Specializing FGPU for Persistent Deep Learning

Rui Ma, Alex Hsu, Tian Tan (The University of Texas at Austin)
Eriko Nurvitadhi, David Sheffield, Aravind Dasu, Rob Pelt,
Martin Langhammer, Jaewoong Sim (Intel)
Derek Chiou (Microsoft / The University of Texas at Austin)

Time-to-Solution

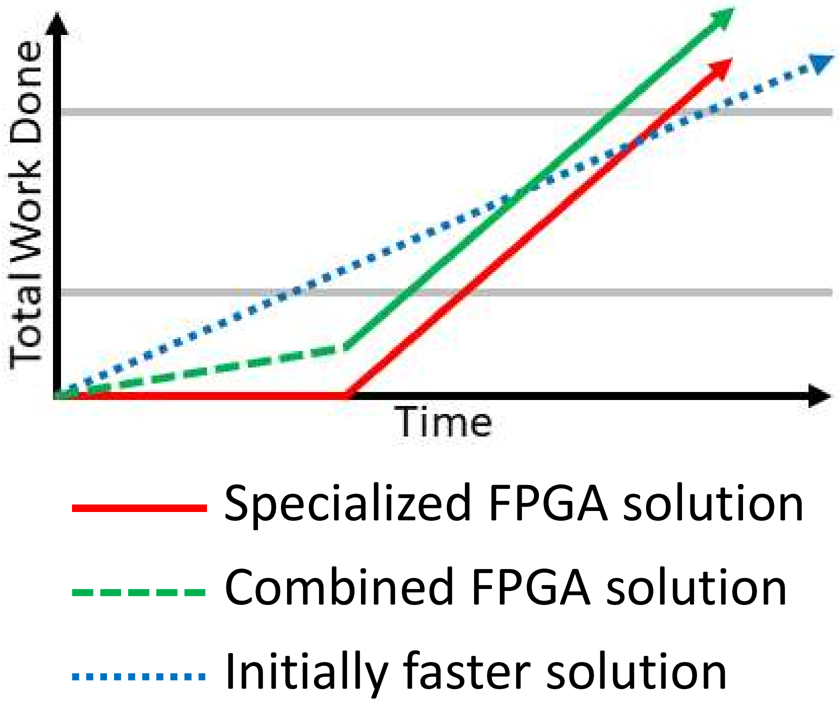
- Time-to-Solution is an important performance metric
  - Includes everything to get all (one to many) needed results
    - E.g., design, implementation, validation, manufacturing, deployment, compilation, and running times
  - Time-to-Solution includes different components depending on approach
    - E.g., software does not include processor development
    - E.g., ASIC includes silicon design and implementation
  - Only if many runs are performed, development time is amortized

- Much of the published work focuses only on kernel run time

- Amdahl's Law is applicable to the total solution

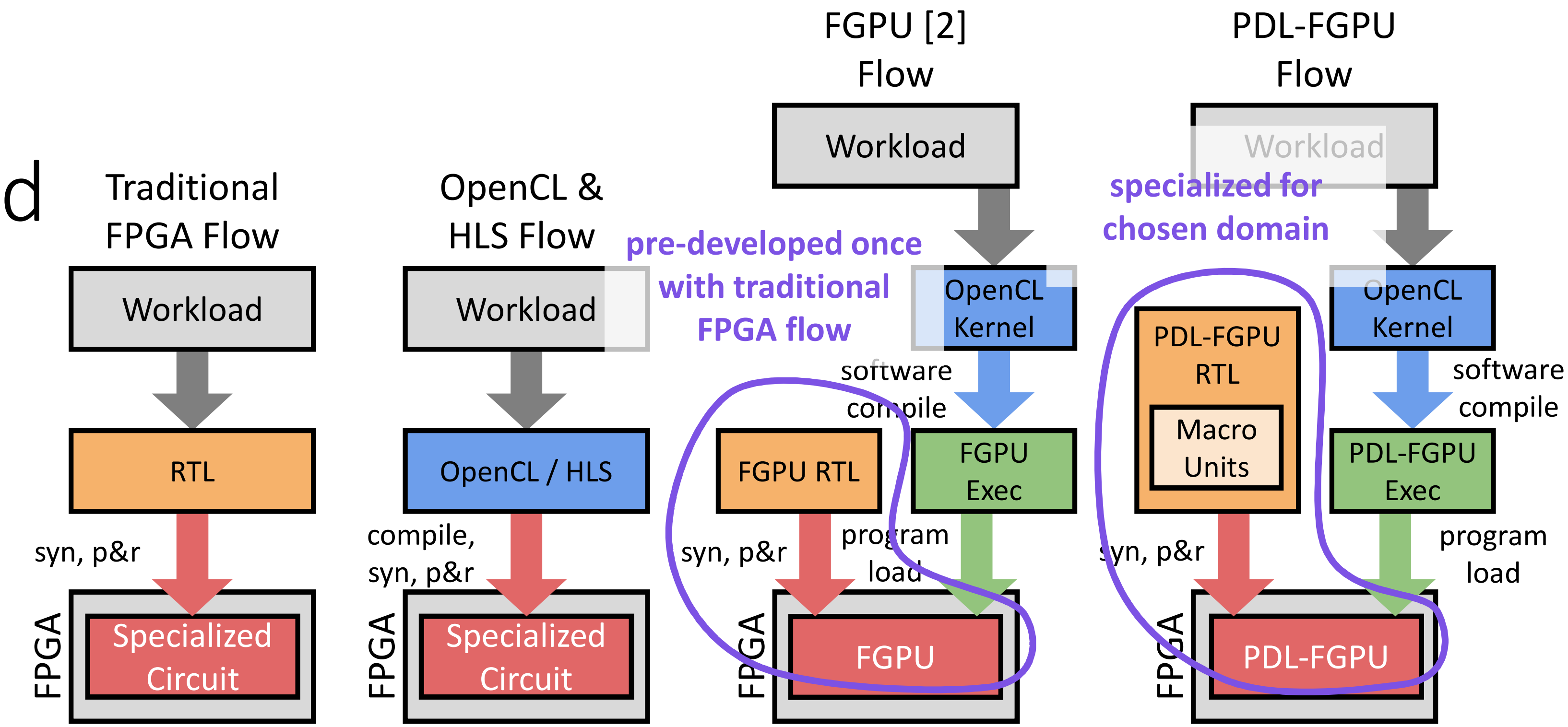
FPGAs High Perf, Slow Development

- Modern FPGAs can achieve industry leading performance [1]
    - Requires high specialization
- Highly-specialized solutions often require long development time
    - Time-to-Solution may be longer than a fast-to-develop even though slower-when-run solution
- Fast dev, reasonable perf solutions used until specialized solution is available
    - May make optimal performance solution unnecessary



[1] Chung, et al. Serving DNNs in Real Rime at Datacenter Scale with Project Brainwave

Solution: Specialized Overlays



General purpose?	No	No	Yes	Yes
Performance	Max	High / Max	Low / Medium	Good
Hardware expertise?	Yes	Yes	No	No
Development time	Weeks - Month	Days - Weeks	Hours - Days	Hours – Days
Compile time	Hours - Days	Hours - Days	Seconds	Seconds

[2] Kadi, Janssen, and Huebner. FGPU: An SIMT-Architecture for FPGAs

Outline

- Time-to-Solution
- PDL-FGPU Architecture and Case Study Workload
- Results
- On-Going Work and Conclusion

Approach

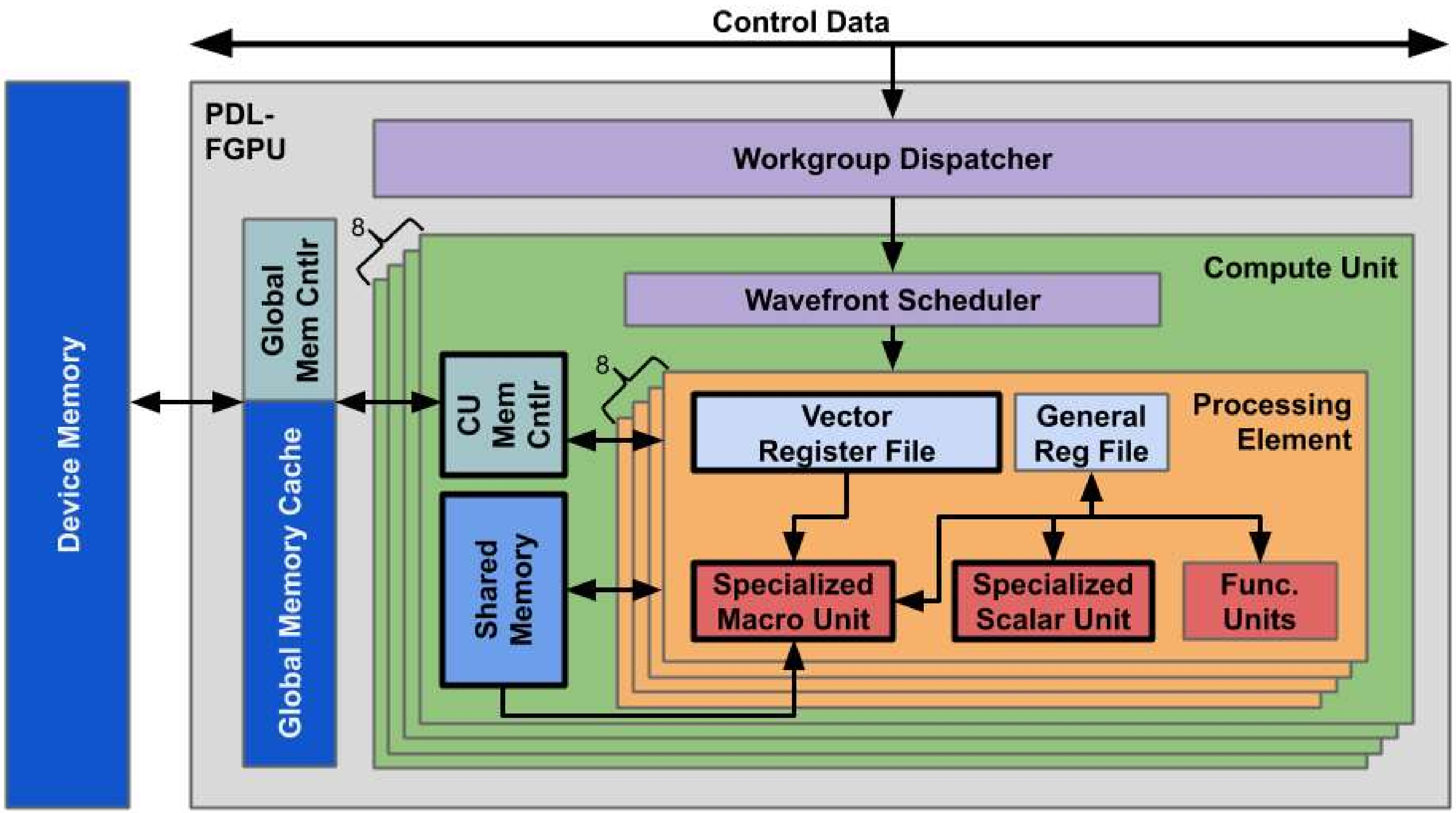
- Start with FGPU [2]
  - Open-source soft GPU programmed with OpenCL-based toolchain

- Specialize FGPU for Persistent RNNs to improve performance

- Target Intel Stratix 10 GX 2800
  - 933,120 ALMs
  - 5,760 DSPs (9.2 FP32 TFLOPS)
  - 11,721 M20Ks (117.2 TB/s BW)
  - 1 GHz

[2] Kadi, Janssen, and Huebner. FGPU: An SIMT-Architecture for FPGAs

Architecture



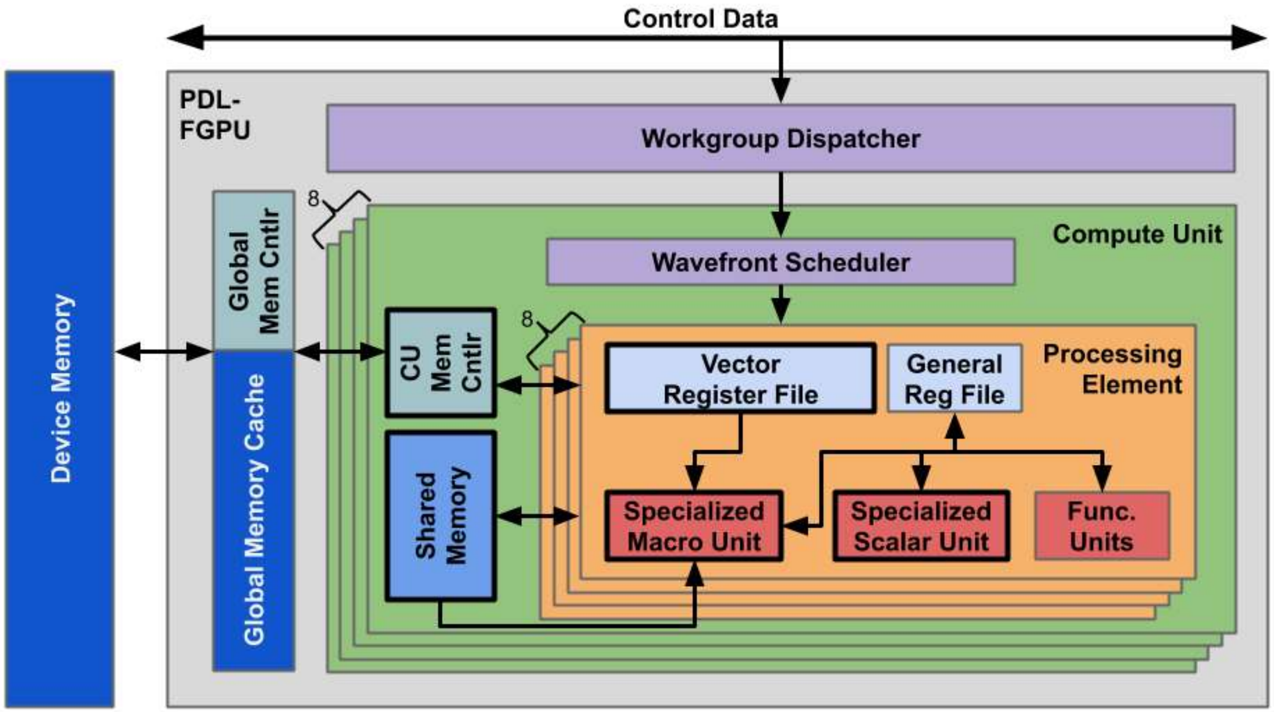
Architecture

Specialized Macro: Dot

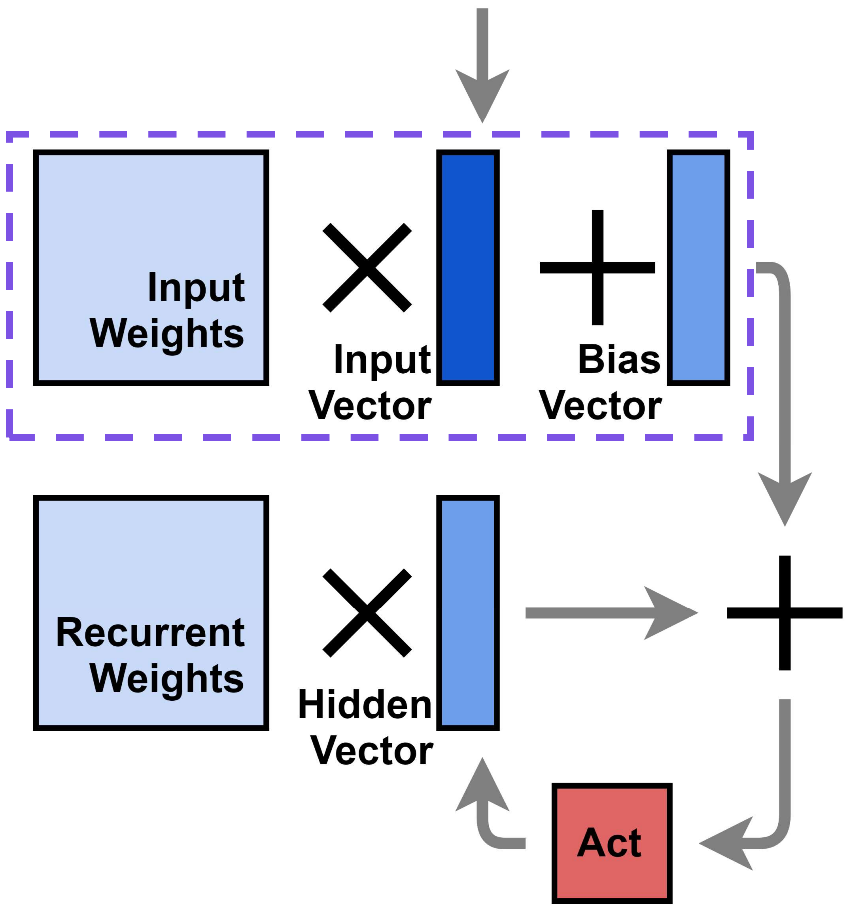
```
dot acc, vec, shr_ptr, shr_off
```

Specialized Scalar: Act

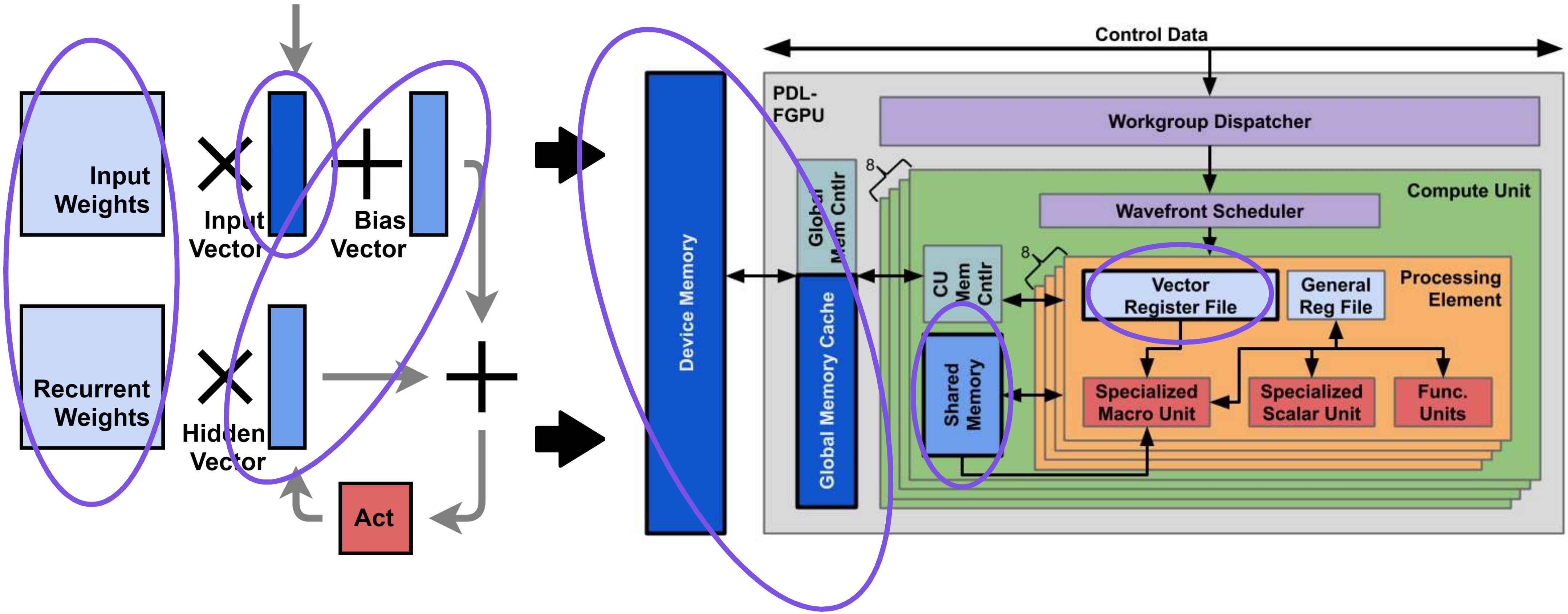
```
sigmoid dest, src
tanh    dest, src
relu    dest, src
```



Persistent RNN Algorithm



Persistent RNN Data Placement



Outline

- Time-to-Solution
- PDL-FGPU Architecture and Case Study Workload
- Results
- On-Going Work and Conclusion

Case Study Workloads

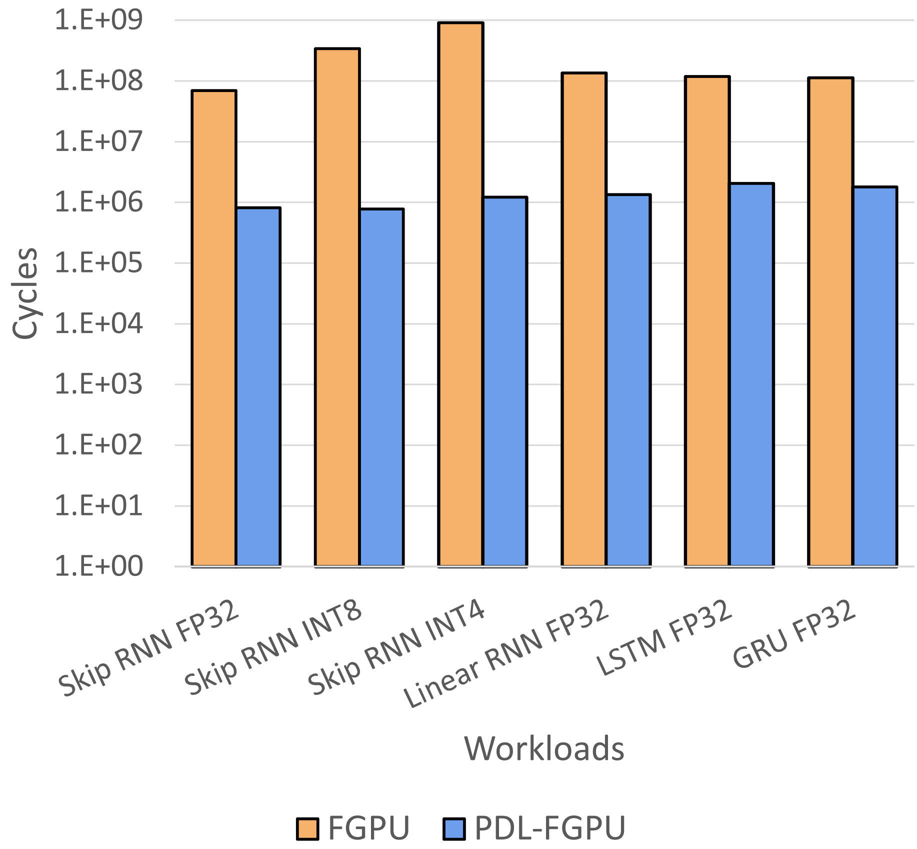
development effort

Algorithm	Precision	Matrix Size	Vector Size	Iters.	Batch	Lines of Code	Engr. Time
RNN (skip input)	FP32	1024x1024	1024	256	1	82	Few hrs
RNN (skip input)	INT8	2048x2048	2048	256	1	75	Few hrs
RNN (skip input)	INT4	4096x4096	4096	256	1	81	Few hrs
RNN (linear input)	FP32	1024x1024	1024	256	1	93	Few hrs
LSTM	FP32	512x512	512	256	1	157	< 1 day
GRU	FP32	512x512	512	256	1	139	< 1 day

PDL-FGPU vs FGPU: Cycles

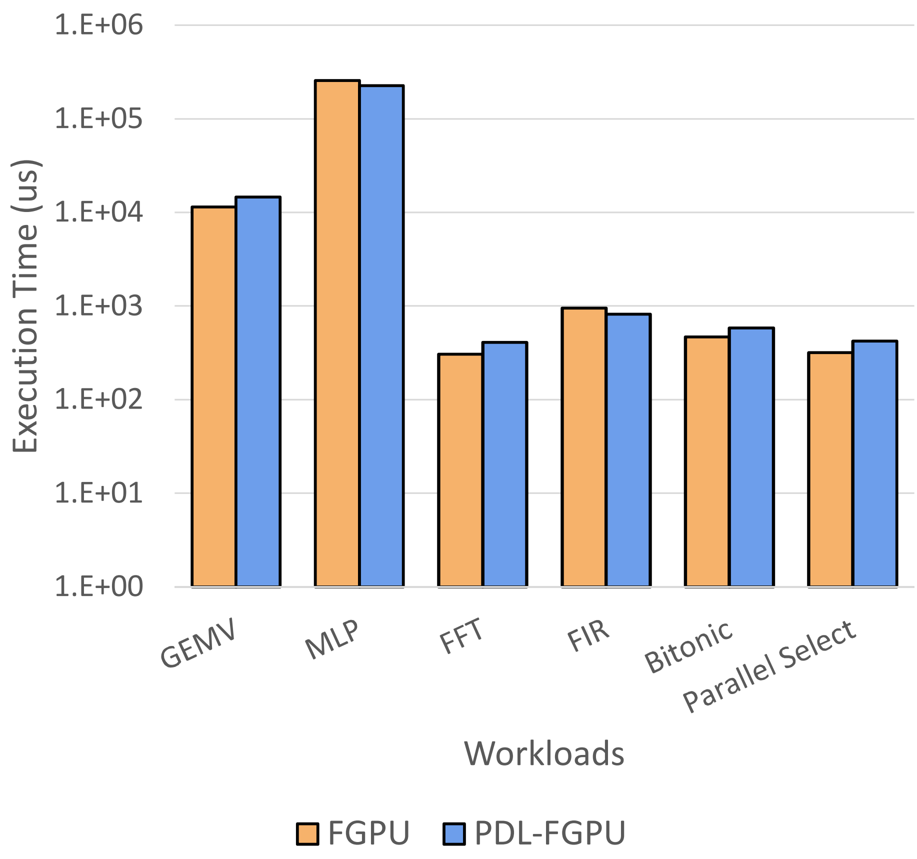
- One to three orders of magnitude performance improvement over baseline
  - 55-727x speedup in single precision and low-precision
- Major reasons for difference

  (85x total on skip input RNN FP32)
  - Vector dot product engine (36x)
  - Keeping weights on-chip (1.7x)
  - Better memory scheduling (1.3x)
  - Improved inter-thread communication (1.05x)



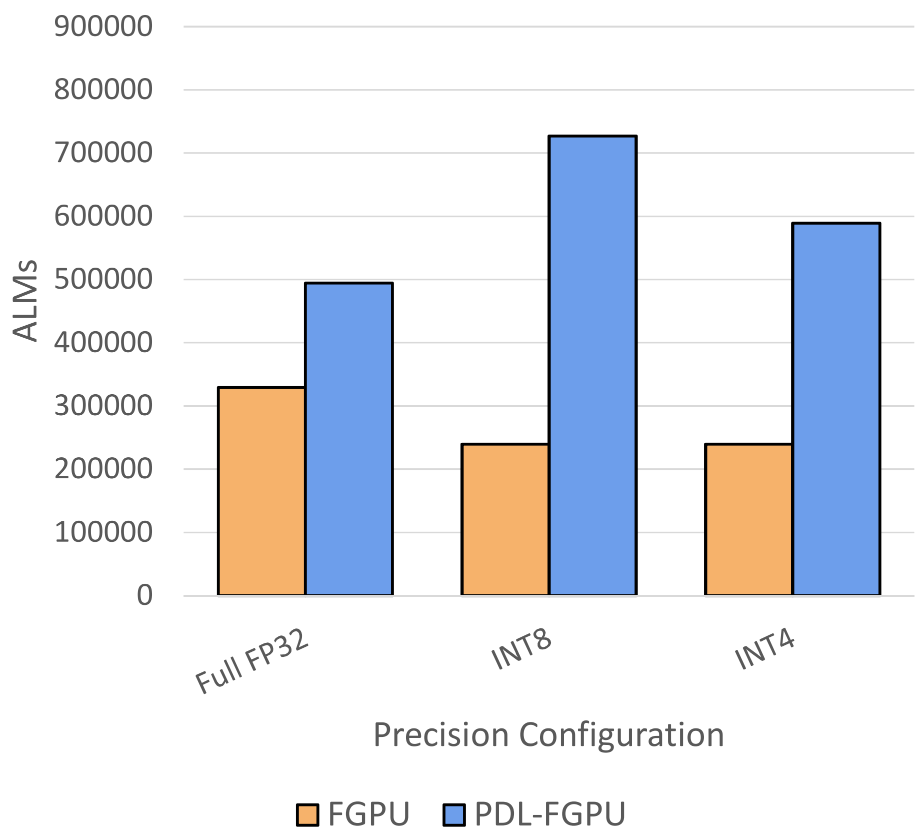
PDL-FGPU vs FGPU: Cycles—Non-PDL

- Generality maintained at close to the same performance
- Cycle reduction mostly due to memory controller scheduling
  - 6% fewer cycles on average
- Execution time increase due to reduced clock frequency
  - 15% slowdown on average



PDL-FGPU vs FGPU: ALM Utilization

- FP32 mode ~1.5x ALM consumption
  - Efficiently leveraged DSPs and on-chip RAM
- Low precision mode has higher ALM consumption
  - Low precision dot product functional units mapped into ALMs
    (at submission time)
  - Improved by packing into DSPs
    (in newer versions)

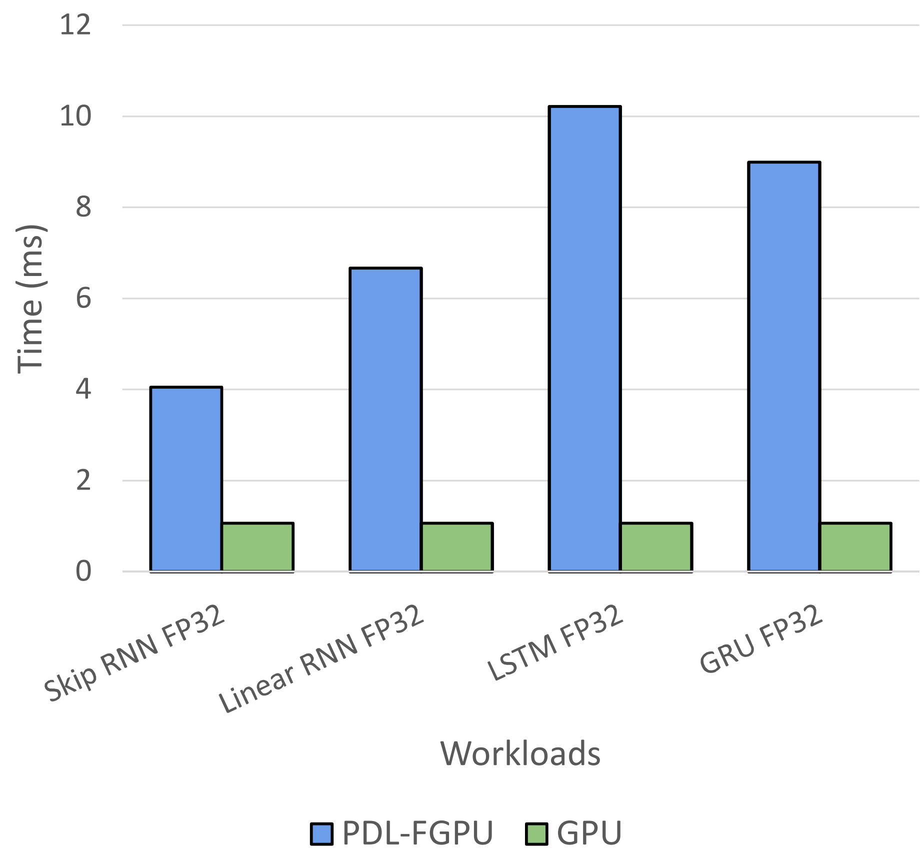


Note: Full FP32 configuration supports all single precision function units: fadd, fmul, fdiv, etc. Each unit can be disabled to save area/improve frequency but requires Quartus compilation.

PDL-FGPU vs V100: Execution Time

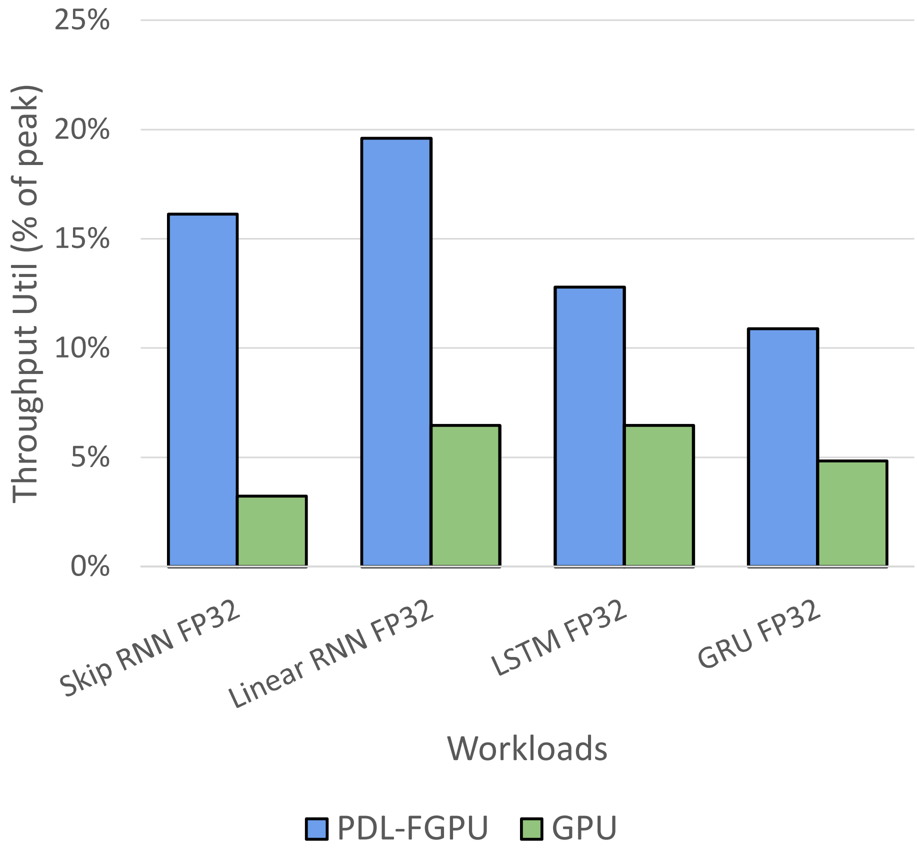
- 3-7x slower than Nvidia V100
  - For measured problems and sizes
- Performance gap factors
  - 5-6x slower frequency
    - ~280 MHz vs ~1500 MHz
  - Fewer floating-point units
    - More DSPs available on S10 than used

Note: cuDNN only supported FP32 kernels at submission time.



PDL-FGPU vs V100: Throughput Utilization

- PDL-FGPU is 2-3x higher in throughput utilization than Nvidia due to higher specialization
- Throughput utilization can be further improved by increasing FPGA resource utilization



Outline

- Time-to-Solution
- PDL-FGPU Architecture and Case Study Workload
- Results
- On-Going Work and Conclusion

On-Going Work

- Continue to optimize
  - Increase number of CUs
  - Increase frequency
  - Improve code generation
- Compare with other OpenCL, HLS, and overlay solutions
- Target other domains
- Improve usability

Conclusions

- Time-to-Solution is an important (but often overlooked) metric
- Using different implementations at different times can improve overall Time-to-Solution
  - Programmability speeds up development
    - Programmable solutions allows quick iteration for functional correctness
    - Domain-specific programmable solutions can minimize runtime
  - Highly-specialized solution maximizes performance once available
- Domain-specific programmable solutions provide higher performance
  - 55-727x speedup on persistent RNNs over baseline
  - Within a factor of 3-7x of Nvidia V100 on persistent RNNs at FP32

Thank you!

Backup Slides

Persistent RNN

- Recurrent neural networks are a class of deep learning networks that have layer(s) that feedback themselves
- Useful for sequential tasks such as speech recognition, text processing, and translation
- In **persistent RNN**, weights are kept in registers and activations are kept in shared memory
  - Leverages the large capacity and high bandwidth of SRAMs on modern FPGA

PDL-FGPU Architecture: Modifications

- Dot product vector instruction
  - Fused shared memory load, dot, and reduction operation
- Activation instructions
  - Reduces instruction pressure
- Synchronization instructions
  - Better inter-thread cooperation
- Conditional memory load/store instructions
  - if reg==0 then ld/st
  - Avoids control flow divergence
- Memory controller improvements
- High bandwidth register file with 1024-bit single-cycle registers
  - 128 bytes / cycle
- High bandwidth shared memory
  - 128 bytes / cycle

PDL-FGPU Configuration

- Hardware
  - 8 Compute Units per PDL-FGPU (16 in progress)
  - 8 Processing Elements per Compute Unit
  - 1024-bit wide operation (32 DSPs) per Processing Element
- Execution
  - 4096 threads in 64-wide SIMD
  - 16x1024-bit & 32x32-bit registers per thread

Hardware Comparison Table

	Nvidia V100	S10-280	S10-210
FP32 throughput	15 TFLOPS	9.2 TFLOPS	6.3 TFLOPS
SRAM size	38 MB	30 MB	30 MB
SRAM bandwidth	145 TB/s	140 + 110 TB/s	65 + 80 TB/s
DRAM bandwidth	1 TB/s (HBM2*4)	64 GB/s (DDR4*4)	0.5 TB/s (HBM2*2)
Frequency	1.4 GHz / 1.67 GHz	1 GHz	1 GHz
I/O	300 GB/s (NVLink)	240 GB/s	240 GB/s
Power	345W	?	?

PDL-FGPU vs FGPU: Resource Utilization

Config	ALM		RAM		DSP		Min Freq (MHz)		Max Freq (MHz)	
	FGPU	PDL	FGPU	PDL	FGPU	PDL	FGPU	PDL	FGPU	PDL
FP32*	329226	494619	1318	5790	768	3552	270	201	322	240
INT8	239714	726823	742	4766	128	128	282	236	335	287
INT4	239714	589425	742	4766	128	128	282	274	335	313

Note: The full FP32 configuration supports all single precision function units: fadd, fmul, fdiv, etc. The design allows any unit to be selectively disabled to save area/improve frequency but requires another full Quartus compilation.

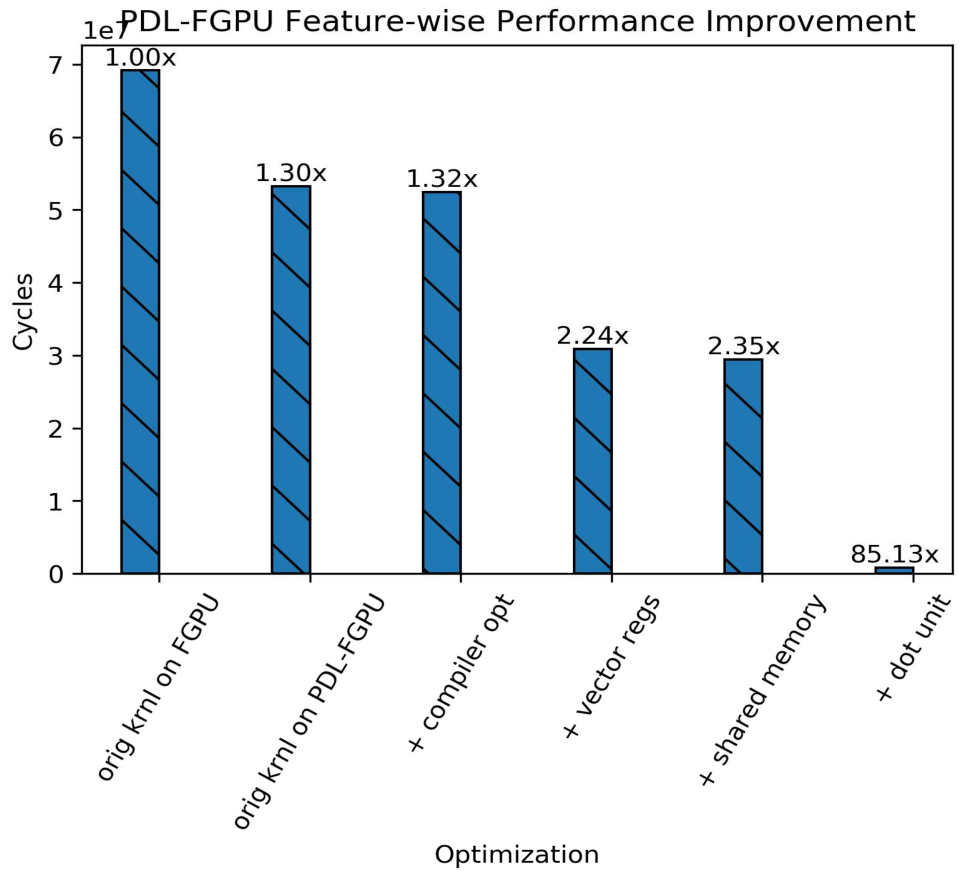
PDL-FGPU vs FGPU: Resource Util Breakdown

FGPU baseline for LSTM / GRU	Global		Per CU			
	global memory controller	workgroup dispatcher	context memory	wavefront scheduler	CU memory controller	CV
ALM	39253	930.3	46	1500	16813	12949
RAM	53	8	2	2	48	56
DSP	0	0	0	0	0	80

PDL-FGPU for LSTM / GRU	Global		Per CU							
	global memory controller	workgroup dispatcher	context memory	wavefront scheduler	CU memory controller	shared memory	CV			
							Total	dot	vector regfile	act
ALM	47510	885	46	1589	2993	14062	27725	8476	3505	3126
RAM	61	8	2	2	55	78	507	0	416	56
DSP	0	0	0	0	0	0	392	280	0	64

Feature-wise Speedup: FP32 RNN (Skip Input)

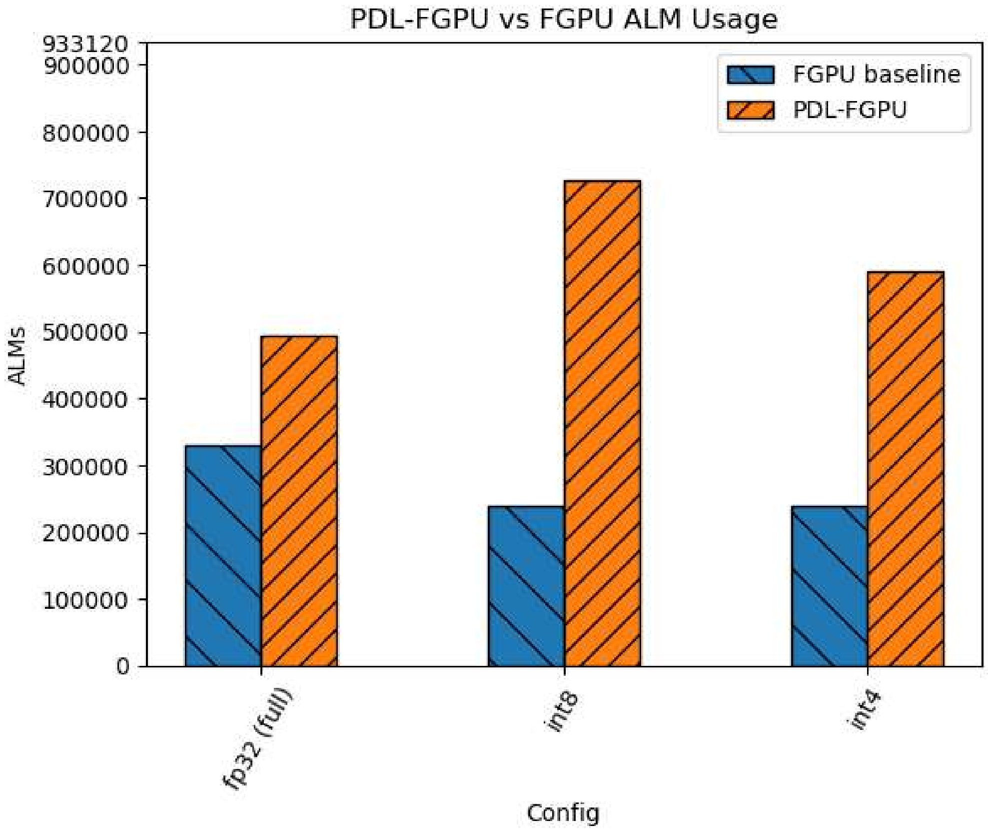
- Domain-specific macro unit (e.g. dot unit) provides the most performance improvement



Even More Backup Slides

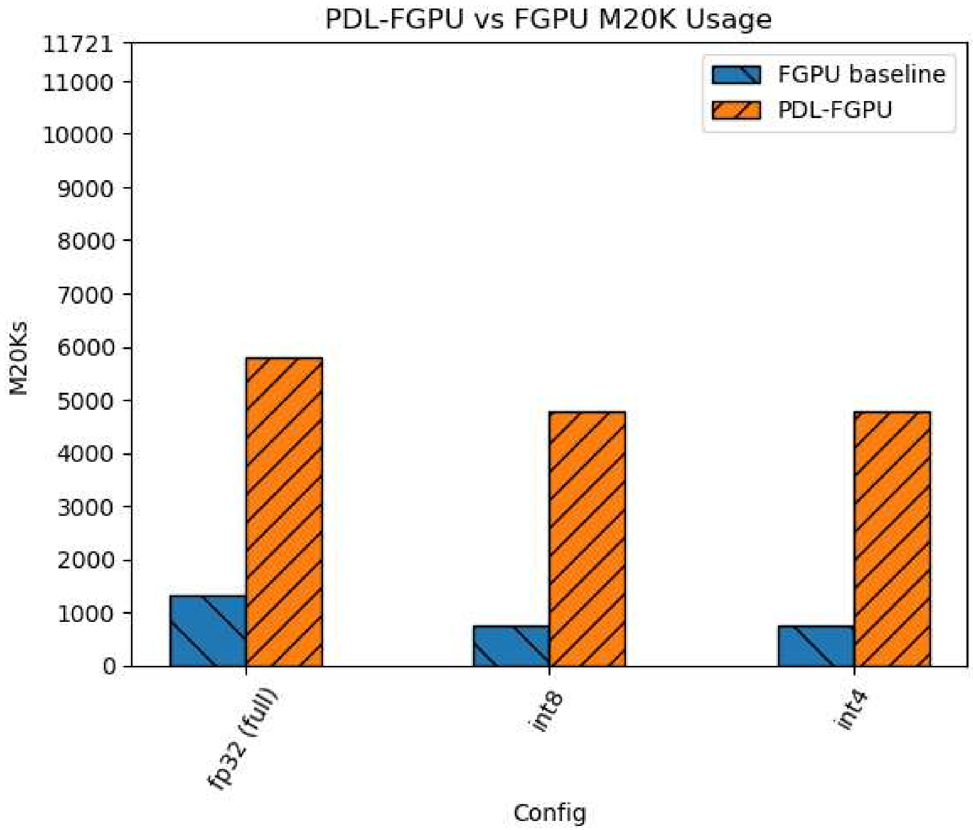
FPGU vs PDL-FGPU: ALMs

- Most configurations ~1.5x ALM consumption
  - Efficiently leverage DSPs and on-chip RAM
- Low precision mode has higher ALM consumption
  - Currently low precision dot function units are mapped into ALMs and could be improved by packing them into DSPs
  - Fixed in new versions



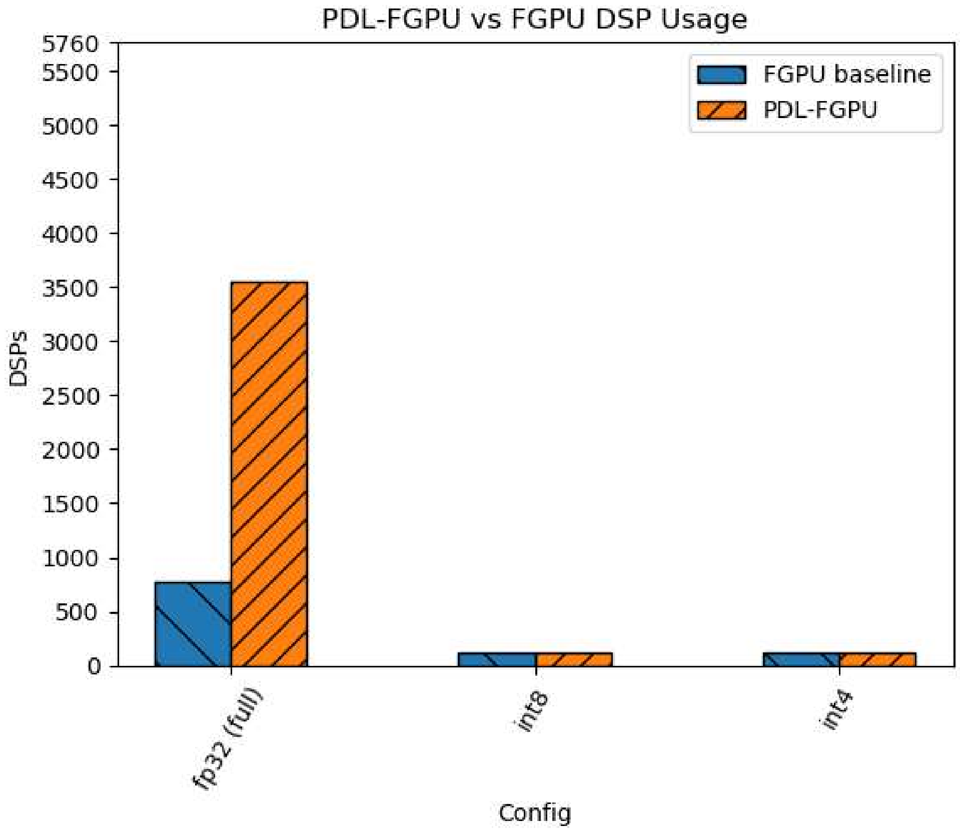
FPGU vs PDL-FGPU: M20ks

- ~5x M20ks consumption
  - Vector register file
  - Shared memory
  - Other microarchitectural changes to better leverage on chip RAM



FPGU vs PDL-FGPU: DSPs

- FP32 configuration ~4.6x DSPs consumption
  - Dot product unit
  - Activation function unit



Configurable FP32 Function Units

Included both in FGPU and PDL-FGPU		Included only in PDL-FGPU	
Function unit	Description	Function unit	Description
FADD	Addition	FFMA	Multiplication and Accumulation
FMUL	Multiplication	SIGMOID	Sigmoid function
FDIV	Division	TANH	Tanh function
FSQRT	Square Root		
FRSQRT	Inverse square root		
UITOFP	Cast unsigned INT to FP32		
FSLT	Comparison, less than		

Performance Evaluation Assumptions

- Exclude
  - host-side compute or data transfers (roughly the same between FPGA/GPU)
  - initialization effects
    - FGPU/PDL-FGPU: ~500 cycles of CU initialization per kernel
    - GPU: one-time JIT compilation of the application DC4
- Nvidia’s terminology is used
  - Skip input RNN assumes the biased input weight activation multiply is precomputed, and thus only 1 GEMV is computed per input per iteration
  - Linear input RNN means both the input and hidden computation are computed

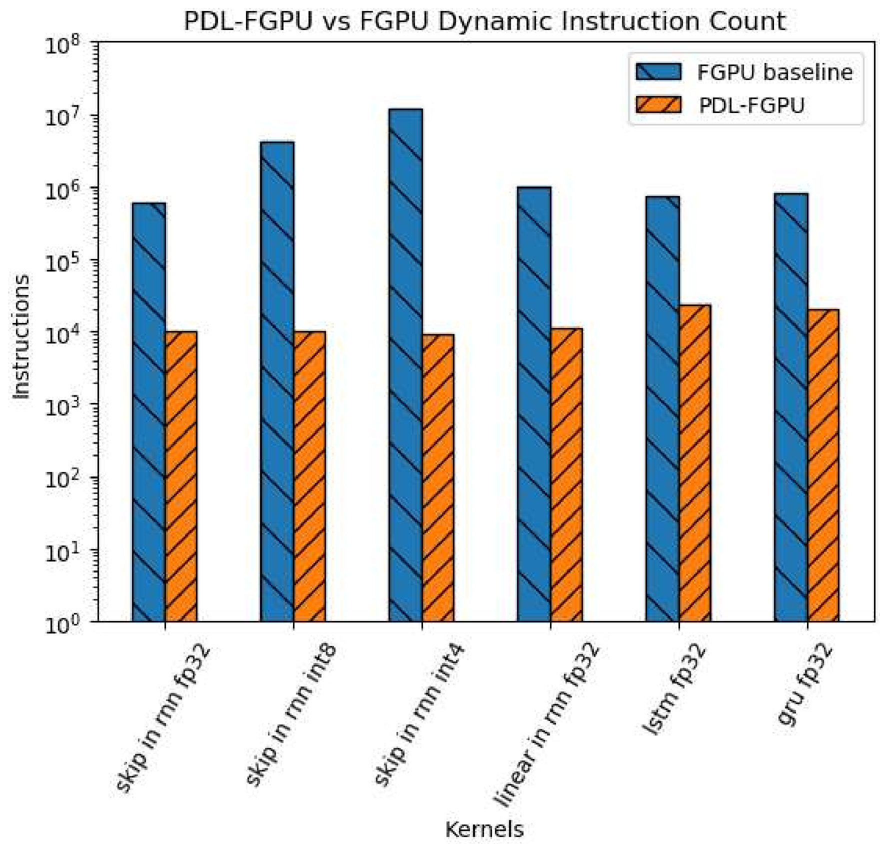
Slide 35

DC4 how much time does this take? Point is to say that they are roughly teh same

Derek Chiou, 8/31/2019

FGPU vs PDL-FGPU: Dynamic Instruction Count

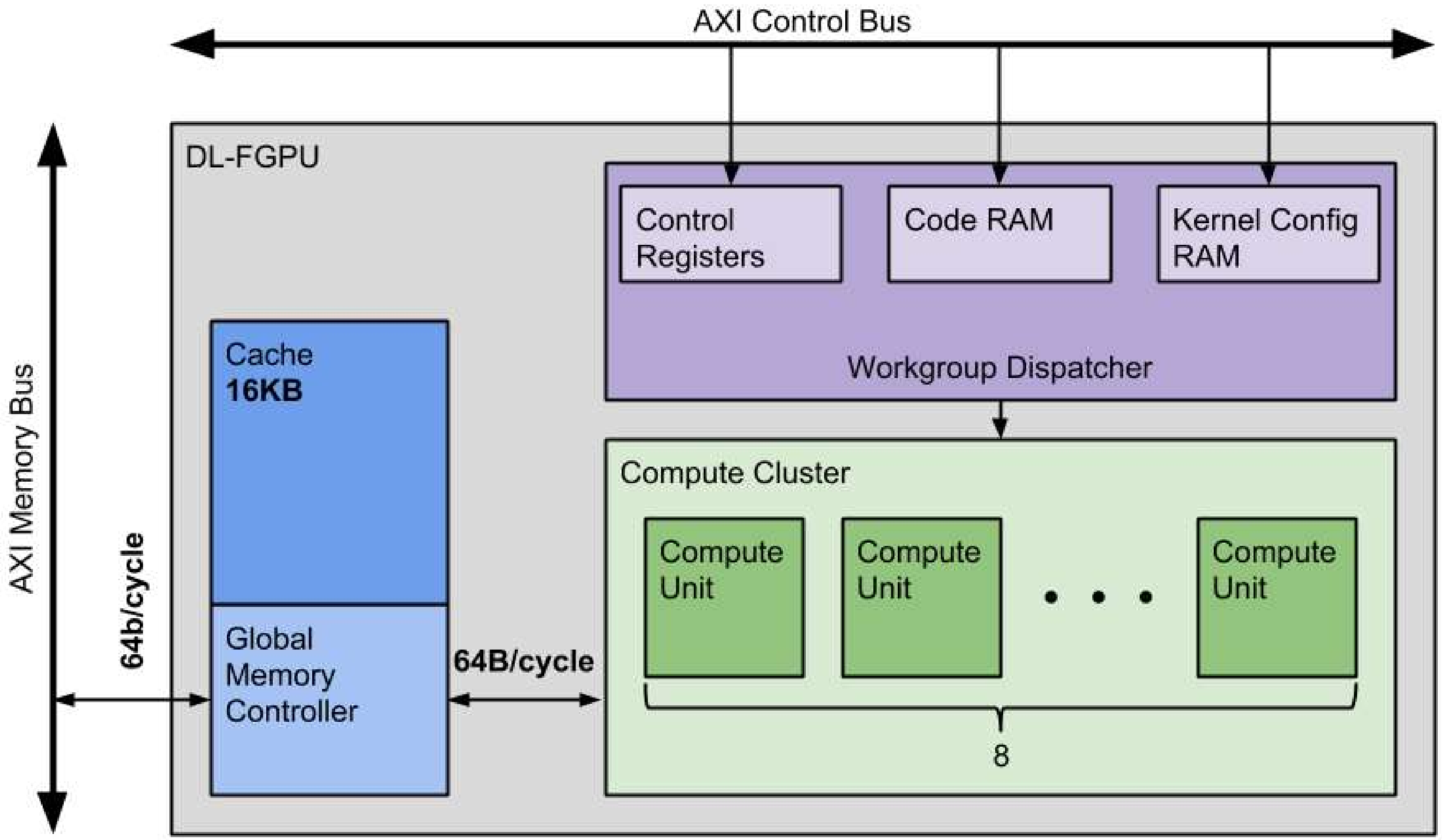
- 30-1342x less instructions than base line
  - Domain-specific instructions reduce instruction pressure



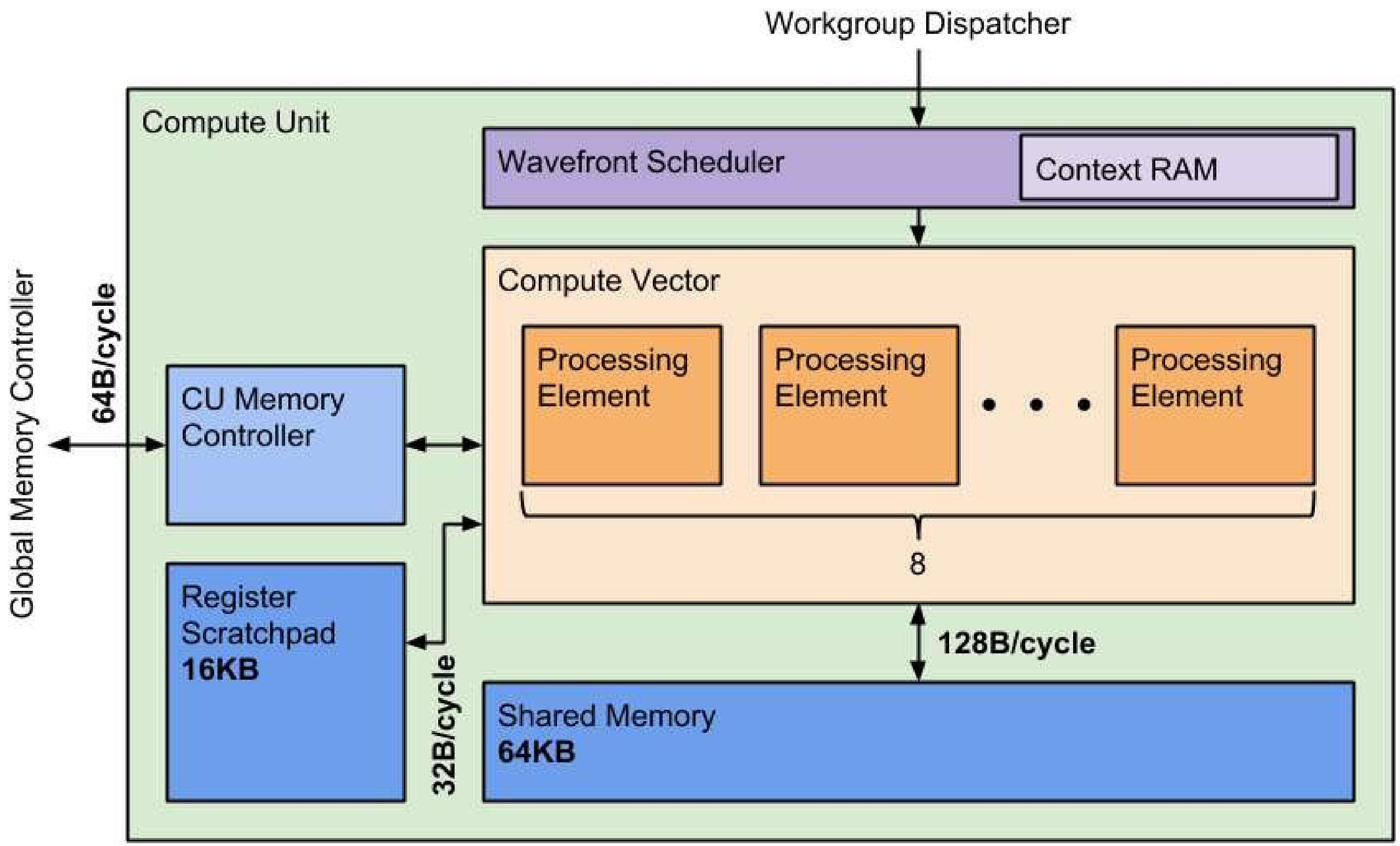
FPGA vs GPU Capabilities

- Flexible precision
  - Densely packed computational resources (Intel)
    - 5760 DSPs on Stratix 10 yield 7 TFLOPS, or 28 TOPS of INT8 arithmetic at 600 MHz
  - 15 TFLOPS on V100, 130 TOPS of INT8 on V100 tensor core
- On-chip memory bandwidth
  - 70 TB/s from M20Ks on Stratix 10 (excluding MLABs)
  - 140 TB/s from register files and shared memories on V100

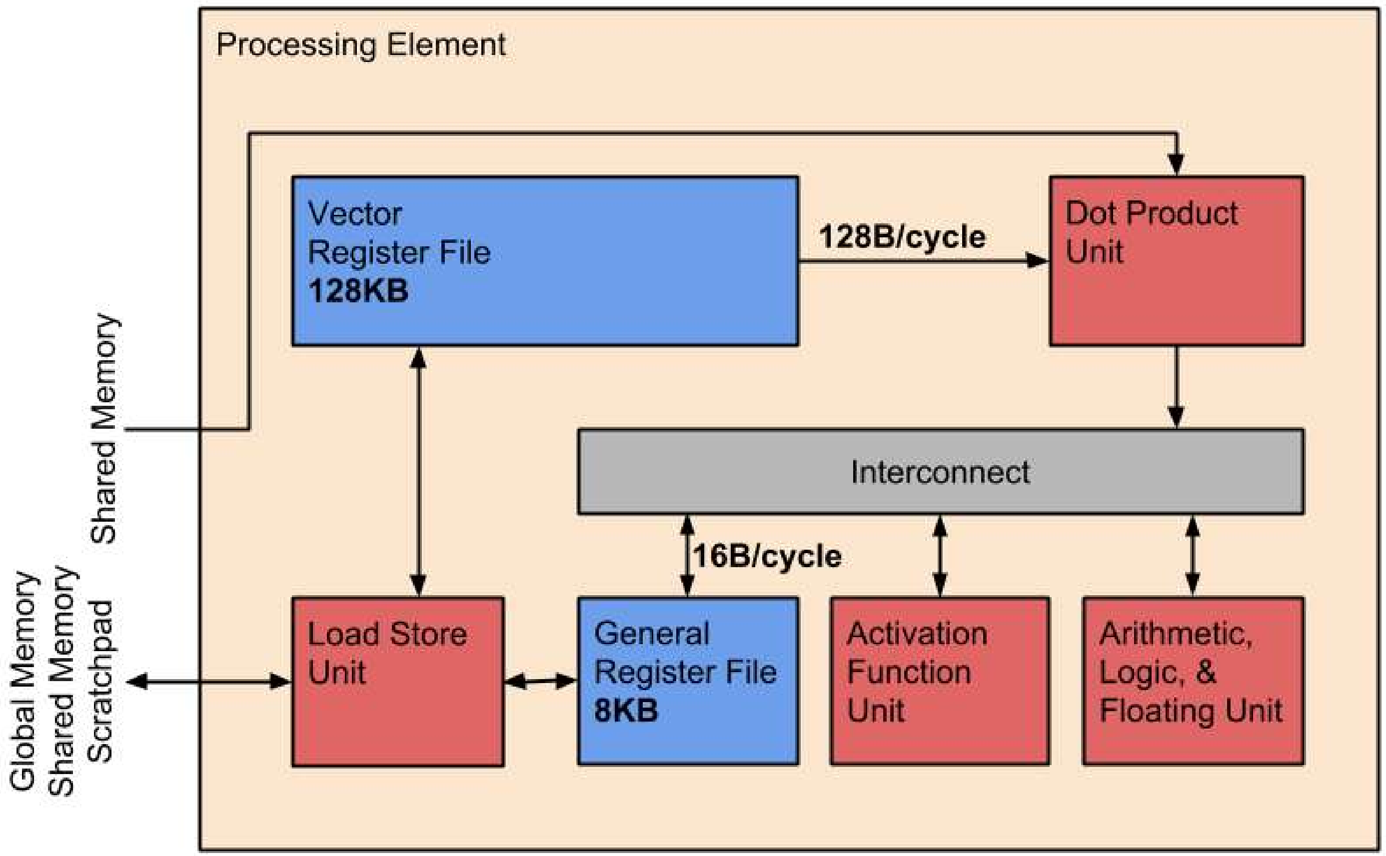
PDL-FGPU Architecture: Chip



PDL-FGPU Architecture: Compute Unit



PDL-FGPU Architecture: Processing Element



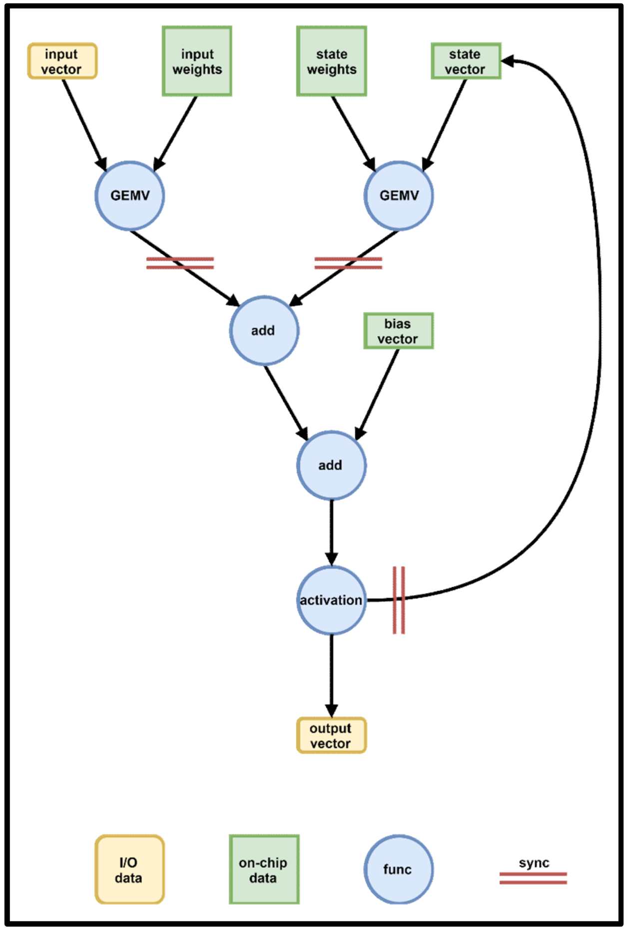
PDL-FGPU Estimated Resource Usage

- DSPs
  - 4,480 DSPs (35 per processing element)
  - 78% utilization on Stratix 10 (280)
- M20Ks
  - ~10,000 M20Ks (~8000 in the vector regfiles and ~500 in the shared memory)
  - ~85% utilization on Stratix 10 (280)
- ALMs
  - ~700,000 ALMs
  - ~75% utilization on Stratix 10 (280)

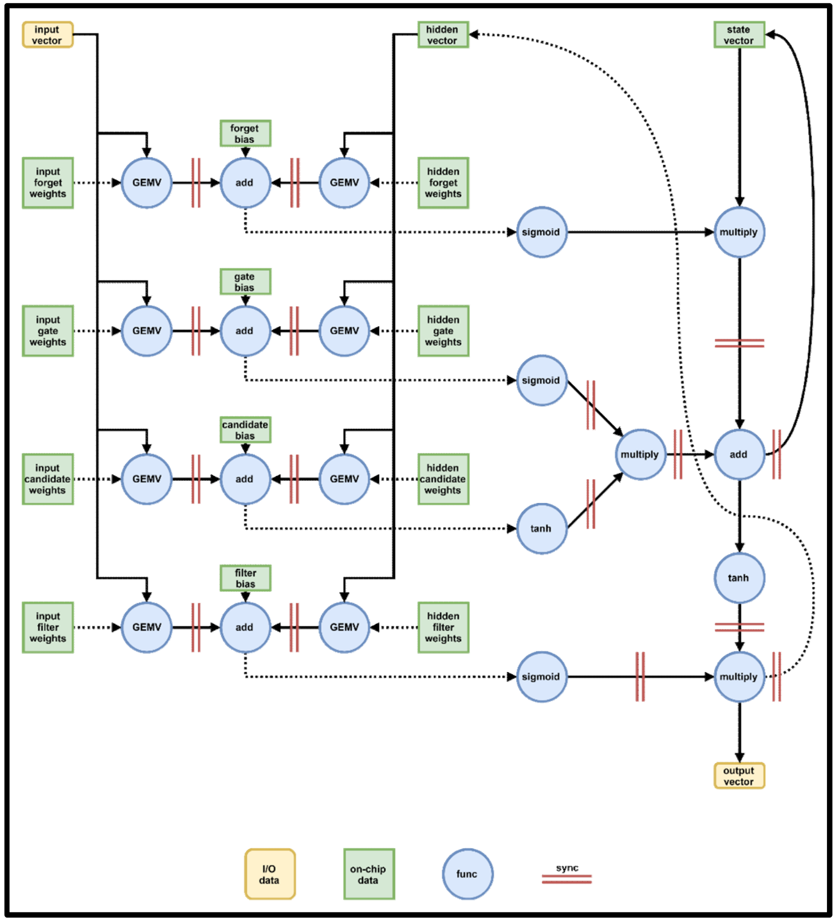
PDL-FGPU Estimated Performance

- INT8
  - $16\ CUs \times 8\ vector\ ops \times 128 \frac{MACCs}{vector\ op} \times 2 \frac{ops}{MACC} \times 500\ MHz \times 50\% \ \frac{instrs}{kernel}$
    $= 8\ INT8\ TOPs$
- FP32
  - $16\ CUs \times 8\ vector\ ops \times 32 \frac{FFMAs}{vector\ op} \times 2 \frac{ops}{FFMA} \times 500\ MHz \times 50\% \ \frac{instrs}{kernel}$
    $= 2\ FP32\ TOPs$

Deep Learning Dataflow: RNN



Deep Learning Dataflow: LSTM



Deep Learning Dataflow: GRU