FPGA AI DATAPATHS

June 2018

Density and Flexibility in Computation

Floating Point & Fixed Point

Integer – many different sizes – including asymmetric - mix and match

Floating Point – FP32 and now BFLOAT16

Mixed Representation – Floating Point without Floating Point resources

Structures – Individual MAC or DOT – of any size

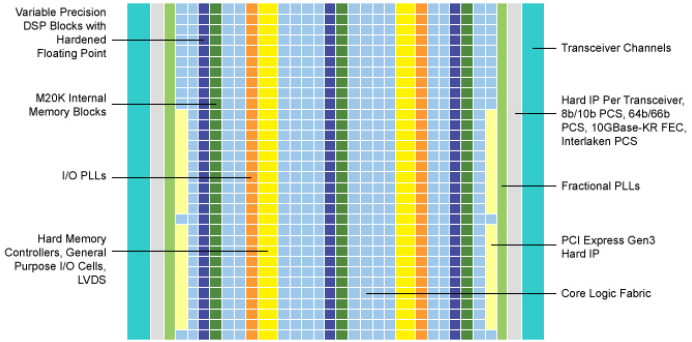
Data Movement – 100% sustained to peak

- Plus massive internal bandwidth

FPGA 101

Customers buy logic

But they pay for routing



Base function: 4/6 LUT + register

AND2 gate = XOR6 gate

But not really

Not enough routing – wire limited

Registers are free

Effective FPGA design is mapping to all of the logic available

And all of the wires

This is much more difficult

FPGA Scale (Current 14nm device)

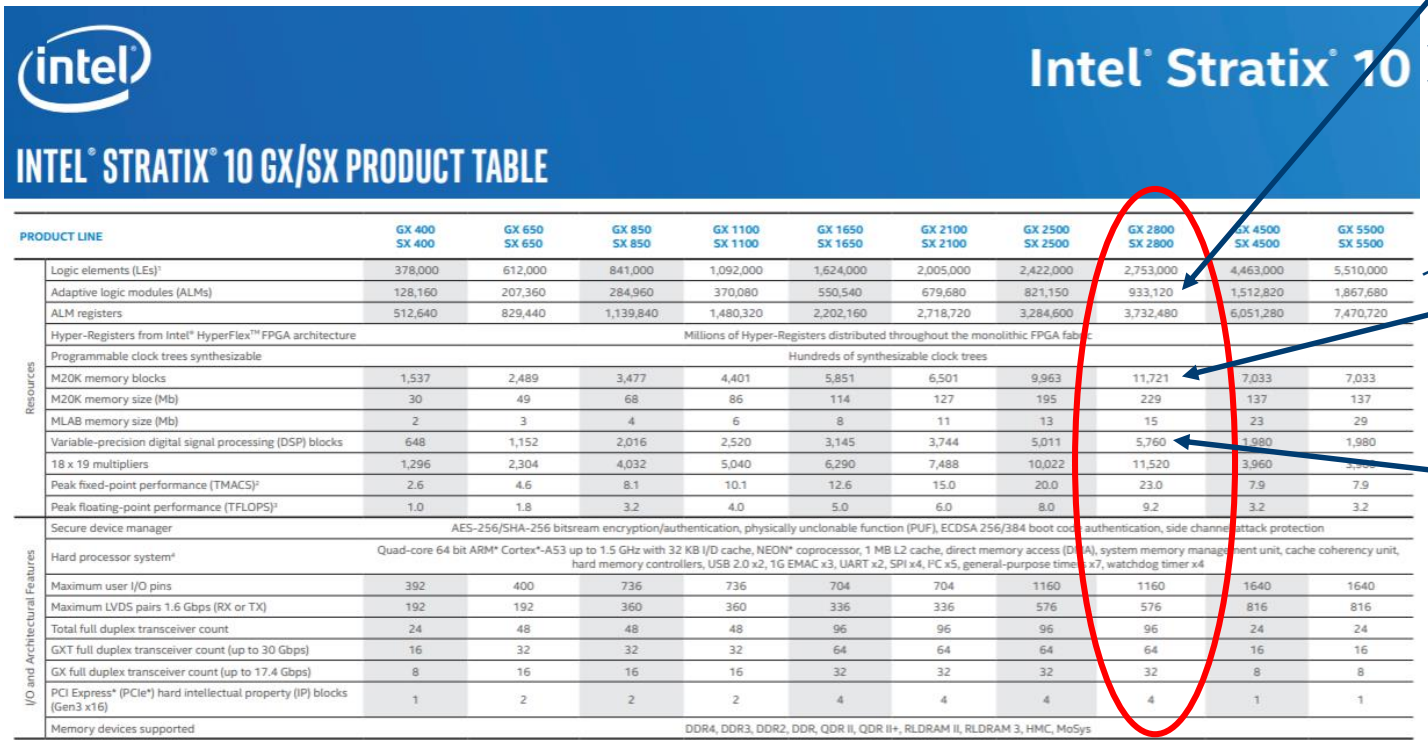
1M 6-LUTs
(2M sum bits)

160K 4x4INT
120TOPs@750MHz

12K Cache Ports

12K FP32
Operators

9TFLOPs@750MHz



Intel® Stratix® 10

INTEL® STRATIX® 10 GX/SX PRODUCT TABLE

	PRODUCT LINE	GX 400 SX 400	GX 650 SX 650	GX 850 SX 850	GX 1100 SX 1100	GX 1650 SX 1650	GX 2100 SX 2100	GX 2500 SX 2500	GX 2800 SX 2800	GX 4500 SX 4500	GX 5500 SX 5500
Resources	Logic elements (LEs)[1]	378,000	612,000	841,000	1,092,000	1,624,000	2,005,000	2,422,000	2,753,000	4,463,000	5,510,000
	Adaptive logic modules (ALMs)	128,160	207,360	284,960	370,080	550,540	679,680	821,150	933,120	1,512,820	1,867,680
	ALM registers	512,640	829,440	1,139,840	1,480,320	2,202,160	2,718,720	3,284,600	3,732,480	6,051,280	7,470,720
	Hyper-Registers from Intel® HyperFlex™ FPGA architecture	Millions of Hyper-Registers distributed throughout the monolithic FPGA fabric									
	Programmable clock trees synthesizable	Hundreds of synthesizable clock trees									
	M20K memory blocks	1,537	2,489	3,477	4,401	5,851	6,501	9,963	11,721	7,033	7,033
	M20K memory size (Mb)	30	49	68	86	114	127	195	229	137	137
	MLAB memory size (Mb)	2	3	4	6	8	11	13	15	23	29
	Variable-precision digital signal processing (DSP) blocks	648	1,152	2,016	2,520	3,145	3,744	5,011	5,760	1,980	1,980
	18 x 19 multipliers	1,296	2,304	4,032	5,040	6,290	7,488	10,022	11,520	3,960	3,960
	Peak fixed-point performance (TMACS)[2]	2.6	4.6	8.1	10.1	12.6	15.0	20.0	23.0	7.9	7.9
	Peak floating-point performance (TFLOPS)[2]	1.0	1.8	3.2	4.0	5.0	6.0	8.0	9.2	3.2	3.2
I/O and Architectural Features	Secure device manager	AES-256/SHA-256 bitstream encryption/authentication, physically unclonable function (PUF), ECDSA 256/384 boot code authentication, side channel attack protection									
	Hard processor system[4]	Quad-core 64 bit ARM* Cortex*-A53 up to 1.5 GHz with 32 KB I/D cache, NEON* coprocessor, 1 MB L2 cache, direct memory access (DMA), system memory management unit, cache coherency unit, hard memory controllers, USB 2.0 x2, 1G EMAC x3, UART x2, SPI x4, I²C x5, general-purpose timers x7, watchdog timer x4									
	Maximum user I/O pins	392	400	736	736	704	704	1160	1160	1640	1640
	Maximum LVDS pairs 1.6 Gbps (RX or TX)	192	192	360	360	336	336	576	576	816	816
	Total full duplex transceiver count	24	48	48	48	96	96	96	96	24	24
	GXT full duplex transceiver count (up to 30 Gbps)	16	32	32	32	64	64	64	64	16	16
	GX full duplex transceiver count (up to 17.4 Gbps)	8	16	16	16	32	32	32	32	8	8
	PCI Express* (PCIe*) hard intellectual property (IP) blocks (Gen3 x16)	1	2	2	2	4	4	4	4	1	1
	Memory devices supported	DDR4, DDR3, DDR2, DDR, QDR II, QDR II+, RLDRAM II, RLDRAM 3, HMC, MoSys									

+100s Tbps local memory bandwidth

FPGA FLOATING POINT

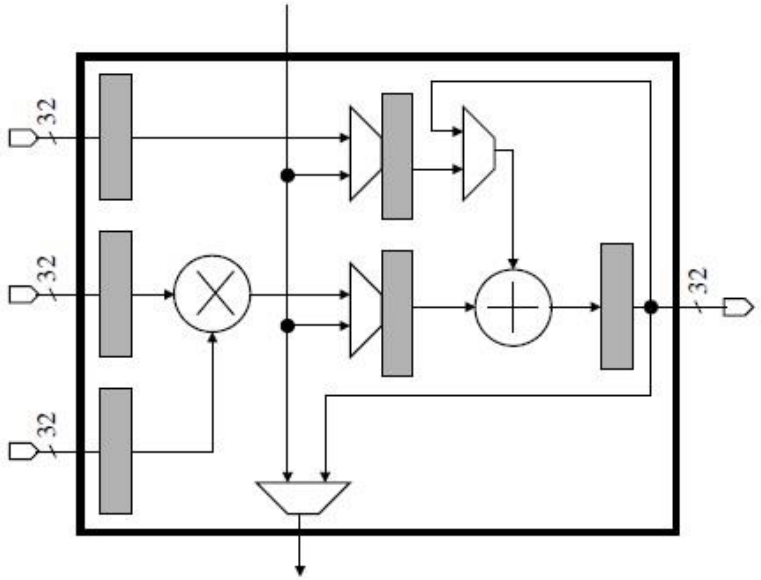
Floating Point DSP Block

Area, aspect ratio and interface largely dictates DSP Block relationship to device

- Wire density key
- Pitch Match to architecture

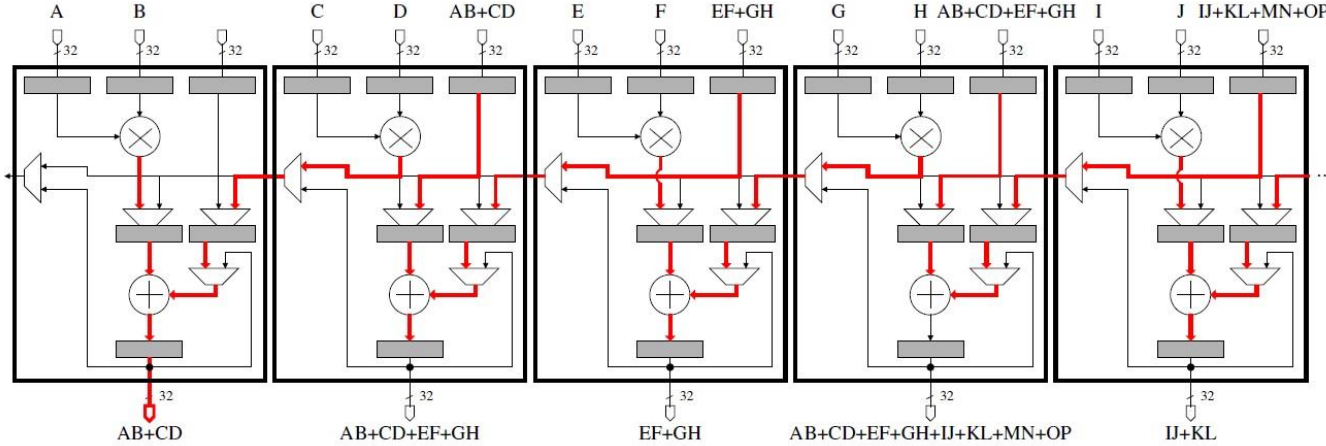
2 major innovations

- Support FP Multiplier RNE inside integer datapath
  - While supporting 40 legacy integer modes
- Vector Mode builds any size recursive reduction trees



Direct Dot Product Support

1. *A,B,C…I,J all arrive at the same time*
2. *AB+CD, EF+GH, IJ+KL computed (Re-use systolic connections)*
3. *Using soft routing, first sum of products fed back to DSP Block inputs*
4. *Re-use sequential connections, calculate next level of tree*



Soft connections can be pipelined to any depth

Yes Virginia, there is a BFLOAT16

- Announced at Intel AI DEVCON in May 2018
  - All Intel products, including FPGA
- What is BFLOAT16?
  - Introduced by Google February 2018
  - FP32 reduced to 16 bits
    - Truncate 16 Mantissa LSBs
  - Same dynamic range – good for vanishingly small numbers for ML training

FPGA FLOATING POINT (WITHOUT FLOATING POINT)

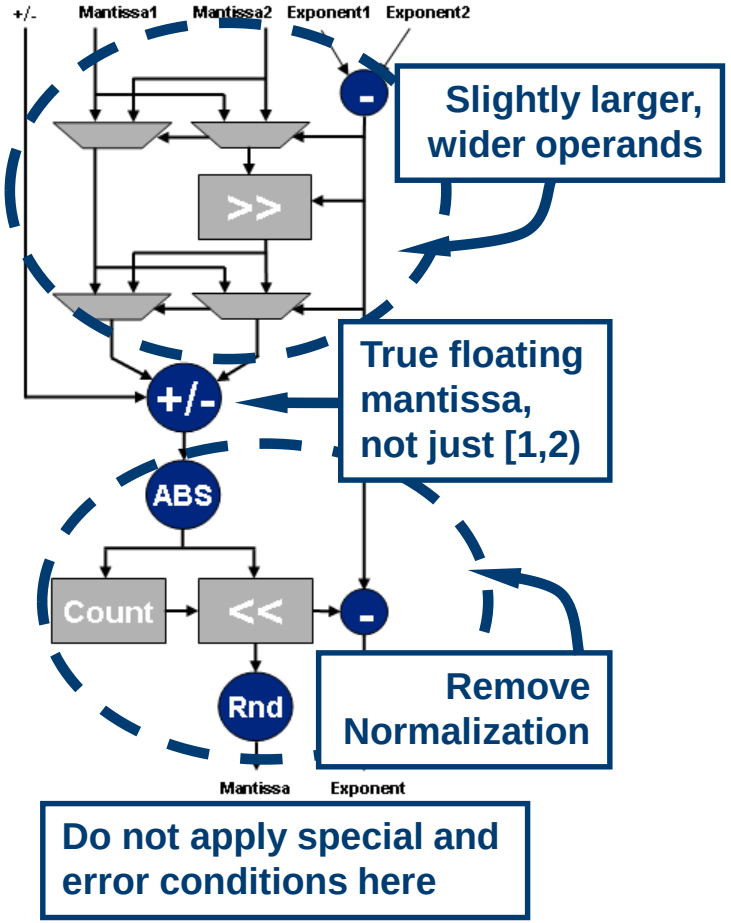
Floating Point Compiler

Automatically extracts inter-operator redundancy in group of floating point operator

- Typically 50% area reduction
- Typically 50% latency reduction
- FPGA floating point system design (soft logic based) becomes possible

Single, Double, or Custom Precision (BFP16)

- Mixed precisions can be directly mixed with optimized CAST() operators



FP without FP

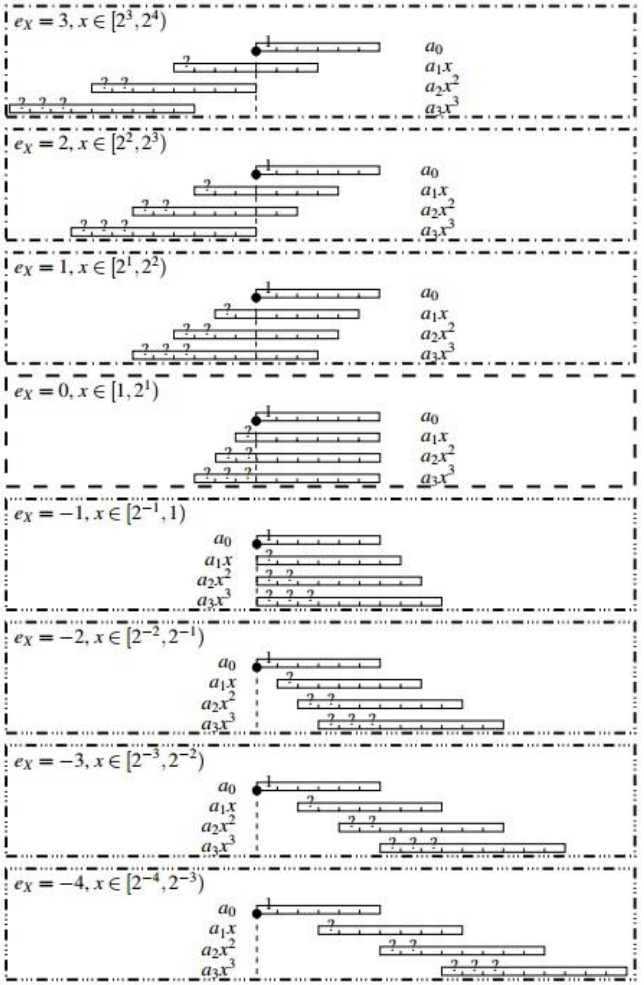
Polynomials can have many operators

- Expensive – power, area, and latency

Most polynomials have monotonic relationship between terms

- First observation : normalization and denormalization redundant
- If relationship between terms is known, all shifts can be pre-computed monotonically

FPGAs filled with small ROMs (6LUTs)



INTEGER - SOFT

Paper Review

Use 100% of logic density

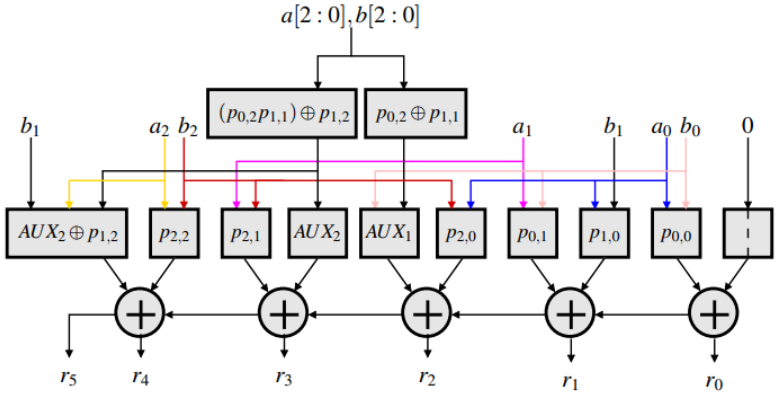
More importantly, use 100% of routing density

- Independent vs. redundant connections

Refactor to greater than 100% logic density

- Use Out-of-band functions

Collapse to single logic level if possible



Soft Logic Multiplier for Free?

3x3 Multiplier 3 ALMs

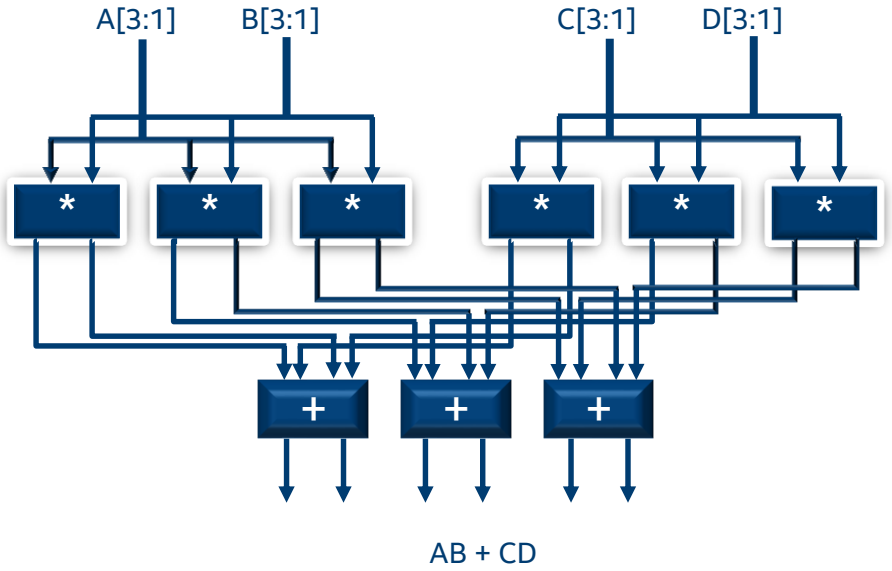
- ALM – two arithmetic bit output

3 ALMs = 6 outputs = min. 3x3

No point in putting 3x3 multipliers in hard logic

- System Cost (routing, logic, power, latency) greater than using inline

Expand to 4x4 and larger



Subset Multiplier Extraction

I thought you just said soft logic multipliers were free?

- Not so for BFLOAT16 or near BFLOAT (15,14,etc)
  - Multiple 6x6, 7x7 – or asymmetric such as 6x7

Can also implement adder tree or portions of it in DSP Block

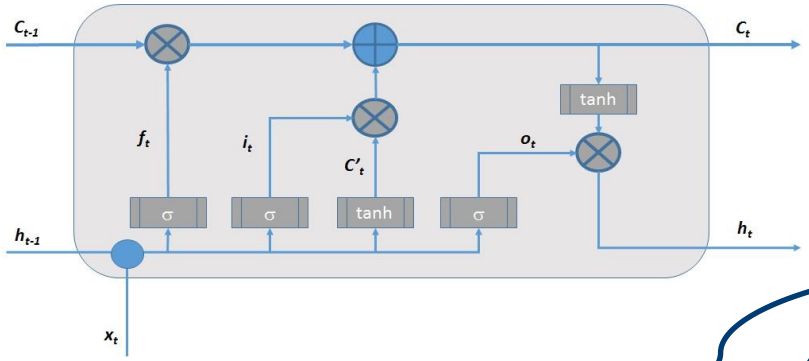
Makes sense if datapath is physically placed near DSP Block

- DSP Block needs to be inline

Mixture of hard and soft logic possible

ELEMENTARY FUNCTIONS

RNN and Hyperbolics



$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_{fi}\right)$$

$$C'_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$C_t = f_t * C_{t-1} + i_t * C'_t$$

$$h_t = o_t * tanh(C_t)$$

Critical Path

$$h_t = o_t * tanh(f_t * C_{t-1} + i_t * tanh(W_c[h_{t-1}, x_t]) + b_c)$$

tanh(x)
12 DSP Blocks.
Latency = 100 total

Matrix-vector multiply
64 DSP Blocks
Latency = 20

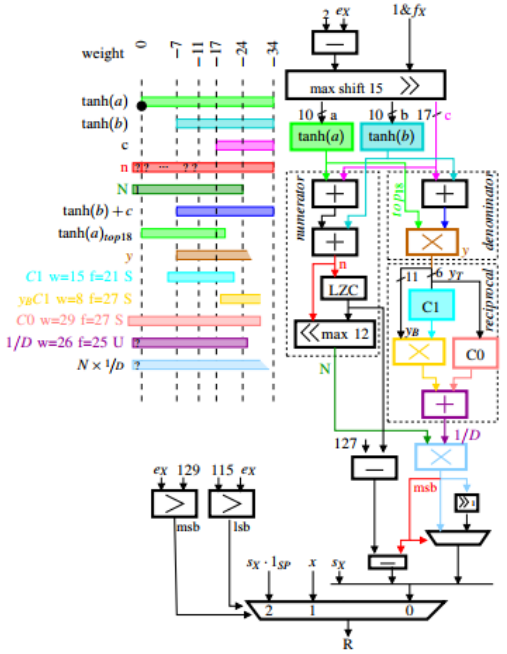
Reducing tanh(x) latency 50% = 70% performance increase!

Hyperbolic Construction

$$\tanh(a+b) = \frac{\tanh(a)+\tanh(b)}{1+\tanh(a)\tanh(b)}$$

$$\tanh(a+b+c) = \frac{\tanh(a)+\frac{\tanh(b)+\tanh(c)}{1+\tanh(b)\,tanh(c)}}{1+\tanh(a)\frac{\tanh(b)+\tanh(c)}{1+\tanh(b)\tanh(c)}}$$

$$\tanh(a+b+c) \approx \frac{\tanh(a)+\tanh(b)+c}{1+\tanh(a)(\tanh(b)+c)}$$



REAL WORLD APPLICATION

Microsoft Brainwave

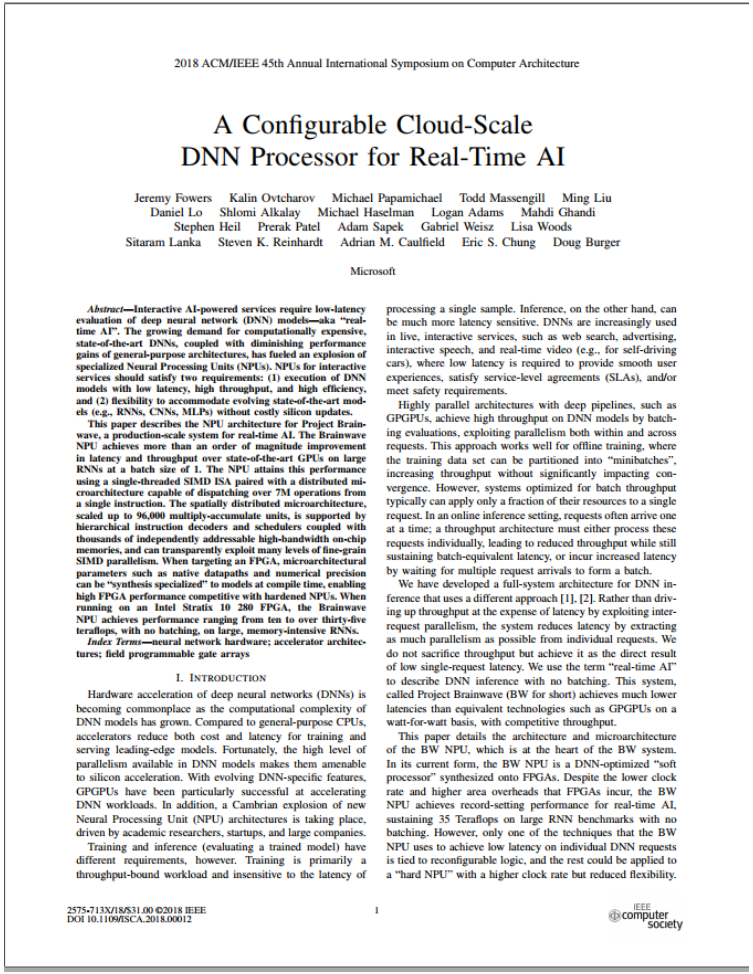
ISCA 2018 Paper

"96,000 multiply-accumulate units"

"287 GFLOPs/W"

"can run all DeepBench layers at under 4ms at batch 1"

"..23% to 75% of peak FLOPs for medium to large LSTM/GRUs (>1500 dimension)"



2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture

A Configurable Cloud-Scale DNN Processor for Real-Time AI

Jeremy Fowers Kalin Ovtcharov Michael Papamichael Todd Massengill Ming Liu
Daniel Lo Shlomi Alkalay Michael Haselman Logan Adams Mahdi Ghandi
Stephen Heil Prerak Patel Adam Sapek Gabriel Weisz Lisa Woods
Sitaram Lanka Steven K. Reinhardt Adrian M. Caulfield Eric S. Chung Doug Burger

Microsoft

***Abstract*—Interactive AI-powered services require low-latency evaluation of deep neural network (DNN) models—aka "real-time AI". The growing demand for computationally expensive, state-of-the-art DNNs, coupled with diminishing performance gains of general-purpose architectures, has fueled an explosion of specialized Neural Processing Units (NPUs). NPUs for interactive services should satisfy two requirements: (1) execution of DNN models with low latency, high throughput, and high efficiency, and (2) flexibility to accommodate evolving state-of-the-art models (e.g., RNNs, CNNs, MLPs) without costly silicon updates.**

This paper describes the NPU architecture for Project Brainwave, a production-scale system for real-time AI. The Brainwave NPU achieves more than an order of magnitude improvement in latency and throughput over state-of-the-art GPUs on large RNNs at a batch size of 1. The NPU attains this performance using a single-threaded SIMD ISA paired with a distributed microarchitecture capable of dispatching over 7M operations from a single instruction. The spatially distributed microarchitecture, scaled up to 96,000 multiply-accumulate units, is supported by hierarchical instruction decoders and schedulers coupled with thousands of independently addressable high-bandwidth on-chip memories, and can transparently exploit many levels of fine-grain SIMD parallelism. When targeting an FPGA, microarchitectural parameters such as native datapaths and numerical precision can be "synthesis specialized" to models at compile time, enabling high FPGA performance competitive with hardened NPUs. When running on an Intel Stratix 10 280 FPGA, the Brainwave NPU achieves performance ranging from ten to over thirty-five teraflops, with no batching, on large, memory-intensive RNNs.

***Index Terms*—neural network hardware; accelerator architectures; field programmable gate arrays**

I. Introduction

Hardware acceleration of deep neural networks (DNNs) is becoming commonplace as the computational complexity of DNN models has grown. Compared to general-purpose CPUs, accelerators reduce both cost and latency for training and serving leading-edge models. Fortunately, the high level of parallelism available in DNN models makes them amenable to silicon acceleration. With evolving DNN-specific features, GPGPUs have been particularly successful at accelerating DNN workloads. In addition, a Cambrian explosion of new Neural Processing Unit (NPU) architectures is taking place, driven by academic researchers, startups, and large companies.

Training and inference (evaluating a trained model) have different requirements, however. Training is primarily a throughput-bound workload and insensitive to the latency of processing a single sample. Inference, on the other hand, can be much more latency sensitive. DNNs are increasingly used in live, interactive services, such as web search, advertising, interactive speech, and real-time video (e.g., for self-driving cars), where low latency is required to provide smooth user experiences, satisfy service-level agreements (SLAs), and/or meet safety requirements.

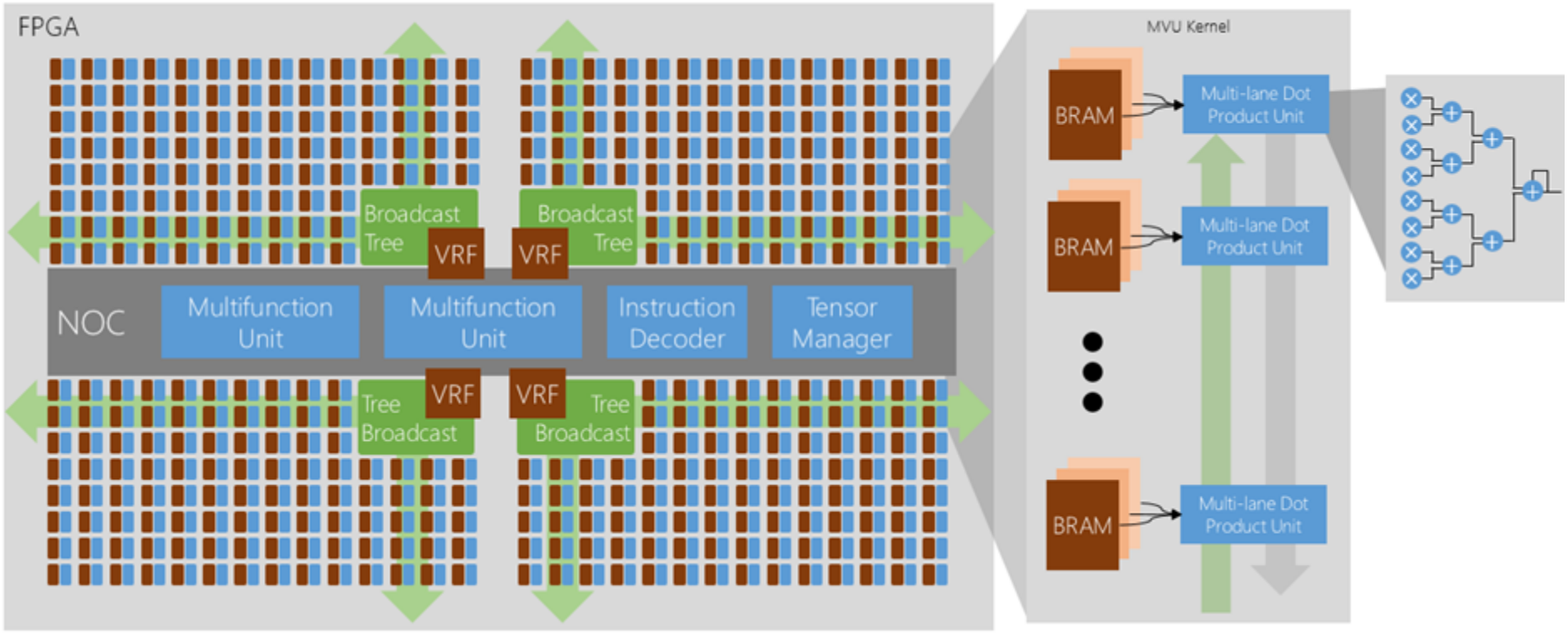
Highly parallel architectures with deep pipelines, such as GPGPUs, achieve high throughput on DNN models by batching evaluations, exploiting parallelism both within and across requests. This approach works well for offline training, where the training data set can be partitioned into "minibatches", increasing throughput without significantly impacting convergence. However, systems optimized for batch throughput typically can apply only a fraction of their resources to a single request. In an online inference setting, requests often arrive one at a time; a throughput architecture must either process these requests individually, leading to reduced throughput while still sustaining batch-equivalent latency, or incur increased latency by waiting for multiple request arrivals to form a batch.

We have developed a full-system architecture for DNN inference that uses a different approach [1], [2]. Rather than driving up throughput at the expense of latency by exploiting inter-request parallelism, the system reduces latency by extracting as much parallelism as possible from individual requests. We do not sacrifice throughput but achieve it as the direct result of low single-request latency. We use the term "real-time AI" to describe DNN inference with no batching. This system, called Project Brainwave (BW for short) achieves much lower latencies than equivalent technologies such as GPGPUs on a watt-for-watt basis, with competitive throughput.

This paper details the architecture and microarchitecture of the BW NPU, which is at the heart of the BW system. In its current form, the BW NPU is a DNN-optimized "soft processor" synthesized onto FPGAs. Despite the lower clock rate and higher area overheads that FPGAs incur, the BW NPU achieves record-setting performance for real-time AI, sustaining 35 Teraflops on large RNN benchmarks with no batching. However, only one of the techniques that the BW NPU uses to achieve low latency on individual DNN requests is tied to reconfigurable logic, and the rest could be applied to a "hard NPU" with a higher clock rate but reduced flexibility.

2575-713X/18/$31.00 ©2018 IEEE
DOI 10.1109/ISCA.2018.00012

Brainwave Floorplan



Source: Microsoft Presentation, HotChips2017

CONCLUSIONS

Summary

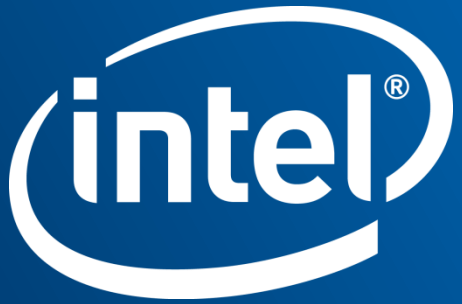
FPGAs can mix IEEE754 FP, custom FP, integer, and combination of numerics simultaneously

Elementary functions – multiple different numerics internally

Can change this from algorithm to algorithm, with multiple different configurations

Very high internal bandwidth and unlimited configurability in connectivity

Computational Density and Flexibility


intel®