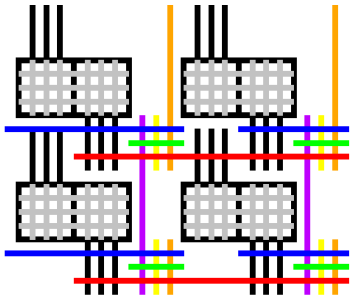
Fast Linking of Separately-Compiled FPGA Blocks without a NoC

Yuanlong Xiao, Syed Tousif Ahmed, and André DeHon
ylxiao,stahmed@seas.upenn.edu, andre@ieee.org

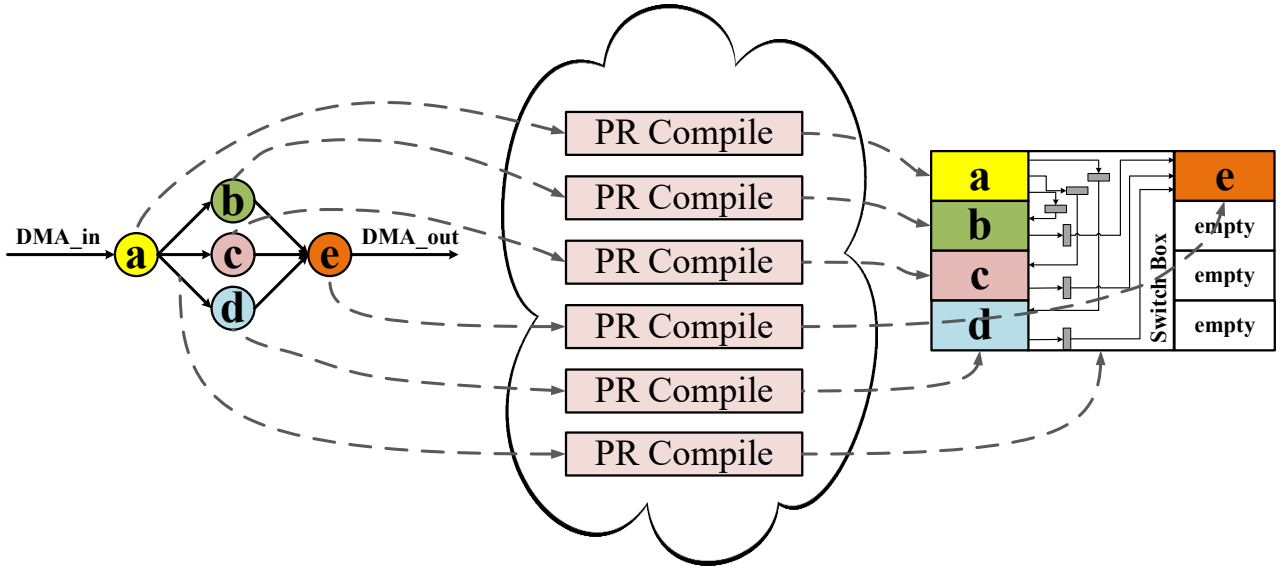
Implementation of Computation Group
University of Pennsylvania
December 12th, 2019



Fast Linking of Separately-Compiled FPGA Blocks without a NoC

Content:

- Motivation
- Approach
- Evaluation
- Future Work
- Conclusion



Motivation

- Today's FPGA compilation is slow
  - 30-178 minutes for Rosetta [1] Benchmarks on Xilinx ZCU102 board [2]
    - Compiled on Google Cloud Computing Engines.
    - 4 Dual-thread, 2.8GHz Intel Xeon Cascade Lake Processors
    - 64GB RAM
    - SDSoC runs with 8 threads

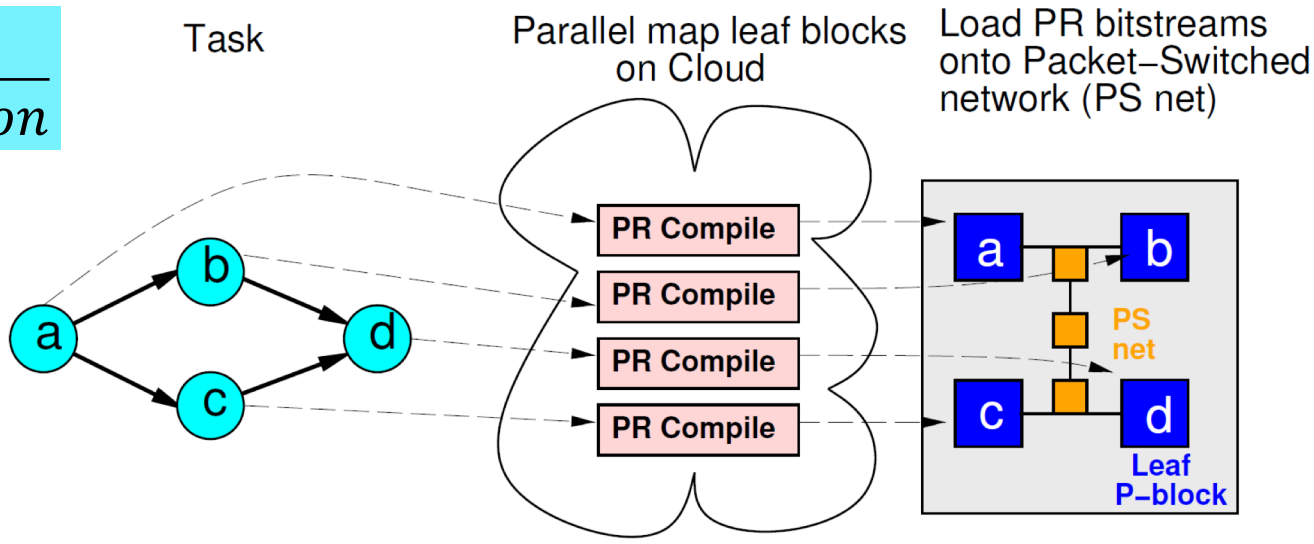
Benchmark[1]	SDSoC Comile Time[2]
Digit Recognition	**41.2 min**
SPAM Filter	**29.5 min**
3-D Rendering	**29.4min**
Optical Flow	**44.3 min**
Binarized Neural Network	**178 min**
Face Detection	**71.9 min**

[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Motivation

- PRFlow[2] can reduce compile time to 10-16 minutes
  - Big Idea: **Divide-and-conquer**
  - Pre-compile Overlay & Compile Separate Blocks in Parallel
  - 10-16 minutes Compile Time

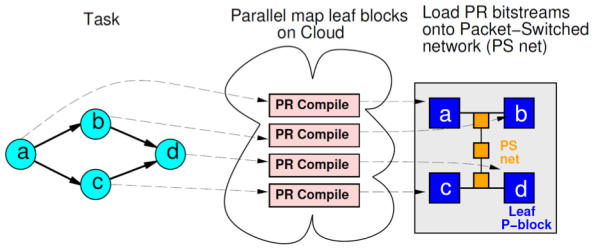
$$T_{new} = \frac{T_{origin}}{\#\ of\ Partition}$$



[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Motivation

- PRFlow[2] can reduce compile time to 10-16 minutes
  - Big Idea: **Divide-and-conquer**
  - Pre-compile Overlay & Compile Separate Blocks in Parallel
  - 10-16 minutes Compile Time



Benchmark[1]	SDSoC Comile Time[2]	PRFlow [2]
Digit Recognition	41.2 min	**10.6 min**
SPAM Filter	29.5 min	**10.9 min**
3-D Rendering	29.4min	**10.9 min**
Optical Flow	44.3 min	**12.4 min**
Binarized Neural Network	178 min	**16.4 min**
Face Detection	71.9 min	**16.2 min**

[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Motivation

- Performance Degradation
  - Slow down by 1.3-12.4 times compared with SDSoC version

Benchmark [1]	Run Time Per Input SDSoC [2]	Run Time Per Input PRFlow [2]
Digit Recognition	13.00 ms	16.58 ms (1.3X)
SPAM Filter	82.13 ms	48.90 ms (0.5X)
3-D Rendering	6.17 ms	1.18 ms (0.19X)
Optical Flow	6.35 ms	25.80 ms (4.1X)
Binarized Neural Network	5.30 ms	17.42 ms (3.3X)
Face Detection	28.19 ms	351.93 ms (12.4X)

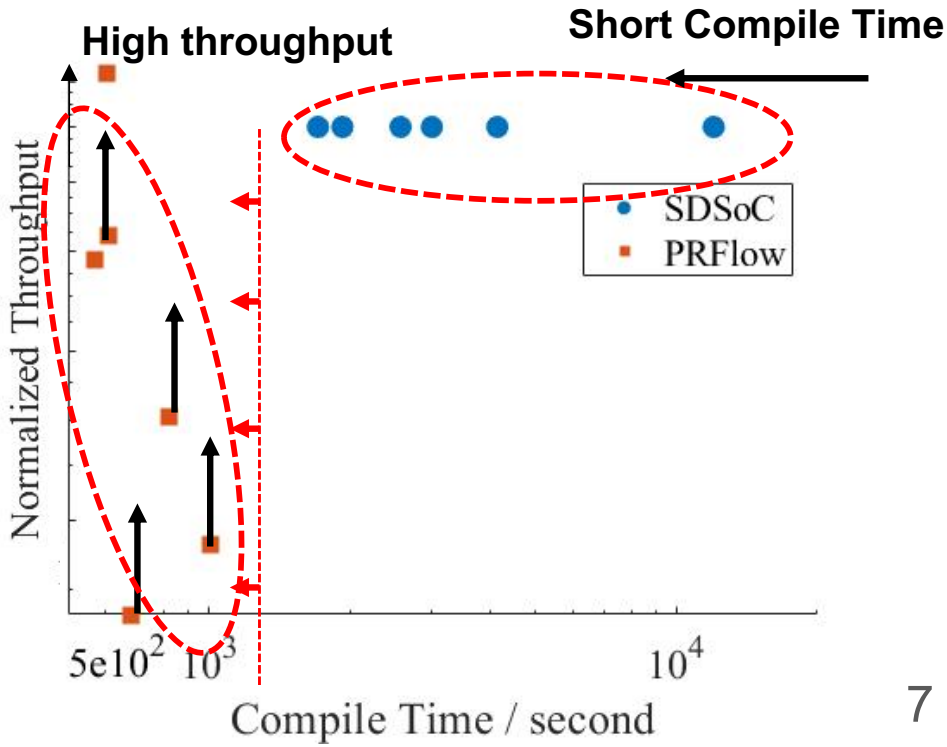
[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Motivation

- Today's FPGA compilation is slow: **30-178 minutes**
- PRFlow[2] can reduce compile time to **10-16 minutes**
- PRFlow[2] can degrade performance by **1.2-12.5 times**

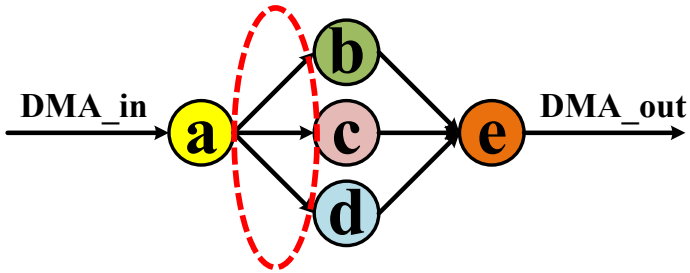
We propose:

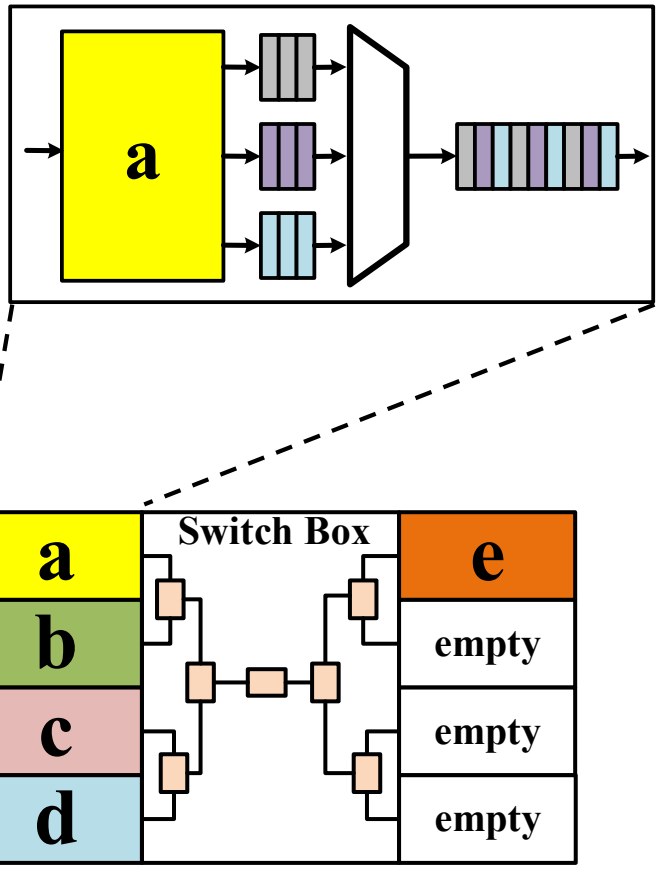
- Keep Separate Compilations
- Exploit More FPGA bandwidth



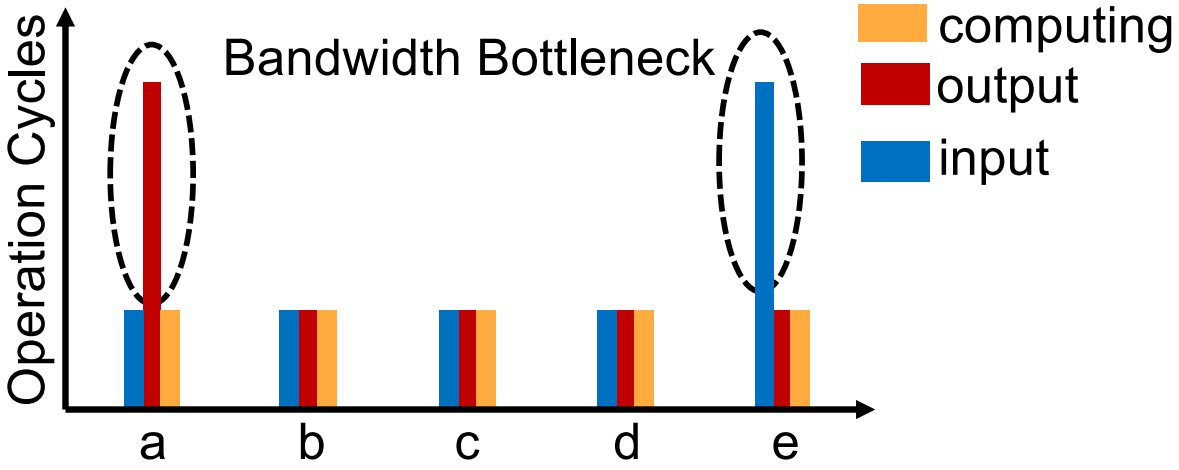
Motivation: Packet-Switched Network-on-Chip (PSNoC) IO Bottleneck

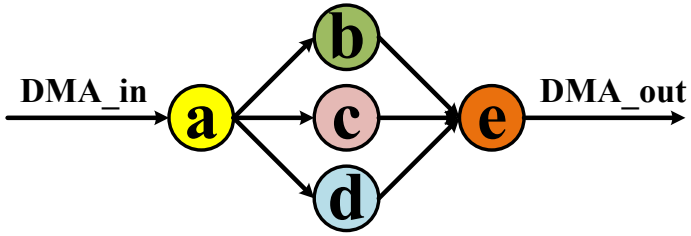
- 1 cycle for computing
- 1 cycle for read/write per link

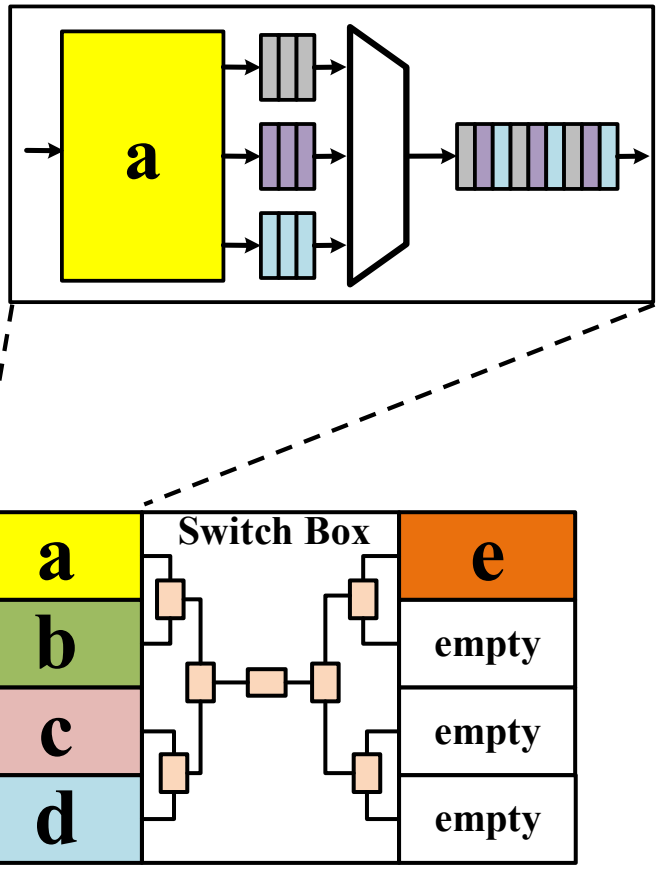




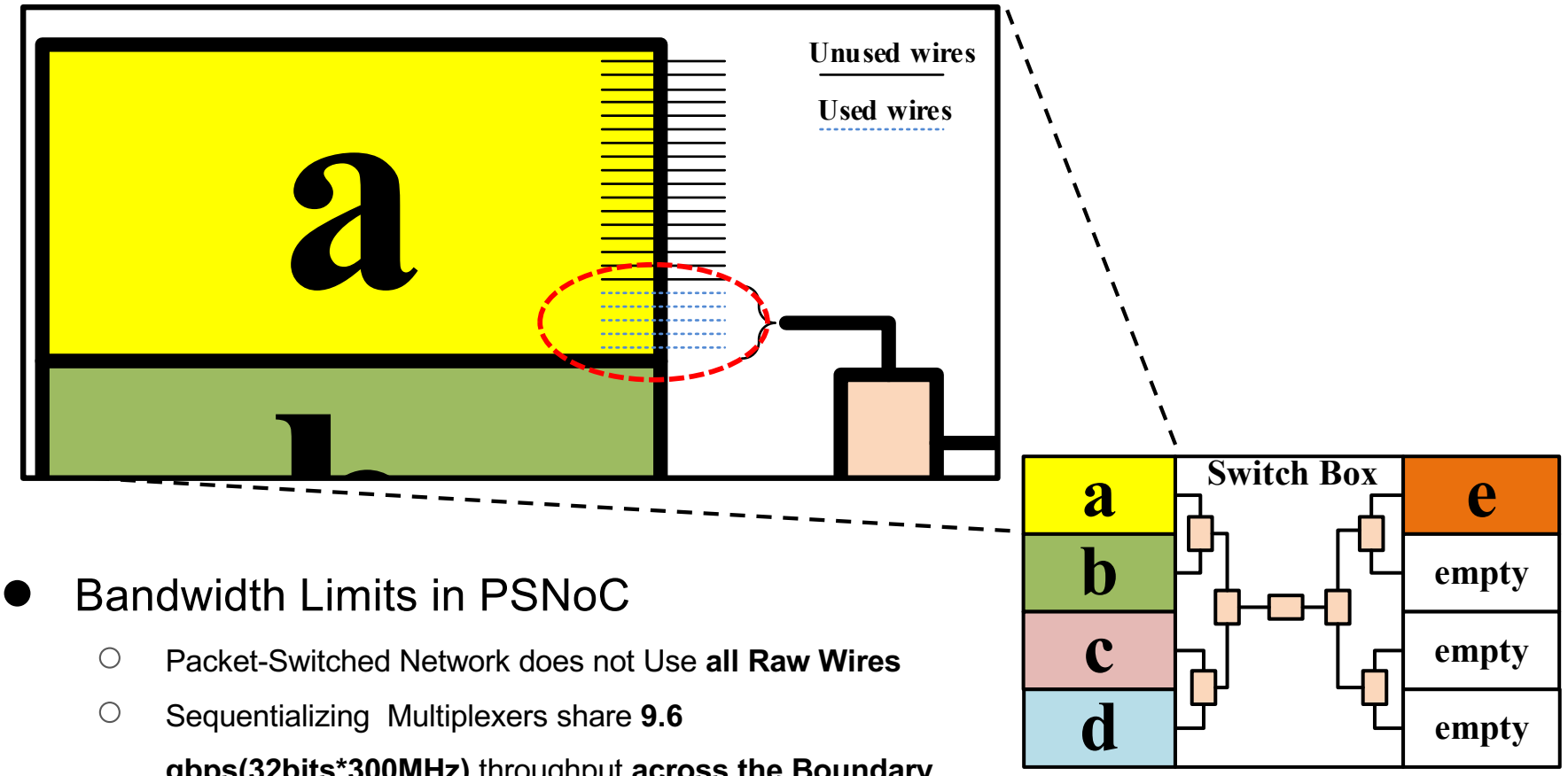
Motivation: Packet-Switched Network-on-Chip (PSNoC) IO Bottleneck







Motivation: Packet-Switched Network-on-Chip (PSNoC) IO Bottleneck



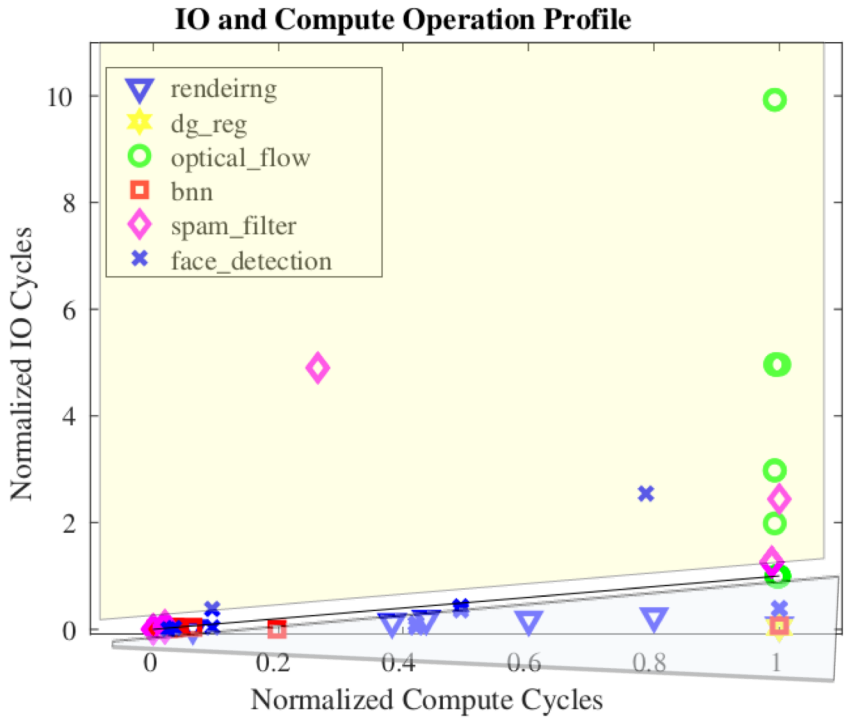
- Bandwidth Limits in PSNoC
  - Packet-Switched Network does not Use **all Raw Wires**
  - Sequentializing Multiplexers share **9.6 gbps(32bits*300MHz)** throughput **across the Boundary**

Motivation: Benchmark[2] Profile on PSNoC

We need more Bandwidth!

$$NormComputeCycles = \frac{ComputeCyles}{Max\{AllComputeCycle\}}$$

$$NormIoCycles = \frac{IoCyles}{Max\{AllComputeCycle\}}$$

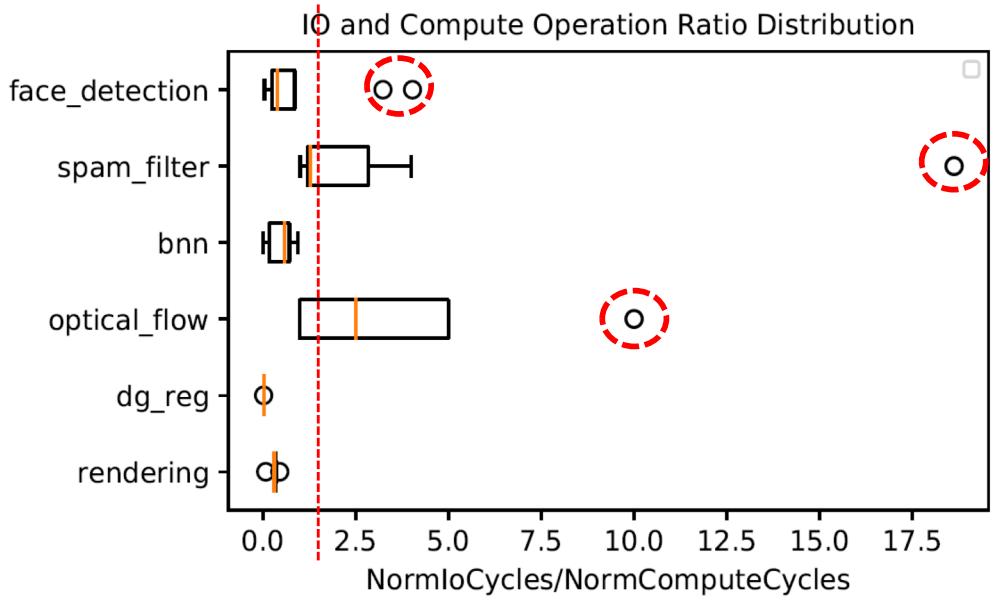


[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Why not increase the PSNoC datawidth?

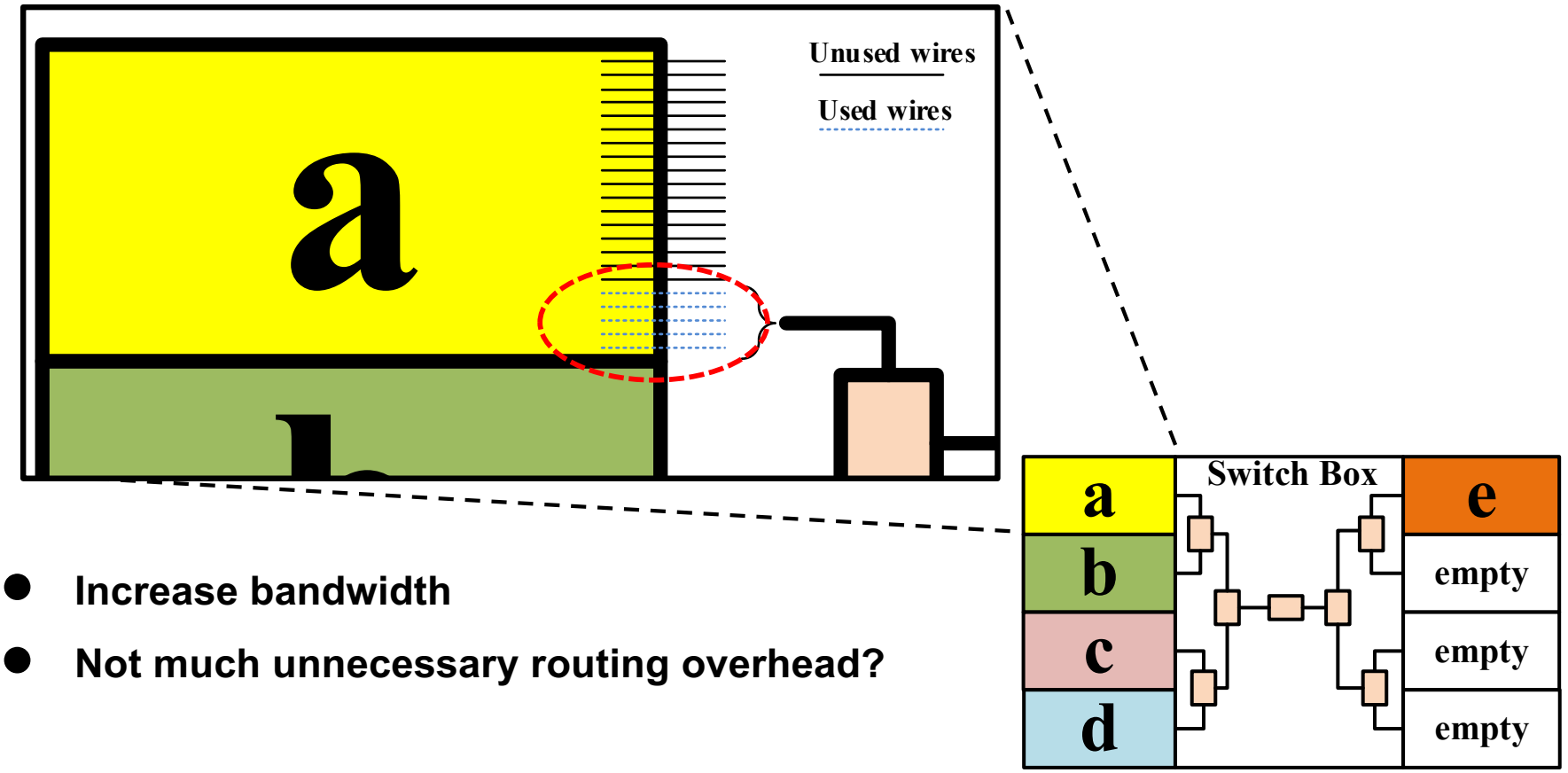
32 bits PSNoC Occupies 20% LUTs

Only limited number of links need more bandwidth



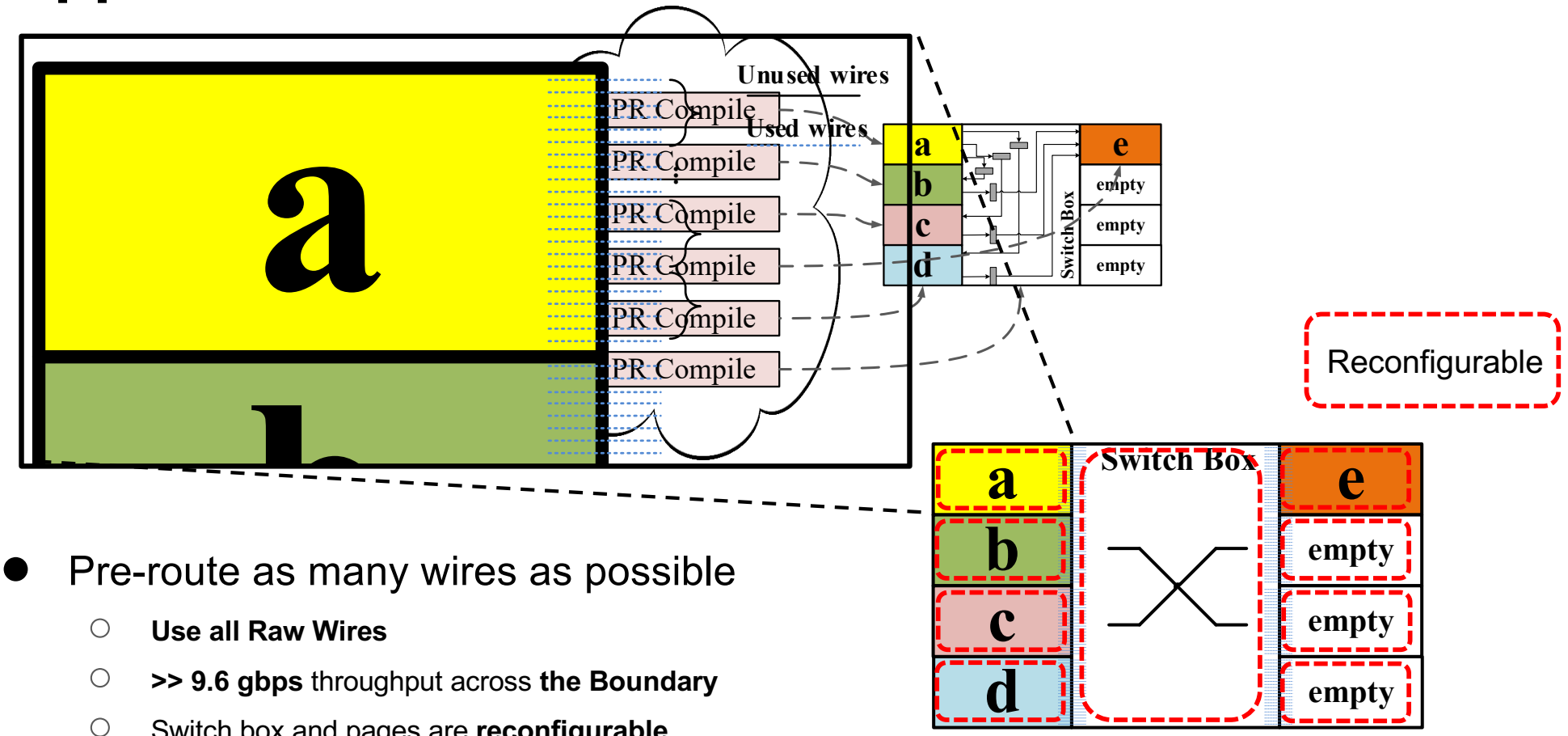
[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Motivation: Packet-Switched Network-on-Chip (PSNoC) IO Bottleneck



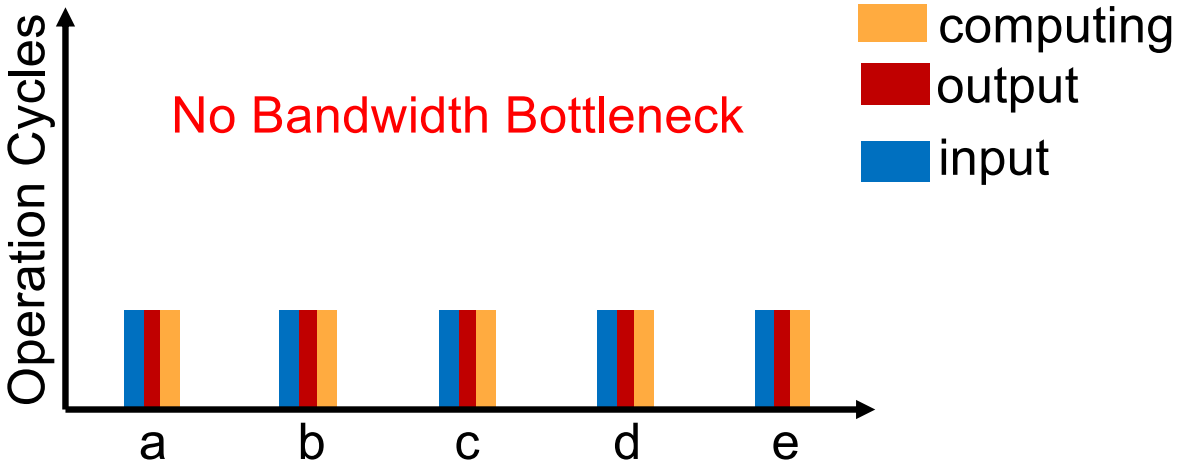
- Increase bandwidth
- Not much unnecessary routing overhead?

Approaches: Direct Wires



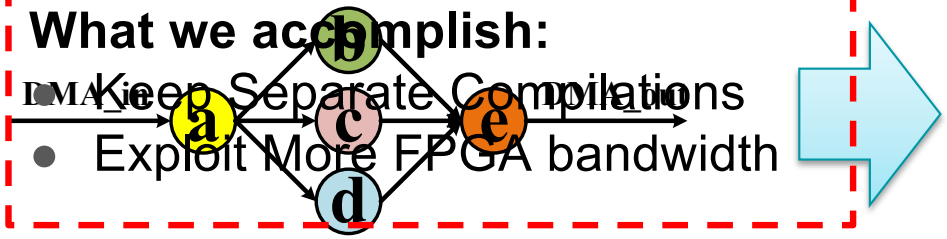
- Pre-route as many wires as possible
  - **Use all Raw Wires**
  - **>> 9.6 gbps** throughput across **the Boundary**
  - Switch box and pages are **reconfigurable**
  - Compiled **in parallel**

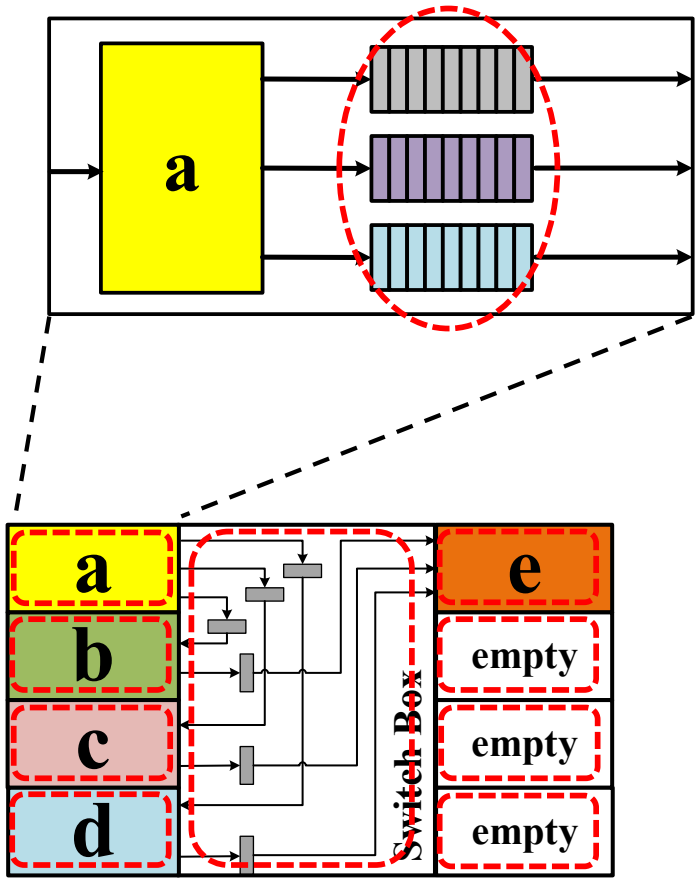
Approaches: Direct Wires



What we accomplish:

- Keep Separate Compilations
- Exploit More FPGA bandwidth





Approaches: Bandwidth Upper Bound

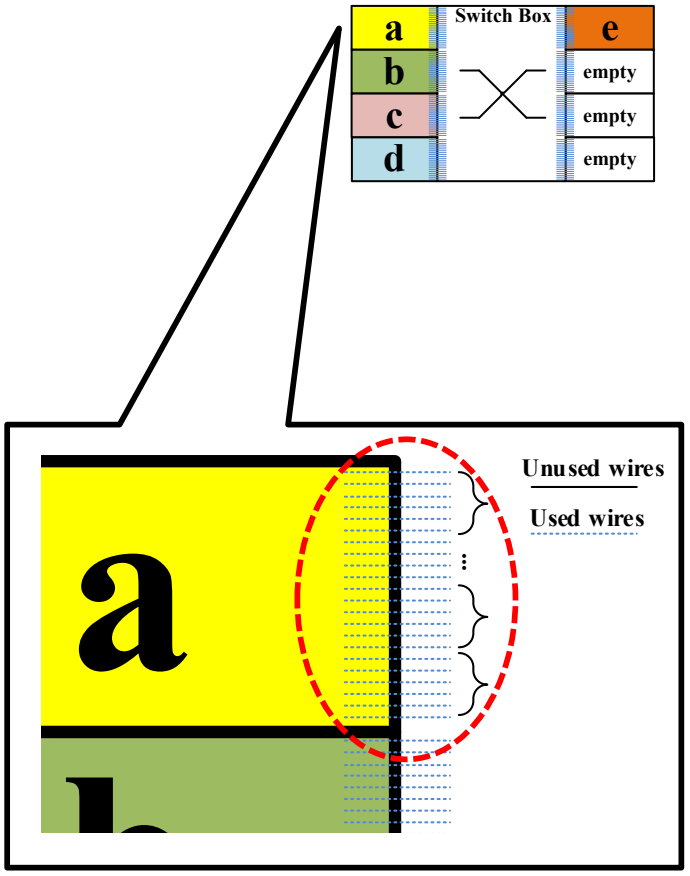
We have more wires!

- Max number of Wire cross the boundary
- Maximum: 86.4Gbps (288X300MHz)

TABLE II: Static Timing Analysis Slack for 60 CLB Boundary

W \ F(MHz)	100	200	250	300	400
32	3.761	0.861	X	X	X
64	1.187	0.284	0.287	X	-0.853
96	3.151	0.438	-0.764	X	X
128	X	X	X	0.021	-1.009
160	3.292	0.536	X	X	-0.905
192	3.262	X	X	X	X
224	X	X	X	-0.01	X
256	X	0.431	0.071	0.002	X
288	3.262	X	X	0.009	X

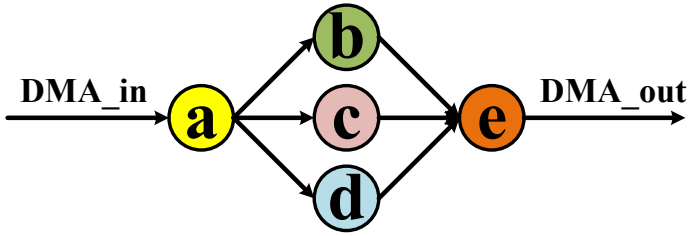
Numbers in cells represent the slack; slack less than zero fail to meet the timing target. X means the routing cannot be completed.

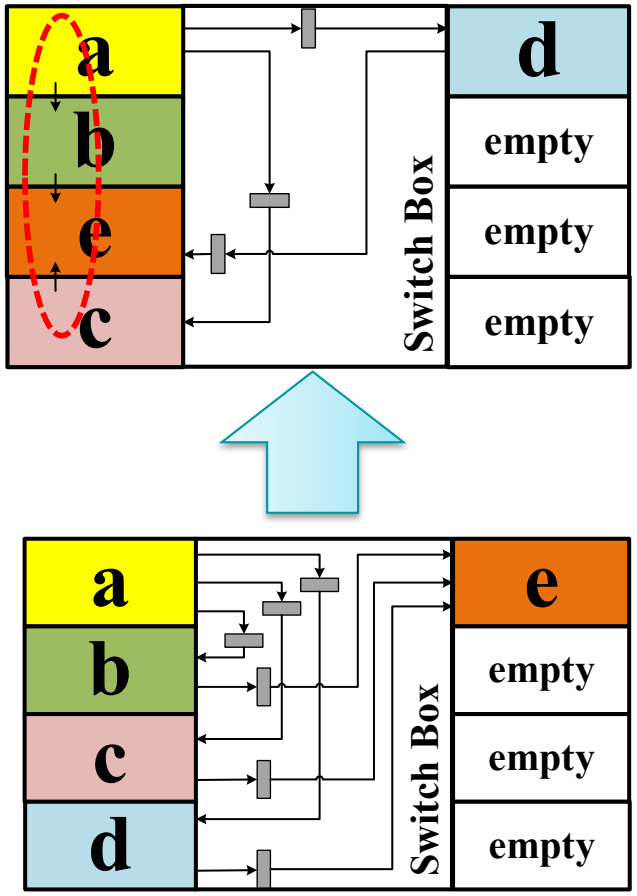


Approaches: A further step

We have more Sides!

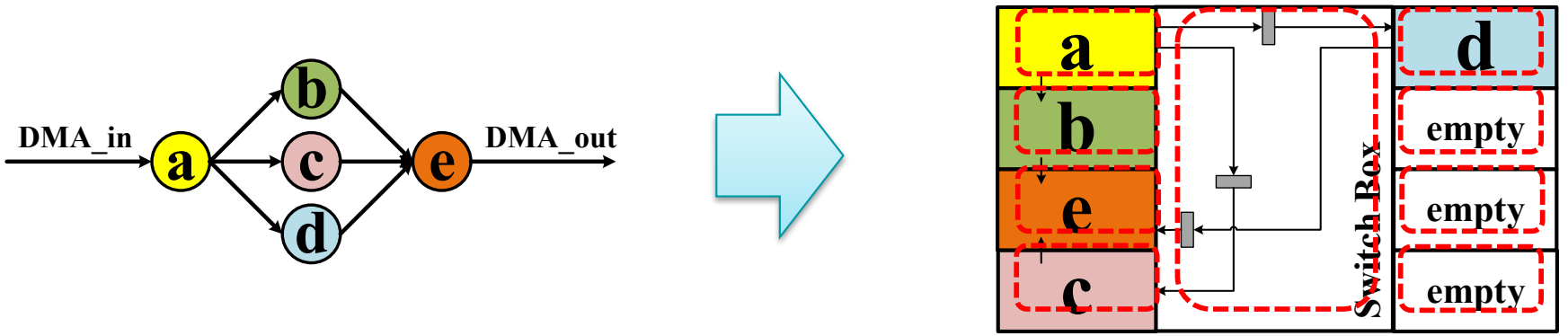
- Pre-route wires between user pages
- Proper placement to use locality.





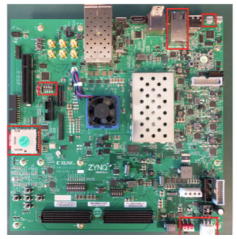
Approach

- Point-to-point Wires (Direct Wires)
  - FPGA has more wires than we have already used for PSNoC
- Compile Separate FPGA Blocks with Divide-and-Conquer
  - Define switch box as reconfigurable
  - Fast Compilation



Evaluation

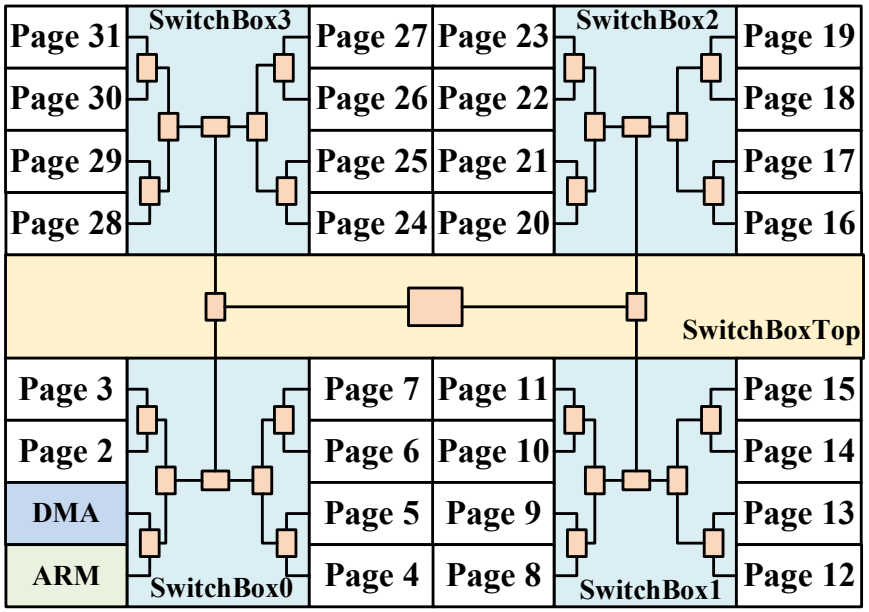
- Compile servers: Google Cloud
  - Each compute node with 4 dual-thread, 2.8GHz Intel Xeon Cascade Lake Processors
- Platform for PRflow on Vivado 2018.2
  - **Xilinx ZCU102 board** with xczu9eg-ffvb1155-2-e MP-SOC chip
  - 274K LUTs, 912 BRAM36, 2520 DSPs
  - 775 MHz clock for Fabric
- Rosetta HLS Benchmark [1]
  - 6 C-based design for High Level Synthesis Benchmark
  - 3-D Rendering, Digit-Recognition, Spam-filter, Optical-flow, BNN, Face-detection
  - We partitioned the benchmarks into small pieces, details in the paper



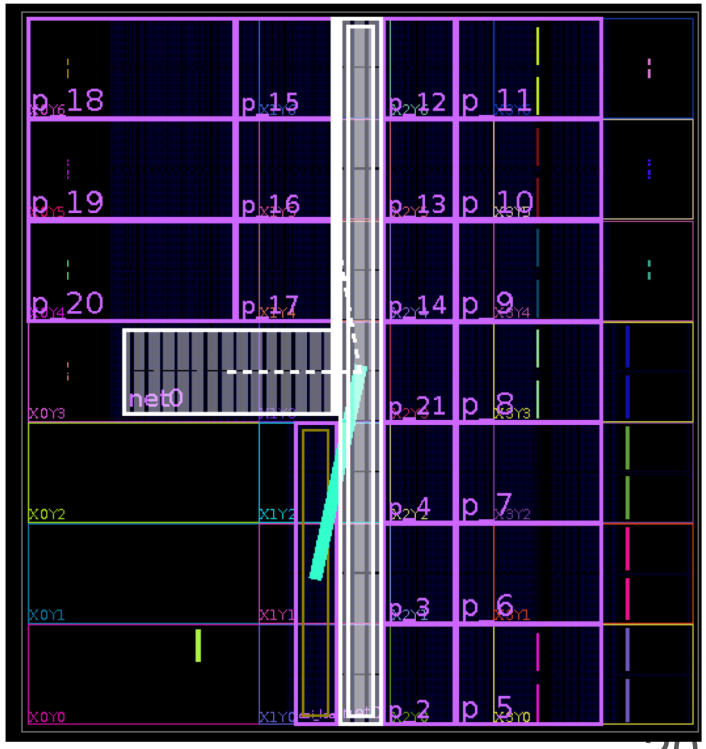
[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18

Evaluation: PSNoC Layout

- 20 leaves for application logic
- 1 leaf for 4-core ARM processor
- 1 leaf for DMA interface
- 20 leaves are connected by BFT

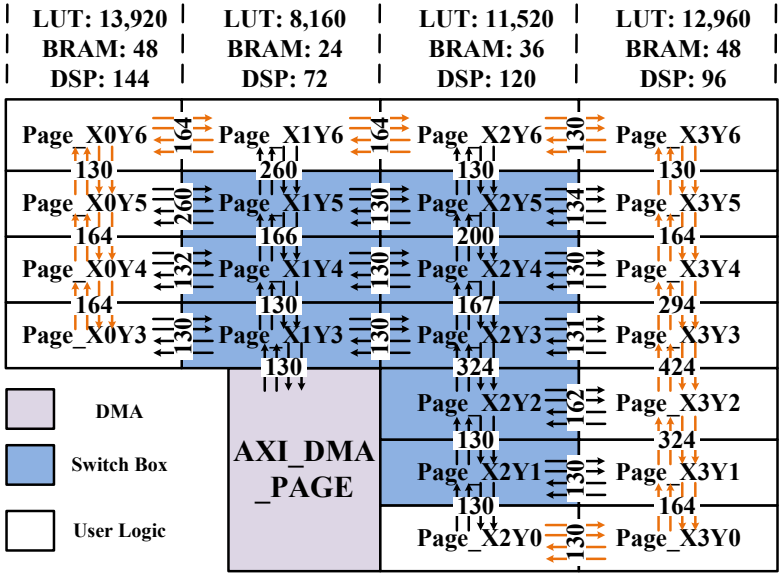


Physical Layout

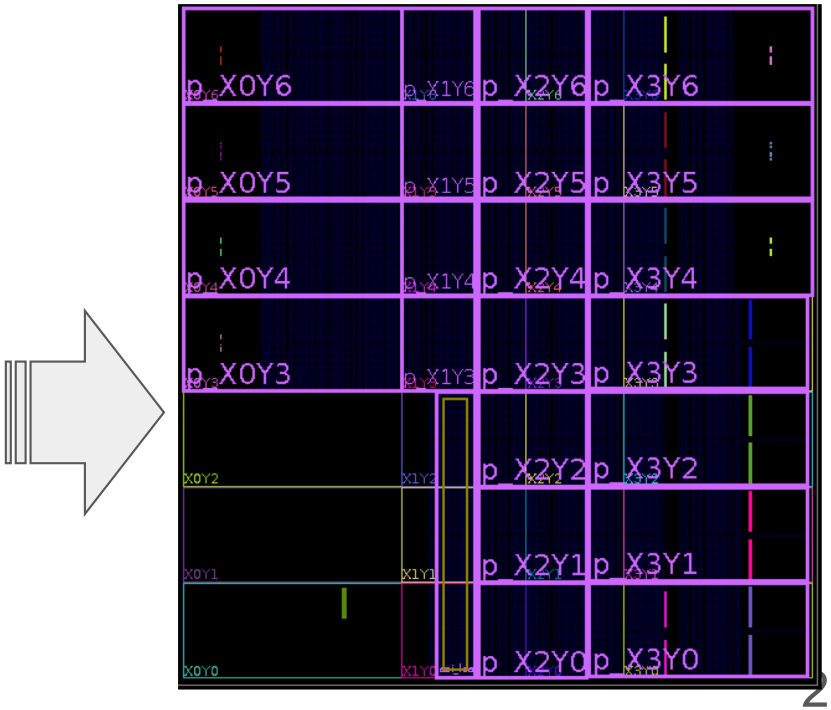


Evaluation: DW Layout

- 14 user pages, 8 switch pages
- User logic can also borrow resource from switch pages
- 1 page for 4-core ARM processor
- 1 page for DMA interface
- Wires numbers vary from 130 to 424

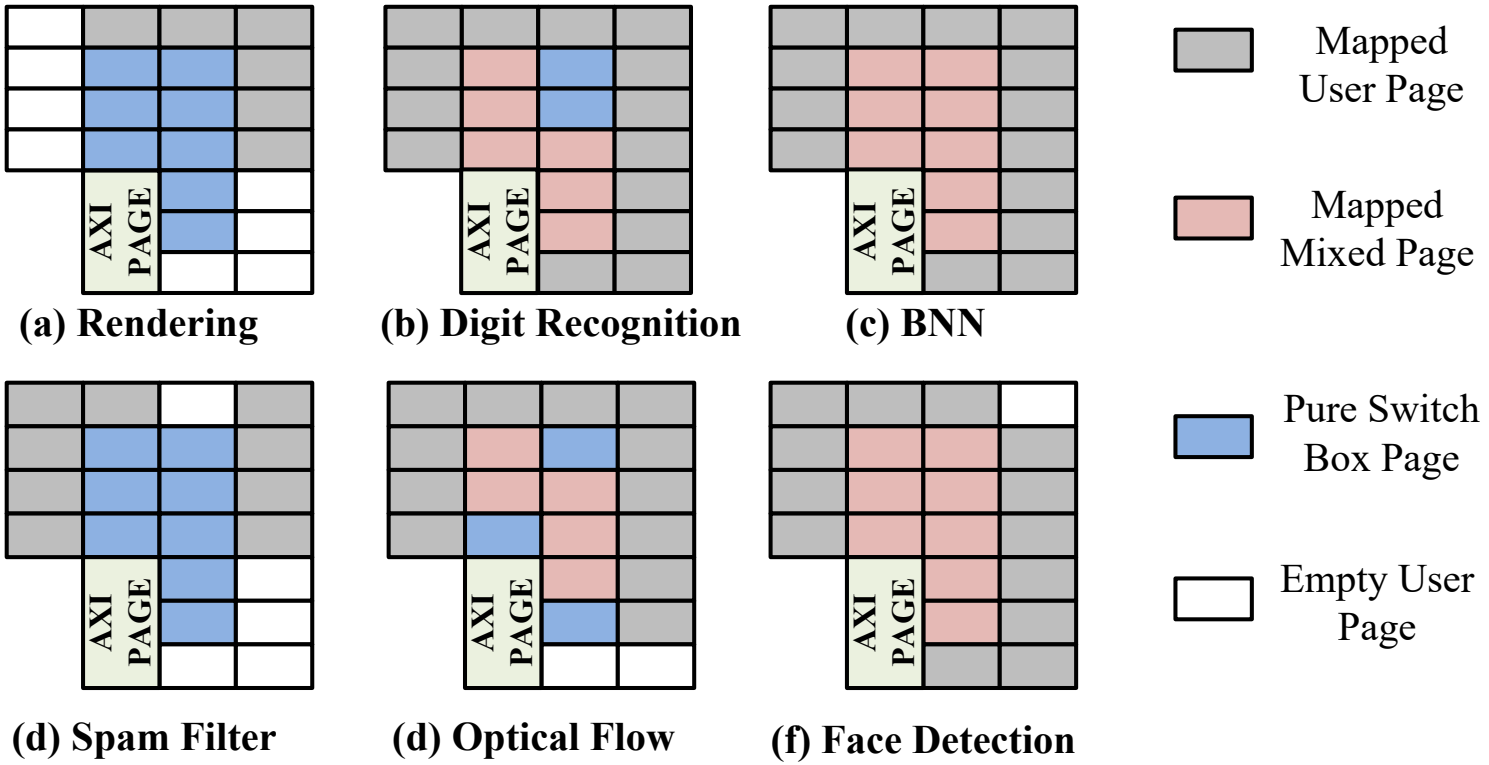


Physical Layout



Evaluation: Benchmark Mapping

- Mapping strategy for all the benchmarks



Evaluation: Compile Time

Unit: second

Benchmark	SDSoC [1]	Mono (no BFT)	PS [2]	DW (our work)	
				Logic	SWB
Rendering	1711	1495	606	737	603
Digit Recognition	2569	2104	610	735	0
Spam Filter	1930	1780	568	695	593
Optical Flow	2997	2792	679	886	685
Binarized Neural Network	12001	11089	1004	1082	611
Face Detection	4136	2981	825	1089	606

- Our user logic compile time is similar to PSNoC[2] Compile time
- The switch box does not increase the compile time
- Digit Recognition has systolic array computing graph, so it does not need to use switch box.

[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Evaluation: Performance Improvements

Unit: ms/input

Benchmark	SDSoC [1]	Mono (no BFT)	PSNoC [2]	DW (our work)
Rendering	1.5	1.4	1.2	1.3 (0.9X)
Digit Recognition	6.9	5.0	10.8	5.4 (2.0X)
Spam Filter	28.2	22.4	48.9	32.2 (1.5X)
Optical Flow	3.5	2.1	25.8	2.6 (9.9X)
Face Detection	18.2	24.3	101.0	33.1 (3.1X)
BNN	5.3	3.6	17.4	22.4 (0.8X)
BNN_new	5.3	3.6	17.4	5.7 (3.0)

- We refactor the code for PSNoC, and improve the performance of Digit Recognition, and Face detection
- Compared with **the improved PSNoC**, DW can improve the performance by **1.5x to 9.9x**
- BNN
  - User can run @250MHz for PSNoC
  - User can only run @187MHz for DW
- **BNN new** (After this paper)
  - Add 2 pipeline stages
  - Increase the Compile Time from 1082s to 1458s

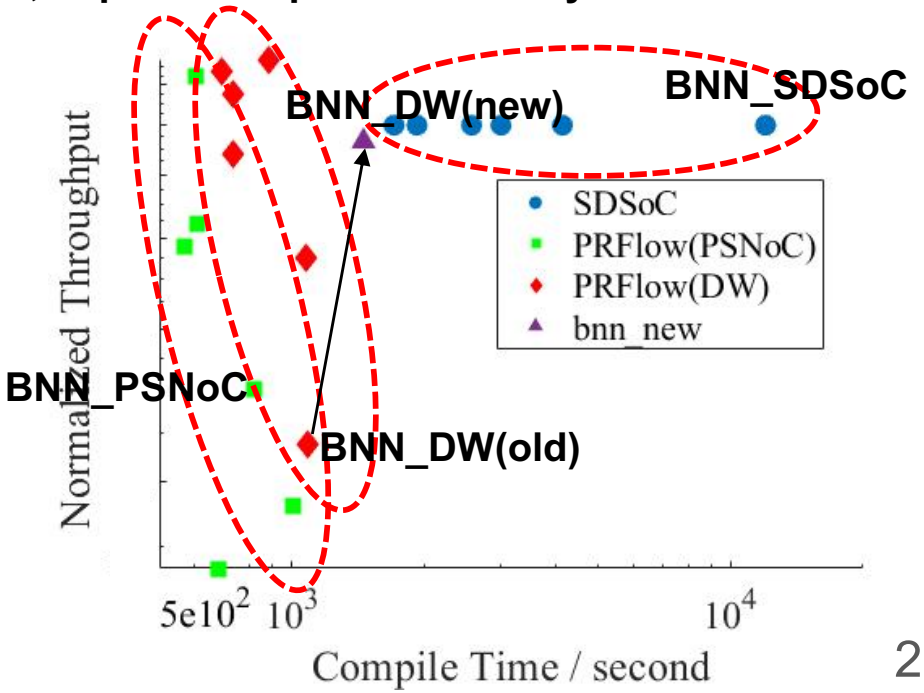
[1] Yuan Zhou et al. Rosetta: *A Realistic High-Level Synthesis Benchmark Suite for Software-Programmable FPGAs*. ISFPGA'18.
[2] Yuanlong Xiao et al. *Reducing FPGA compile time with separate compilation for FPGA building blocks*. ICFPT'19.

Evaluation: Throughput vs. Compile Time

- Today's FPGA compilation is slow: **30-178 minutes**
- PRFlow[2] can reduce compile time to **10-16 minutes, but degrade the performance at most 12.5 times**
- DW can keep compile time to **12-18 minutes, improve the performance by at most 10 times**

We propose:

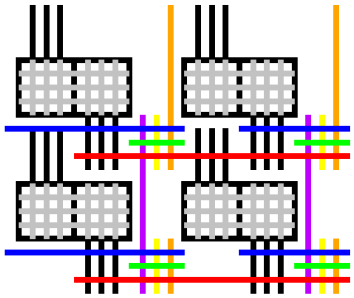
- Keep Separate Compilations
- Exploit More FPGA bandwidth



Future work:

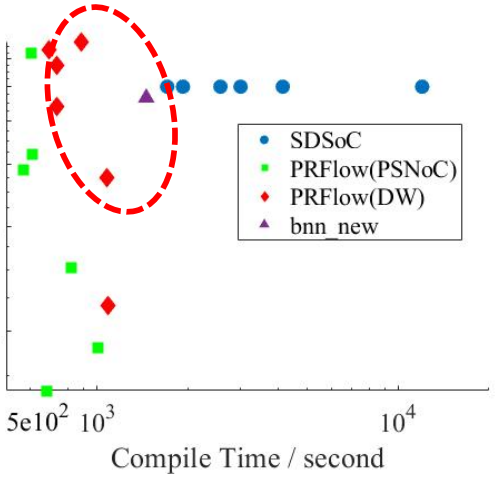
- User Page and switch box page assignment automation
- PSNoC and DW hybrid overlay for scalability

Penn



Conclusion

- Improve the Application performance by at most 10X
- Able to keep the compile time whin **18 minutes**
- Decrease the interface area by 47-86%.



Thank you

Q&A