multipliers

Let a, b be 16-bit signed to multiply.
Split a, b into 8-bit slices $\{a_1, a_0\}$ and $\{b_1, b_0\}$ such that:

$$a = a_1 2^8 + a_0$$
$$b = b_1 2^8 + b_0$$

The product ab requires four 8-bit-wide multiplications:

$$\begin{aligned} ab &= (a_1 2^8 + a_0)(b_1 2^8 + b_0) \\ &= a_1 b_1 2^{16} + (a_1 b_0 + a_0 b_1) 2^8 + a_0 b_0 \end{aligned}$$

$a_0 b_0$	: 8-bit unsigned multiplication
$a_1 b_0$, $a_0 b_1$	: 8-bit mixed-sign multiplication
$a_1 b_1$	: 8-bit signed multiplication.

Generating Larger Multipliers

Let a, b be 16-bit signed to multiply.
Split a, b into 8-bit slices $\{a_1, a_0\}$ and $\{b_1, b_0\}$ such that:

$$a = a_1 2^8 + a_0$$
$$b = b_1 2^8 + b_0$$

The product ab requires four 8-bit-wide multiplications:

$$\begin{aligned} ab &= (a_1 2^8 + a_0)(b_1 2^8 + b_0) \\ &= a_1 b_1 2^{16} + (a_1 b_0 + a_0 b_1) 2^8 + a_0 b_0 \end{aligned}$$

$a_0 b_0$	: 8-bit unsigned multiplication
$a_1 b_0$, $a_0 b_1$	: 8-bit mixed-sign multiplication
$a_1 b_1$	: 8-bit signed multiplication.

NX AI Tensor Block only supports 8-bit **signed** multiplications!

Sign-Byte-Pairs (Tuples)

Represent a, b using **pairs of signed 8-bit values** (a_1, a_0) and (b_1, b_0):

$$a = a_1 2^8 + a_0$$

$$b = b_1 2^8 + b_0$$

The product ab requires four signed 8-bit multiplications:

$$ab = a_1 b_1 2^{16} + (a_1 b_0 + a_0 b_1) 2^8 + a_0 b_0$$

Sign-Byte-Pairs (Tuples)

Represent a, b using **pairs of signed 8-bit values** (a_1, a_0) and (b_1, b_0):

$$a = a_1 2^8 + a_0$$

$$b = b_1 2^8 + b_0$$

The product ab requires four signed 8-bit multiplications:

$$ab = a_1 b_1 2^{16} + (a_1 b_0 + a_0 b_1) 2^8 + a_0 b_0$$

How to convert to SBP?

- start off with slices $\{a_1, a_0\}$
- if $a_0 \in [0, 127]$ then return (a_1, a_0)
- if $a_0 \in [128, 255]$ then return $(a_1 + 1, a_0 - 256)$

Sign-Byte-Pairs (Tuples)

Represent a, b using **pairs of signed 8-bit values** (a_1, a_0) and (b_1, b_0):

$$a = a_1 2^8 + a_0$$

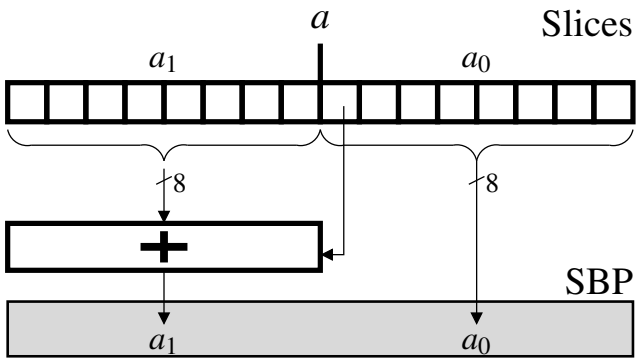
$$b = b_1 2^8 + b_0$$

The product ab requires four signed 8-bit multiplications:

$$ab = a_1 b_1 2^{16} + (a_1 b_0 + a_0 b_1) 2^8 + a_0 b_0$$

How to convert to SBP?

- start off with slices $\{a_1, a_0\}$
- if $a_0 \in [0, 127]$ then return (a_1, a_0)
- if $a_0 \in [128, 255]$ then return $(a_1 + 1, a_0 - 256)$



Range and Mapping

SPB Range is different than 2's complement

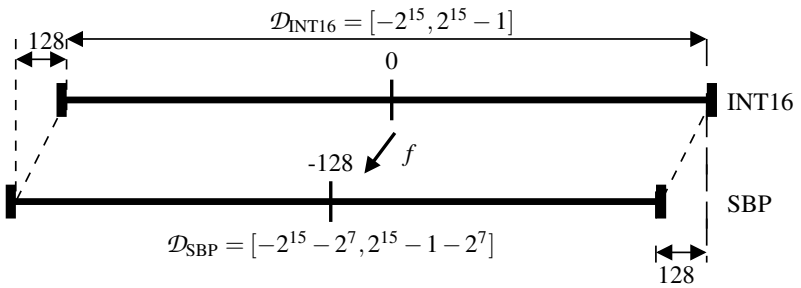
- 16-bit 2's complement: $[-2^{15}, 2^{15} - 1]$
- 16-bit SBP: $[-2^{15} - 2^{7}, 2^{15} - 2^{7} - 1]$

Range and Mapping

SPB Range is different than 2's complement

- 16-bit 2's complement: $[-2^{15}, 2^{15}-1]$
- 16-bit SBP: $[-2^{15}-2^{7}, 2^{15}-2^{7}-1]$

Cheap (1-bit not gate) mapping function $\rightarrow$ convert to SBP and offset

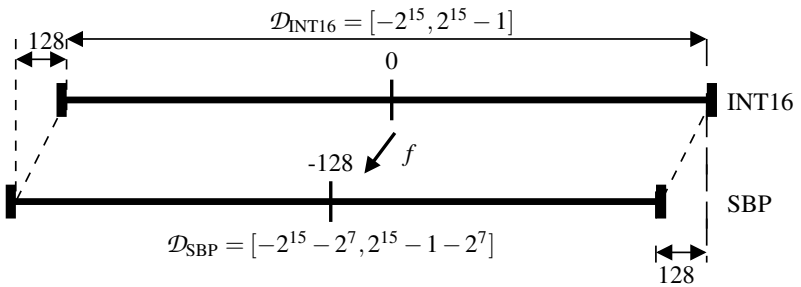


Range and Mapping

SPB Range is different than 2's complement

- 16-bit 2's complement: $[-2^{15}, 2^{15}-1]$
- 16-bit SBP: $[-2^{15}-2^{7}, 2^{15}-2^{7}-1]$

Cheap (1-bit not gate) mapping function $\rightarrow$ convert to SBP and offset



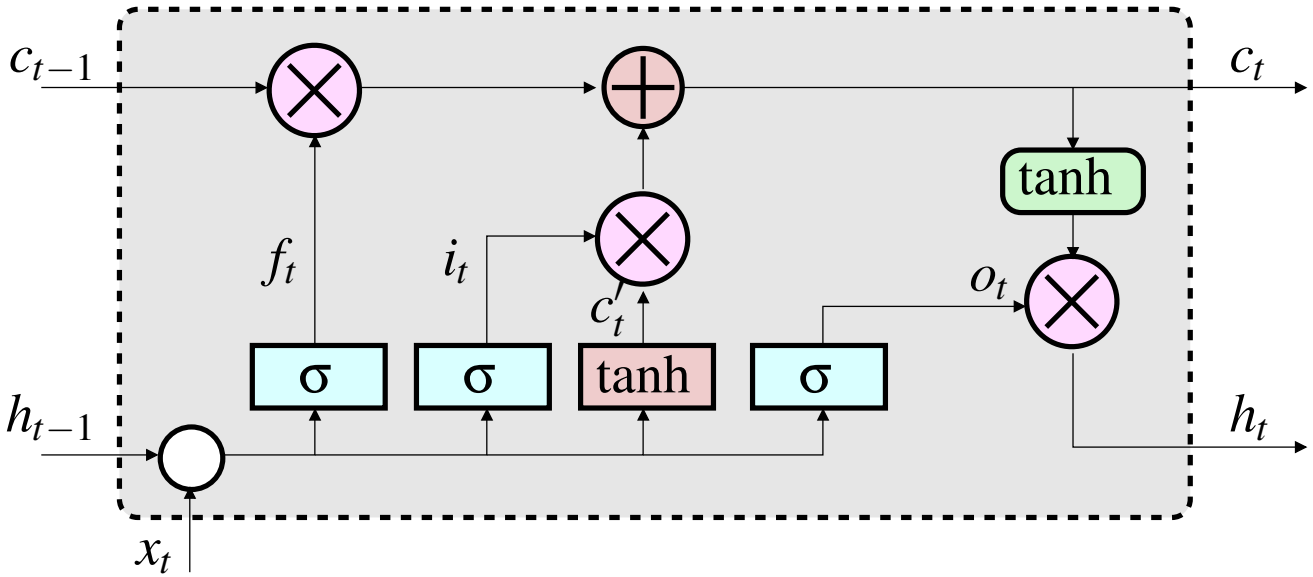
We compute $ax = a(x-128) + 128a$

- a is already in SBP form
- x gets mapped to SBP $x-128$
- product value recovered by adding 128a in post-processing

Activation Functions

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = 1 - \frac{2}{e^{2x} + 1}$$

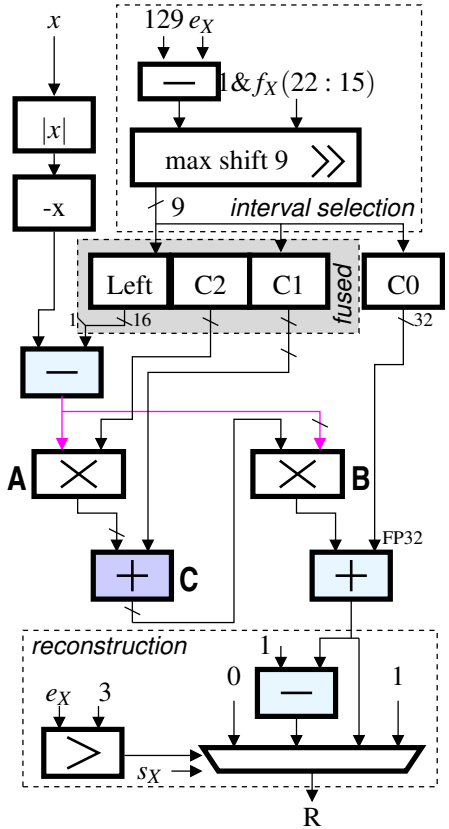
$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$



Activation Functions

f	Prec.	Lat.	Resources	Perf.
Sigmoid	HP	16	211 ALMs, 4 AIs, 0 M20Ks	526 MHz
	SP†	39	501 ALMs, 10 AIs, 4 M20Ks	521 MHz
	SP‡	36	426 ALMs, 8+1/3 AIs 4 M20Ks	526 MHz
	SP§	34	382 ALMs, 6+1/2 AIs 4 M20Ks	526 MHz
Tanh	HP	16	154 ALMs, 4 AIs, 0 M20Ks	526 MHz
	SP	47	705 ALMs, 14 AIs, 3 M20Ks	526 MHz

Arch.	A	B	C	Acc.
SP†	(8,23)	(8,23)	(8,23)	4 ULPs
SP‡	(8,13)	(8,23)	(8,23)	6 ULPs
SP§	(8,13)	(8,15)x(8,23)	(8,15)	9 ULPs



Some results

	Stratix 10NX	Stratix 10MX 2100	Stratix 10SX 2800	XCUV7P	XCUV9P
Blocks	3960	3960	5760	4560	6840
INT8	118800	7920	11520	4560	6840
INT8 (extracted)	118800	15840	23040	9120	13680

Some results

	Stratix 10NX	Stratix 10MX 2100	Stratix 10SX 2800	XCUV7P	XCUV9P
Blocks	3960	3960	5760	4560	6840
INT8	118800	7920	11520	4560	6840
INT8 (extracted)	118800	15840	23040	9120	13680

Device	**Process**	**DSP Blocks**	**Multipliers**
Xilinx VUP9	TSMC 16nm	6840	6840 INT27x18
Stratix 10 2800	Intel 14nm	5760	11520 INT18
Stratix 10 NX	Intel 14nm	3960 Tensor AI	29700 INT16
Achronix AC7T6000	TSMC 7nm	1760	7040 INT16
Xilinx Versal VC1902	TSMC 7nm	400 AI Engines + 1968 DSPs	12800 INT16 + 1968 INT27x18
Intel Agilex AGF027	Intel 10nm	8528	17056 INT18

Conclusion

- new AI Tensor Blocks makes FPGAs competitive in inference
- new architectural features of the AI Tensor Block can fuel new research for non-AI algorithm mappings
- new set of resources $\rightarrow$ different techniques for elementary function implementations.