<u>SimBSP</u> Enabling RTL Simulation for Intel FPGA OpenCL Kernels

Ahmed Sanaullah, Chen Yang, Daniel Crawley and <u>Martin C. Herbordt</u> Department of Electrical and Computer Engineering, Boston University







The Intel OpenCL Toolflow



Code Development Challenges

- There are a number of challenges associated with code development, such as:
 - Reducing development time
 - Implementing efficient pipelines
 - Even codes which took a long time to develop are not guaranteed to be efficient
 - Reducing lines of code needed to express designs
 - Maintaining designs with relative ease across toolflow/API/SDK updates
 - Expertise required and learning curves



Augmentations

Problems to be addressed -

- 1. Performance-programmability gap: Optimizing OpenCL code is hard
- 2. No RTL simulation capabilities in the standard OpenCL toolflow
- 3. Hard to integrate OpenCL generated pipelines into existing structures

Solution

- Cannot rewrite a new compiler to address these shortcomings
- Hence, we augment the existing Intel toolflow

Benefits of Our Work

- Systematic optimizations that are easy to apply, and can even be automated
- Rapid turnaround necessary to be able to evaluate optimizations
- Integration of OpenCL pipelines into existing structures





Apply optimization methods to parallel computing dwarfs

A. Sanaullah, R. Patel, and M. Herbordt, "An Empirically Guided Optimization Framework for FPGA OpenCL," in Proc. IEEE Conf. on Field Programmable Technology, 2018.

Benchmarks	Dwarf	Problem Size
NW	Dynamic Programming	16K x16K Integer Table
FFT	Spectral Methods	64 point Radix-2 1D FFT, 8192 Vectors
Range Limited	N-Body	180 particles per cell, 15% pass rate
PME	Structured Grids	1,000,000 Particles, 32 ³ grid, 3D Tri-Cubic Interpolation
MMM	Dense Linear Algebra	1K x 1K Matrix, Single Precision
SpMV	Sparse Linear Algebra	1K x 1K Matrix, Single Precision, 5%-Sparsity, NZ=51122
CRC	Combinational Logic	100MB CRC32

Ver.	Optimizations
0	(GPU code for porting to FPGA OpenCL)
1	Single thread code with cache optimization
2	Implement task parallel computations in sep-
	arate kernels and connect them using channels
	Fully unroll all loops w/ #pragma unroll
	Minimize variable declaration outside com-
	pute loops – use temps where possible
	Use constants for problem sizes and data values
	 instead of relying on off-chip memory access
	Coalesce memory operations
3	Implement the entire computation within a sin-
	gle kernel and avoid using channels
4	Reduce array sizes to infer pipeline registers as
	registers, instead of BRAMs
5	Perform computations in detail, using tempo-
	rary variables to store intermediate results
6	Use predication instead of conditional branch
	statements when defining forks in the data path

Benchmarks	V-1	V-2	V-3	V-4	V-5	V-6
NW	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark
FFT	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	
Range Limited	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark
PME	\checkmark	\checkmark	\checkmark			\checkmark
MMM	\checkmark	\checkmark	\checkmark		\checkmark	
SpMV	\checkmark	\checkmark	\checkmark		\checkmark	\checkmark
CRC	\checkmark	\checkmark	\checkmark		\checkmark	\checkmark

Characterization of Optimizations





Execution Time Comparison for Different MMM Baselines

Speedup of Code Versions Relative to Version 1 (Baseline)



Comparison with Different Platforms

- Outperform existing CPU by **1.2x** on average
- Outperform previous FPGA OpenCL by 5x ٠ on average
- Within **12%** of average hand tuned HDL ٠ performance
- We estimate a 4x increase in performance of ٠ our OpenCL designs using Intel Stratix 10





Normalized Execution Time Comparison for Different Platforms



Usage of Compute Pipeline HDL

- Analytics for optimizing kernel code
 - RTL simulation
 - Determine latency
 - Verify functional correctness
 - Remember to include all the files from the source file directory
 - Compilation
 - Post-synthesis resource usage
 - More reliable than post compilation log file of normal toolflow
 - Post place&route resource usage and frequency
 - Create custom wrapper and fit design to board
 - Significantly smaller fitting time due to no BSP
 - Faster design iterations based on feedback
- Custom deployment
 - Integration of compute pipelines into existing HDL codes

Finding Our Source File

- Compilation Breakpoint
 - Perform full compilation with the –v flag
 - Terminate once successful source file generation is displayed (can also be automated by modifying tcl files)

Source Files

- Located in [Path to Kernel File]/<kernel_filename>/kernel_subsystem/<kernel_filename>_system_140/synth/
- Folder contains the implemented kernel file, <kernel_filename>.v, as well as additional modules needed for compilation (including custom RTL)

~~~~~ "Path to .cl file"~~~	<pre>/fft64/kernel_system/fft64_</pre>	_sy_tem_140/synth\$ ls	$\wedge$
acl_address_to_bankaddress.v	acl_ic_slave_wrp.v	adl_work_group_dispatcher.v	
<pre>acl_aligned_burst_coalesced_lsu.v</pre>	acl_ic_to_avm.v	acwork_group_limiter.v	
acl_arb2.v	acl_ic_wrp_reg.v	acl_work_item_iterator.v	
acl_arb_intf.v	acl_id_iterator.v	TTL64_system.v	
acl_atomics_arb_stall.v	acl_kernel_finish_detector.v	fft64.v	
acl atomics nostall.v	acl_ll_fifo.v	lsu atomic.v	
acl_avm_to_ic.v	acl_ll_ram_fifo.v	lsu_basic_coalescer.v	
acl_data_fifo.v	acl_loop_limiter.v	lsu_bursting_load_stores.v	
acl_debug_mem.v	acl_mem1x.v	lsu_burst_master.v	
acl_enable_sink.v	acl_mem_staging_reg.v	lsu_enabled.v	
acl_fifo.v	acl_multistage_accumulator.v	lsu_ic_top.v	
<pre>acl_finish_signal_chain_element.v</pre>	acl_multistage_adder.v	lsu_non_aligned_write.v	
acl_fp_add_a10.v	acl_optimized_clz_27.v	lsu_pipelined.v	
acl_fp_custom_clz.v	acl_pipeline.v	lsu_prefetch_block.v	
acl_fp_dot2_a10.v	acl_pop.v	lsu_read_cache.v	
acl_fp_mdot2_a10.v	acl_push.v	lsu_simple.v	
acl_fp_sub_a10.v	acl_shift_register.v	lsu_streaming_prefetch.v	
acl_full_detector.v	acl_staging_reg.v	lsu_streaming.v	
acl_ic_intf.v	acl_stall_free_sink.v	lsu_top.v	
acl_ic_local_mem_router.v	<pre>acl_start_signal_chain_element.v</pre>	lsu_wide_wrapper.v	
acl_ic_master_endpoint.v	<pre>acl_task_copy_finish_detector.v</pre>	six_three_comp.v	
acl_ic_rrp_reg.v	acl_toggle_detect.v	ternary_add.v	
acl_ic_slave_endpoint.v	acl_token_fifo_counter.v	thirtysix_six_comp.v	
acl_ic_slave_rrp.v	acl_valid_fifo_counter.v		

• Change .v to .sv before using

# Finding Our Code Within the File

- Basic blocks are modules used to implement the kernel
  - Construct logic using basic behavioral functions and Altera IP blocks
  - A single kernel can generate multiple basic blocks
  - Number and function of these modules depends on kernel implementation
- Observed rules of typical basic block generation
  - Each normal loop generates a basic block module
  - Nested normal loops generate independent modules and connect to their parent loop module
  - Unrolled loops will also generate a separate module.
    - However, consecutive unrolled loops, or any unrolled nested loop within an unrolled loop, will not generate a new module
- All compute pipelines are within the same basic block
  - Since all compute loops are unrolled





#### FPGA

#### Results

- Intel[®] Arria[®] 10AX115H3F34E2SGE3
  - 427,200 ALMs
  - 1506K Logic Elements
  - 1518 DSP Blocks
  - 53Mb On-chip Memory
- Intel[®] SDK for OpenCL[™] 16.0
- Intel[®] FFT IP Core
- CPU
  - 2.7 GHz Intel[®] Xeon[®] E5-2680
    - eight core
  - Intel[®] C++ Compiler
  - Intel[®] MKL DFTI
- GPU
  - NVIDIA Tesla P100 PCIe 12GB
    - 3584 CUDA Cores
    - 549 GB/s Off-Chip Memory (HBM2)
  - CUDA 8.0
  - cuFFT

#### EXECUTION TIME (US) FOR 3D FFT IMPLEMENTATIONS

Design	$16^{3}$	$32^{3}$	$64^{3}$
CPU	22.0	55.0	288.0
GPU	20.7	23.6	43.1
IP Core	1.8	6.8	31.1
OpenCL-HDL	1.8	6.6	25.8

 TABLE I

 LATENCY AND RESOURCE USAGE FOR OPENCL-HDL 1D FFT

FFT Size	Latency (cycles)	ALM	DSP
8	20	1,849(<1%)	56(4%)
16	37	4,387(1%)	168(11%)
32	41	7,237(2%)	456(30%)
64	53	18,705(4%)	1160(76%)

 TABLE II

 LATENCY AND RESOURCE USAGE FOR IP-CORE 1D FFT

FFT Size	Latency (cycles)	ALM	DSP
8	16	26,759(6%)	96(6%)
16	32	11,132(3%)	256(17%)
32	64	63,322(15%)	832(55%)
64	128	176,285(41%)	1412(93%)



IP core resource usage with respect to OpenCL-HDL. OpenCL-HDL designs consume both fewer ALMs and DSPs.



## Emulation

- Used to simulate kernel code for functional verification.
- Compiling for emulation allows the compiler to
  - generate CPU equivalent code for FPGA-specific constructs
    - such as channels
  - execute the entire computation in software.
- This is useful for:
  - ensuring that computation and memory accesses have been correctly defined
  - identify run-time faults
    - such as occurrences of deadlocks.

## Emulation

- Used to simulate kernel code for functional verification.
- Compiling for emulation allows the compiler to
  - generate CPU equivalent code for FPGA-specific constructs
    - such as channels
  - execute the entire computation in software
  - Emulation does not provide any information regarding kernel code mapping to hardware or estimated performance
    - ensuring that computation and memory accesses have been correctly defined
    - identify run-time faults
      - such as occurrences of deadlocks.

### Reports

- Generated automatically during the initial compilation (C-HDL translation)
- Give the following information
  - Loop analysis
    - Used to determine initiation intervals (II) for loops in the kernel and the dependencies causing high IIs.
    - Resolving these dependencies allows loops to operate stall free.
  - Area analysis
    - Provides estimates of resource usage and implementation details for data structures.
    - This is particularly useful for determining if the compiler:
      - has correctly inferred the optimal hardware based on access patterns.
      - is resorting to sub-optimal, high-resource "safe" options such as memory replication and barrel shifters.

#### System viewer

- Gives a graphical overview of the kernel computation and memory accesses.
- Kernel execution is represented as sequential blocks, with each block carrying out a varying number of operations
  - such as memory transactions, channel calls and loop iterations.
- Details provided include
  - latencies, stalls, types and sizes of Load-Store units created for each memory transaction,
  - the dependencies between blocks.
- Kernel memory viewer
  - Gives a graphical overview of the connectivity of Load-Store units with external memory banks.
  - Can be used to verify that the compiler has correctly inferred off-chip access patterns.

### Reports

- Generated automatically during the initial compilation (C-HDL translation)
- Gives the following information
  - Loop analysis
    - Used to determine initiation intervals (II) for loops in the kernel and the dependencies causing high IIs.
    - Resolving these dependencies allows loops to operate stall free.
  - Area analysis
    - Provides estimates of resource usage and implementation details for data structures.
    - This is particularly useful for determining if the compiler:
      - has correctly inferred the optimal hardware based on access patterns.

#### Kernel codes with no loop dependencies, initialization intervals equal to 1, efficient memories and low latencies can still be sub-optimal

- Gives a graphical overview of the kernel computation and memory accesses.
- Kernel execution is represented as sequential blocks, with each block carrying out a varying number of operations
  - such as memory transactions, channel calls and loop iterations.
- Details provided include
  - latencies, stalls, types and sizes of Load-Store units created for each memory transaction,
  - the dependencies between blocks.
- Kernel memory viewer
  - Gives a graphical overview of the connectivity of Load-Store units with external memory banks.
  - Can be used to verify that the compiler has correctly inferred off-chip access patterns.

# What is a BSP?

- BSP = the files needed to wrap user specified kernel logic
- i) compilation scripts
- ii) board.qsys (the Shell)
- iii) XML files which are used to tell the compiler, among other things, which scripts to pick up and execute
- iv) some HDL files for "top" and "freeze_wrapper" modules.

To modify

- For the XML file, we link to a main "TCL" compilation script (called "import_compile.tcl" in most BSPs) which ends up calling the rest. These compilation scripts are responsible for all operations after the C-to-HDL translation.
- ii) The last script takes the bitstream (.sof file) and packages it into the OpenCL bitstream (.aocx file).

## SimBSP

Very lightweight – no "board.qsys", just

- XML files
- Compilation scripts
- Testbench template



## Matrix Multiply Kernel Code

#define SIZE 16

}

}

```
_kernel void mmm(_global float* restrict a, __global float* restrict b, __global
float* restrict c){
  for (int i = 0; i < SIZE; i++){
    for (int j =0; j < SIZE; j++){
      float temp = 0;
      for (int k =0; k < SIZE; k++){
        temp += (a[i*SIZE+k] * b[k*SIZE+j]);
      }
      c[i*SIZE+j] = temp;
  }
}</pre>
```

After linking to SimBSP, we compile the kernel as usual. Since the compilation forks after generating HDL, i.e. to set up the sim environment, no .sof file is generated.

```
E:\Research\PAPPA\kernel_file>aoc -v mmm.cl
aoc: Environment checks are completed successfully.
You are now compiling the full flow!!
aoc: Selected default target board al0gx
aoc: Running OpenCL parser....
aoc: OpenCL parser completed successfully.
aoc: Compiling....
aoc: Compiling....
aoc: Linking with IP library ...
aoc: First stage compilation completed successfully.
aoc: Hardware generation completed successfully.
aoc: Warning: Cannot find a FPGA programming (.sof) file
```

## Testbench: Interfaces



Name	Туре	Interface Description
clock_clk	Clock	Kernel clock
clock_reset_n	Reset	Active low kernel reset
cc_snoop	Streaming	Not used
kernel_cra	Memory Mapped	Interface to configuration registers
kernel_irq	Interrupt	Interrupt to host machine
kernel_mem0	Memory Mapped	Interface to global memory

## **Testbench: Configuration Registers**



Address	Bits [63:32]	Bits [31:0]
0x0	-	Start (Bit 0)
0x28	Workgroup_Size	Workgroup_Dimensions
0x30	Global_Size[1]	Global_Size[0]
0x38	Number_of_Workgroups[0]	Global_Size[2]
0x40	Number_of_Workgroups[2]	Number_of_Workgroups[1]
0x48	Local_Size[1]	Local_Size[0]
0x50	Global_Offset[0]	Local_Size[2]
0x58	Global_Offset[2]	Global_Offset[1]
0x60 - end	Argument_Pointer[63:32]	Argument_Pointer[31:0]

## Waveforms: Configuration

- 🍐	/tb_mmm/kernel_system_inst/mm	St0												
- 🍝	/tb_mmm/clock	1												
- 🍝	/tb_mmm/resetn	1												
<b></b>	/tb_mmm/address	000	0000	0000				xxxxxX0				00000000		
- 4	/tb_mmm/read	0												
- 🔞	/tb_mmm/write	0												
<b></b>	/tb_mmm/data_out	000	000000000000000000000000000000000000000									000000000	000000000	000000
<b></b>	/tb_mmm/byte_enable	000	ffffff	fffffffff								000000000	0000000	
<b>H</b> -4	/tb_mmm/burst_count	00	01									00		
- 🍐	/tb_mmm/kernel_irg	0												
- 4	/tb_mmm/avs_cra_readdatavalid	0												
<b>H</b> -4	/tb_mmm/avs_cra_readdata	000								000000010	0000001			
- 🗳	/tb_mmm/avs_cra_write	0												
- 🗳	/tb_mmm/avs_cra_waitrequest	0												
<b>H</b>	/tb_mmm/avs_cra_address	000	0000	0000	00000028		00000030		00000038		00000040		00000048	
<b></b>	/tb_mmm/avs_cra_writedata	000			000000010	0000001								

000000	000000000000000000000000000000000000000	0000000000	000000000	000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000	000000000	0000000000	h	
000000		00000000	000000000	000000000					000000000		000000000		
						00000000	0000001	000000000	000000	00000000	1000000	00000000	8000
						00000000	0000001	.00000000	,0000000	00000000	0000000	.000000000	5000
	00000050		00000059		00000060		00000069		00000070		00000000		
	00000050	0000001	00000058	0000000	00000060	0000000	00000068	0000000	00000070	0000000	00000000	0000001	00
	000000000	0000001	,000000000	00000000	00000000	<del>10000000</del> ,	000000000	, 0000000	00000000000		000000000	0000001	,00

## Waveforms: Configuration

- <u></u>	/t	th mmm/ker	nel sv	vstem inst	/mm 9	sto												
- X	/t	tb_mmm/cloc	k	, o . <u>.</u>		1			-				-					
- X	/t	tb_mmm/res	etn			1												
-	/t	tb_mmm/add	ress	ress 0			. 0000	0000				xxxxxxX(	)			00000000		
- 🏹	/t	tb_mmm/rea	/read		)	0000						<b>'</b>						
- 🍝	/t	tb_mmm/writ	te			)												
₽-∲	/t	tb_mmm/dat	a_c	Address			Bits [63:3	2]		Bi	ts [31:0]		Value			0000000000	00000	
+->	/t	tb_mmm/byt	e_e	0x28			W	orkgroup_	Size		Workgrou	up_Dimens	sions	00000	0000001 0000001			
*~~~	/t	tb_mmm/keri	nel_	0x30			G	lobal_Size	e[1]		Glob	oal_Size[0]		00000	00000			
- 🔶	/t	tb_mmm/avs	_a	0x38		N	lumber	r_of_Work	groups[0]		Glob	oal_Size[2]		00000	0000001 00000001			
+->	/t /t	tb_mmm/avs tb_mmm/avs	_a	0x40		N	lumber	r_of_Work	groups[2]		Number_o	f_Workgro	ups[1]	00000	0000001_00000001			
4	/t	tb_mmm/avs	_a	0x48			L	_ocal_Size	[1]		Local_Size[0]				0000001_00000001			
• 🔶	/t	tb_mmm/avs	_cr	0x50			Gl	obal_Offse	et[0]		Local_Size[2]				0000000_0000001			
<u>+</u>	· /t	to_mmm/avs	nmm/avs_cr0x58				Global_Offset[2]				Global_Offset[1]				00000000_00000000			
				0x60						Pointer "a	inter "a"				0000000_4000000			
				0x68						Pointer "b	inter "b"				00000000_80000000			
				0x70						Pointer "c	pinter "c"				000_C000	0000		
				0x00				-			Sta	art (Bit 0)		00000	0000_0000	0001		
			20000	000000000	0000000	000000	00000	000000000	000000000000000000000000000000000000000	000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000	000000000000000000000000000000000000000			
		0	0000	000000000	00000000	000000	00000	000000000		000000000		0000000000		000000000	000000000000000000000000000000000000000	,		
										00000000	00000001	00000000	0000000	00000000	10000000	00000008	000	
				00000050		000	00058		00000060		00000068		00000070		00000000			
				00000000	0000000	1 000	000000	0000000	<u>, 00000000</u>	10000000	<u>, 00000000</u>	\$0000000	00000000	0000000	000000000	0000001	00	

## Waveforms: Kernel Execution



# **THANK YOU**

#### Programming Model?

This is an issue because OpenCL is not sufficiently expressive to allow programmer to explicitly express all Intel FPGA capabilities. Several alternatives:

Choice 1: Program Intel FPGA as if it were a GPU

Rationale: Since OpenCL maps well to GPUs, and the GPU model is a subset of the Intel FPGA model, and GPU code is efficient for many applications, this should be reasonable, if not perfect.

Problem: Gives very poor performance

Choice 2: Program Intel FPGA as if it were an Intel[®] FPGA

Rationale: Although support is a subset of HDL support, there are Intel[®] FPGA-specific extensions, including channels and embedded HDL.

Problem: Gives very poor performance

Choice 3: Program Intel FPGA as if were a single threaded CPU This means: Single work item/group AND single work group Rationale: Huh?

Rationale: The Intel OpenCL compiler is REALLY GOOD – turn it loose!

Problems (solved here): Long compile times, need systematic optimizations

#### **Optimizing OpenCL Kernels**

- V1: Cache Optimized CPU Code
  - CPU architecture resembles FPGAs more than GPUs
- V2: Typical FPGA Optimizations / Recommended Best Practices
  - SWI kernels, Channels, Constants, Loop unrolling, Coalesced loops
- V3: Single Kernel Implementation
  - Remove channels
    - Channels isolate resources and prevent global optimizations
    - Kernels are launched sequentially, potentially resulting in race conditions, deadlocks and high synchronization costs
  - Instead, let Intel OpenCL compiler infer task parallelism within a single kernel
    - Use code fragments that are tied to the same loop iterator but have no data dependencies
    - Intel OpenCL compiler utilizes delay modules instead of blocking channels for synchronization of data paths which is more efficient

#### Optimizing OpenCL Kernels, cont.

#### • V4: Infer Pipeline Registers as Registers

- Large arrays can be inferred as BRAMs instead of registers
  - Rapidly increases resource usage due to memory replication to meet throughput
- Avoid using large variable arrays where possible
  - Use scripts to generate and use individual variables

#### V5: Explicit computations

- Provide as much detail as possible regarding the computation
  - Even if it is an apparent suboptimal practice such as
    - Un-coalescing loops to help infer access patterns
    - Small constant arrays instead of single large one
    - Intermediate variables for complex computations
    - Manually unrolling a loop and specifying tree based computations using parenthesis
    - Extra kernel parameters that are pointers to the same memory space
      - One per task
      - manually guarantee no RAW data dependencies between tasks
- Helps the Intel OpenCL compiler infer the desired architecture with greater accuracy

V6: Avoid conditional statements e.g. if, else

Use conditional assignments instead

#### execFFT basic block 3

		avm local bb3 ld memcoalesce null load 0 address[310]
		avm_local_bb3_ld_memcoalesce_null_load_0_burstcount
		avm local bb3 ld memcoalesce null load 0 byteenable[2550]
		avm_local_bb3_ld_memcoalesce_null_load_0_enable
avm local bb3 ld memcoalesce null load 0 readdata[20470]		avm_local_bb3_ld_memcoalesce_null_load_0_read
avm_local_bb3_ld_memcoalesce_null_load_0_readdatavalid		avm local bb3 ld memcoalesce null load 0 writedata[20470]
avm_local_bb3_ld_memcoalesce_null_load_0_waitrequest		avm_local_bb3_ld_memcoalesce_null_load_0_write
avm_local_bb3_ld_memcoalesce_null_load_0_writeack		avm local bb3 ld memcoalesce null load 02 address[310]
avm local bb3 ld memcoalesce null load 02 readdata[20470]		avm_local_bb3_ld_memcoalesce_null_load_02_burstcount
avm_local_bb3_ld_memcoalesce_null_load_02_readdatavalid		avm local bb3 ld memcoalesce null load 02 byteenable[2550]
avm_local_bb3_ld_memcoalesce_null_load_02_waitrequest		avm_local_bb3_ld_memcoalesce_null_load_02_enable
avm_local_bb3_ld_memcoalesce_null_load_02_writeack		avm_local_bb3_ld_memcoalesce_null_load_02_read
avm local bb3 st memcoalesce null insertValue 63 readdata[20470]		avm local bb3 ld memcoalesce null load 02 writedata[20470]
avm_local_bb3_st_memcoalesce_null_insertValue_63_readdatavalid		avm_local_bb3_ld_memcoalesce_null_load_02_write
avm_local_bb3_st_memcoalesce_null_insertValue_63_waitrequest		avm local bb3 st memcoalesce null insertValue 63 address[310]
avm_local_bb3_st_memcoalesce_null_insertValue_63_writeack		avm_local_bb3_st_memcoalesce_null_insertValue_63_burstcount
avm local bb3 st memcoalesce null insertValue 63137 readdata[20470]		avm local bb3 st memcoalesce null insertValue 63 byteenable[2550]
avm_local_bb3_st_memcoalesce_null_insertValue_63137_readdatavalid		avm_local_bb3_st_memcoalesce_null_insertValue_63_enable
avm_local_bb3_st_memcoalesce_null_insertValue_63137_waitrequest		avm_local_bb3_st_memcoalesce_null_insertValue_63_read
avm_local_bb3_st_memcoalesce_null_insertValue_63137_writeack		avm local bb3 st memcoalesce null insertValue 63 writedata[20470]
clock2x		avm_local_bb3_st_memcoalesce_null_insertValue_63_write
clock		avm local bb3 st memcoalesce null insertValue 63137 address[310]
input ap ctr addr[630]	_	avm_local_bb3_st_memcoalesce_null_insertValue_63137_burstcount
input ap ctr5 addr[630]		avm local bb3 st memcoalesce null insertValue 63137 byteenable[2550]
input c0 exe6[30]	_	avm_local_bb3_st_memcoalesce_null_insertValue_63137_enable
input c0 exe9[30]	_	avm_local_bb3_st_memcoalesce_null_insertValue_63137_read
input c0 exe10[30]		avm local bb3 st memcoalesce null insertValue 63137 writedata[20470]
input c0 exe11[310]	_	avm_local_bb3_st_memcoalesce_null_insertValue_63137_write
input c0 exe463[30]		Mb bb3 add436[310]
input_c0_exe4		Mb bb3 c0 exe178[310]
input c0 exit59 c0 exit2[4470]		Wb bb3 c0 exe279[310]
input_forked		Mb bb3 c0 exe380[630]
resetn		Mb bb3 c0 exe481[630]
stall in		lvb_bb3_c0_exe12
start		Mb c0 exe6[30]
valid_in	_	Mb c0 exe9[30]
32'h1 workgroup_size[310]		Mb c0 exe10[30]
		Wb c0 exe463[30]
		Mb_c0_exe4
		lvb forked

(execFFT_basic_block_3)

stall out valid_out

execFFT basic block 3

#### Control Ports

ce null load 0 readdata[20470]	avm local bb3 ld memco
alesce_null_load_0_readdatavalid	avm_local_bb3_ld_me
oalesce_null_load_0_waitrequest	avm_local_bb3_ld_m
emcoalesce_null_load_0_writeack	avm_local_bb3_lo
e null load 02 readdata[20470]	avm local bb3 ld memcoa
esce_null_load_02_readdatavalid	avm_local_bb3_ld_mem
alesce_null_load_02_waitrequest	avm_local_bb3_ld_me
ncoalesce_null_load_02_writeack	avm_local_bb3_id
nsertValue 63 readdata[20470]	avm local bb3 st memcoalesce r
ull_insertValue_63_readdatavalid	avm_local_bb3_st_memcoales
_null_insertValue_63_waitrequest	avm_local_bb3_st_memcoale
sce_null_insertValue_63_writeack	avm_local_bb3_st_memo
rtValue 63137 readdata[20470]	avm local bb3 st memcoalesce null
insertValue_63137_readdatavalid	avm_local_bb3_st_memcoalesce_
_insertValue_63137_waitrequest	avm_local_bb3_st_memcoalesce
null_insertValue_63137_writeack	avm_local_bb3_st_memcoale
clock2x	
clock	
input ap ctr addr[630]	
input ap ctr5 addr[630]	
input c0 exe6[30]	
input c0 exe9[30]	
input c0 exe10[30]	
input c0 exe11[310]	
input c0 exe463[30]	
input_c0_exe4	
input c0 exit59 c0 exi12[4470]	
input_forked	
resetn	
stall in	
start	
valid in	
00%1	

32'h1 workgroup_size[31..0]

	aum local bh3 ld mamcoalasce null load 0 address[31 0]
T	aum local bb3 ld memocalesce null load 0 burstoount
+	avm_local_bb3_id_memcoalesce_null_load_0_butseenable[255_0]
T	avm local bb3 id memoalesce null load 0 enable
+	avm_local_bb3_ld_memcoalesce_null_load_0_enable
+	avm_local_bb3_ld_memoalesce_null_load_0_vetad
t	avm local bb3 ld memoalesce null load 0 write
+	avm_local_bb3_ld_memcoalesce_null_load_02_address[31_0]
T	avm local bb3 ld memcoalesce null load 02 burstcount
+	avm_local_bb3_ld_memcoalesce_null_load_02_byteenable[255.0]
T	avm local bb3 ld memcoalesce null load 02 enable
	avm local bb3 ld memcoalesce null load 02 read
	avm local bb3 ld memcoalesce null load 02 writedata[20470]
T	avm local bb3 ld memcoalesce null load 02 write
	avm local bb3 st memcoalesce null insertValue 63 address[310]
٦	avm local bb3 st memcoalesce null insertValue 63 burstcount
	avm local bb3 st memcoalesce null insertValue 63 byteenable[2550]
T	avm local bb3 st memcoalesce null insertValue 63 enable
	avm_local_bb3_st_memcoalesce_null_insertValue_63_read
	avm local bb3 st memcoalesce null insertValue 63 writedata[20470]
Т	avm_local_bb3_st_memcoalesce_null_insertValue_63_write
	avm local bb3 st memcoalesce null insertValue 63137 address[310]
$\Box$	avm_local_bb3_st_memcoalesce_null_insertValue_63137_burstcount
	avm local bb3 st memcoalesce null insertValue 63137 byteenable[2550]
	avm_local_bb3_st_memcoalesce_null_insertValue_63137_enable
	avm_local_bb3_st_memcoalesce_null_insertValue_63137_read
4	avm local bb3 st memcoalesce null insertValue 63137 writedata[20470]
_	avm_local_bb3_st_memcoalesce_null_insertValue_63137_write
4	lvb bb3 add436[310]
4	Nb bb3 c0 exe178[310]
4	Nb bb3 c0 exe279[310]
4	Nb bb3 c0 exe380[630]
4	Nb bb3 c0 exe481[630]
4	Nb_bb3_c0_exe12
4	Wb c0 exe6[30]
4	Wb c0 exe9[30]
-	Wb c0 exe10[30]
4	Wb c0 exe463[30]
-	lvb_c0_exe4
-	lvb_forked
-	stall out
-	valid_out

(execFFT_basic_block_3)

#### **Control Ports**

- Consists of clock, resetn, Stall and Valid ports
- Stall ports used by a basic block to stall upstream modules
- Valid ports used to stall downstream basic blocks

T_basic_block_3	execFF
+	
avm local	
avm local	
avm local	
avm_local	
avm local	avm local bb3 ld memcoalesce null load 0 readdatai2047.01
avm local	avm local bb3 ld memcoalesce null load 0 readdatavalid
avm local	avm_local_bb3_ld_memcoalesce_null_load_0_waitrequest
avm local	avm local bb3 ld memcoalesce null load 0 writeack
- Avm local	avm local bb3 ld memcoalesce null load (12 readdata(2047 01
avm local	avm local bb3 ld memcoalesce null load 02 readdatavalid
avm local	avm local bb3 ld memcoalesce null load 02 waitrequest
avm local	avm local bb3 ld memcoalesce null load 02 writeack
avm local	avm local bb3 st memcoalesce null insertValue 63 readdata[2047_0]
aum local	avm local bb3 st memcoalesce null insertValue 63 readdatavalid
avm local	avm local bb3 st memcoalesce null insertValue 63 waitrequest
avm local	avm local bb3 st memcoalesce null insertValue 63 writeack
avm local	avm local bb3 st memcoalesce null insertValue 63137 readdata[20470]
avm local	avm local bh3 st memocalesce null insertValue 63137 readdatavalid
avm_local	avm local bb3 st memcoalesce null insertValue 63137 waitrequest
avm local	avm local bb3 st memcoalesce null insertValue 63137 writeack
avm local	clock2x
avm local	clock
avm local	input ap ctr addr[630]
avm local	input ap ctr5 addr[630]
avm local	input c0 exe6[30]
avm local	input c0 exe9[30]
avm local	input c0 exe10[30]
avm local	input c0 exe11[310]
IVb bb3 a	input c0 exe463[30]
IVb bb3 c	input_c0_exe4
IVb bb3 c	input c0 exit59 c0 exi12[4470]
IVb bb3 c	input_forked
IVb bb3 o	resetn
Ivb_bb3_c	stall in
lvb c0 ex	start
lvb c0 ex	valid in
	001-1

32'h1 workgroup_size[31..0]

alesce null load 0 address[31 0] hh3 ld memonalesce null load 0 hurstcount memonalesce null load 0 byteenable[255 bh3 ld memocalesce null load 0 enable hh3 ld memonalesce null load 0 write nemcoalesce null load 02 address[31 bh3 ld memonalesce null load 02 bvf memonalesce null load 02 e bh3 ld memonalesce null load 02 read memonalesce null load 02 writedata[2047 0 nalaeca null incortValua 63 addrace[31 0] palesce null insertValue 63 hurstoour memonalesce null insertValue 63 hyteenable[255 memonalesce null insertValue 63 enab bb3_st_memcoalesce_null_insertValue_63_read monalesce null insertValue 63 writedata[2047... alecce null incert/alue 63137 address[31 bh3 st memonalesce null insertValue 63137 writedata[2047 bb3 st memcoalesce null insertValue 63137 write 0 exe178[31..0] 0 exe279[31..0 0 exe380[63..0 0 exe481[63..0] :0 exe12 e6[3..0] e9[3..0] c0 exe10[3..0] c0_exe463[3_0 vb c0 exe4

- Control Ports
- Load Store Unit Ports

#### Load Store Unit (LSU) Ports

- Consists of Avalon interfaces to LSU modules
- LSU modules sink and source data to compute pipelines
- Remove LSU modules and interface pipelines with required data-buses

(execFFT_basic_block_3)

lvb forked

chool i	Ŧ	
	<u>ا ا</u>	
		avm local bb3 ld memcoalesce null load 0 address[31.0]
		avm local bb3 ld memcoalesce null load 0 burstcount
		avm local bb3 ld memcoalesce null load 0 byteenable[2550]
		avm_local_bb3_ld_memcoalesce_null_load_0_enable
idata[20470]		avm local bb3 ld memcoalesce null load 0 read
readdatavalid		avm local bb3. Id. memopalesce. pull load 0. writedatal2047. 01
0_waitrequest		avm local bb3 ld memcoalesce null load 0 write
ud 0 writeack		avm local bb3 ld memcoalesce null load 02 address[310]
idata[2047_0]		avm.local.bb3.ld.memcoalesce.null.load.02.burstcount
readdatavalid		avm local bb3 ld memcoalesce null load 02 byteenable[2550]
2 waitrequest		avm local bb3 ld memcoalesce null load 02 enable
d 02 writeack		avm local bb3 ld memcoalesce null load 02 read
idata[2047_0]		avm local bb3 ld memcoalesce null load 02 writedata[20470]
readdatavalid		aum local bh3 ld memocalesce pull load 02 write
3 waitrequest		avm local bb3 st memonalesce null insertValue 63 address[31_0]
e 63 writeack		avm local bb3 st memocalesce null insertValue 63 burstcount
idata[20470]		avm local bb3 st memcoalesce null insertValue 63 byteenable(255.0)
readdatavalid		avm local bb3 st memocalesce null insertValue 63 enable
7 waitrequest		avm local bb3 st memcoalesce null insertValue 63 read
137 writeack		avm local bb3 st memcoalesce null insertValue 63 writedata/204701
clock2x		avm local bb3 st memocalesce null insertValue 63 write
clock		avm local bb3 st memopalesce pull insertValue 63137 address[31,0]
tr addr[630]	<b>-</b> 7	avm local bb3 st memcoalesce null insertValue 63137 burstcount
5 addr[630]		avm local bb3 st memcoalesce null insertValue 63137 byteenable[2550]
c0_exe6[3_0]	F 7	avm local bb3 st memocalesce null insertValue 63137 enable
c0_exe9[3_0]	<b>-</b> -	avm local bb3 st memopalesce pull insertValue 63137 read
0 exe10[3.0]		avm local bb3 st memonalesce null insertValue 63137 writedata[2047.0]
exe11[31_0]		avm local bb3 st memopalesce null insertValue 63137 write
eve463[3_0]		Mb bb3 add436[3] 0]
nut c0 evel		ht bb3 c0 evel 78[3] 0]
evi12[447_0]		Mb bb3 c0 exe279[31.0]
input forked		Mb bb3 c0 exe380[63 0]
receto	-	Mb bb3 c0 exe481/63 01
stall in	-	bb bb3 c0 eva12
start	-	Mb_00_co_cxc12
volid in		
n cizof21 0	-	
ip_size[310]		ND C0 exet0[3.0]
	-	ND CU EXE400[00]
	-	IVD_CU_EXE4
	-	
	-	untid out
	-	wand our

avm local bb3 ld memcoalesce null load 0 readdata[20470]
avm local bb3 ld memcoalesce null load 0 readdatavalid
avm_local_bb3_ld_memcoalesce_null_load_0_waitrequest
avm local bb3 ld memcoalesce null load 0 writeack
avm local bb3.ld memcoalesce pull load 02 readdata[2047.0]
avm local bb3 ld memcoalesce null load 02 readdatavalid
avm local bb3 ld memcoalesce null load 02 waitrequest
avm local bb3 ld memcoalesce null load 02 writeack
avm_local_bb3_st_memcoalesce_null_insertValue_63_readdata[2047_0]
avm local bb3 st memcoalesce null insertValue 63 readdatavalid
avm local bb3 st memcoalesce null insertValue 63 waitreguest
avm local bb3 st memcoalesce null insertValue 63 writeack
avm local bb3 st memcoalesce null insertValue 63137 readdata[20470]
avm local bh3 st memonalesce null insertValue 63137 readdatavalid
avm local bb3 st memcoalesce null insertValue 63137 waitrequest
avm local bb3 st memcoalesce null insertValue 63137 writeack
clock2x
clock
input ap ctr addr[630]
input ap ctr5 addr[630]
input c0 exe6[30]
input c0 exe9[30]
input c0 exe10[30]
input c0 exe11[310]
input c0 exe463[30]
input cD exe4
input cu exits9 cu exit2[4470]

32'h1 worka

(execFFT basic block 3)

- Control Ports
- Load Store Unit Ports
- Feedback Ports

#### **Feedback Ports**

- Used to select between initial and steady state values of state registers
- Hardwired User cannot modify at run time
- Paired output-input ports connect to each other while individual inputs have a constant value

execEET basic	block 3
	avm local bb3 ld memcoalesce null load 0 address[3
	avm local bb3 ld memcoalesce null load 0 burstcoun
	avm local bb3 ld memcoalesce null load 0 byteenabl
	avm_local_bb3_ld_memcoalesce_null_load_0_enable
memcoalesce null load 0 readdata[2047.0]	avm local bb3 ld memcoalesce null load 0 read
3 ld memcoalesce null load 0 readdatavalid	avm local bb3 ld memocalesce pull load 0 writedatal
bb3_ld_memcoalesce_null_load_0_waitrequest	avm local bb3 ld memcoalesce null load 0 write
al bb3 ld memcoalesce null load 0 writeack	avm local bb3 ld memcoalesce null load 02 address
memcoalesce pull load 02 readdata[2047_0]	avm local bh3 ld memocalesce null load 02 hurstoou
Id memocalesce null load 02 readdatavalid	avm local bb3 ld memcoalesce null load 02 byteenat
b3 ld memcoalesce null load 02 waitrequest	avm local bb3 ld memcoalesce null load 02 enable
bb3 ld memcoalesce null load 02 writeack	avm local bb3 ld memcoalesce null load 02 read
palesce null insertValue 63 readdata[2047_0]	avm local bb3 ld memcoalesce null load 02 writedate
emcoalesce null insertValue 63 readdatavalid	aum local bh3 ld memocalesce pull load 02 write
nemcoalesce null insertValue 63 waitrequest	avm local bh3 st memocalesce null insertValue 63 ar
st memcoalesce null insertValue 63 writeack	avm local bb3 st memcoalesce null insertValue 63 b
sce null insertValue 63137 readdata[20470]	avm local bb3 st memcoalesce null insertValue 63 b
palesce null insertValue 63137 readdatavalid	avm local bb3 st memcoalesce null insertValue 63 er
coalesce null insertValue 63137 waitrequest	avm_local_bb3_st_memcoalesce_null_insertValue_63_re
nemcoalesce null insertValue 63137 writeack	avm local bb3 st memcoalesce null insertValue 63 w
clock2x	avm local bh3 st memocalesce null insertValue 63 w
clock	avm local bb3 st memcoalesce null insertValue 6313
input ap ctr addr[630]	avm local bb3 st memcoalesce null insertValue 6313
input ap ctr5 addr[630]	avm local bb3 st memcoalesce null insertValue 6313
input_c0_exe6(3_0)	avm local bh3 st memocalesce null insertValue 6313
input_c0_exe9(3_0)	avm local bh3 st memocalesce null insertValue 6313
input c0 exe10(30)	avm local bb3 st memcoalesce null insertValue 6313
input c0 exe11[310]	avm local bb3 st memcoalesce null insertValue 6313
input_c0_exe463[3_0]	Vb_bb3_add436/3101
input c0 exe4	Wb_bb3_c0_exe178(310)
input c0 exit59 c0 exi12[4470]	wh_bh3_c0_exe279[31_0]
input_forked	Mb_bb3_c0_exe380[63_0]
resetn	Wb bb3 c0 exe481[630]
stall in	Wh bb3 c0 exe12
start	Wb_c0_exe6[30]
valid in	Mb_c0_exe9[3_0]
32'h1 workgroup_size[310]	Mb_c0_exe10[3_0]
	Wb_c0_exe46313_01
	Mb_c0_exe4
	Wh forked
	stall out
	valid_out

(execFFT basic block 3)

Control Ports

- Load Store Unit Ports
- Feedback Ports
- Don't Care Ports

#### **Don't Care Ports**

- Typically correspond to a variety of logic such as
- Parallel control and data paths that do not interact with compute pipelines
- Logic that interfaces load store unit (LSU) modules e.g. address computation
- Since we remove LSU modules, and since there is no interaction with compute pipelines, we can leave these pins unconnected

T_basic_block	execFF
±	
- avi	
- avii	
avin	
avm	sum local bb2 id memocalesce sull load 0 readdate[2047_0]
	avm local bb3 ld memocalesce null load 0 readdataualid
avm	avm local bb3 ld memonalesce null load 0 waitrequest
avm	avm local bb3 ld memcoalesce null load 0 writeack
avm	avm local bb3 ld memcoalesce null load 02 readdata[2047 0]
avm	avm local bb3 id memocalesce null load 02 readdatavalid
avm	avm local bb3 ld memcoalesce null load 02 waitrequest
avm	avm local bb3 ld memonalesce null load 02 writeack
avm	avm local bb3 st memcoalesce null insertValue 63 readdata(2047 0)
-	avm local bb3 st memcoalesce pull insertValue 63 readdatavalid
avm	avm local bb3 st memcoalesce null insertValue 63 waitrequest
avm	avm local bb3 st memcoalesce null insertValue 63 writeack
avm	avm local bb3 st memcoalesce null insertValue 63137 readdata[2047.0]
avm	avm local bh3 st memonalesce null insertValue 63137 readdatavalid
avm	avm local bb3 st memoalesce null insertValue 63137 waitrequest
avm	avm local bb3 st memcoalesce null insertValue 63137 writeack
ave	clock2x
avm	clock
avm	input ap ctr addr[630]
avm	input ap ctr5 addr[630]
Lavn	input_c0_exe6(3_0)
avm	input_c0_exe9[3.0]
avm	input c0 exe10[30]
avm	input c0 exe11[310]
Mb	input_c0_exe463[3_0]
Mb	input c0 exe4
Mh	input c0 exit59 c0 exi12[4470]
Mh	input_forked
Mb	resetn
Mb	stall in
Mb	start
Mh	valid in
Mh	32'h1 workgroup_size[310]
Mb	
Mh	

3 local bh3 ld memonalesce null load 0 hurstcount Id memonalesce null load 0 hyteenable[2 local bh3 ld memocalesce null load 0 enable ative 0 beal llug asselection bl 8de memonalesce null load 02 address[3 al bh3 ld memonalesce null load 02 h al bb3 ld memonalesce null load 02 ocal bh3 ld memocalesce null load 02 read al bh3 ld memonalesce null load 02 writedata[2 al bh3 et memonalecce null incert/jalue 63 addrecel3 nomenalessa pull incort/alus 62 h cal bb3 st memcoalesce null insertValue 63 hvt et memonalacce null incert/value 63 e local bh3 st memocalesce null insertValue 63 read al bb3 st memonalesce null insertValue 63 voalesce null insertValue 63137 adv local bh3 st memonalesce null insertValue 63137 local bb3 st memonalesce null insertValue 63137 b3_add436[31 bb3_c0_exe178[31_0 bb3_c0_exe279[31_0 bb3_c0_exe380[63_0 bb3 c0 exe481[63.0 bb3_c0_exe12 c0_exe6[3_0] c0_exe9[3_0] c0_exe10[3_0 c0_exe463[3] cO exe forker

Control Ports
 Load Store Unit Ports

#### **Direct Ports**

(not shown)

Direct Kernel Input : if the kernel has a constant input. Appears as a N bit input to the kernel based on variable type

Direct Data : Data is available as a direct bus. Typically occurs when compiler moves LSU unit out of the basic block due to:

- outer loop variable not used in index/address computations
- Problem size is too small

Initial Values

(execFFT_basic_block_3)

