A Dog by Any Other Name Would Run Just as Slow

Computational and Hardware Complexity in Software Defined Radio

John S. Brunhaver II

School of Electrical Computer Energy Engineering
Fulton Schools of Engineering
Arizona State University

Feb 2nd 2020

Algorithms change to suit their hardware context

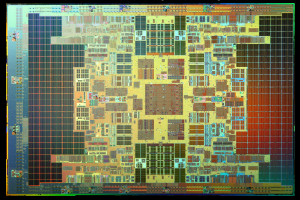
ASU
Ira A.Fulton Schools of Engineering
Arizona State University

Who am I, to say this?

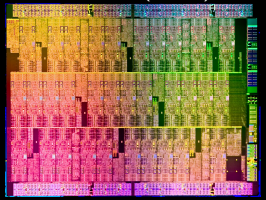
I am a digital hardware engineer

ASU
Ira A.Fulton Schools of Engineering
Arizona State University

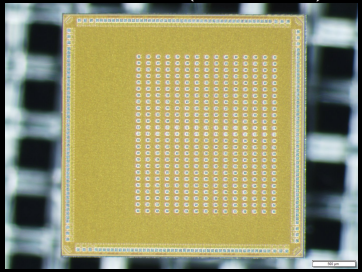
Tukwila (65nm)



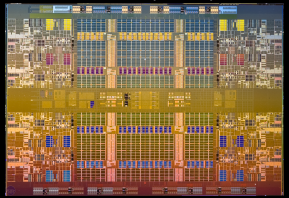
Larrabee (45nm)



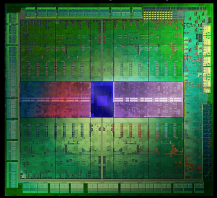
ASU-SNL-TC1 (12nm FinFet)



Beckton (32nm)

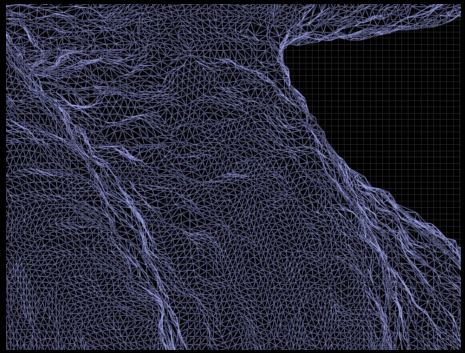


Kepler (28nm)

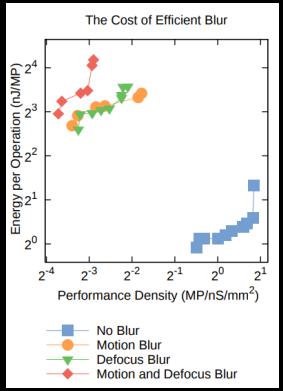


I started research with hw/sw co-design

ASU
Ira A.Fulton Schools of Engineering
Arizona State University

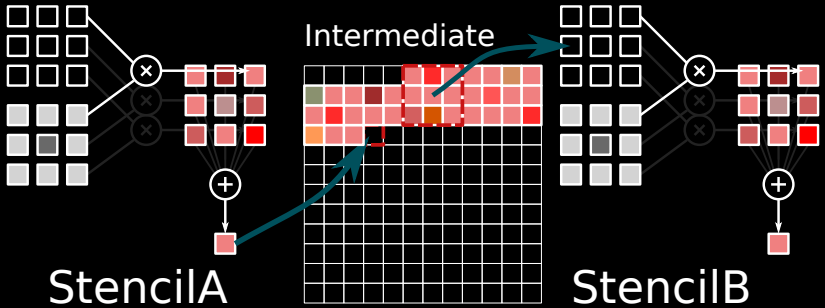


ack. Kayvon Fatahalian



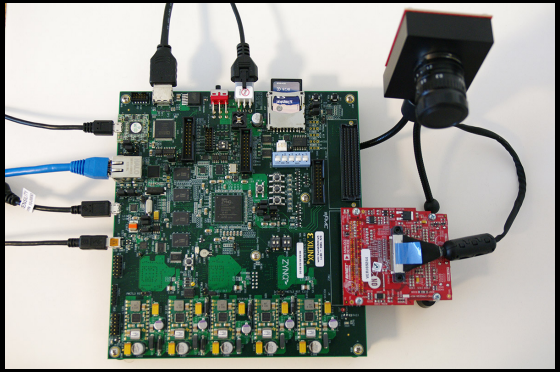
- Brunhaver, John S., et. al. "Hardware implementation of micropolygon rasterization with motion and defocus blur." HPG 2010.
- Fatahalian, Kayvon, et al. "Data-parallel rasterization of micropolygons with defocus and motion blur." HPG 2009

I Moved to DSL to RTL for Image Processing



- John, S. Brunhaver. Design and optimization of a stencil engine. Diss. Stanford University, 2015.
- Hegarty, James, et al. "Darkroom: compiling high-level image processing code into hardware pipelines." ACM Trans. Graph. 33.4 (2014): 144-1.
- https://halide-lang.org/

We demoed on FPGAs



- Pu, Jing, et al. "Programming heterogeneous systems from an image processing DSL." ACM Transactions on Architecture and Code Optimization (TACO) 14.3 (2017): 1-25.
- https://github.com/jingpu/Halide-HLS
- Xilinx Zynq-7000 SoC ZC702 Evaluation Kit

Currently we work on the general problem

- Given a set of code written by domain experts, can you
  - **Identify** kernels in an applications
  - Procedurally **label** kernels, identifying equivelance
  - Discover the **taxonomy** of kernels
  - **Predict and arbitrage** accelerators and their organization

Currently we work on the general problem

- Given a set of code written by domain experts, can you
  - **Identify** kernels in an applications
  - Procedurally **label** kernels, identifying equivelance
  - Discover the **taxonomy** of kernels
  - **Predict and arbitrage** accelerators and their organization
  - Produce FPGA code

We currently focus on discovering kernels in code

Static 

- Preprint – "Automated Parallel Kernel Extraction from Dynamic Application Traces" – http://arxiv.org/abs/2001.09995
- CodeOcean – https://codeocean.com/capsule/2cb73b4e-11f9-4547-8fe3-4b4956d3d251/tree
- Git – https://github.com/ruhrie/TraceAtlas

We currently focus on discovering kernels in code

Static

- Preprint – "Automated Parallel Kernel Extraction from Dynamic Application Traces" – http://arxiv.org/abs/2001.09995
- CodeOcean – https://codeocean.com/capsule/2cb73b4e-11f9-4547-8fe3-4b4956d3d251/tree
- Git – https://github.com/ruhrie/TraceAtlas

We currently focus on discovering kernels in code

Static

Dynamic

- Preprint – "Automated Parallel Kernel Extraction from Dynamic Application Traces" – http://arxiv.org/abs/2001.09995
- CodeOcean – https://codeocean.com/capsule/2cb73b4e-11f9-4547-8fe3-4b4956d3d251/tree
- Git – https://github.com/ruhrie/TraceAtlas

We currently focus on discovering kernels in code

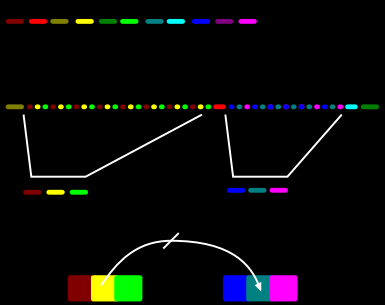
Ira A. Fulton Schools of Engineering
Arizona State University

Static

Dynamic

Cluster

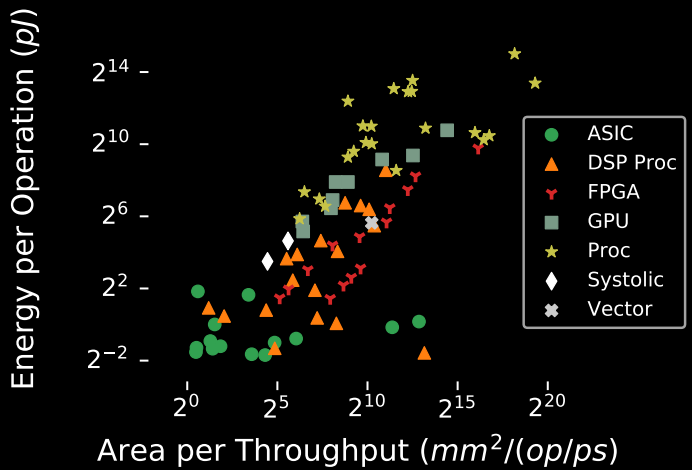
Data Flow



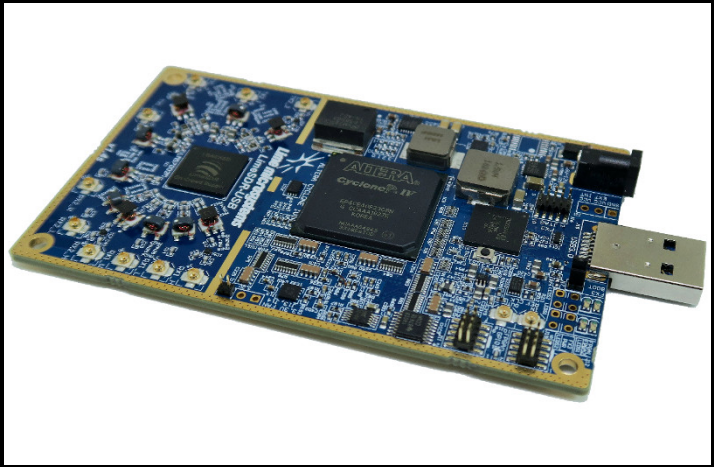
- Preprint – "Automated Parallel Kernel Extraction from Dynamic Application Traces" – http://arxiv.org/abs/2001.09995
- CodeOcean – https://codeocean.com/capsule/2cb73b4e-11f9-4547-8fe3-4b4956d3d251/tree
- Git – https://github.com/ruhrie/TraceAtlas

FPGAs are just like puppies

FPGAs are just like puppies



FPGAs are just like puppies



Now you have a puppy – Systems, atoms, and accelerators

ASU
Ira A.Fulton Schools of Engineering
Arizona State University

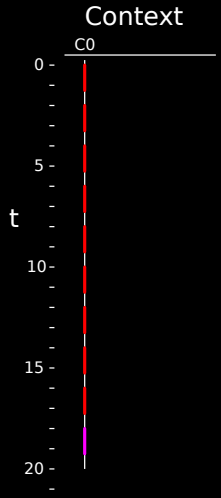
- System Pitfalls
  - Feed -and- Care of a Dog
- Structural FPGA Design
  - Atoms of traick
- Modelling Accelerator Structure
  - Putting the trick together

Now you have a puppy – Systems, atoms, and accelerators

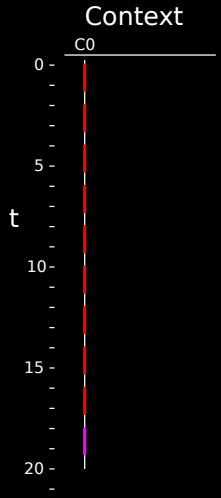
ASU
Ira A.Fulton Schools of Engineering
Arizona State University

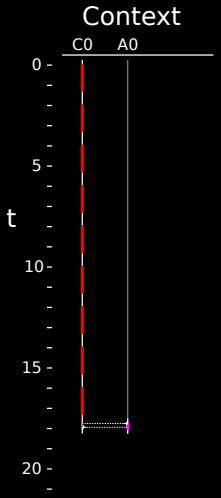
- **System Pitfalls**
  - Feed -and- Care of a Dog
- Structural FPGA Design
  - Atoms of traick
- Modelling Accelerator Structure
  - Putting the trick together

Understand your application's composition

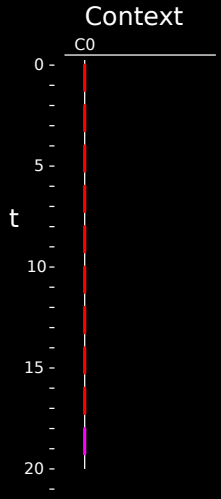


Understand your application's composition

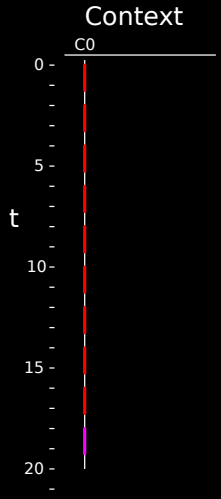




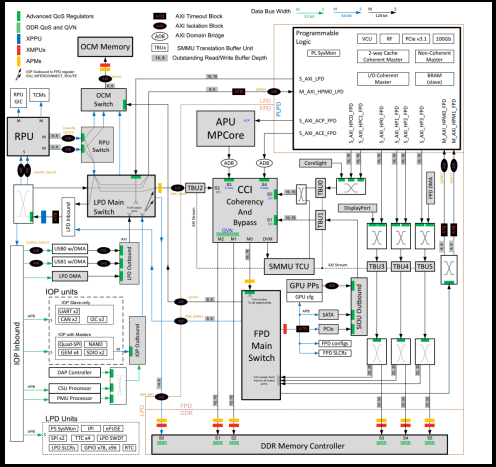
Optimize what matters



Optimize what matters

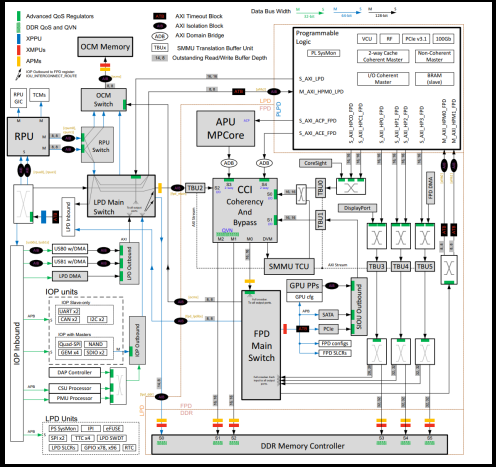


Understand your sources of latency

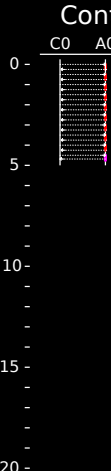


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

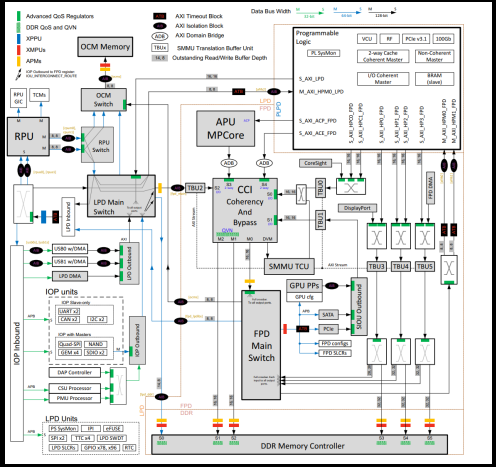
Understand your sources of latency



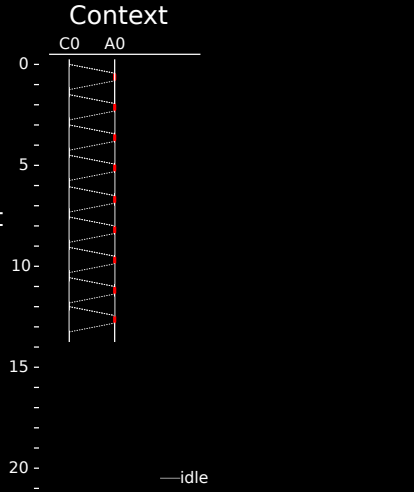
Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019



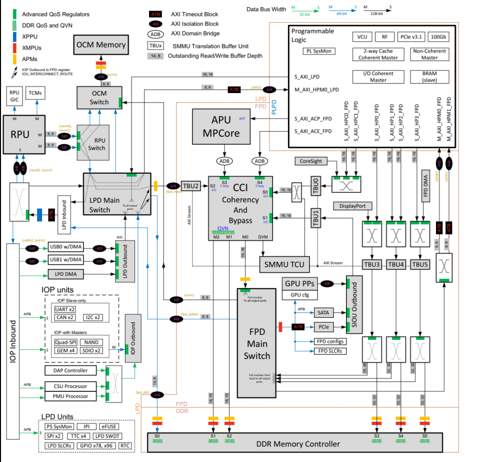
Understand your sources of latency



Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019



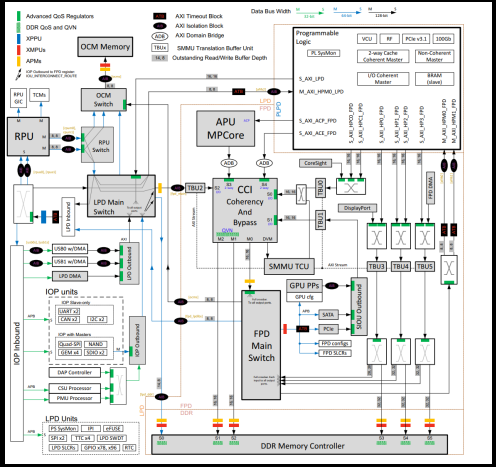
Understand your sources of latency



Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

For Dramatic effect only ...

Understand your sources of latency

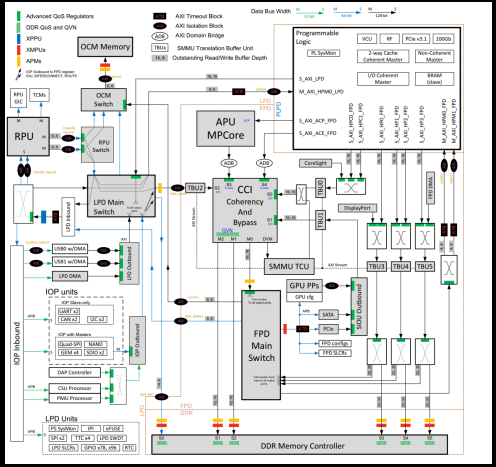


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver

Understand your sources of latency

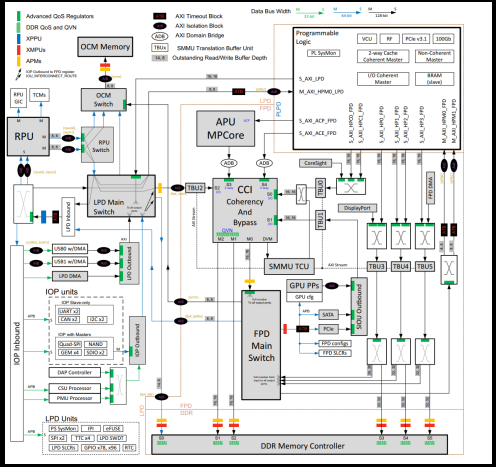


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush

Understand your sources of latency

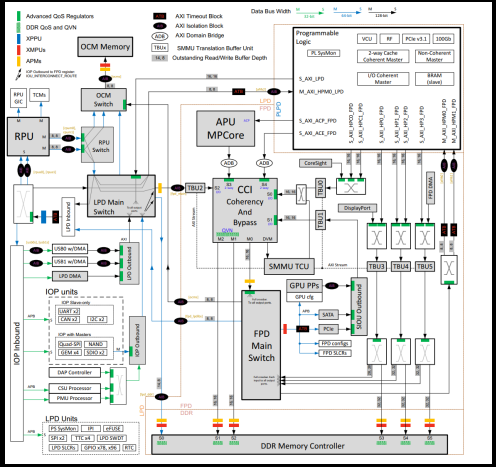


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy

Understand your sources of latency

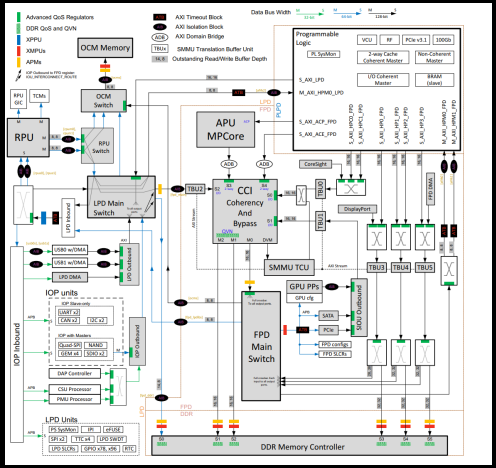


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA

Understand your sources of latency

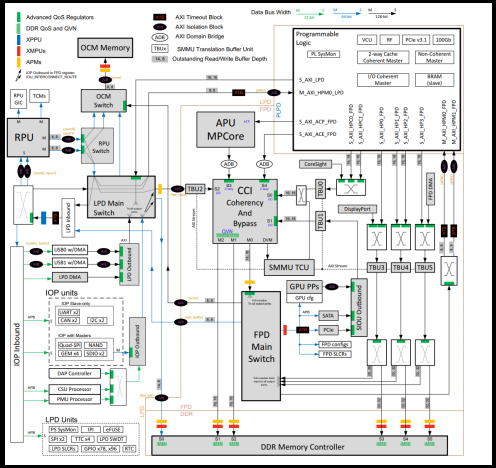


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete

Understand your sources of latency

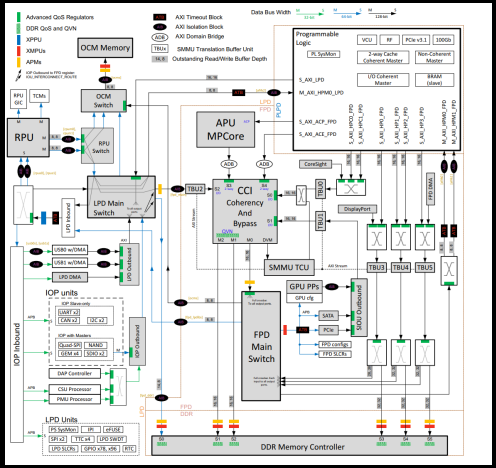


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel

Understand your sources of latency

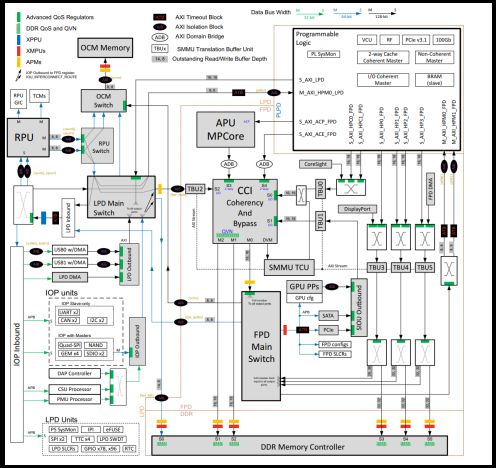


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel
- Driver - Poll Accel for complete

Understand your sources of latency

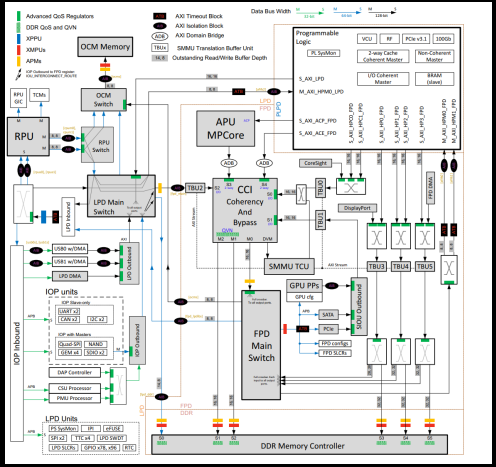


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1)
August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel
- Driver - Poll Accel for complete
- Driver - Command DMA

Understand your sources of latency

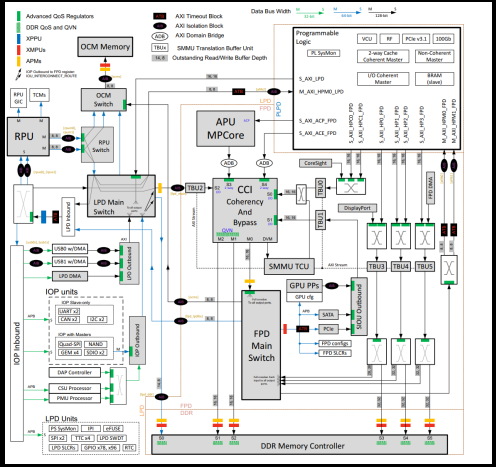


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel
- Driver - Poll Accel for complete
- Driver - Command DMA
- Driver - Poll DMA

Understand your sources of latency

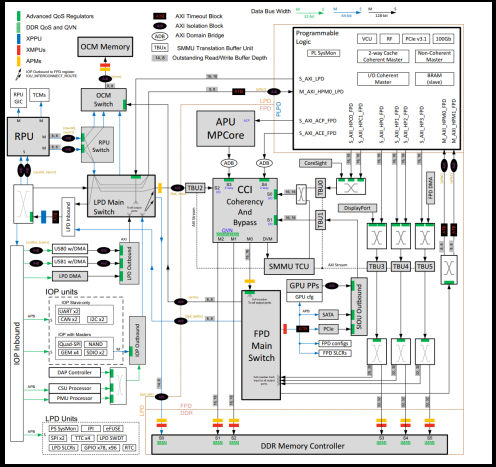


Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel
- Driver - Poll Accel for complete
- Driver - Command DMA
- Driver - Poll DMA
- OS - Restore

Understand your sources of latency



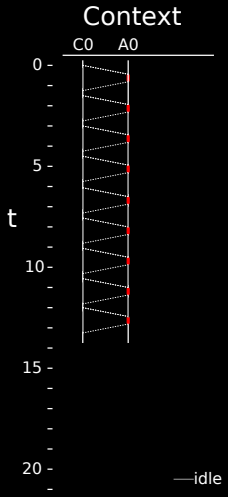
Zynq UltraScale+ Device Technical Reference Manual UG1085 (v2.1) August 21, 2019

For Dramatic effect only ...

- User - Blocking OS Call to use driver
- OS - Forced cache flush
- Driver - Forced data copy
- Driver - Send command to DMA
- Driver - Poll DMA for complete
- Driver - Command Accel
- Driver - Poll Accel for complete
- Driver - Command DMA
- Driver - Poll DMA
- OS - Restore

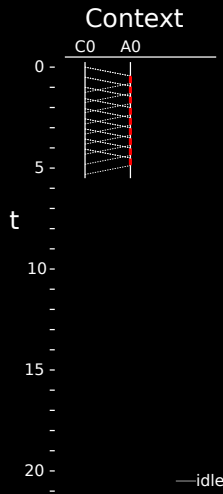
Hide your latency

- Overlap transfer and operation
- Sequence operation in PL
- Execute and overlap many operations



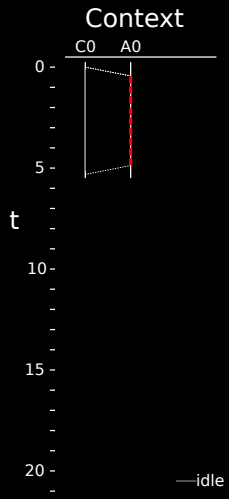
Hide your latency

- Overlap transfer and operation
- Sequence operation in PL
- Execute and overlap many operations
- Avoid blocking, stream instead
  - Skip False memory barriers
  - True memory barriers



Hide your latency

- Overlap transfer and operation
- Sequence operation in PL
- Execute and overlap many operations
- Avoid blocking, stream instead
  - Skip False memory barriers
  - True memory barriers



Understand your applications data-rate

Kernel	Expression	Compute	Bandwidth	Ratio
Axpy	$y_i = \alpha x_i + b$	N	N	1
Inner Product	$z = \sum_{i=1}^{N} x_i y_i$	N	N	1
FIR-1D	$y_i = \sum_{l=1}^{M} \alpha_l x_{i-k}$	NM	N	M
FIR-2D	...	N^2M^2	N^2	M^2
FIR-3D	...	N^3M^3	N^3	M^3
Bubble Sort	$y_i : y_{i+1} \geq y_i \geq y_{i-1}$	N^2	N	N
GEMM	$z_{i,j} = \sum_{k=1}^{N} x_{i,k} y_{i,k}$	N^3	N^2	N
FFT	$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi}{N} kn}$	$NlogN$	N	logN

Understand your applications data-rate

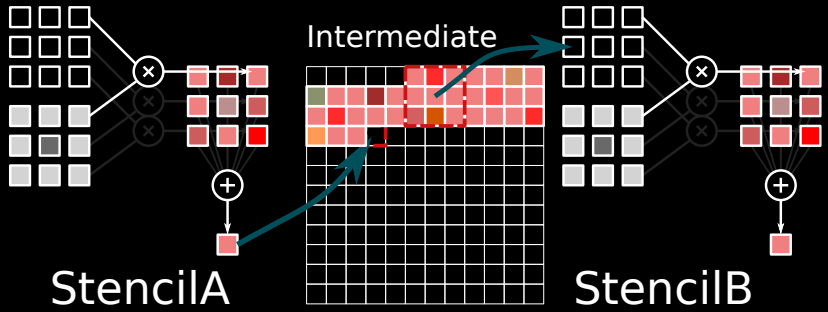
Resource	Count	Unit rate	Total bandwidth
DRAM Ports	2	$128b \cdot 300MHz$	77 Gbps
BRAM Ports	$2k$	$32b \cdot 600MHz$	38 Tbps
DSP	$2k$	$3 \cdot 32b \cdot 600MHz$	114 Tbps
Reg	$0.5M$	$1 \cdot 600MHz$	300 Tbps

Roughly – Zynq UltraScale+ XCZU9EG-2FFVB1156 MPSoC

Exploit locality

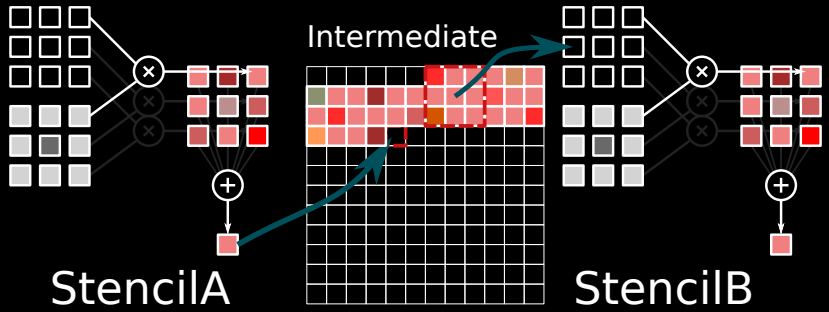
- Avoid the memory wall
  - Registers mitigate band-limit in BRAM
  - BRAM mitigates band-limit in DRAM
  - Avoid writing back intermediates

Exploit locality



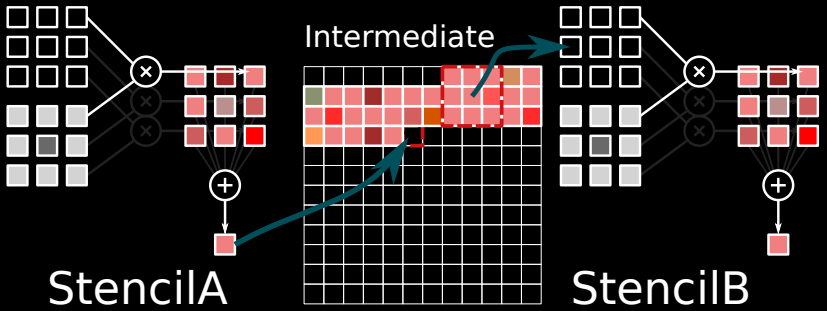
Exploit locality

Ira A.Fulton Schools of Engineering
Arizona State University



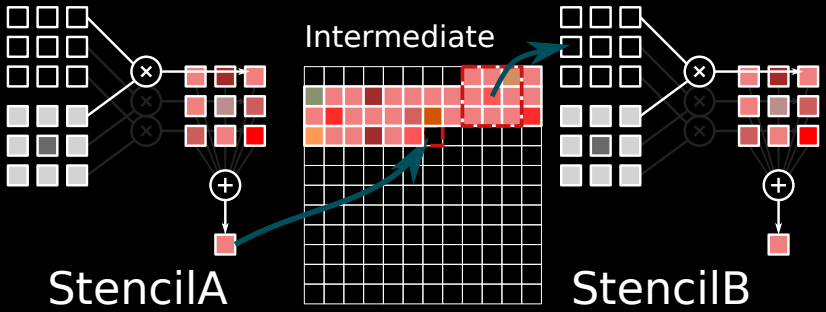
Exploit locality

Ira A. Fulton Schools of Engineering
Arizona State University

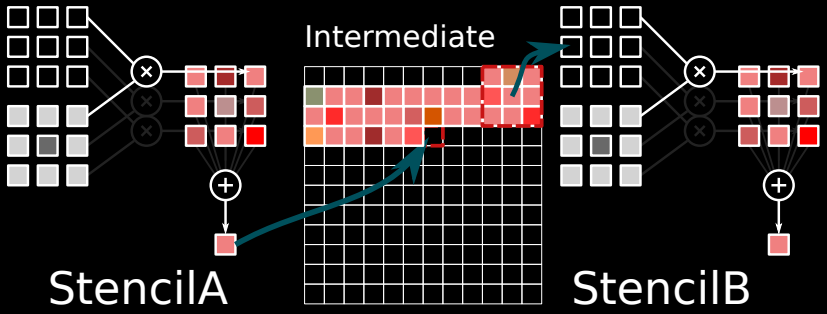


Exploit locality

Ira A.Fulton Schools of Engineering
Arizona State University

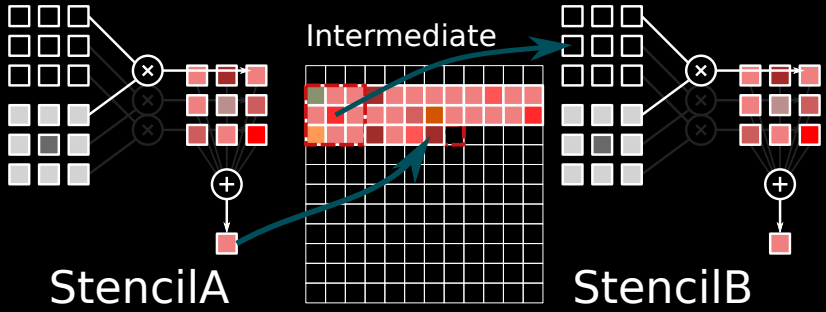


Exploit locality

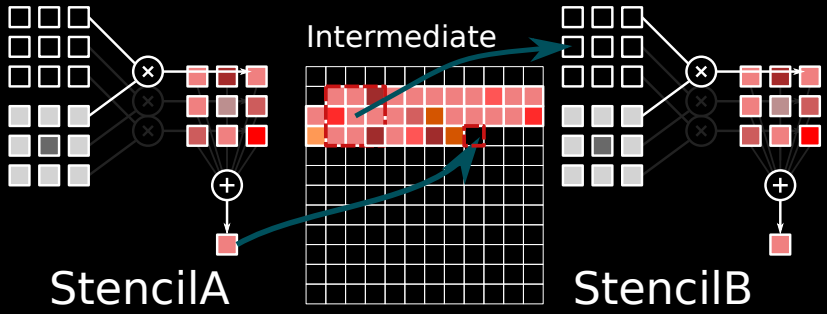


Exploit locality

Ira A.Fulton Schools of Engineering
Arizona State University



Exploit locality



Train actions then tricks – Understand FPGA structures

Ira A.Fulton Schools of Engineering
Arizona State University

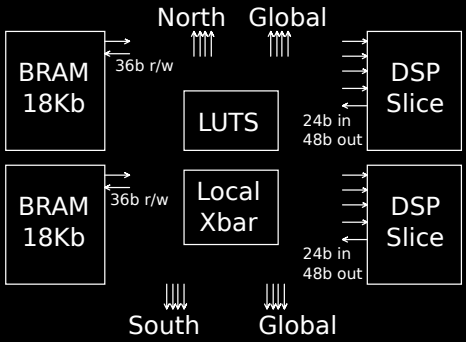
- System Pitfalls
  - Feed -and- Care of a Dog
- Structural FPGA Design
  - Atoms of traick
- Modelling Accelerator Structure
  - Putting the trick together

Train actions then tricks – Understand FPGA structures

ASU
Ira A.Fulton Schools of Engineering
Arizona State University

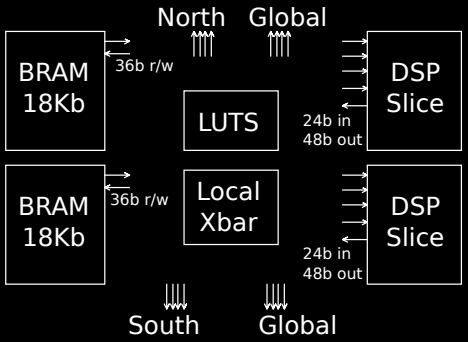
- System Pitfalls
  - Feed -and- Care of a Dog
- **Structural FPGA Design**
  - Atoms of traick
- Modelling Accelerator Structure
  - Putting the trick together

Think structurally



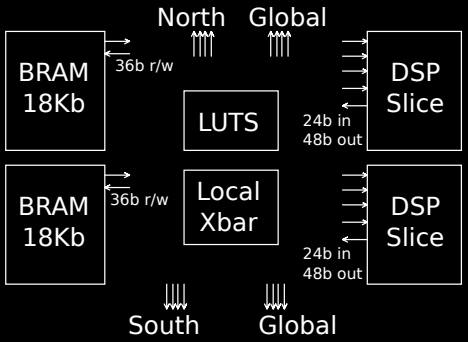
- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing

Think structurally



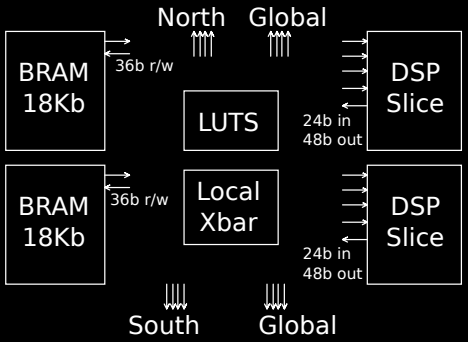
- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing
- Minor misconceptions

Think structurally



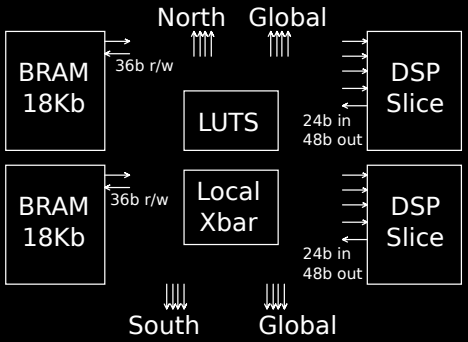
- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing
- Minor misconceptions
  - Not a Verilog accelerator

Think structurally



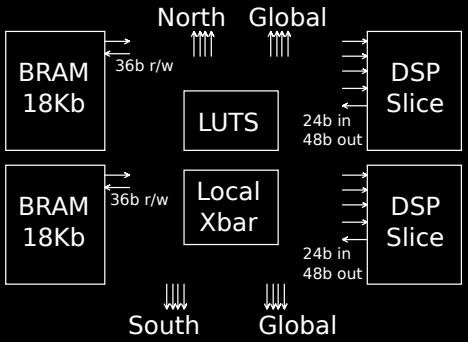
- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing
- Minor misconceptions
  - Not a Verilog accelerator
  - Slices, not LUTs and Regs

Think structurally



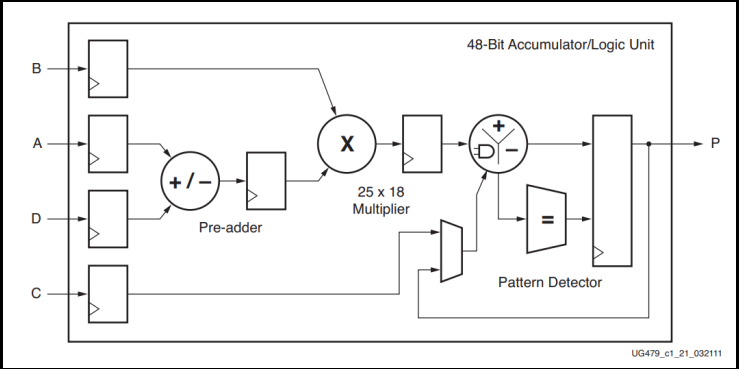
- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing
- Minor misconceptions
  - Not a Verilog accelerator
  - Slices, not LUTs and Regs
  - Wires are expensive, not gates

Think structurally



- Resources
  - BRAM - 2× 18kb / 36b RW
  - DSP - 2× 24b MAC
  - LUTs - 300× 5:2 fx
  - Regs - 300× 1b state
  - Routing
- Minor misconceptions
  - Not a Verilog accelerator
  - Slices, not LUTs and Regs
  - Wires are expensive, not gates
  - Global wires are worse

Pipeline your DSP



UltraScale Architecture DSP Slice User Guide UG579 (v1.9) September 20, 2019

Pipeline your DSP

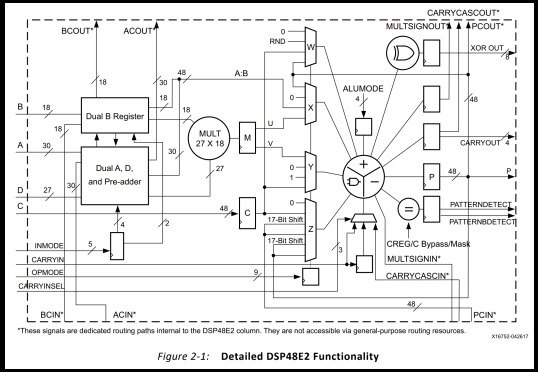
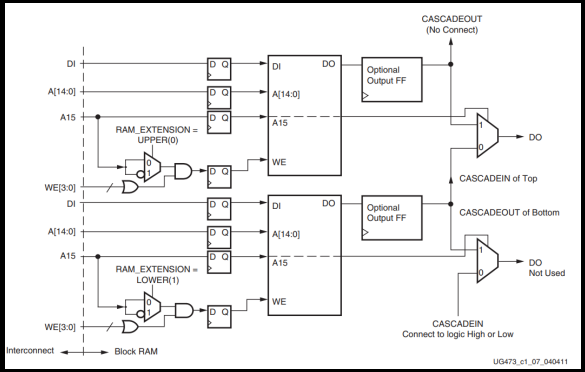


Figure 2-1: **Detailed DSP48E2 Functionality**

UltraScale Architecture DSP Slice User Guide UG579 (v1.9) September 20, 2019

Pipeline your BRAM



UltraScale Architecture Memory Resources User Guide UG573 (v1.10) February 4, 2019

Maximize resource utilization

- Assets:
  - DSP - Rate, Location
  - BRAM - Capacity, Ports, Bandwidth, Location
  - DRAM - Bandwidth, Ports
- Goals:
  - Spatial utilization ($70 - 90\%$)
  - Temporal utilization ($> 99\%$)
  - Minimize regret

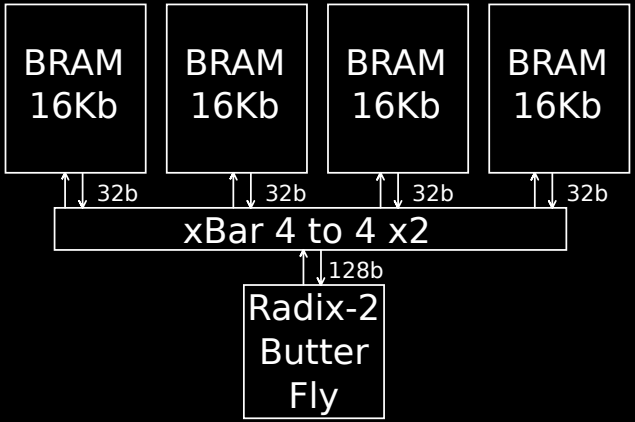
Plan a trick from atoms – Predict Accelerator Structure

- System Pitfalls System Pitfalls
  - Feed -and- Care of a Dog
- Structural FPGA Design
  - Atoms of traick
- Modelling Accelerator Structure
  - Putting the trick together

Plan a trick from atoms – Predict Accelerator Structure

- System Pitfalls System Pitfalls
  - Feed -and- Care of a Dog
- Structural FPGA Design
  - Atoms of traick
- **Modelling Accelerator Structure**
  - Putting the trick together

In Place FFT


BRAM
16Kb
BRAM
16Kb
BRAM
16Kb
BRAM
16Kb
32b
32b
32b
32b
xBar 4 to 4 x2
128b
Radix-2
Butter
Fly

Make a "good" FOSS Verilog library

ASU
Ira A.Fulton Schools of Engineering
Arizona State University

- Utilize best practices from software
  - Compact API
  - Parametric calls
  - Well documented
  - Unit tested
  - Benchmarked

Putting it all together

- Optimize what matters
- Hide your latency
- Exploit locality
- Think structurally

Putting it all together

- Optimize what matters
- Hide your latency
- Exploit locality
- Think structurally
- Algorithms change to suit their hardware context

Thank you

Questions, Thoughts, Concerns, Hopes, Dreams, Desires?