DYNAMIC HARDWARE CONVOLUTION ACCELERATION DESIGN

Designed/Presented by David Cain

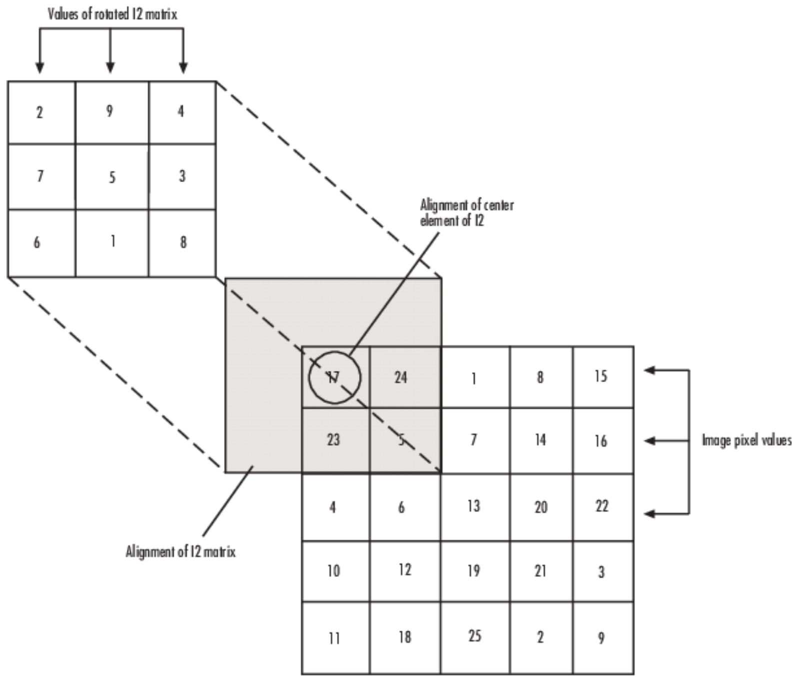
Special thanks to Omar Eddash and Adam Frost for mentorship

Overview

Design Objective

Develop an FPGA based reconfigurable design to accelerate matrix convolution

- This is a computationally intensive step used in many image processing algorithms
  - can benefit from hardware acceleration
- The design must accept multiple data types
  - Integer data values
  - Floating point data values
  - Fixed point data values
- Each of these data types can be dynamically reconfigured on device


Values of rotated I2 matrix
Alignment of center element of I2
Image pixel values
Alignment of I2 matrix

Intro to Vivado

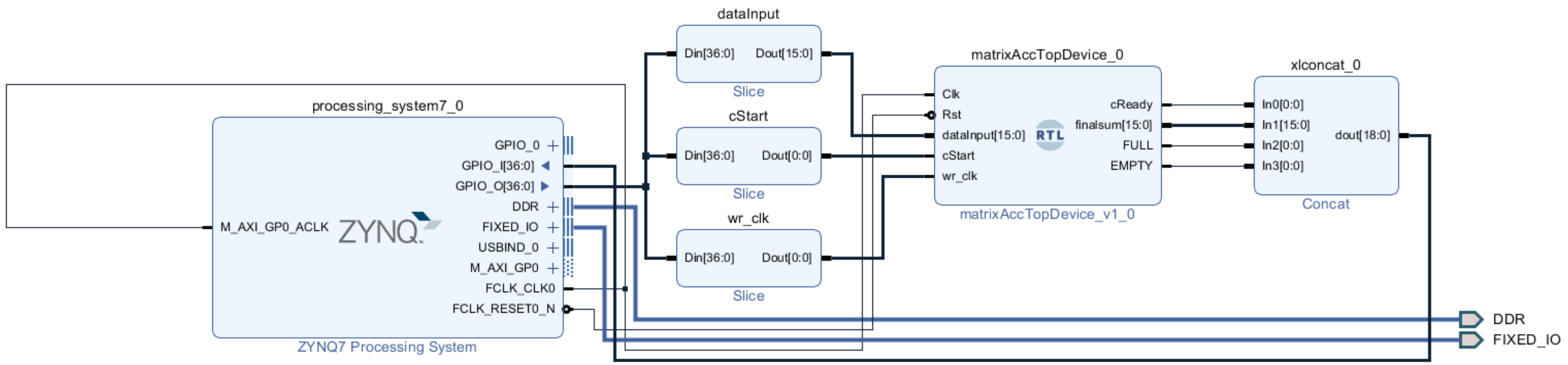
- Vivado is the primary software suite used when working with Xilinx based FPGAs such as the Pynq-Z2

- Vivado offers design development using HDL or the built in IP Integrator using block diagrams
  - These block diagrams can compose of either base IPs provided by Xilinx or user custom IPs that are imported/created within the project
  - The HDL defining these Xilinx based IPs are not viewable by end-user

- Often both development forms are used when working on a design

- Example of block diagram interface

  **For synthesis, block diagrams must be packaged with HDL wrappers generated by Vivado*

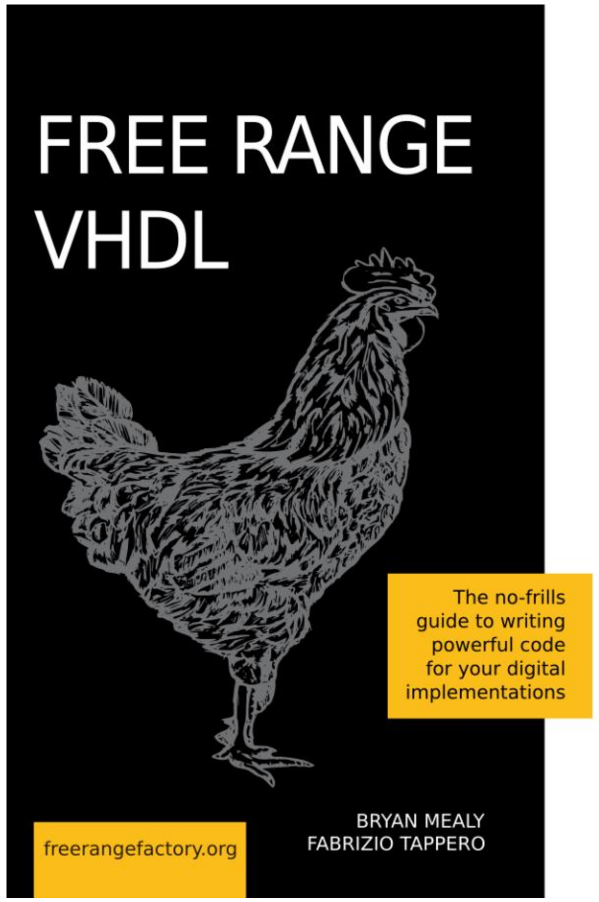


Learn Verilog

- Upon entering the project, I was referenced this book and several tutorials to get a feeling of Verilog
  - *Free Range VHDL*, Bryan Mealy, 2018

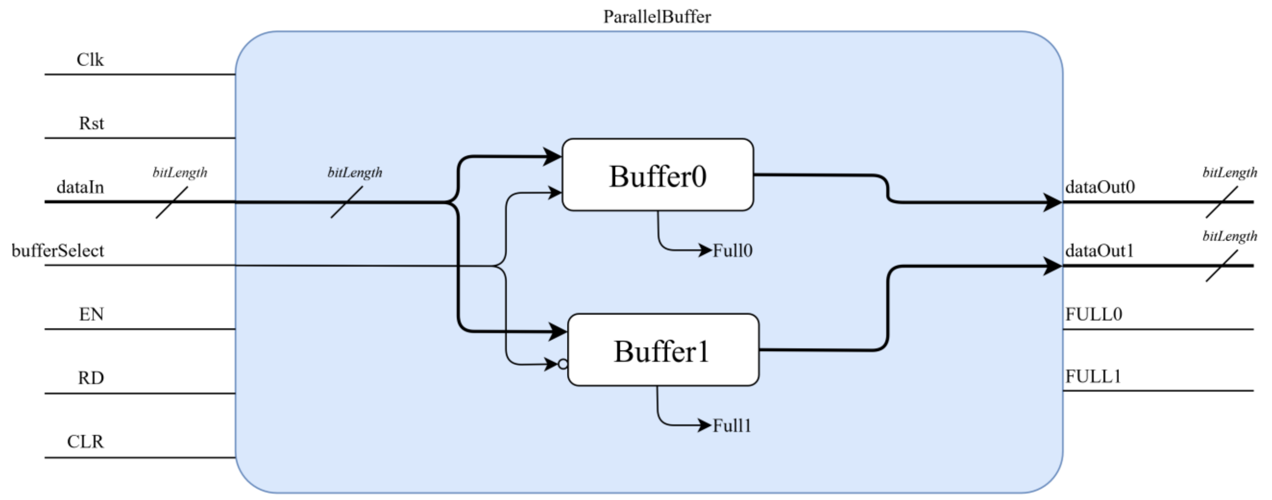
- Xilinx also has tutorials with pre-built designs to be imported and demonstrate concepts of how to use the hardware available

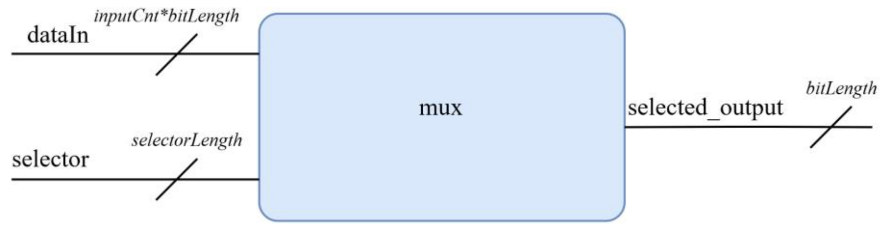
- YouTube tutorials were also helpful for workflow within Vivado


FREE RANGE VHDL
The no-frills guide to writing powerful code for your digital implementations
freerangefactory.org
BRYAN MEALY
FABRIZIO TAPPERO

- With a simple understanding of Verilog, basic designs were created and tested:

1.	Multibit flipflop register	(FlipFlop.v)
2.	Parallel multibit flipflop	(ParallelBuffer.v)
3.	Multibit port MUX	(mux.v)
4.	Variable data slicer	(dataSplit.v)



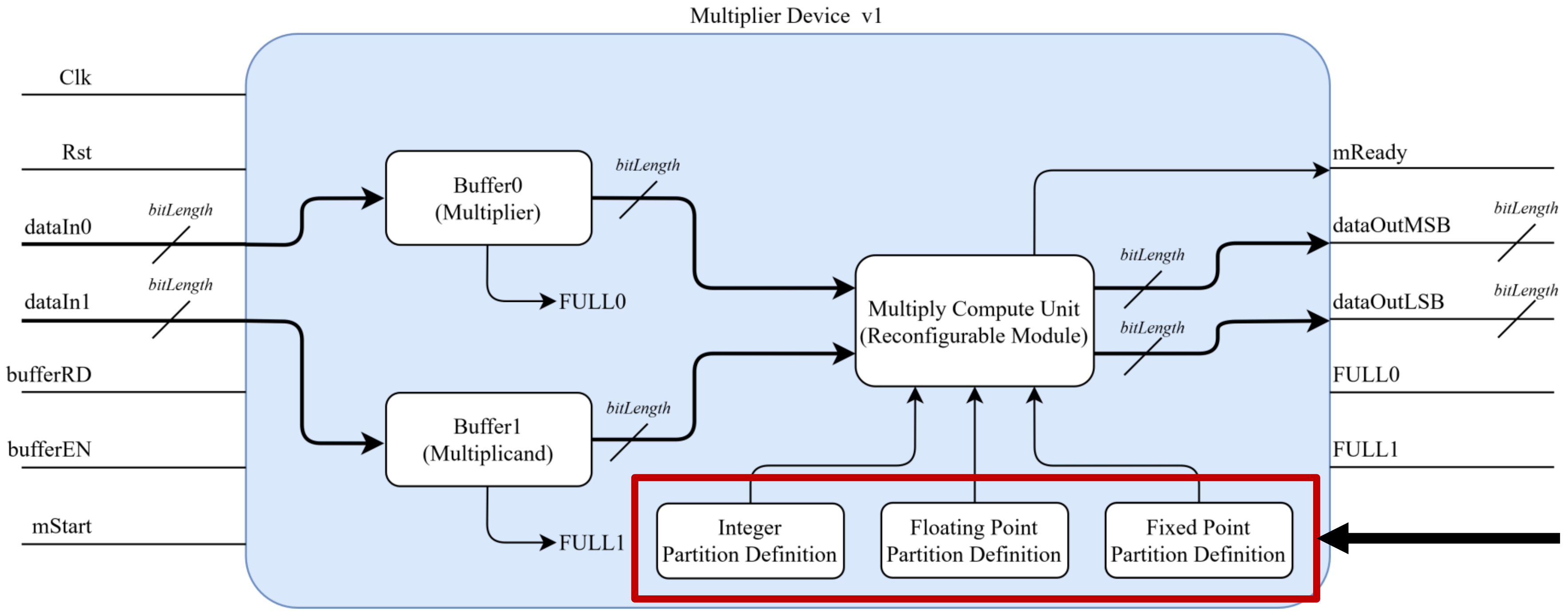


Partial Reconfig Multipliers

- A first step was to implement devices that could intake 2 data values, multiply them, then return the product
  - This needed to be designed for three separate data types, with separate algorithms to complete each computation type

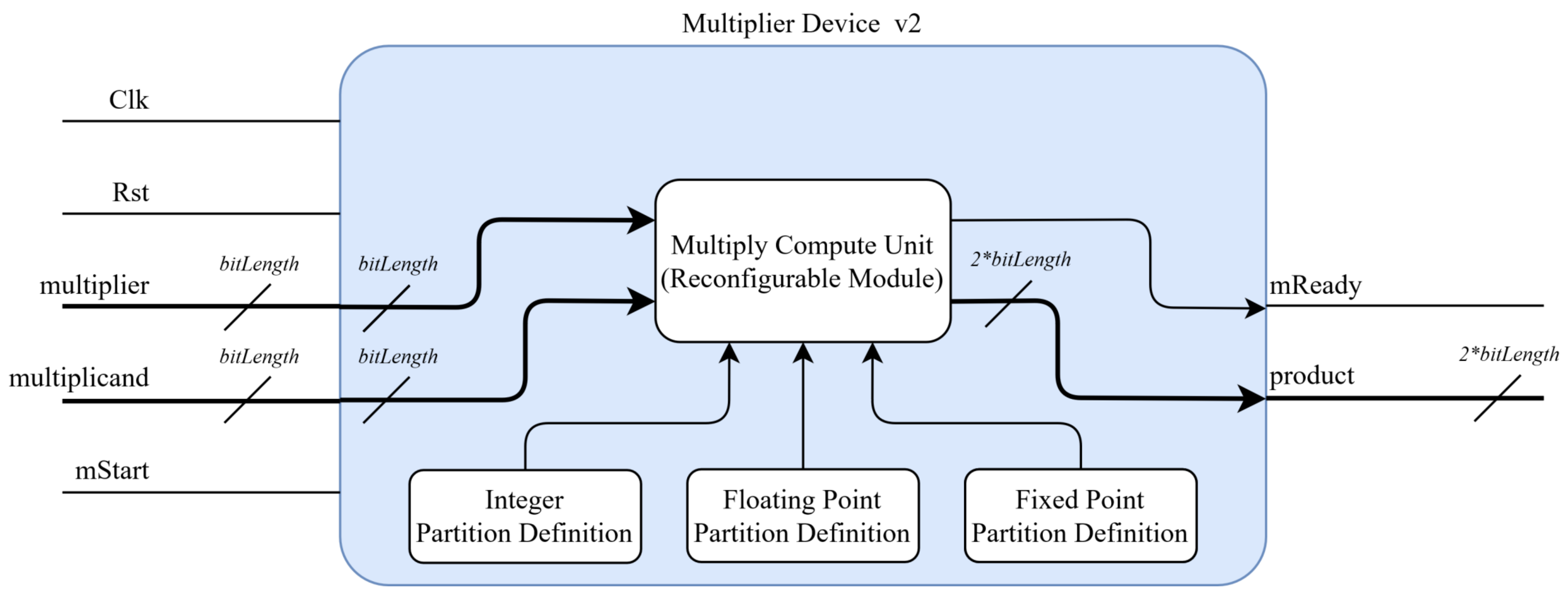
- This was initially designed to buffer input data internally, allowing for parallel computations. The buffered input was removed due to adding undesirable delay while pipelining data streams

- Block diagram of multiplier v1



Reconfigurable Partition Definitions

- Block diagram of multiplier v2



Crossbar Data Switch

- A data link was needed to interface many parallel multiplier outputs to many parallel adder devices

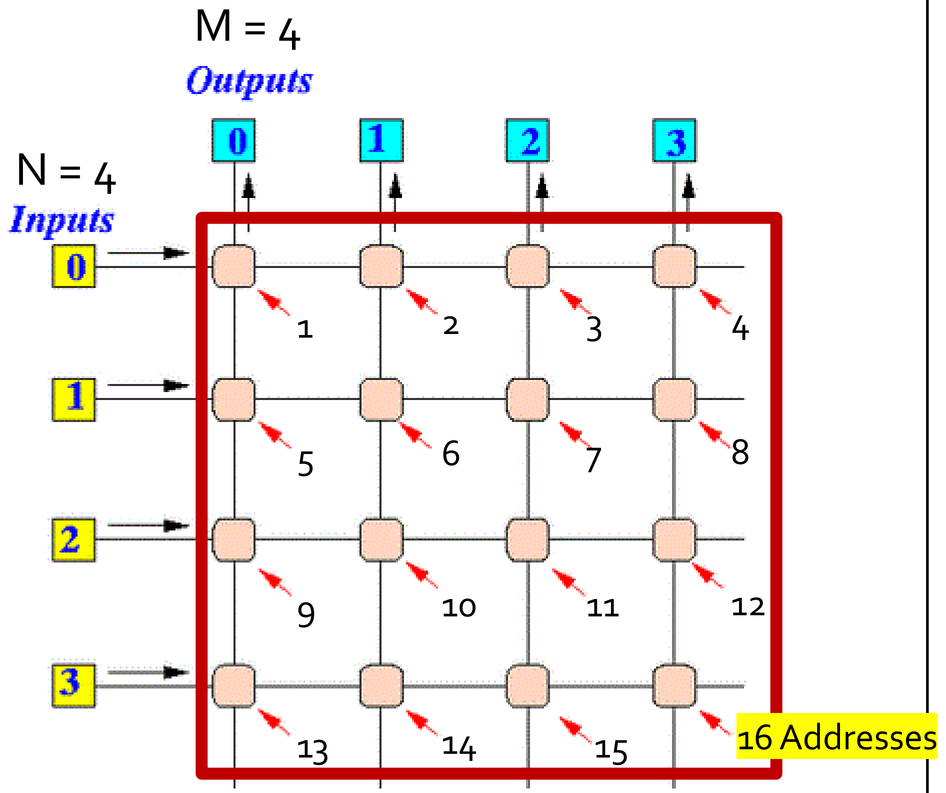
- A crossbar switch was designed and implemented. This switch allows for any input port to be selected and connected to any output port.
  - The crossbar was written from scratch to have variable input and output port counts
  - These ports are also variable bit length

- The data connection is asynchronous, mimicking wire connections

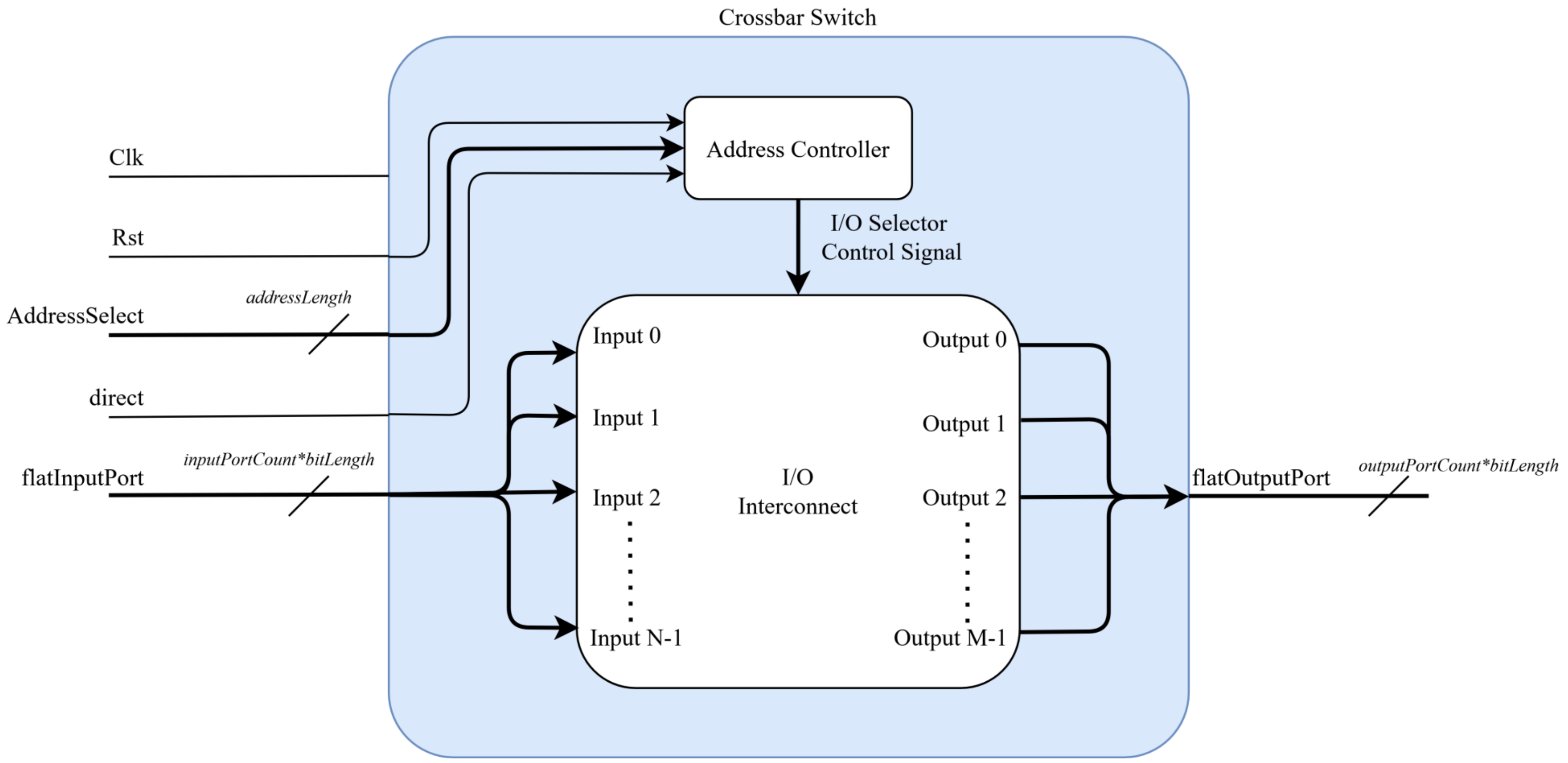
- The crossbar operates on a grid connection method. With N inputs and M outputs, this generates NxM valid address selections

- At positive edge clocks, the address on AddressSelect port is toggled

- If direct is HIGH, equal I/O ports will be connected.
  - i.e. Output0 = Input0; Input1 = Output1;…



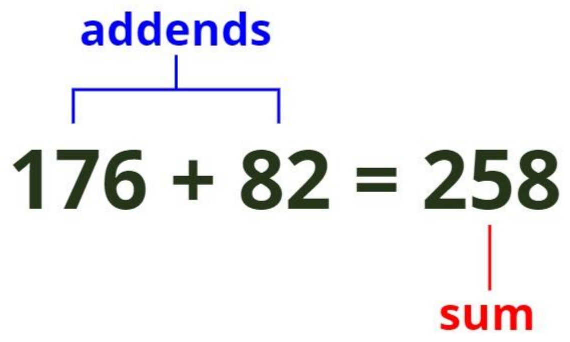
- Block diagram of crossbar



Partial Reconfig Adder

- With a method to now compute many products in parallel and sort them, a device was needed to accumulate the sum of products

- 2 Partition definitions were created:
  - Integer/Fixed Point Adder
  - Floating Point Adder

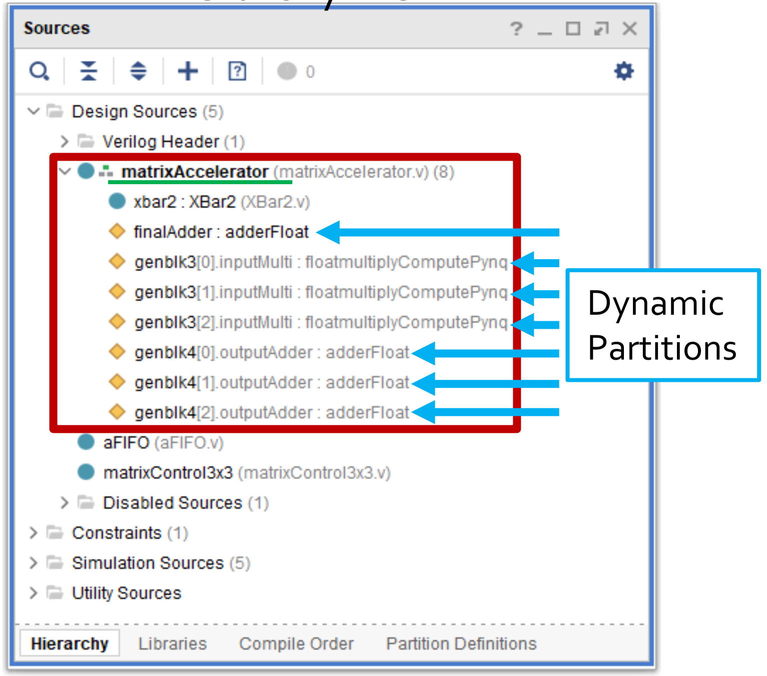


- Floating point needs much more logic due to needing to check the magnitude difference, then normalize the values

Matrix Accelerator

- The devices were then packaged into a single IP, matrixAcclerator
  - This device has a lot of I/O and no control logic
  - It must be packaged with a controller IP

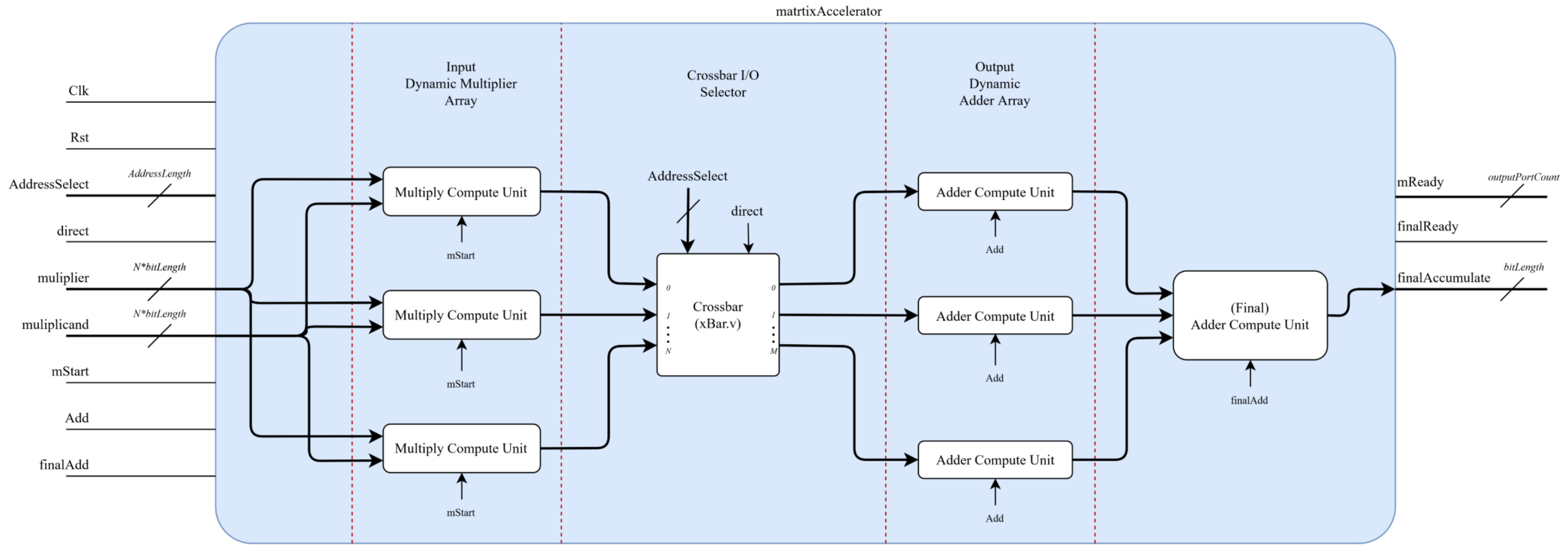
Hierarchy View


Sources

Design Sources (5)
- Verilog Header (1)
- matrixAccelerator (matrixAccelerator.v) (8)
  - xbar2 : XBar2 (XBar2.v)
  - finalAdder : adderFloat
  - genblk3[0].inputMulti : floatmultiplyComputePynq
  - genblk3[1].inputMulti : floatmultiplyComputePynq
  - genblk3[2].inputMulti : floatmultiplyComputePynq
  - genblk4[0].outputAdder : adderFloat
  - genblk4[1].outputAdder : adderFloat
  - genblk4[2].outputAdder : adderFloat
- aFIFO (aFIFO.v)
- matrixControl3x3 (matrixControl3x3.v)
- Disabled Sources (1)

Constraints (1)
Simulation Sources (5)
Utility Sources

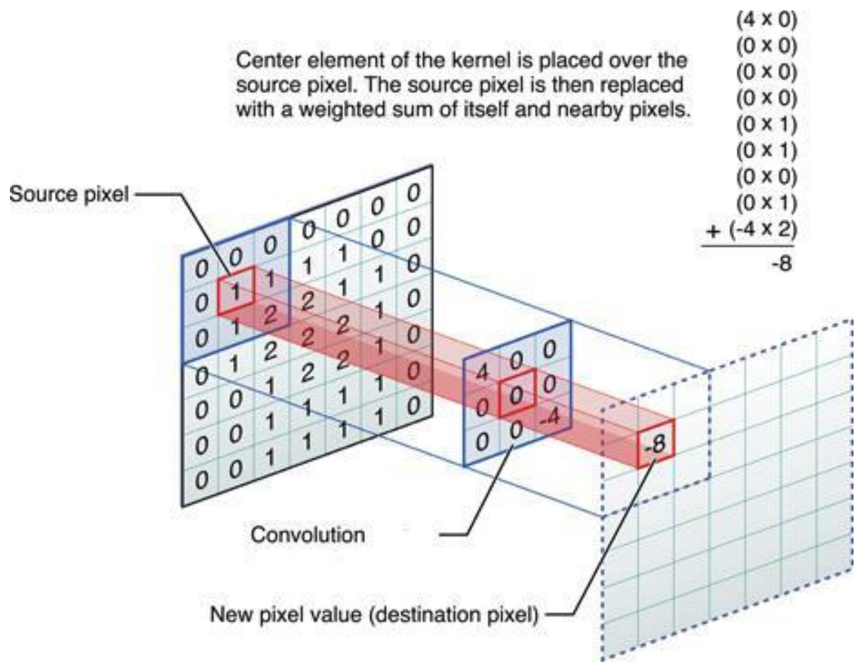
Block diagram of matrix accelerator

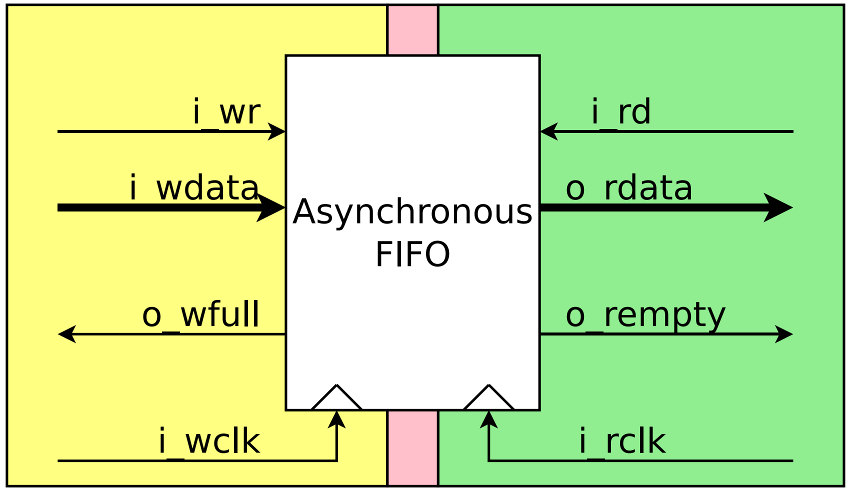


Asynchronous FIFO

- The convolution accelerator needs much data input
  - Data Set (Image Values)
  - Filter Set (Filter Values)

- Since the program will not be able to load values and operate the FPGA properly, an asynchronous FIFO buffer was used to input a data set across the clock domains

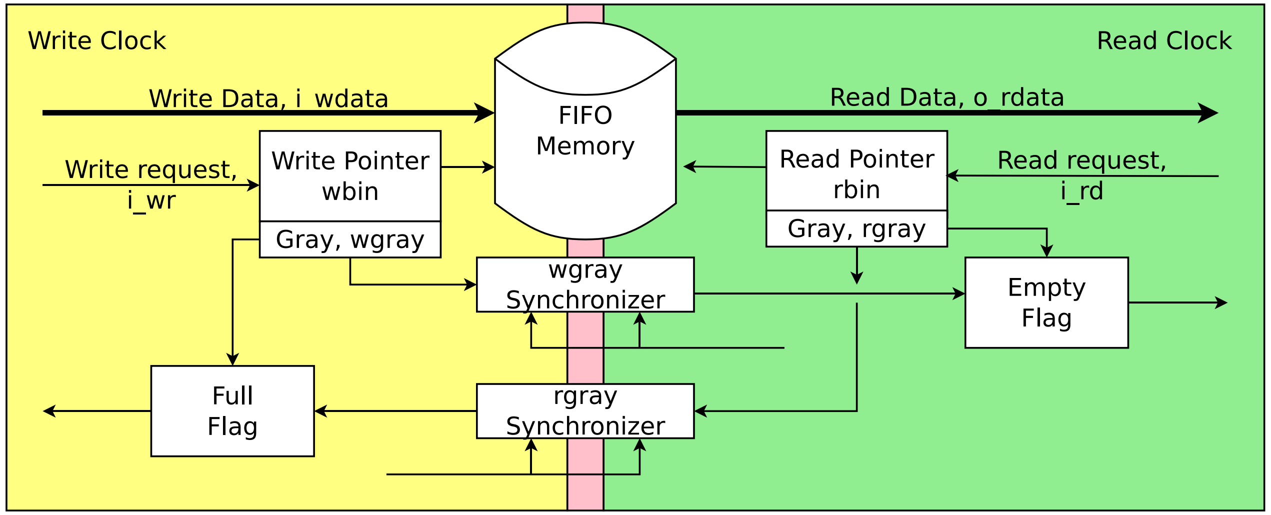

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution
New pixel value (destination pixel)



- There was a helpful online project:
  - *Crossing Clock Domains with an Asynchronous FIFO*, Dan Gisselquist, 2018
  - https://zipcpu.com/blog/2018/07/06/afifo.html

- I attempted to write this IP from scratch, but the problem was difficult

- This write-up by Gisselquist was an extensive overview of the device problems, behavior, and similar work

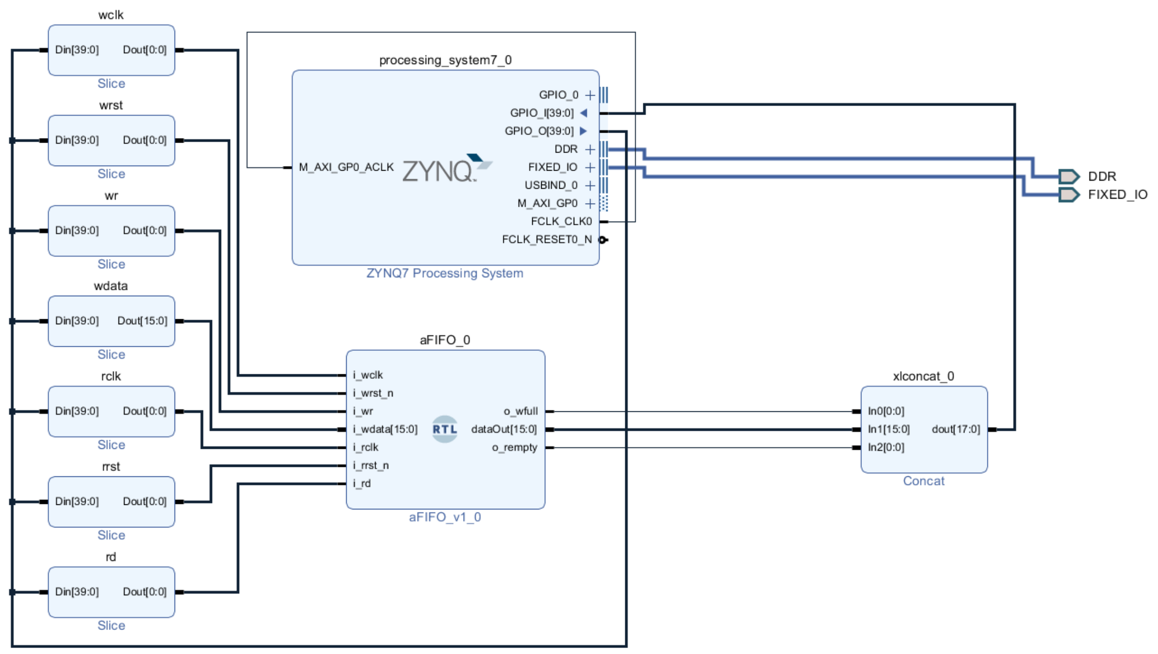


- To verify this FIFO design was suitable, it was first tested in Vivado simulation
- Next, a bitstream and block diagram was generated to program the FPGA with the Python3 interface

- In the Python environment, the buffer would be filled with a random data set

- Next, the buffer data would be extracted and compared with original input

- The load/compare times were also tracked



Matrix Controller

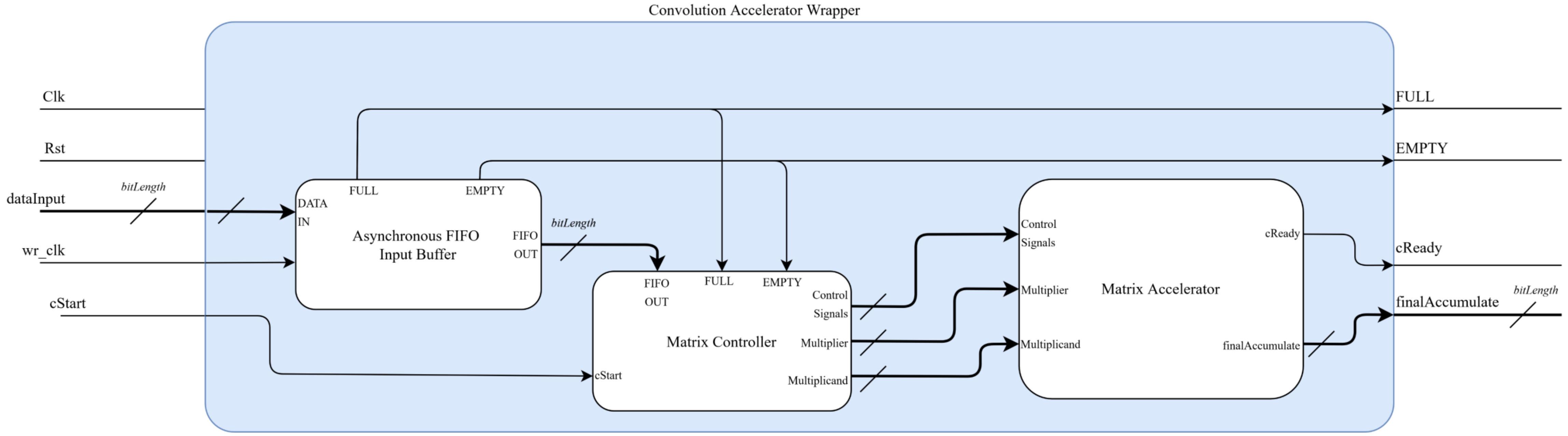
- Now with hardware to allow for external data input, a state machine controller is needed to complete computation

- The controller needs to:
  - Read data in the input buffer
  - Pipeline data into the matrix accelerator
  - Use matrix accelerator control signals to generate convolution sum

- The controller has three main states:
  - Reading data from the buffer
  - Multiplying the input values
  - Add the resulting products

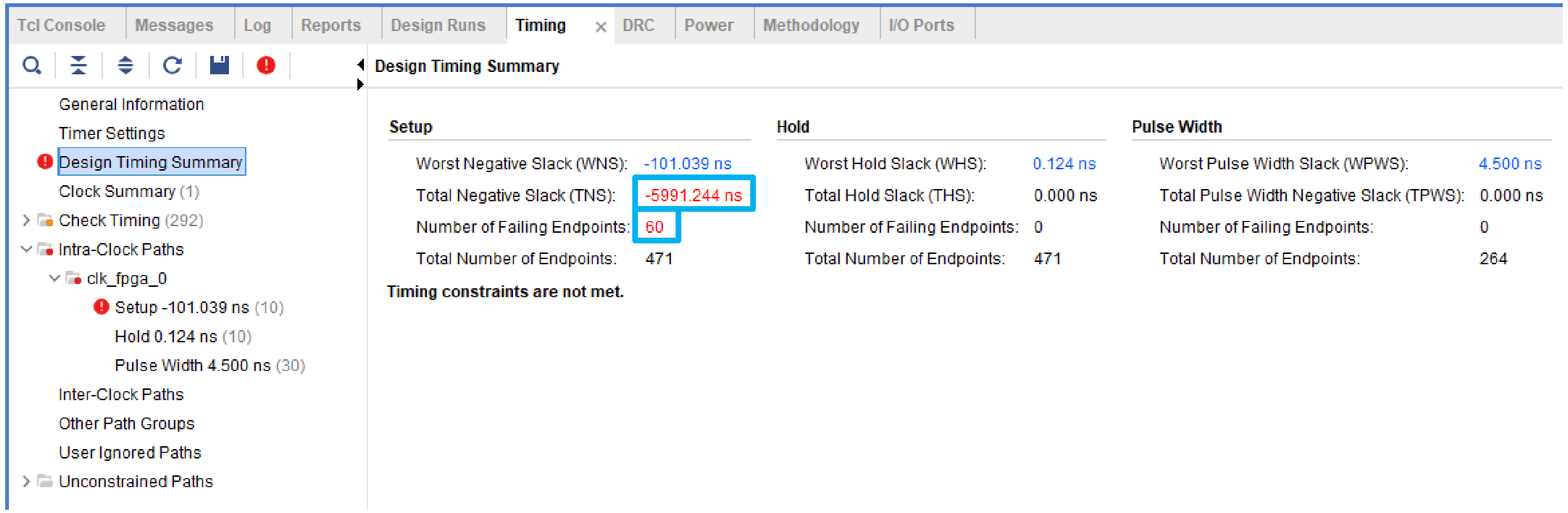
Convolution Accelerator Top Wrapper

- A top-level package was created to wrap the input buffer, matrix accelerator, and controller into one design
- Convolution Accelerator block diagram
- This device has three dynamic configurations: Integer, Floating Point, Fixed



- This package is currently being prototyped and fails the implementation phase due to timing issues

- Timing issues are generally caused by unstable signals or undesirable combinational logic delay


Tcl Console | Messages | Log | Reports | Design Runs | Timing | DRC | Power | Methodology | I/O Ports

General Information
Timer Settings
Design Timing Summary
Clock Summary (1)
Check Timing (292)
Intra-Clock Paths
clk_fpga_0
Setup -101.039 ns (10)
Hold 0.124 ns (10)
Pulse Width 4.500 ns (30)
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-101.039 ns	Worst Hold Slack (WHS):	0.124 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	-5991.244 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	60	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	471	Total Number of Endpoints:	471	Total Number of Endpoints:	264

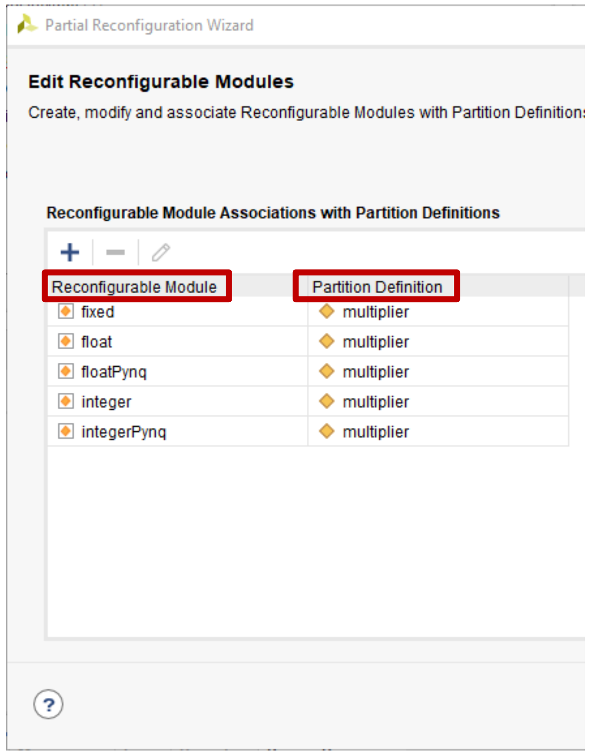
Timing constraints are not met.

Enabling Partial Reconfig

- Using Vivado’s partial reconfiguration wizard, designs can become dynamically reconfigurable

- This allows for partitions of the FPGA to be reprogrammed during runtime

- When converting a project to partially reconfigurable, simulation is no longer available
  - It is recommended to flesh out much of a PR design while static, then create a new project for a dynamic version



Edit Reconfigurable Modules

Create, modify and associate Reconfigurable Modules with Partition Definitions

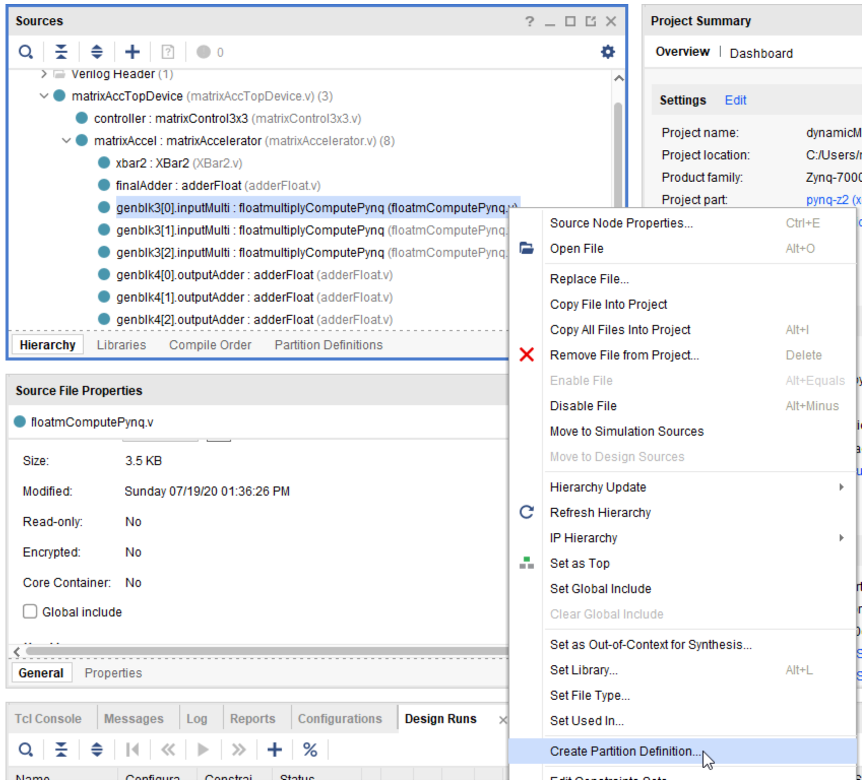
Reconfigurable Module Associations with Partition Definitions

Reconfigurable Module	Partition Definition
fixed	multiplier
float	multiplier
floatPynq	multiplier
integer	multiplier
integerPynq	multiplier

- Backup project

- Enable Partial Reconfiguration
  - Tools->Enable Partial Reconfiguration…

- Select a device in top to create a partition definition



Sources

Verilog Header (1)
matrixAccTopDevice (matrixAccTopDevice.v) (3)
controller : matrixControl3x3 (matrixControl3x3.v)
matrixAccel : matrixAccelerator (matrixAccelerator.v) (8)
xbar2 : XBar2 (XBar2.v)
finalAdder : adderFloat (adderFloat.v)
genblk3[0].inputMulti : floatmultiplyComputePynq (floatmComputePynq.v)
genblk3[1].inputMulti : floatmultiplyComputePynq (floatmComputePynq.
genblk3[2].inputMulti : floatmultiplyComputePynq (floatmComputePynq.
genblk4[0].outputAdder : adderFloat (adderFloat.v)
genblk4[1].outputAdder : adderFloat (adderFloat.v)
genblk4[2].outputAdder : adderFloat (adderFloat.v)

Hierarchy | Libraries | Compile Order | Partition Definitions

Source File Properties

floatmComputePynq.v

Size:	3.5 KB
Modified:	Sunday 07/19/20 01:36:26 PM
Read-only:	No
Encrypted:	No
Core Container:	No

☐ Global include

General | Properties

Tcl Console | Messages | Log | Reports | Configurations | Design Runs

Name | Configura | Constrai | Status

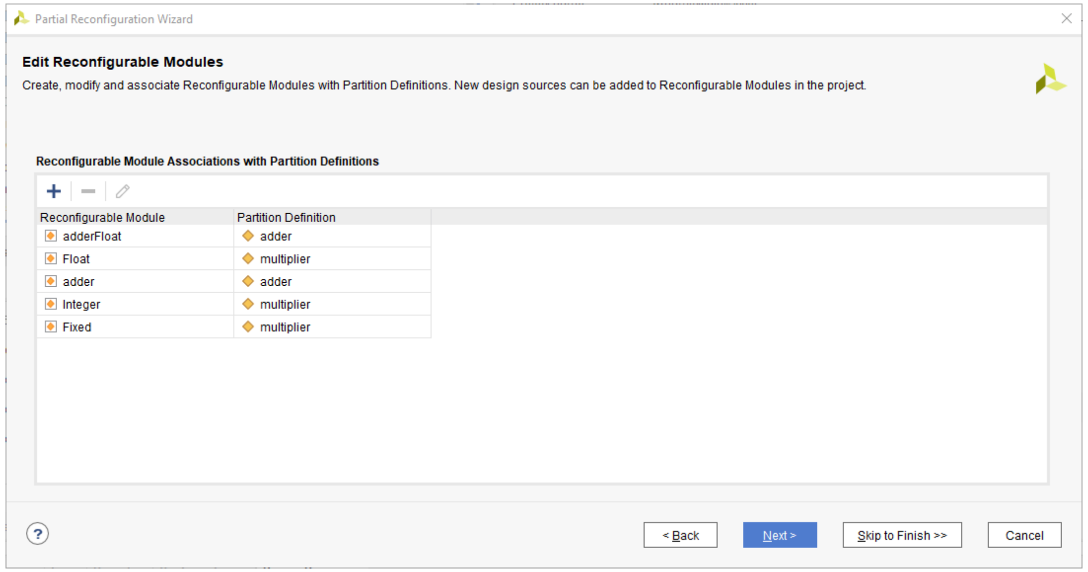
Project Summary

Overview | Dashboard

Settings Edit

Project name:	dynamicM
Project location:	C:/Users/
Product family:	Zynq-700
Project part:	pynq-z2 (x

- Add any needed partitions and modules
- Create a configuration with modules set as needed

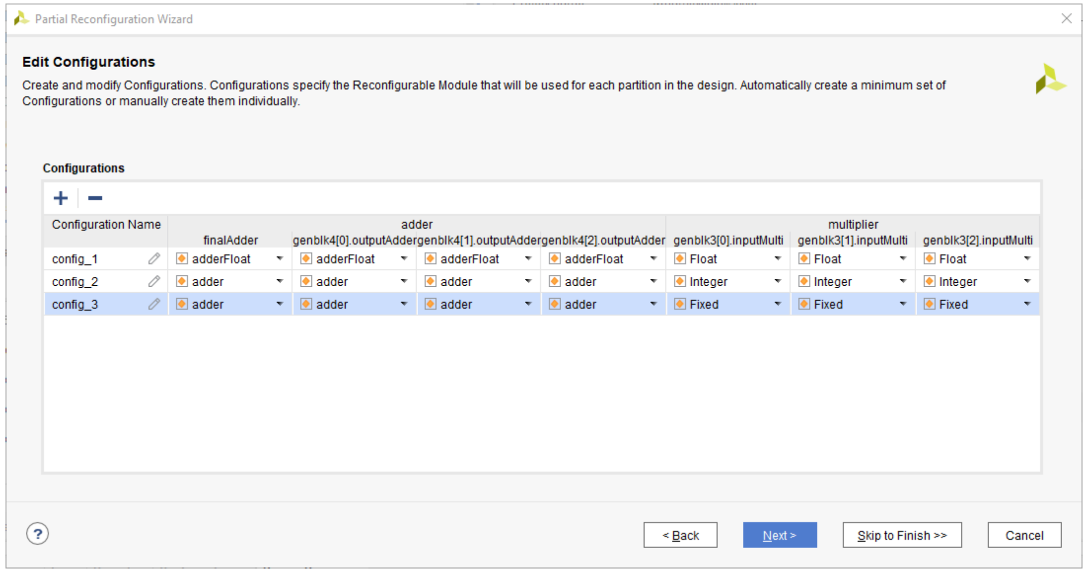


Edit Reconfigurable Modules

Create, modify and associate Reconfigurable Modules with Partition Definitions. New design sources can be added to Reconfigurable Modules in the project.

Reconfigurable Module Associations with Partition Definitions

Reconfigurable Module	Partition Definition
adderFloat	adder
Float	multiplier
adder	adder
Integer	multiplier
Fixed	multiplier



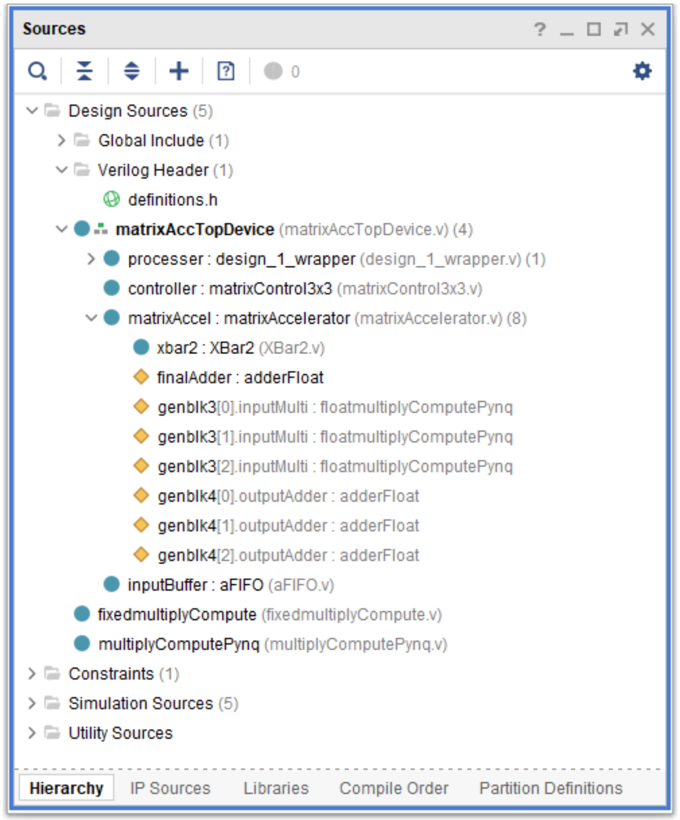
Edit Configurations

Create and modify Configurations. Configurations specify the Reconfigurable Module that will be used for each partition in the design. Automatically create a minimum set of Configurations or manually create them individually.

Configurations

Configuration Name	adder				multiplier		
	finalAdder	genblk4[0].outputAdder	genblk4[1].outputAdder	genblk4[2].outputAdder	genblk3[0].inputMulti	genblk3[1].inputMulti	genblk3[2].inputMulti
config_1	adderFloat	adderFloat	adderFloat	adderFloat	Float	Float	Float
config_2	adder	adder	adder	adder	Integer	Integer	Integer
config_3	adder	adder	adder	adder	Fixed	Fixed	Fixed

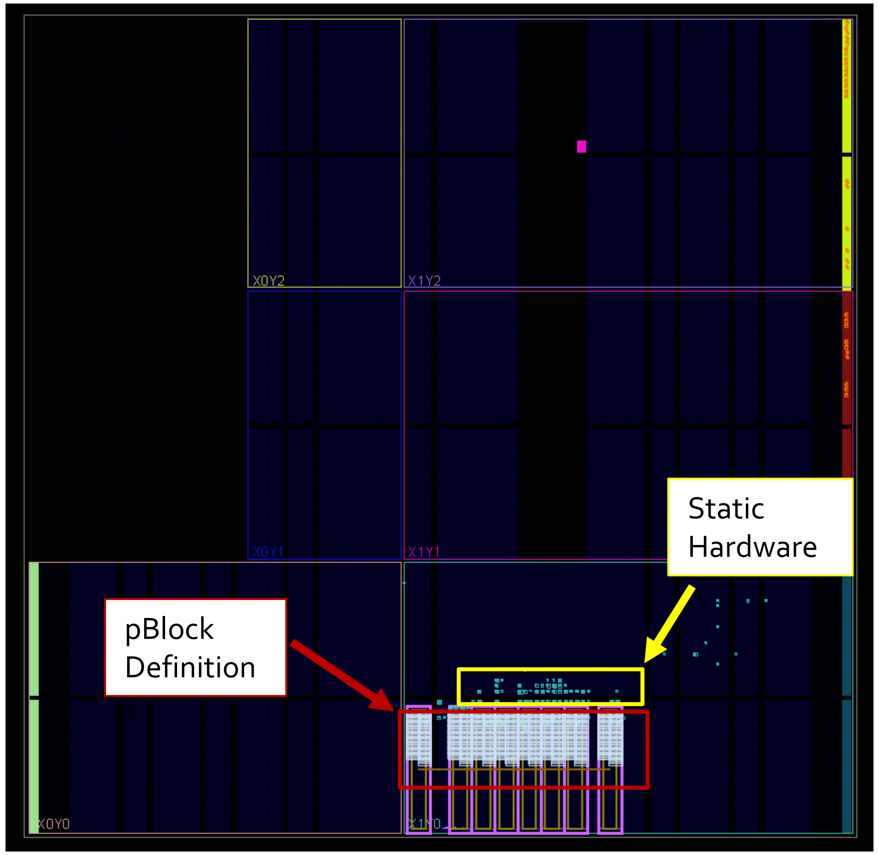
- Hierarchy view of design wrapped together with partial reconfiguration enabled



Sources

- Design Sources (5)
  - Global Include (1)
  - Verilog Header (1)
    - definitions.h
  - matrixAccTopDevice (matrixAccTopDevice.v) (4)
    - processer : design_1_wrapper (design_1_wrapper.v) (1)
    - controller : matrixControl3x3 (matrixControl3x3.v)
    - matrixAccel : matrixAccelerator (matrixAccelerator.v) (8)
      - xbar2 : XBar2 (XBar2.v)
      - finalAdder : adderFloat
      - genblk3[0].inputMulti : floatmultiplyComputePynq
      - genblk3[1].inputMulti : floatmultiplyComputePynq
      - genblk3[2].inputMulti : floatmultiplyComputePynq
      - genblk4[0].outputAdder : adderFloat
      - genblk4[1].outputAdder : adderFloat
      - genblk4[2].outputAdder : adderFloat
    - inputBuffer : aFIFO (aFIFO.v)
  - fixedmultiplyCompute (fixedmultiplyCompute.v)
  - multiplyComputePynq (multiplyComputePynq.v)
- Constraints (1)
- Simulation Sources (5)
- Utility Sources

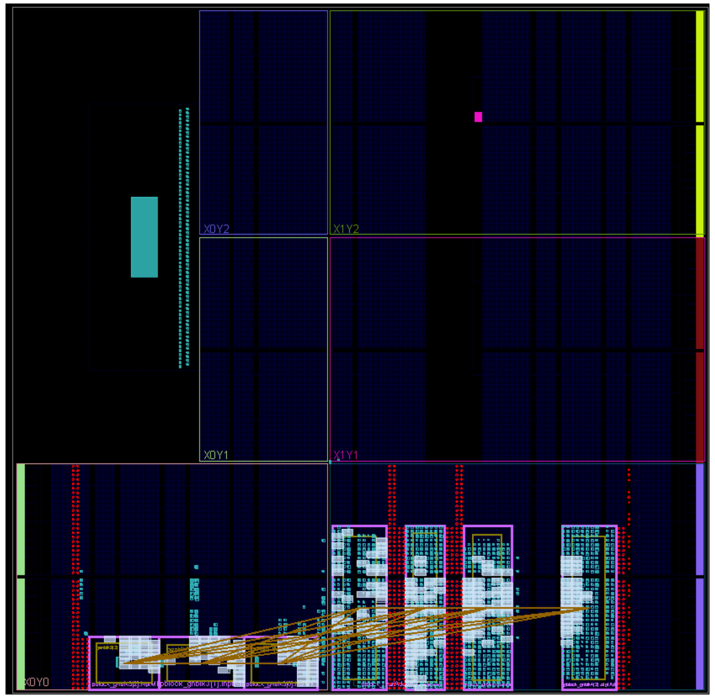
- Each reconfigurable partition requires a dedicated pBlock to dictate the hardware available for reconfiguration
- This pBlock should contain only the dynamic module. Static modules will be placed and routed automatically
- This is a screenshot of the floorplan for a design with 8 pBlocks to enable 8 separate reconfigurable multiply compute partitions.



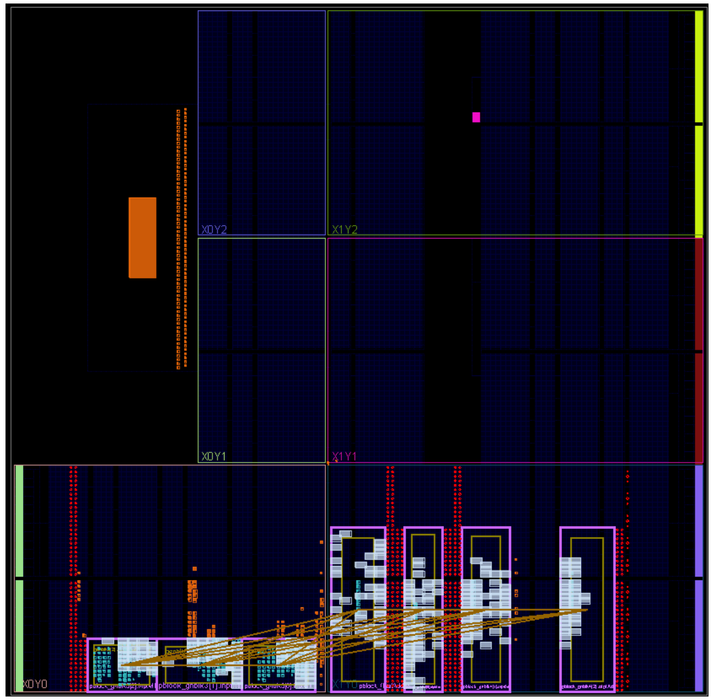
Dynamic Resources Floorplan

Dynamic Resources (Floorplan)

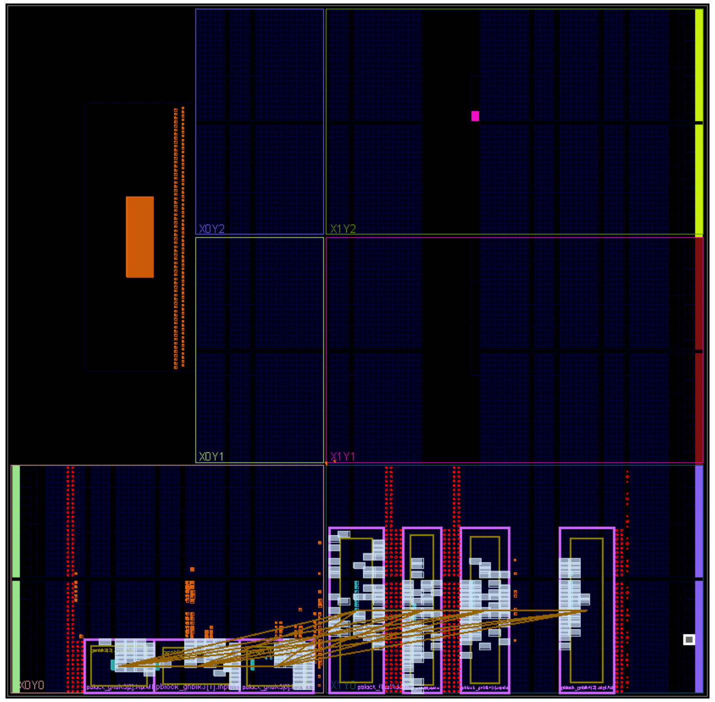
Floating Point Configuration Floorplan



Fixed Point Configuration Floorplan

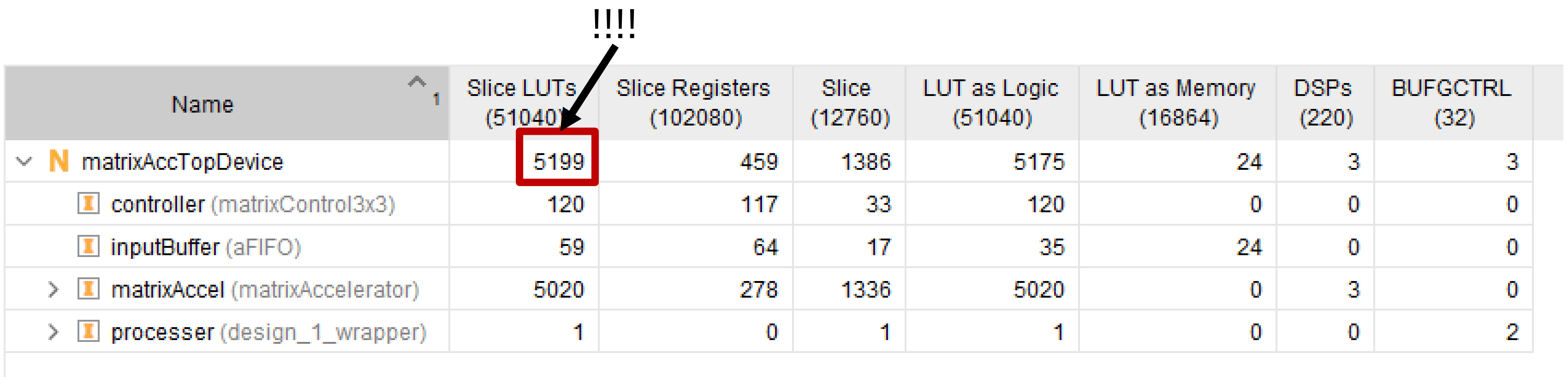


Integer Configuration Floorplan



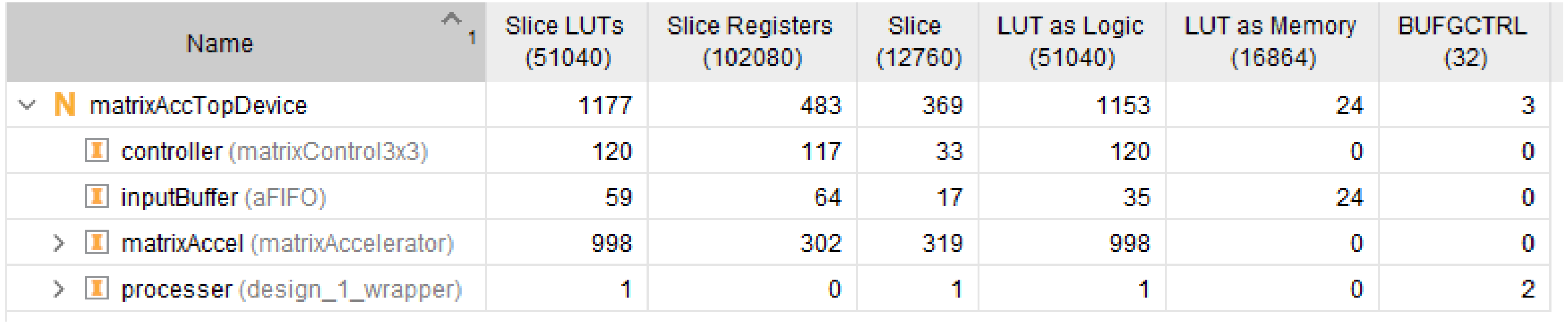
Dynamic Resources
Resource Utilization

Floating Point Configuration Resource Utilization



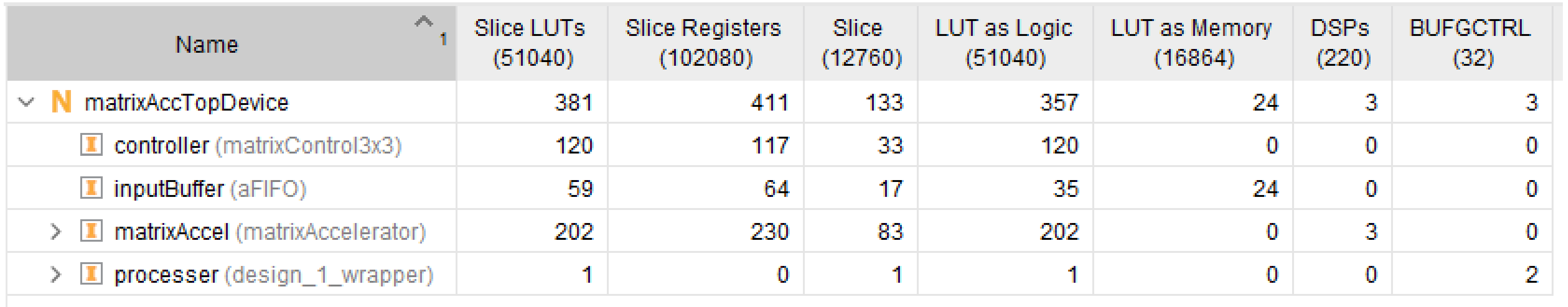
Name	Slice LUTs (51040)	Slice Registers (102080)	Slice (12760)	LUT as Logic (51040)	LUT as Memory (16864)	DSPs (220)	BUFGCTRL (32)
matrixAccTopDevice	$\boxed{5199}$	459	1386	5175	24	3	3
controller (matrixControl3x3)	120	117	33	120	0	0	0
inputBuffer (aFIFO)	59	64	17	35	24	0	0
matrixAccel (matrixAccelerator)	5020	278	1336	5020	0	3	0
processer (design_1_wrapper)	1	0	1	1	0	0	2

Fixed Point Configuration Resource Utilization



Name	Slice LUTs (51040)	Slice Registers (102080)	Slice (12760)	LUT as Logic (51040)	LUT as Memory (16864)	BUFGCTRL (32)
matrixAccTopDevice	1177	483	369	1153	24	3
controller (matrixControl3x3)	120	117	33	120	0	0
inputBuffer (aFIFO)	59	64	17	35	24	0
matrixAccel (matrixAccelerator)	998	302	319	998	0	0
processer (design_1_wrapper)	1	0	1	1	0	2

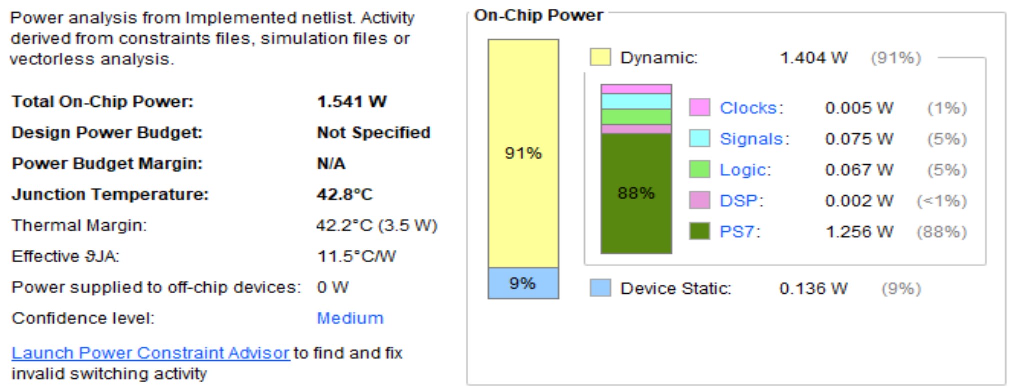
Integer Configuration Resource Utilization



Name	Slice LUTs (51040)	Slice Registers (102080)	Slice (12760)	LUT as Logic (51040)	LUT as Memory (16864)	DSPs (220)	BUFGCTRL (32)
matrixAccTopDevice	381	411	133	357	24	3	3
controller (matrixControl3x3)	120	117	33	120	0	0	0
inputBuffer (aFIFO)	59	64	17	35	24	0	0
matrixAccel (matrixAccelerator)	202	230	83	202	0	3	0
processer (design_1_wrapper)	1	0	1	1	0	0	2

Dynamic Resources
Power Report

Floating Point Configuration Power Report



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

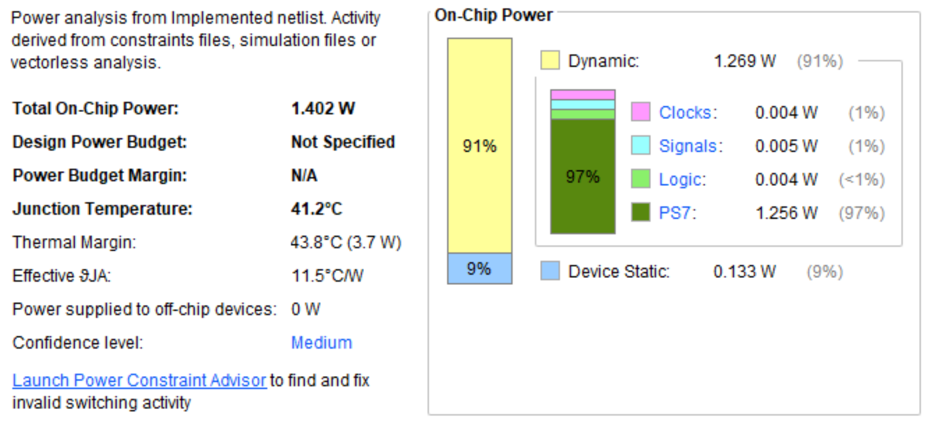
Total On-Chip Power:	**1.541 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**42.8°C**
Thermal Margin:	42.2°C (3.5 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	1.404 W	(91%)
Clocks:	0.005 W	(1%)
Signals:	0.075 W	(5%)
Logic:	0.067 W	(5%)
DSP:	0.002 W	(<1%)
PS7:	1.256 W	(88%)
Device Static:	0.136 W	(9%)

Fixed Point Configuration Power Report



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

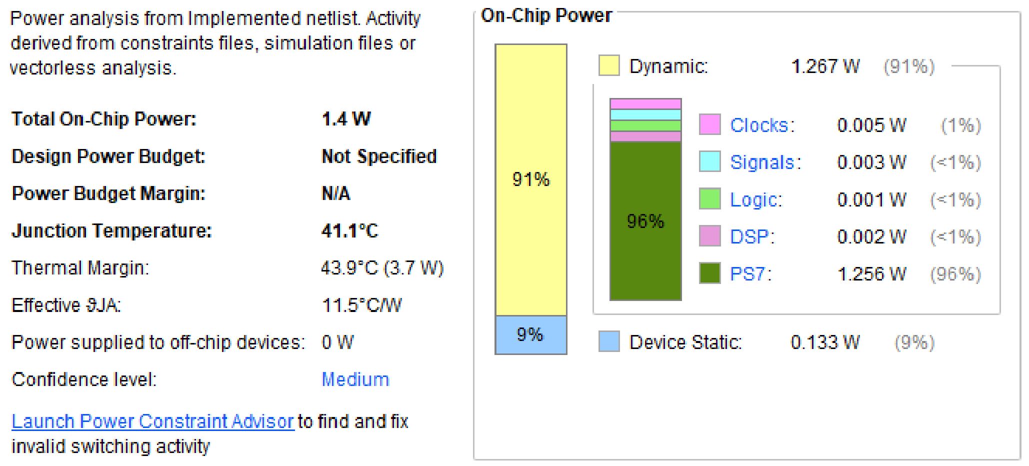
Total On-Chip Power:	**1.402 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**41.2°C**
Thermal Margin:	43.8°C (3.7 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	1.269 W	(91%)
Clocks:	0.004 W	(1%)
Signals:	0.005 W	(1%)
Logic:	0.004 W	(<1%)
PS7:	1.256 W	(97%)
Device Static:	0.133 W	(9%)

Integer Configuration Power Report



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.4 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**41.1°C**
Thermal Margin:	43.9°C (3.7 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	1.267 W	(91%)
Clocks:	0.005 W	(1%)
Signals:	0.003 W	(<1%)
Logic:	0.001 W	(<1%)
DSP:	0.002 W	(<1%)
PS7:	1.256 W	(96%)
Device Static:	0.133 W	(9%)

Conclusions

- While much of the development process for RTL has streamlined by Xilinx, many times development and debugging could be quite strenuous
  - Much of the development process is very well documented for users of Tcl but nothing else
    - This often led to scrolling through forum posting for information rather than being able to rely on the official documentation

- The Convolution Accelerator design needs to be refined
  - Due to the mentioned timing issues, the design would not function properly on the Pynq-Z2
  - The floating-point adder uses much more resources than any other device in the project

Thank You