



FPGA Virtualization Strategies for Mainstream High-level Synthesis



HLS4HPC Workshop: HIPEAC 2013

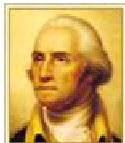
UF UNIVERSITY of
FLORIDA

Virginia
Tech

VIRGINIA POLYTECHNIC INSTITUTE
AND STATE UNIVERSITY

BYU

BRIGHAM YOUNG
UNIVERSITY



THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON DC

Greg Stitt

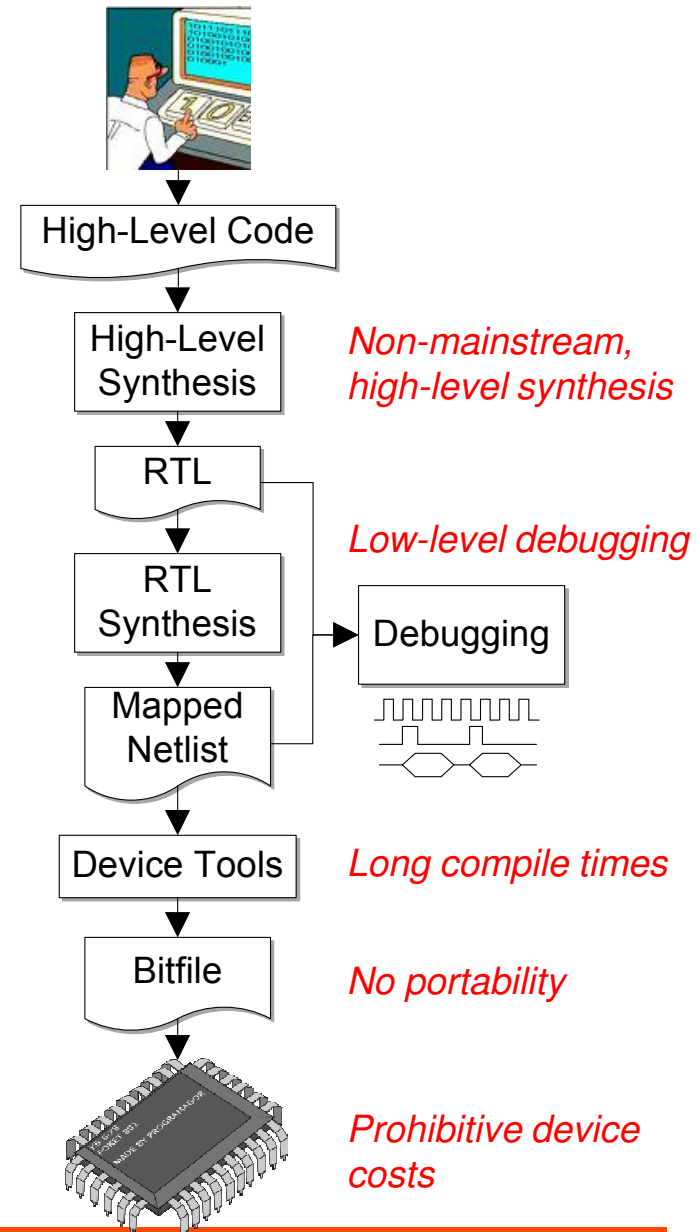
Assistant Professor of ECE
University of Florida

PhD Students: James Coole, Aaron Landy,
Robert Kirchgessner

*This work is supported by National Science Foundation grant
CNS-1149285 and the I/UCRC Program of the National Science
Foundation under Grant No. EEC-0642422.*

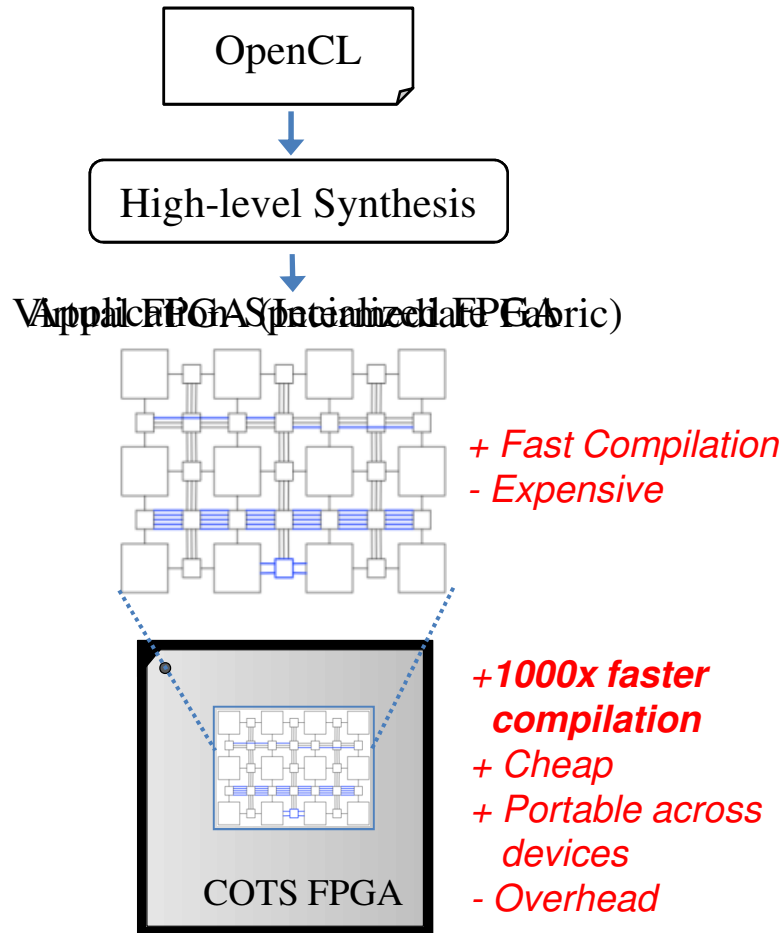
Introduction

- FPGAs advantages are widely known
 - But, GPUs & CPUs used even when FPGAs have significant benefits
- Problem: low productivity
 - Limits usage to FPGA experts
 - **Mainstream designers miss out on FPGA benefits**
- Barriers to mainstream usage
 - Prohibitive device costs
 - Specialized languages, low-level debugging
 - Addressed by OpenCL, CUDA research
 - **Long compile times (hours to days)**
 - Prevents mainstream methodologies
 - **No portability**
 - Prevents design reuse
- **Potential solution: virtualization**

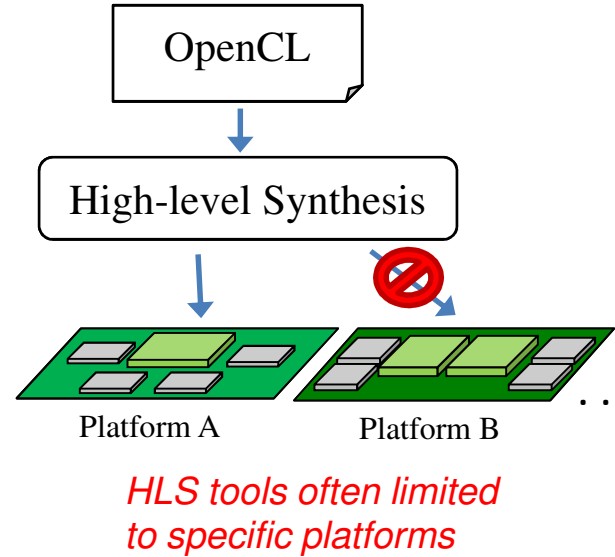


Virtualization Overview

Device Virtualization

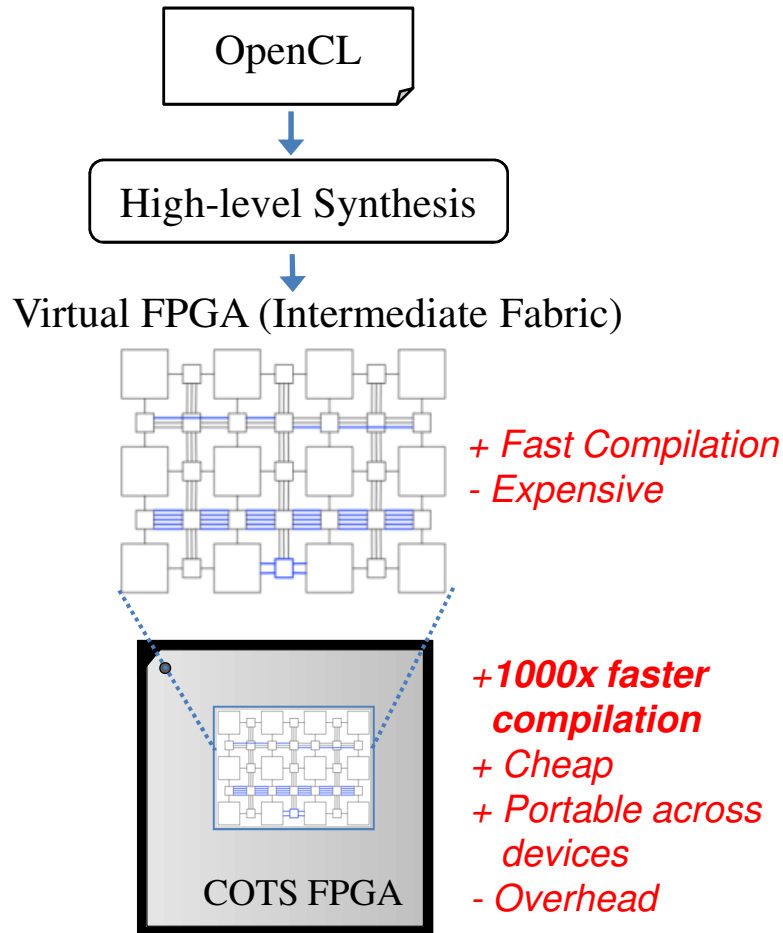


Platform Virtualization

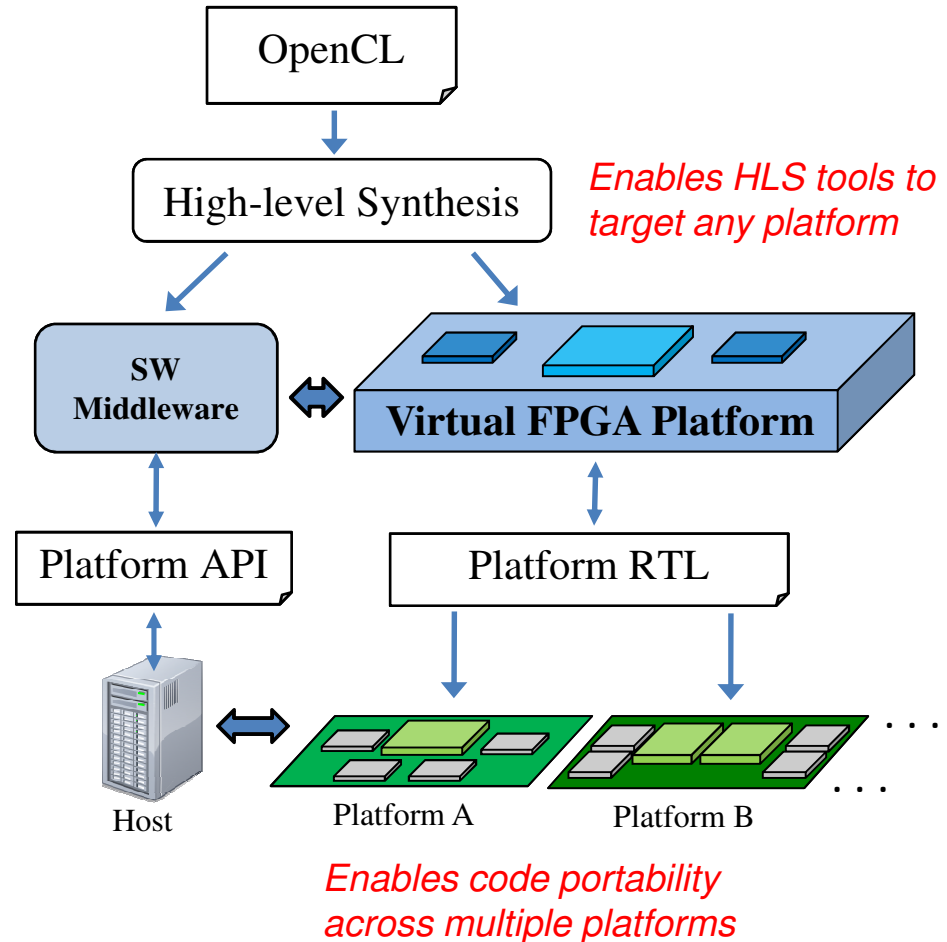


Virtualization Overview

Device Virtualization

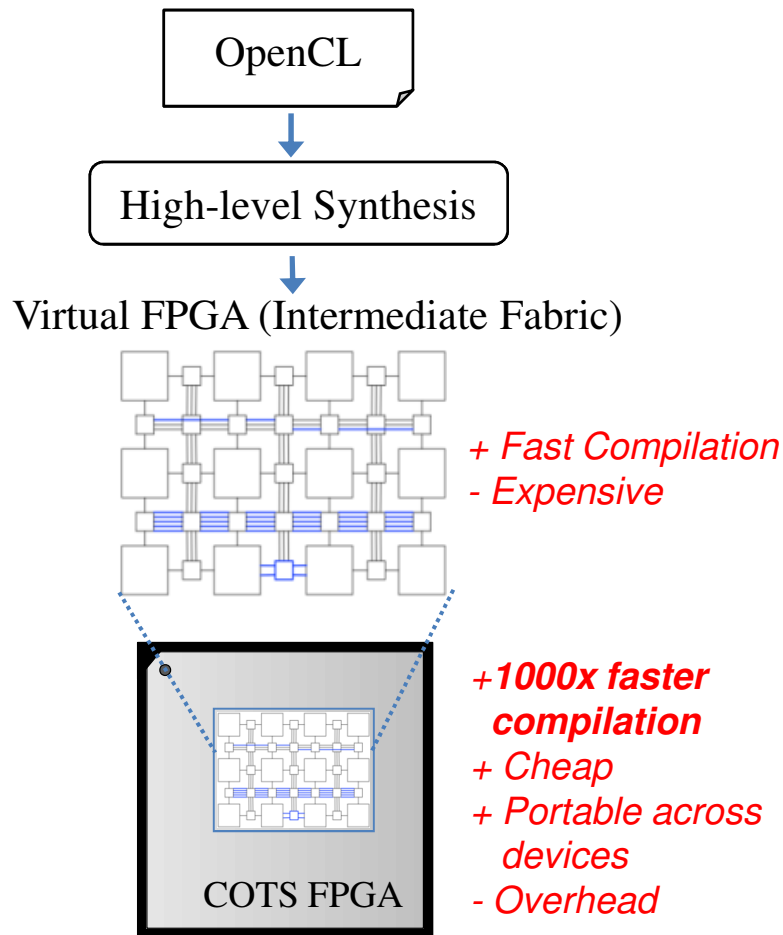


Platform Virtualization

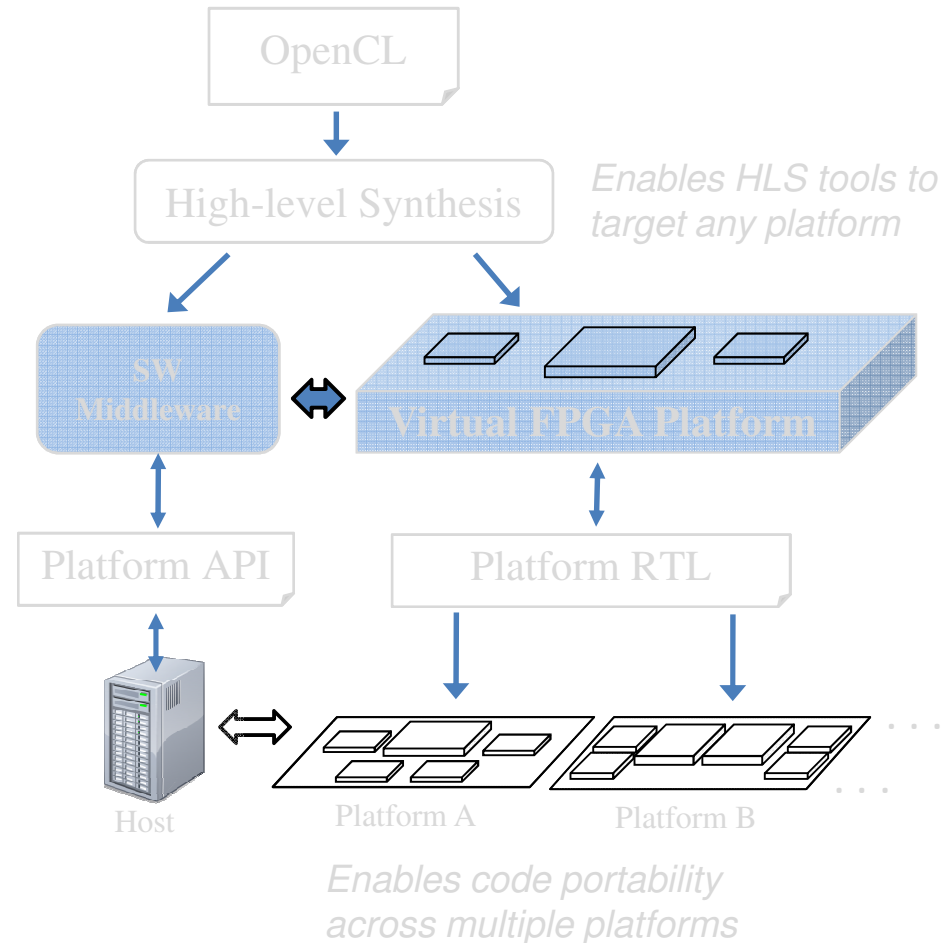


Outline

Device Virtualization

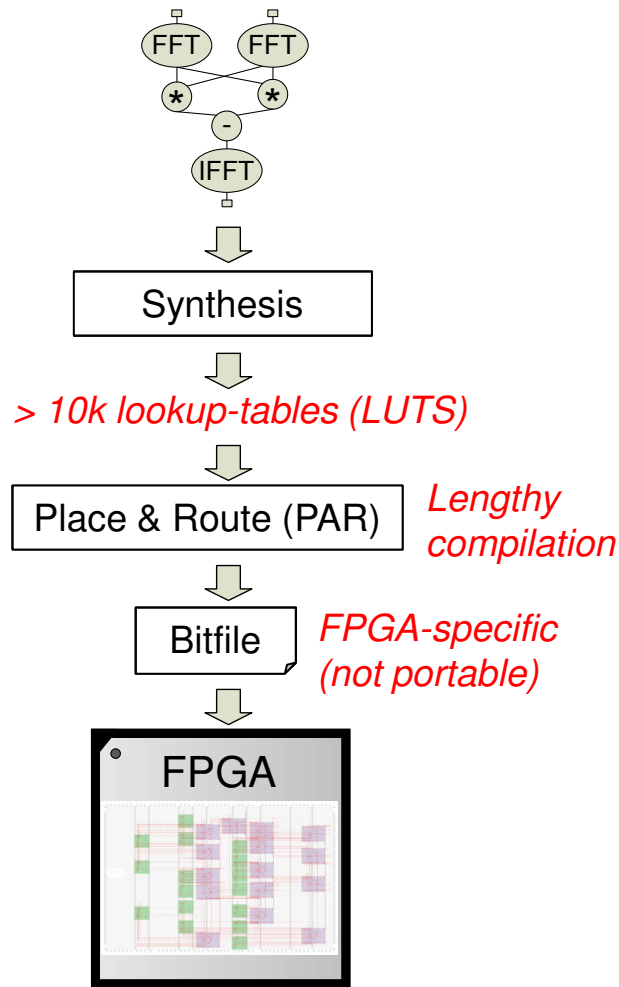


Platform Virtualization

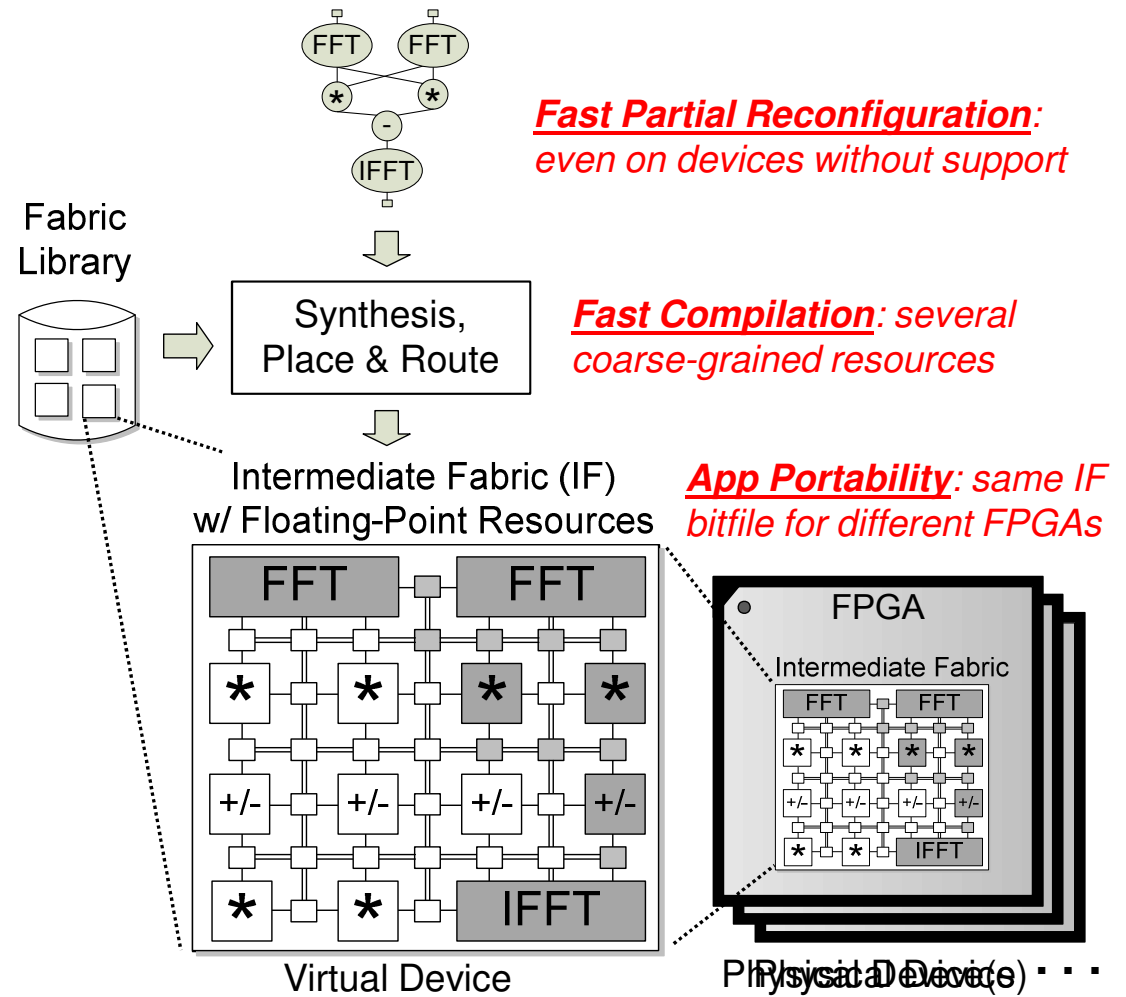


Intermediate Fabric (IF) Overview

Traditional FPGA Tool Flow



Intermediate Fabric Tool Flow

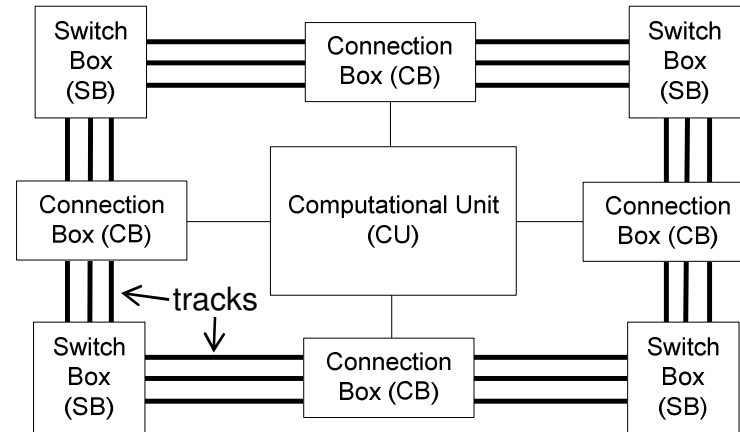


Main Research Challenge: Minimizing Overhead

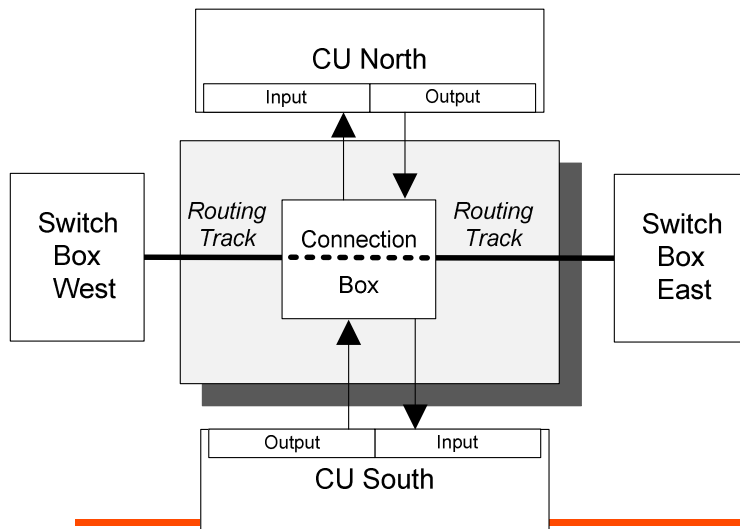
Intermediate Fabric (IF) Architecture

- Fabric can implement *any* architecture
 - Currently focus on island style layout
 - Switch boxes, connection boxes, tracks
 - App. specialized computational units (CUs)
 - FFTs, floating-point resources, filters, etc.
 - Specialized track widths

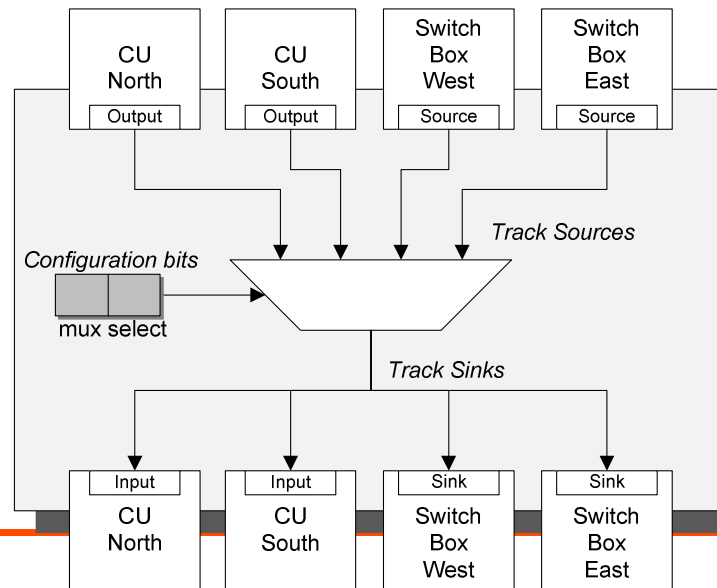
Island-Style Layout



Virtual Track



“Soft” RTL Track Implementation

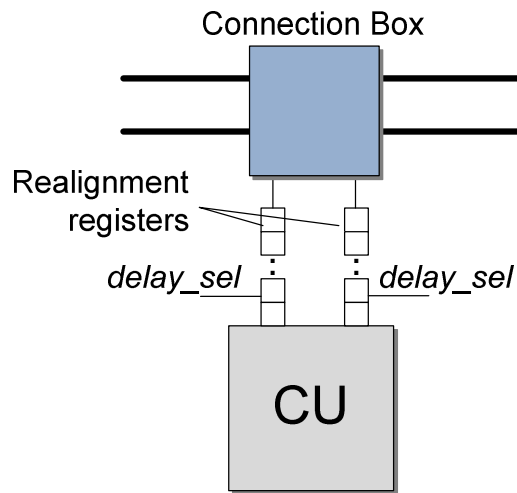
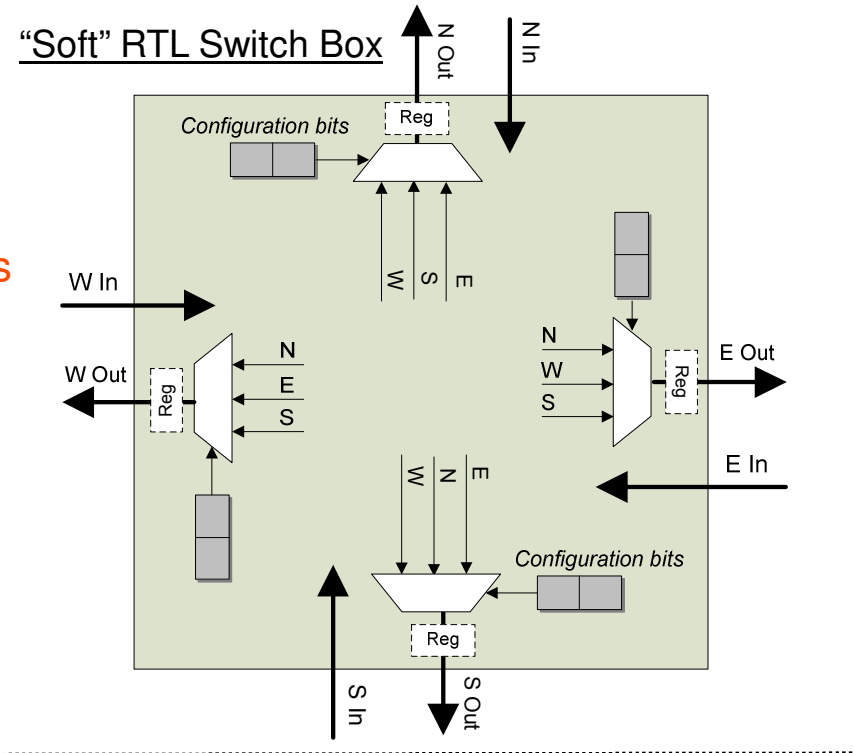


For a n -bit track with m sources, circuit uses a $m:1$, n -bit mux

Many tracks in IF, largest source of overhead

Intermediate Fabric (IF) Architecture, Cont.

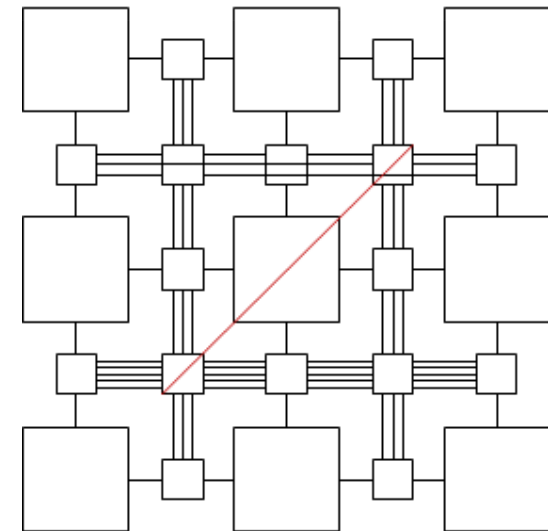
- Switch boxes implemented similarly
 - Mux defines every connection
 - Supports any topology
 - Specialized to application requirements
- Optional registers on outputs
 - Eliminates combinational loops
 - Minimizes delays across muxes



- Pipelined interconnect can require complicated routing
 - Ensures routing paths have same # of hops
- For pipelined circuits, avoid by using realignment registers
 - Lengthens shorter path, adds pipeline stages
 - Enables use of traditional place & route algorithms

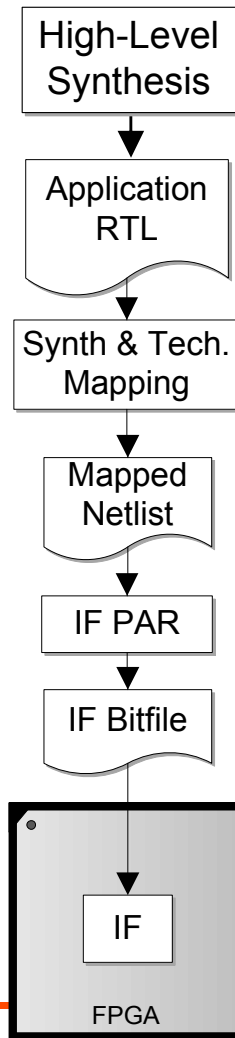
Area Optimization

- Hard resources
 - ❑ Map virtual routing resources directly onto physical resources
 - ❑ Eliminates all routing muxes (> 90% of resource utilization)
 - ❑ But, reduces flexibility and portability
 - Partial reconfiguration not possible unless FPGA supports it
 - Bitfile is specific to each FPGA
- Fabric Specialization
 - ❑ Creating a new virtual fabric is basically free
 - Only cost is time for single FPGA place & route
 - ❑ Therefore, fabrics can be highly specialized to application requirements
 - Both CUs and routing resources
- Specialization techniques
 - ❑ Global: # of CUs, type of CUs, track density, connection box flexibility, switch box topology, long tracks, etc.
 - ❑ Local: wide channels, jump tracks

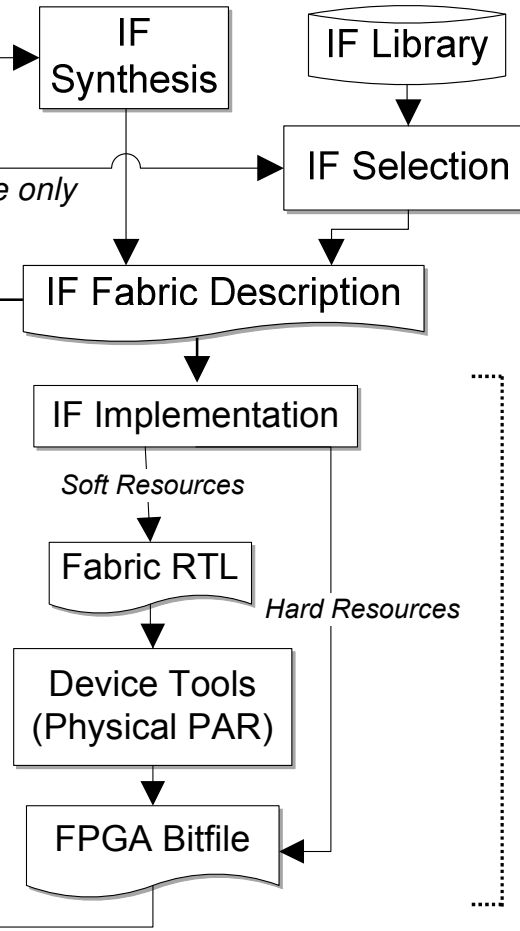


Intermediate Fabric (IF) Tool Flow

App Design Flow



IF Creation Flow



Choose appropriate fabric:

1) Synthesize custom fabric

- + Low area overhead
- - Requires one physical PAR
or

2) Select fabric from library

- + Fabric instantly available
- - Possibly no appropriate IF

Implement IF on FPGA:

1) Soft resources implement virtual fabric as RTL code

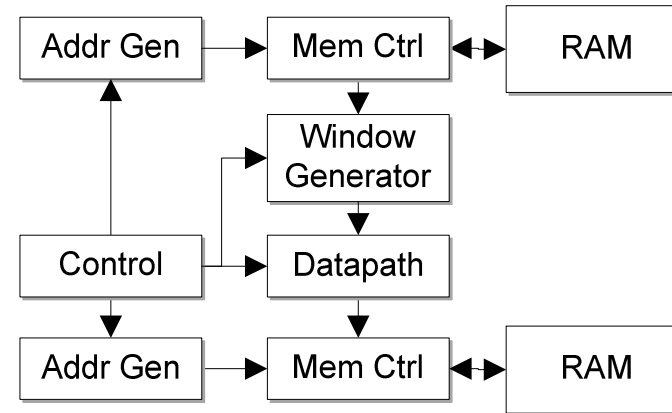
- + Portable, flexible
 - - More overhead
- 2) Hard resources directly use physical routing resources
- + Less overhead
 - - Less portable, flexible

Image-Processing Case Study

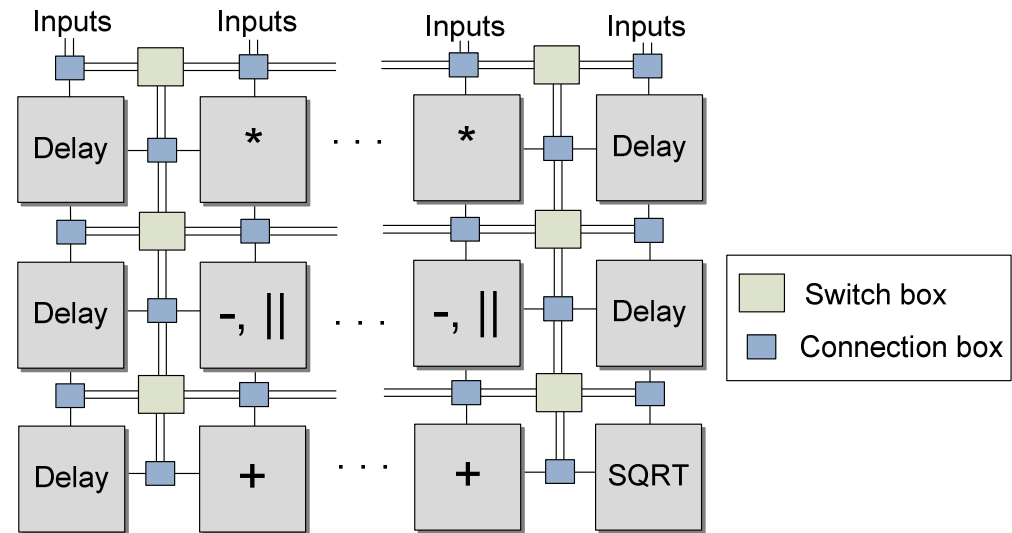
- Intermediate fabric for sliding-window image processing
 - Reconfigurable controller
 - Window generator
 - Provides windows each cycle to datapath
 - Pipelined datapath
 - Address generators
 - RAMs, memory controllers

- Datapath resources
 - Island-style fabric
 - 64 multipliers
 - 64 subtractors with optional absolute value
 - 64 adders
 - 5 delays, up to 9,000 cycles
 - 1 square root
 - 128 inputs, 1 output (not shown)
 - All operations 16-bit fixed point
 - Also, created smaller 32-bit floating-point fabric

Overall Intermediate Fabric Architecture



Pipelined Datapath Fabric



Experimental Setup

- Evaluated 3 sliding-window image-processing circuits
 - Sum-of-absolute differences
 - 2D convolution
 - Sobel edge detection
- Intermediate fabric
 - 16-bit fixed-point fabrics supported up to 8x8 windows
 - Floating-point examples limited to 5x5 windows
 - Configured using FPGA block RAM to store intermediate fabric bitfile
- Target system
 - GiDEL PROCStar III PCIe x8 FPGA accelerator card
 - Altera Stratix III E260 FPGA
 - 2.26 GHz quad-core Intel E5520, used for data transfer
- Comparisons
 - Circuit area and performance using IF vs. directly on FPGA
 - Place & route times of IF tools vs. FPGA tools
 - Quartus II 9.1, SP2 on 2.66 MHz Intel W3520, 12 GB of RAM
 - Application speedup compared to sequential software
 - C++ compiled with g++ 4.4.3 with -O3 optimizations

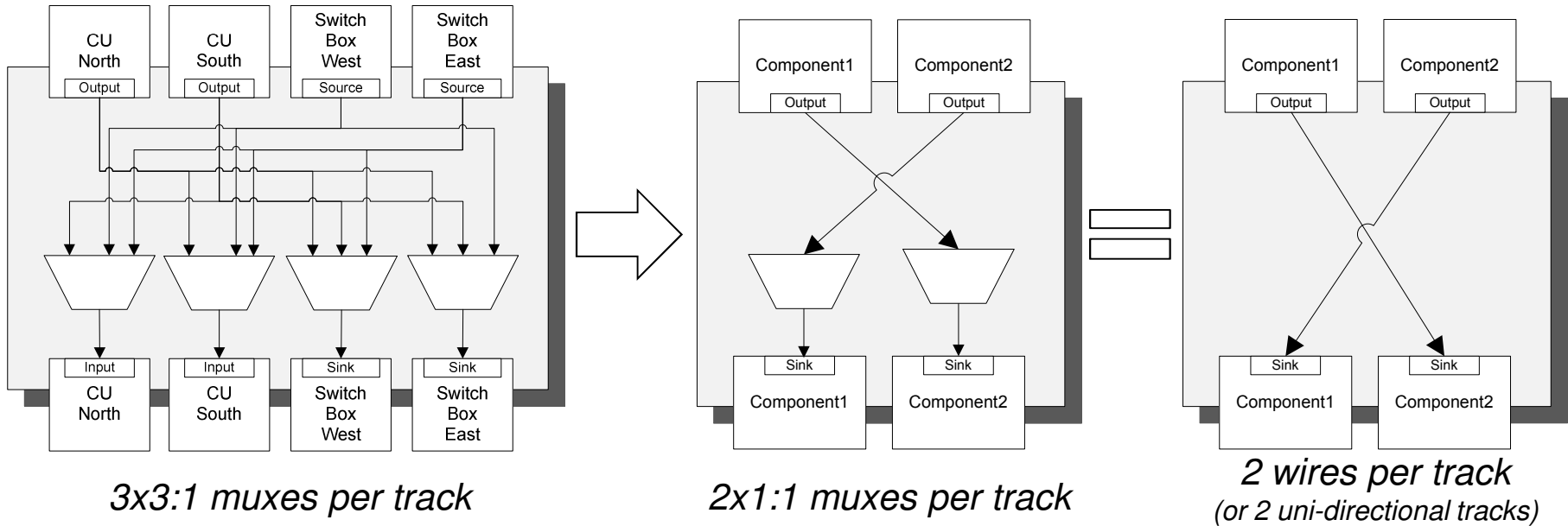
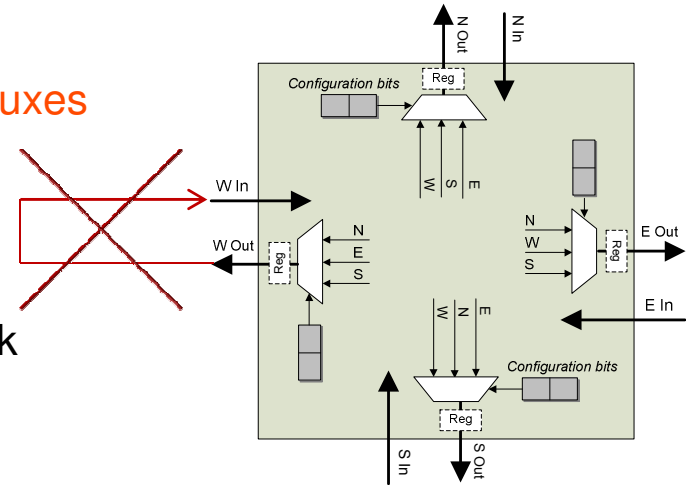
Experimental Results

	Place and Route Times			Performance					Area Utilization		
	<i>IF</i>	<i>Quartus 9.1</i>	<i>Speedup</i>	<i>Clk FPGA</i>	<i>Overhead</i>	<i>Speedup</i>		<i>Perf. Overhead</i>	<i>LUT</i>	<i>REG</i>	<i>DSP</i>
						<i>IF</i>	<i>FPGA</i>				
Conv 3x3	0.9s	14min 48s	943	150 MHz	17%	3.2	3.5	9%	13%	14%	1%
Conv 4x4	1.5s	15min 06s	613	148 MHz	16%	5.9	6.3	6%	13%	14%	2%
Conv 5x5	2.1s	15min 33s	447	146 MHz	15%	8.0	8.5	6%	13%	14%	4%
Conv 6x6	3.0s	15min 41s	312	151 MHz	18%	11.1	11.9	7%	13%	15%	5%
Conv 7x7	4.0s	16min 19s	243	139 MHz	11%	14.7	15.5	5%	13%	15%	8%
Conv 8x8	5.3s	16min 08s	184	146 MHz	15%	18.8	20.0	6%	13%	15%	8%
Sobel	4.2s	14min 56s	214	154 MHz	19%	0.53	0.58	9%	13%	14%	1%
SAD 8x8	5.3s	16min 51s	190	143 MHz	13%	18.6	19.5	5%	15%	16%	0%
Conv 5x5 (float)	1.7s	25min 28s	919	148 MHz	23%	5.2	5.5	5%	21%	29%	13%
Sobel (float)	1.5s	18min 58s	759	144 MHz	21%	0.32	0.36	11%	16%	19%	3%
SAD 5x5 (float)	0.6s	30min 43s	2880	140 MHz	19%	5.3	5.5	4%	25%	38%	0%
Average	2.7s	18min 14s	700	146 MHz	17%	8.3	8.8	7%	15%	18%	4%
IF				124 MHz					57%	58%	17%
IF (float)				114 MHz					59%	61%	13%

- Avg. place & route speedup: 700x
- Avg. performance overhead: 7%
- IF used 2.2x to 4.4x more LUTs than FPGA circuits
 - However, IF pessimistically implemented with only soft routing resources
 - IF also capable of implementing numerous circuits with fast partial reconfiguration (28-72 cycles)
 - Compared to an FPGA circuit with 3-4 kernels, IF saves area (assuming kernels are not concurrent)

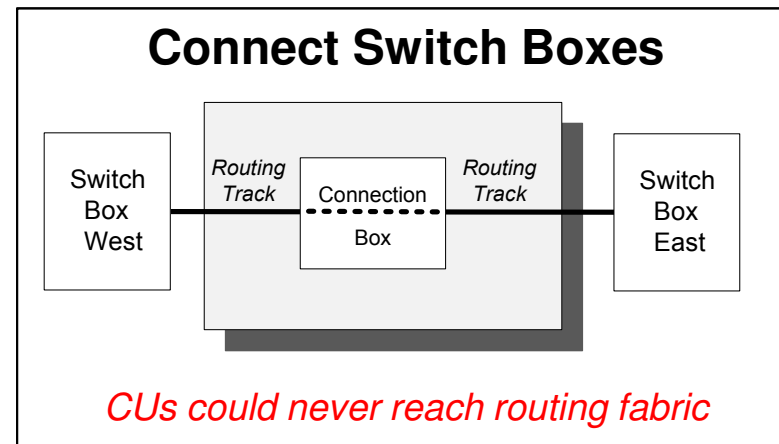
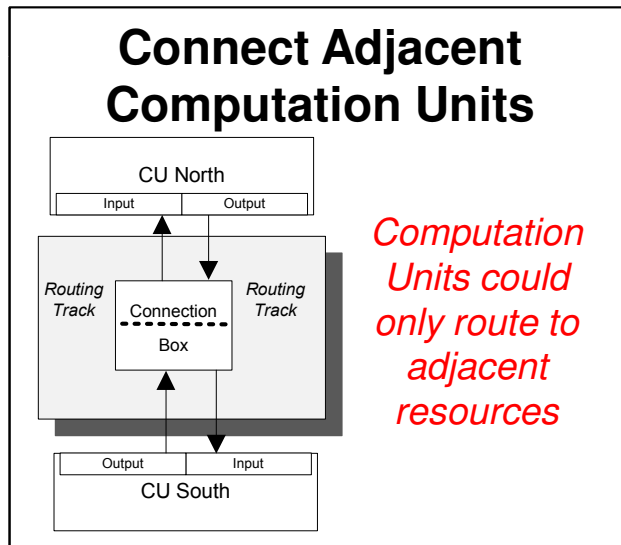
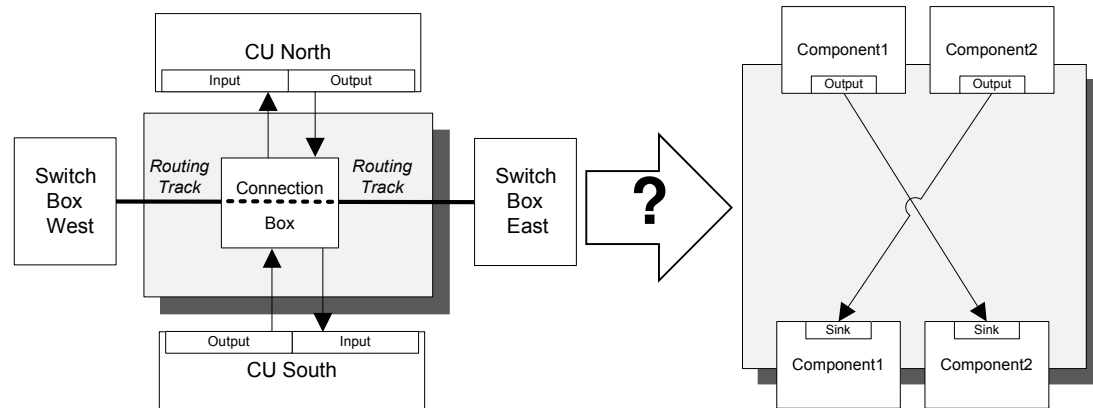
Low-Overhead Interconnect

- Results show muxes are the largest IF overhead
 - **Goal: low-overhead interconnect that reduces muxes**
- Eliminate redundant connections
 - **We can replace 1 n:1 mux with n, n-1:1 muxes**
 - e.g. 4:1 mux on each track -> 4 3:1 muxes
- Reduce number of source components to 2 per track
 - **Equivalent to two uni-directional wires**



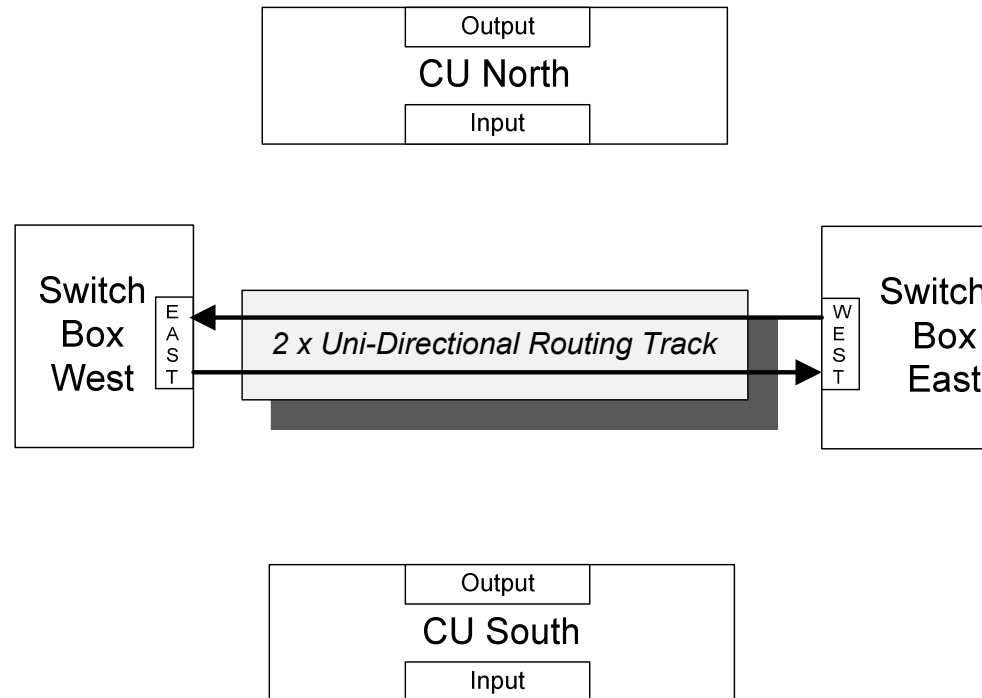
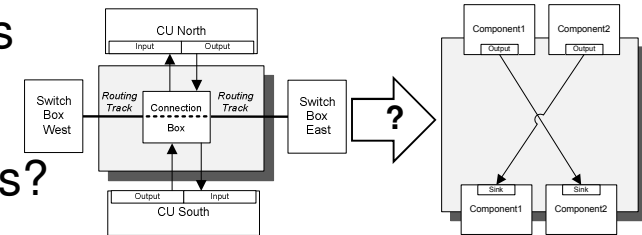
How to Achieve Two-Source Tracks?

- How can we reduce the number of track sources while preserving routability?



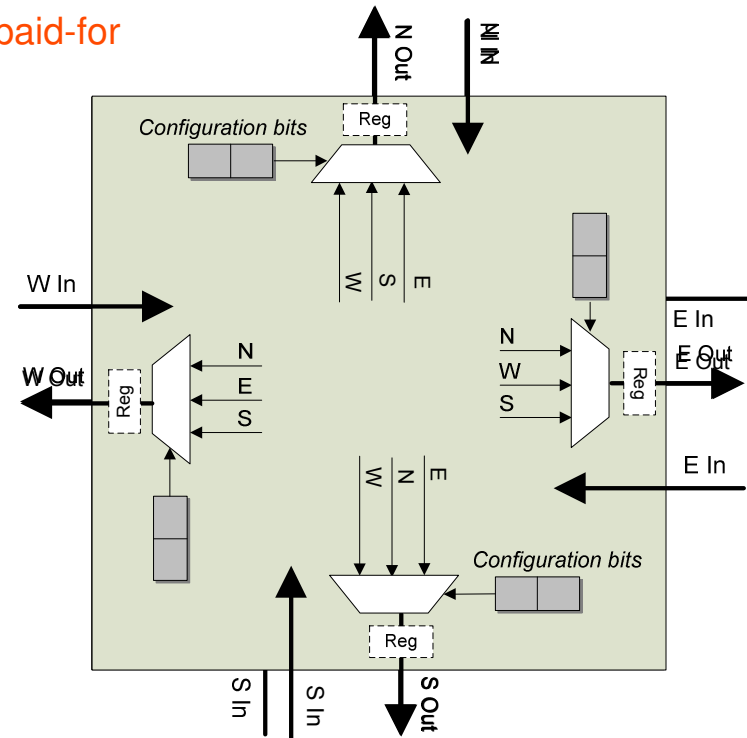
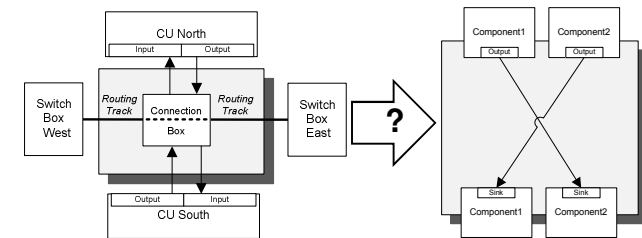
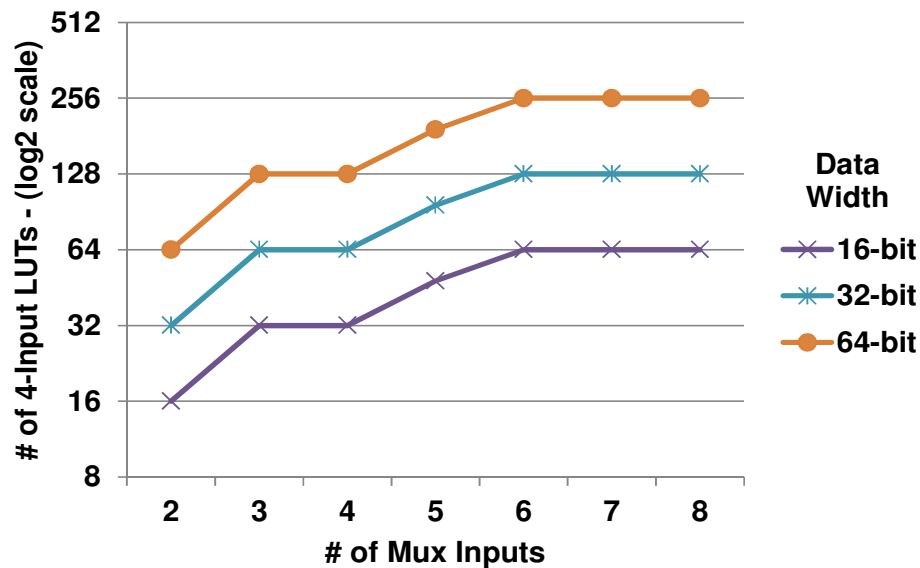
How to Achieve Two-Source Tracks?

- Need to connect two Switch Boxes and two CUs
- Begin by connecting adjacent Switch Boxes
- Now how can we route the remaining CU signals?



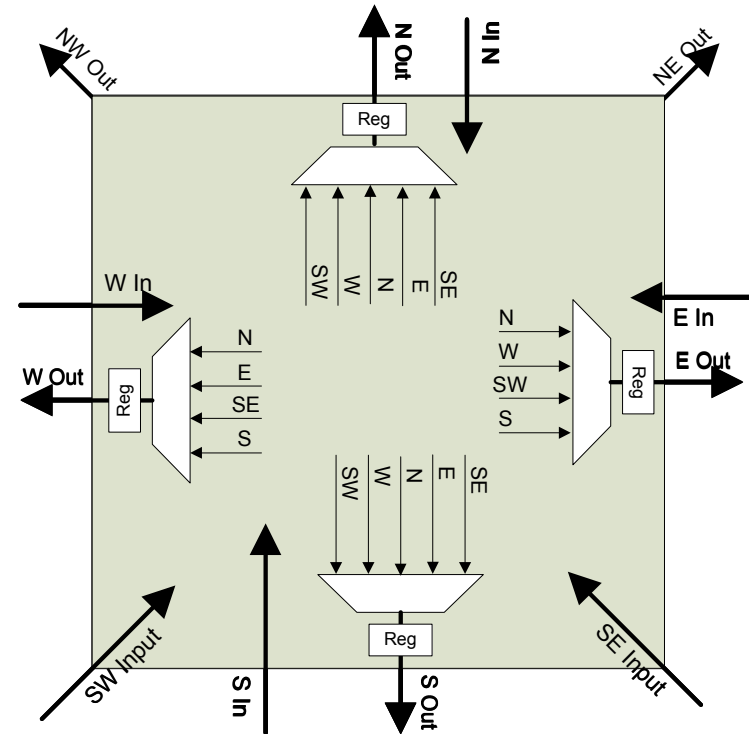
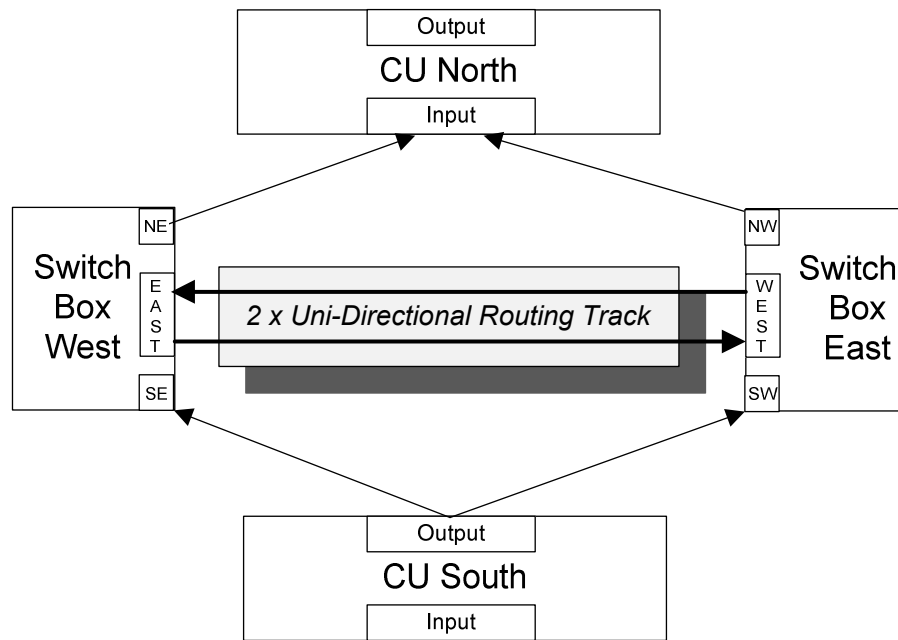
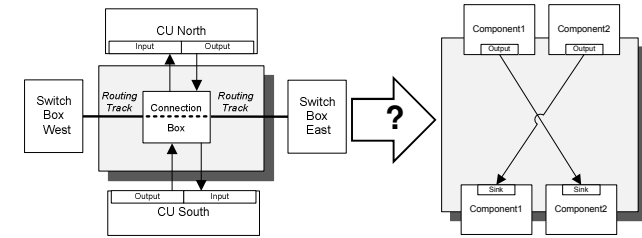
Mux Area Optimization

- FPGA-implemented mux area utilization plateaus as number of inputs grows
- For Xilinx Virtex 4 and later, no penalty to move from 3-input to 4-input muxes
- Switchboxes contain 4x3-input muxes
 - Therefore each has one unutilized input already paid-for
- Can we use these to route CU signals?



How to Achieve Two-Source Tracks?

- Connect CUs directly to switch box using free mux inputs
- Now, tracks require no muxes
 - Switch box muxes may increase depending on routing requirements



Results: Low-Overhead Interconnect

- Average Results
 - ~47% area reduction
 - Uncertain impacts on routability and clock speed due to large outliers and high variability
 - 2.4x PAR speedup vs. previous interconnect
 - 1350x PAR speedup vs. FPGA

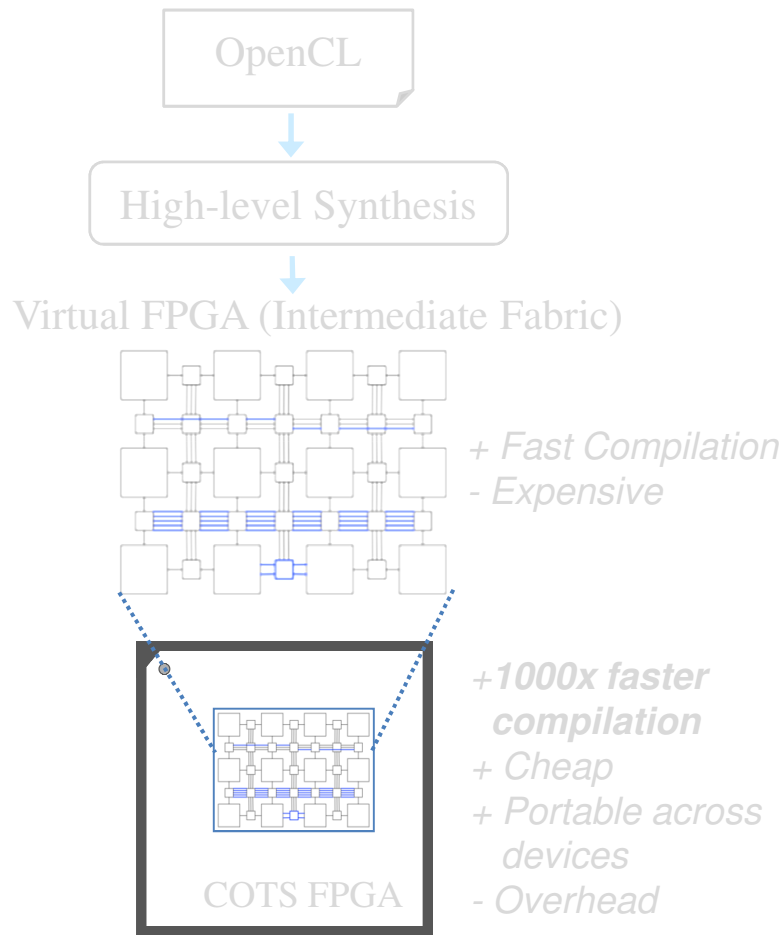
	Place-and-Route Time					Area and Routability			Clock Speed		
	IF Prev	IF New	FPGA	Speedup Prev	Speedup New	LUT Savings	Flip-Flop Savings	Routability Overhead	IF Prev	IF New	Clock Overhead
<i>Matrix multiply FXD</i>	0.6s	0.6s	1min 08s	112x	112x	56%	60%	1%	170 MHz	186 MHz	-9%
<i>Matrix multiply FLT</i>	0.6s	0.6s	6min 06s	602x	602x	59%	59%	1%	184 MHz	222 MHz	-21%
<i>FIR FXD</i>	0.6s	0.6s	0min 33s	54x	58x	45%	41%	5%	174 MHz	158 MHz	9%
<i>FIR FLT</i>	0.6s	0.6s	4min 36s	454x	484x	35%	35%	5%	203 MHz	215 MHz	-6%
<i>N-body FXD</i>	0.5s	0.2s	0min 57s	126x	300x	40%	32%	1%	185 MHz	165 MHz	11%
<i>N-body FLT</i>	0.5s	0.2s	3min 42s	491x	1168x	37%	26%	1%	218 MHz	200 MHz	8%
<i>Accum FXD</i>	0.1s	0.02s	0min 26s	280x	1733x	52%	53%	0%	186 MHz	187 MHz	-1%
<i>Accum FLT</i>	0.1s	0.02s	0min 30s	323x	2000x	52%	50%	0%	225 MHz	241MHz	-7%
<i>Normalize FXD</i>	0.2s	0.3s	1min 10s	299x	241x	66%	71%	-63%	178 MHz	162 MHz	9%
<i>Normalize FLT</i>	0.2s	0.3s	6min 44s	1726x	1393x	43%	54%	-63%	197 MHz	222 MHz	-13%
<i>Bilinear FXD</i>	0.3s	0.3s	1min 08s	230x	213x	51%	47%	0%	184 MHz	165 MHz	10%
<i>Bilinear FLT</i>	0.3s	0.3s	8min 48s	1784x	1650x	41%	42%	0%	206 MHz	200 MHz	3%
<i>Floyd-Steinberg FXD</i>	0.1s	0.1s	1min 27s	621x	926x	53%	50%	2%	182 MHz	169 MHz	7%
<i>Floyd-Steinberg FLT</i>	0.1s	0.1s	5min 37s	2407x	3585x	48%	44%	2%	196 MHz	179 MHz	9%
<i>Thresholding</i>	1.4s	1.3s	0min 33s	24x	26x	44%	36%	5%	167 MHz	181MHz	-8%
<i>Sobel</i>	0.3s	0.4s	2min 28s	500x	344x	44%	31%	2%	181MHz	162 MHz	10%
<i>Gaussian Blur</i>	3.3s	2.2s	3min 19s	60x	90x	39%	41%	-42%	170 MHz	181MHz	-6%
<i>Max Filter</i>	0.2s	0.03s	1min 16s	444x	2533x	48%	41%	0%	186 MHz	176 MHz	5%
<i>Mean Filter 3x3</i>	0.2s	0.01s	2min 30s	962x	10714x	52%	52%	10%	185 MHz	187 MHz	-1%
<i>Mean Filter 5x5</i>	1.9s	1.9s	3min 25s	110x	108x	64%	65%	-1%	169 MHz	161MHz	5%
<i>Mean Filter 7x7</i>	8.9s	4.7s	5min 03s	34x	64x	39%	40%	-38%	157 MHz	183 MHz	-17%
Average	1.0s	0.7s	2min 56s	554x	1350x	48%	46%	-8%	186 MHz	186 MHz	0%

Intermediate Fabric Summary

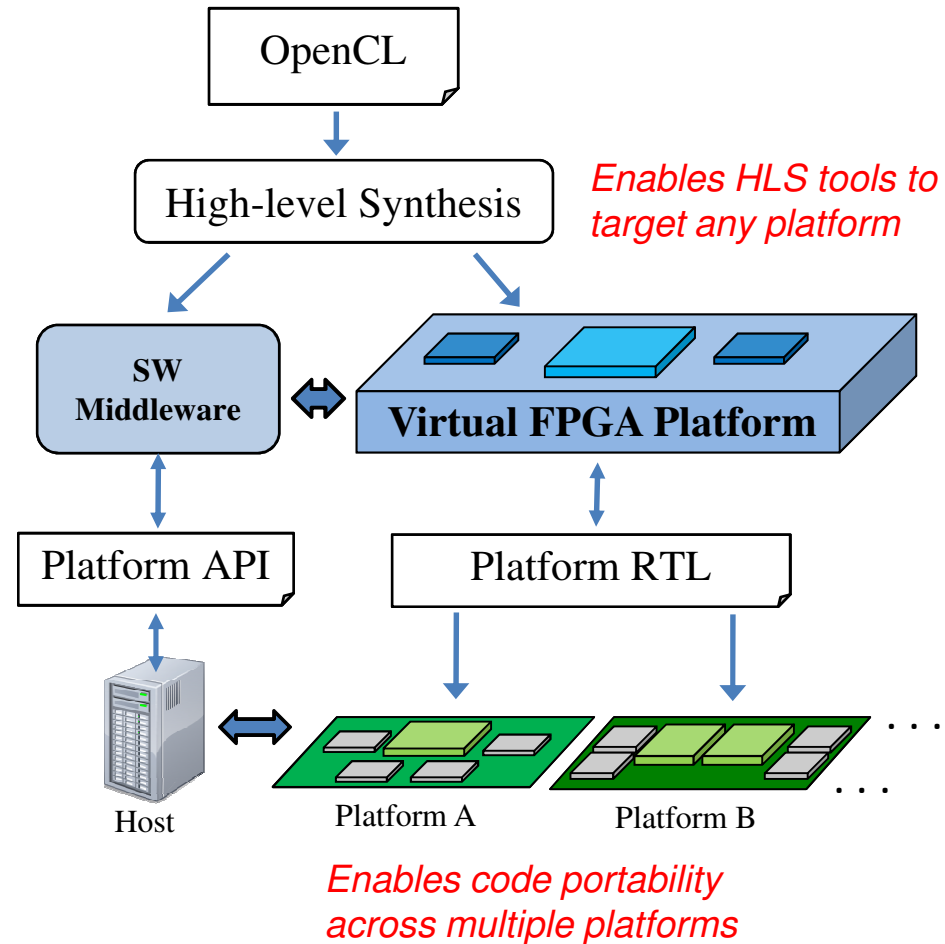
- Intermediate fabrics are application-specialized, virtual FPGAs
 - ~1000x faster place-and-route than vendor tools
 - Application portability across physical FPGAs
- Main limitation: overhead
 - Initial overhead was 2.2x-4.4x larger than individual FPGA circuits
 - Worse for small, embedded designs
 - Optimizations to interconnect reduced overhead by 50%
- Future work
 - Integrate with OpenCL synthesis
 - Investigate novel virtual architectures (non-island-style)
 - Preliminary results show 10x reduction in overhead

Outline

Device Virtualization

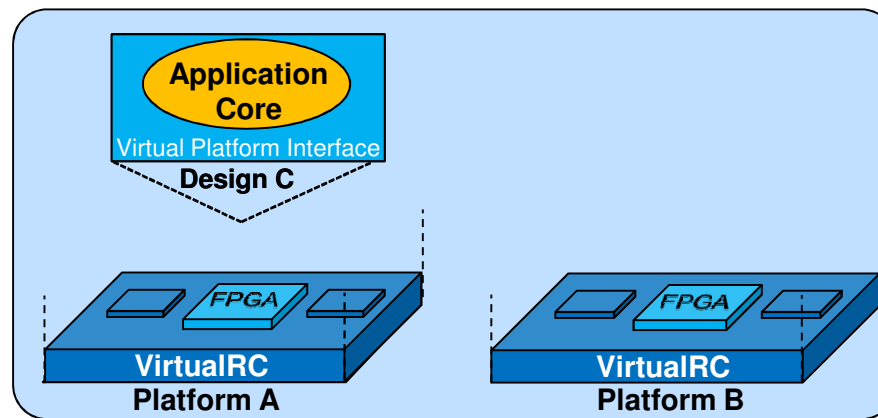


Platform Virtualization



Platform Virtualization

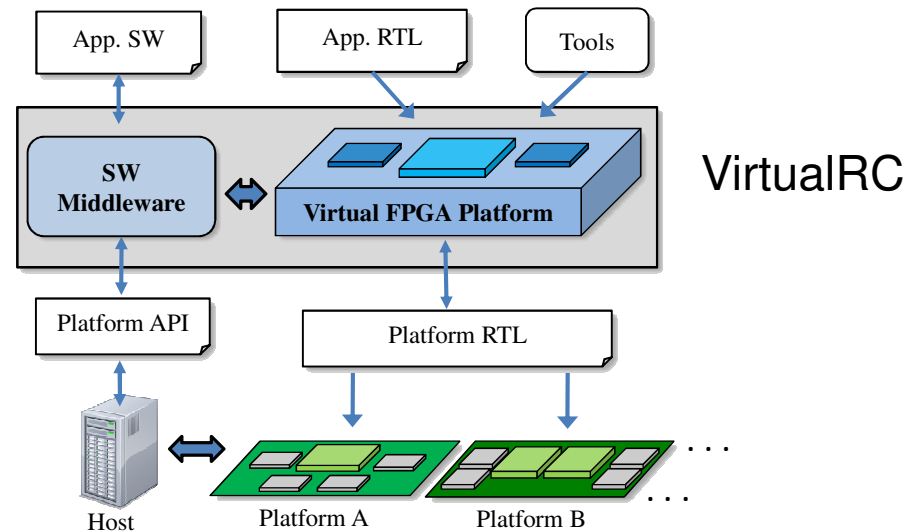
- Introduced VirtualRC virtual platform
 - Abstracts away platform-specific details
 - Same virtual interface provided across different platforms



- Enables application and tool portability across heterogeneous platforms
 - VirtualRC handles underlying virtual-to-physical translation

VirtualRC

- VirtualRC is composed of two major components:
 - *SW Middleware* provides a portable software API to virtual resources
 - *Virtual FPGA Platform* provides virtual interfaces to platform resources



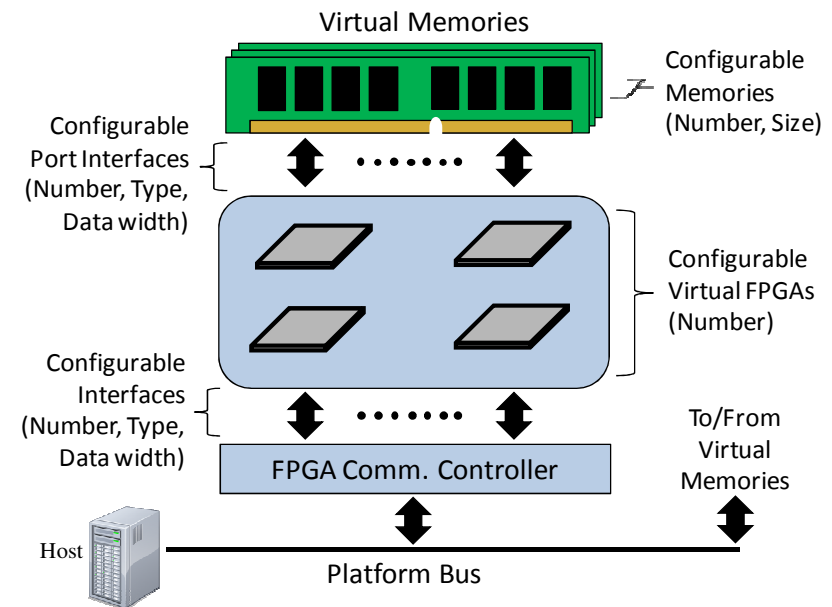
- VirtualRC integrated into RC Middleware (RMCW) CHREC Project
 - Available to public

Virtual Platform Interface

- VirtualRC's virtual platform interface is user customizable
 - Developers can customize the virtual platform based on application requirements

- Customizable options include:







- Virtual memories
 - Number, size, and interface
- Customizable virtual FPGAs
 - Number of top-level interfaces
- Customizable host interface
 - Number, type and data width



- For example, a developer working on convolution may use two virtual memory inputs and one virtual memory output
 - Also simplifies application development overhead

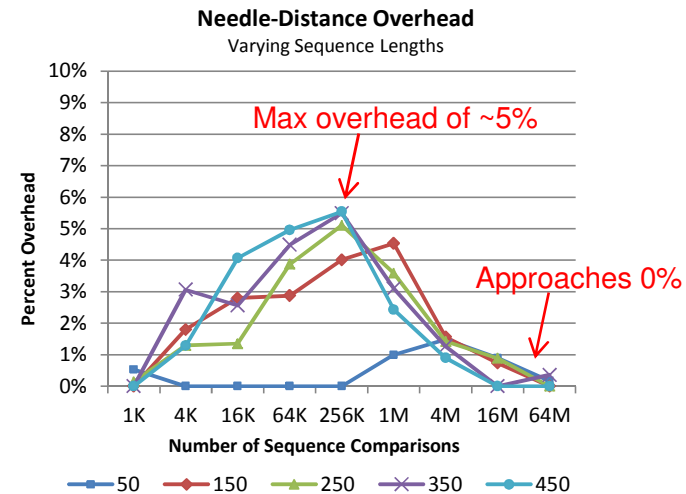
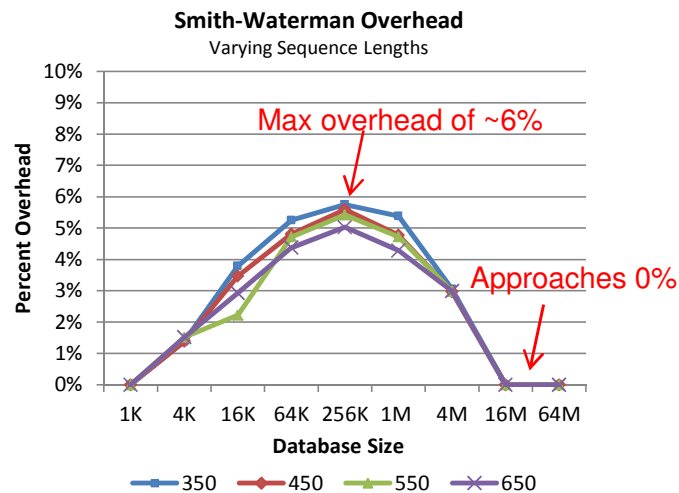
Experimental Setup

- Evaluated VirtualRC performance overhead, resource overhead, and portability
 - Three significantly different FPGA platforms:
 - GiDEL PROCStar III, Nallatech H101 and Pico Computing M501

  GiDEL PROCStar III	  Pico M501	  Nallatech H101
4 FPGAs/board	1 FPGA/board	1 FPGA/board
Altera Stratix-III	Xilinx Virtex-6	Xilinx Virtex-4
3 memory banks/FPGA: - 256/512 MB DDR2 - 2 x 2GB DDR2	- 512MB DDR3	- 512MB DDR2 - 4 x 4MB DDR2 SRAM
PCIe x8	PCIe x8	133MHz PCI-x

Overhead Analysis

- Compared native platform vs. VirtualRC
 - Two computational biology applications



- Benchmarks demonstrate less than 1% overhead for large transfers from FPGA to ext. memory and host to ext. memory
- Average resource overhead of less than 1% measured for application case-studies

Portability Analysis

- Demonstrated application and tool portability using a variety of applications and kernels
 - Applications and kernels were created for VirtualRC or obtained from OpenCores (www.opencores.org)

Application Studies

	PROCStar III		M501		H101	
	Freq. (MHz)	Time (ms)	Freq. (MHz)	Time (ms)	Freq. (MHz)	Time (ms)
<i>1D Convolution FP</i>	125	39.29	125	247.90	100	91.06
<i>2D Convolution FP</i>	106	13.18	106	15.18	100	43.25
<i>Option Pricing</i>	125	12.15 s	125	14.40 s	-	-
<i>Sum Abs. Differences</i>	98	14.72	98	15.62	98	86.71
<i>Needle Distance</i>	125	194.00	125	116.20	100	199.51
<i>Smith Waterman</i>	125	116.00	125	133.00	100	225.00
<i>Image Segmentation</i>	125	12.40	125	16.39	100	4.81
<i>OpenCores SHA256</i>	125	64.05	125	120.49	100	25.97
<i>OpenCores FIR</i>	125	24.51	125	413.80	100	106.16
<i>OpenCores AES128</i>	125	25.33	125	503.78	100	126.18
<i>OpenCores JPEG Enc.</i>	125	15.29	125	23.93	100	21.24

Tool Studies

	PROCStar III		M501		H101	
	Freq. (MHz)	Time (ms)	Freq. (MHz)	Time (ms)	Freq. (MHz)	Time (ms)
<i>ROCCC 8pt FFT</i>	125	15.66	125	16.61	100	39.91
<i>ROCCC 5-tap FIR</i>	125	17.78	125	18.73	100	40.57
<i>AutoESL Convolution</i>	125	4.29	125	7.31	100	2.49

- Applications worked on all platforms without any changes
- ROCCC and AutoESL transparently compiled to three different platforms

Conclusions

- Long compilation times and portability prevent mainstream high-level synthesis
- Demonstrated that virtualization can address these problems
- Device virtualization (Intermediate Fabrics)
 - 1000x faster place-and-route
 - Application portability across devices
 - Area overhead can be significant (but improving)
- Platform virtualization (VirtualRC, CHREC RCMW)
 - Allows code portability across platforms
 - Enables high-level synthesis to target any platform
 - Overhead is minimal
 - Future work: maximize usage of platform-specific resources



Questions?

- COOLE, J., AND STITT, G. Intermediate fabrics: Virtual architectures for circuit portability and fast placement and routing. In *CODES/ISSS '10: Proceedings of the IEEE/ACM/IFIP international conference on hardware/Software codesign and system synthesis* (October 2010), pp. 13–22.
- STITT, G., AND COOLE, J. Intermediate fabrics: Virtual architectures for near-instant FPGA compilation. *Embedded Systems Letters, IEEE* 3, 3 (sept. 2011), 81 –84.
- LANDY, A. AND STITT, G. A low-overhead interconnect architecture for virtual reconfigurable fabrics. In *CASES'12: Proceedings of the 2012 international conference on compilers, architectures and synthesis for embedded systems (October 2012)*, pp. 111-120.
- KIRCHGESSNER, R., STITT, G., GEORGE, A., AND LAM, H. VirtualRC: a virtual FPGA platform for applications and tools portability. In *FPGA '12: Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays* (New York, NY, USA, February 2012), FPGA '12, ACM, pp. 205–208.