

# Case for Fast FPGA Compilation using Partial Reconfiguration

Dongjoon(DJ) Park, Yuanlong Xiao, Nevo Magnezi, and André DeHon

Implementation of Computation Group

University of Pennsylvania



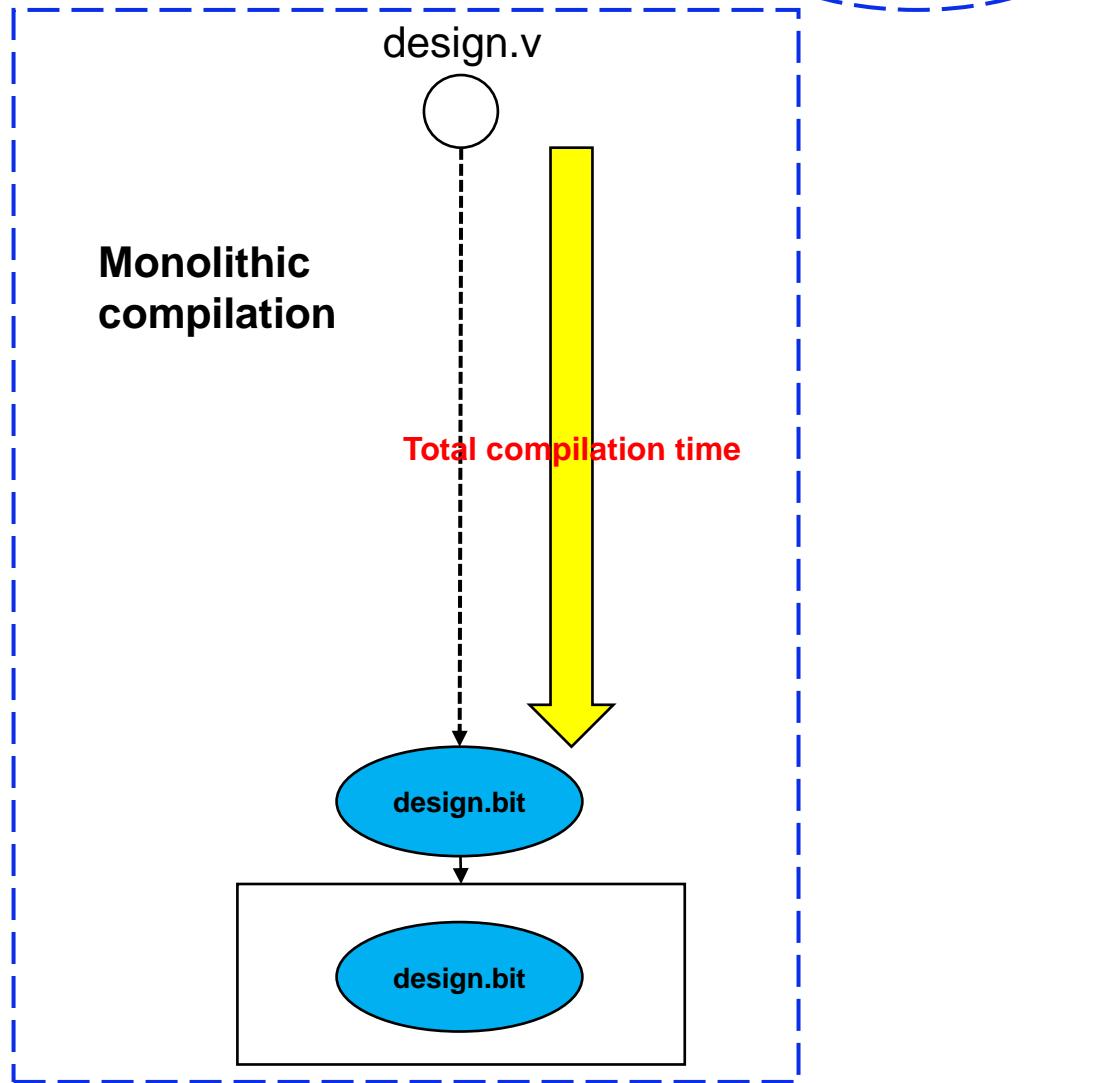
# Introduction

- **Problem:** FPGA's long compilation
- **Methodology:** Divide-and-Conquer
  - Compile design blocks **in parallel**
  - **Partial Reconfiguration (PR)** for separate compilations
  - Connect design blocks through an **overlay network**

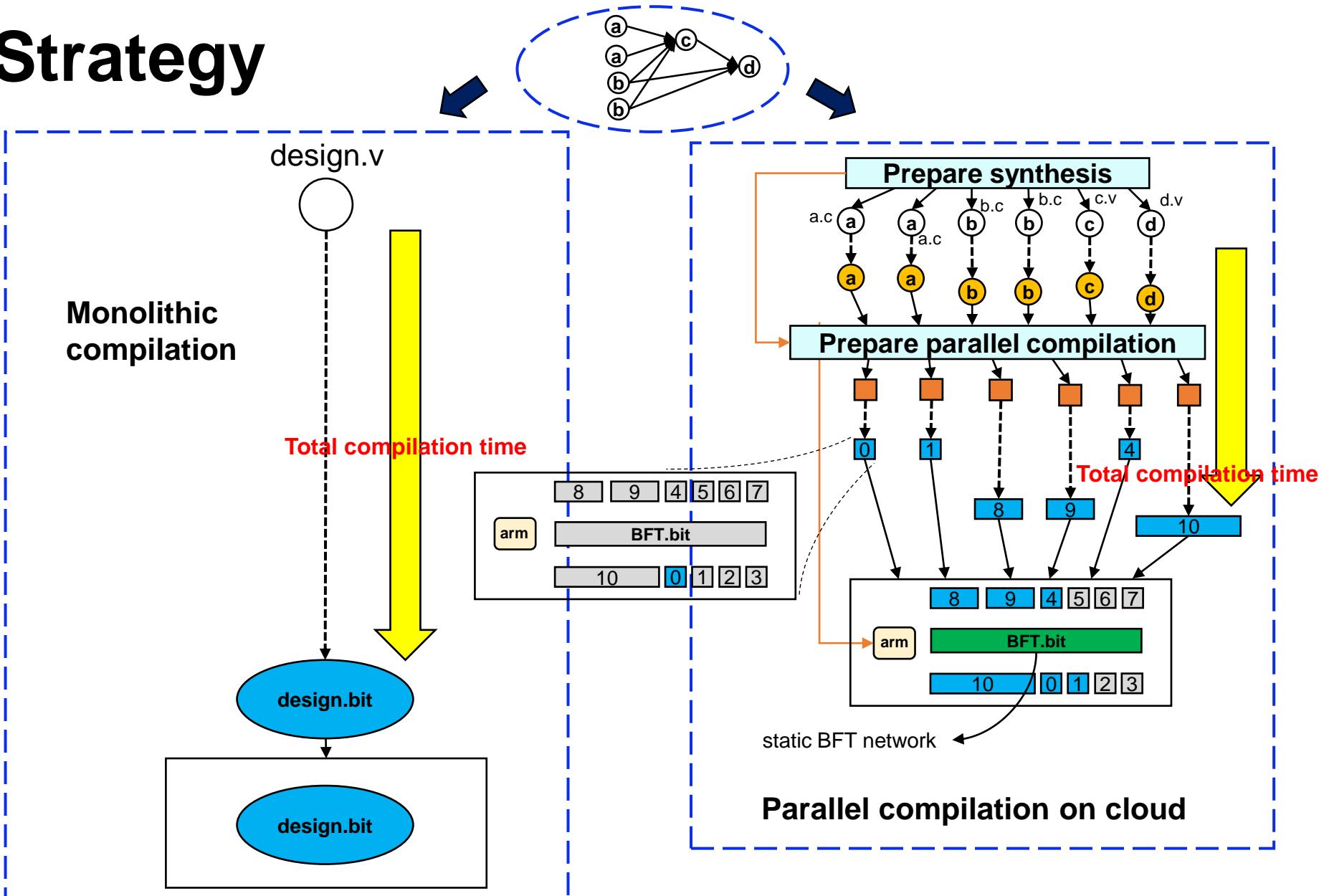
# Overlay Network

- Overlay network:  
**Packet-switched Butterfly Fat Tree (BFT)**  
network as a static design of PR
  - Support arbitrary connectivity among separately-compiled components
  - Fixed and pre-computed
    - not contribute to user-design mapping time
  - Packet switched
    - no need to configure overlay network

# Strategy

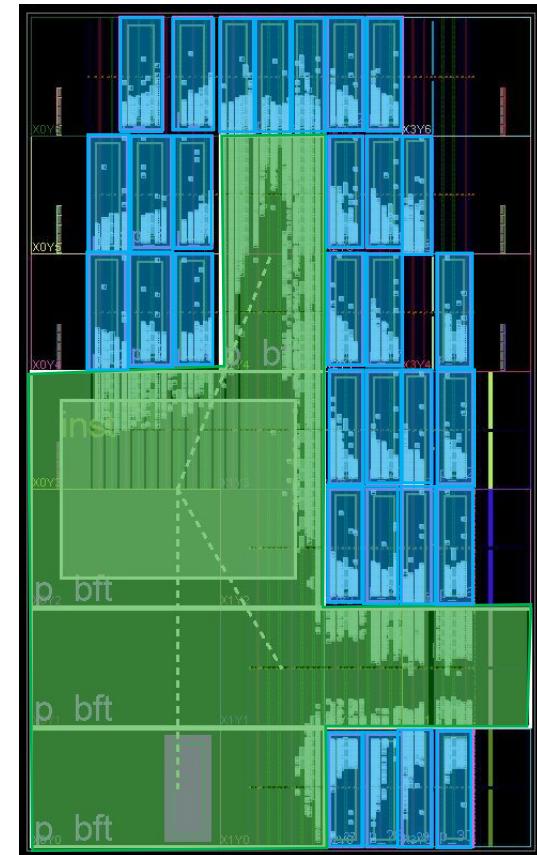
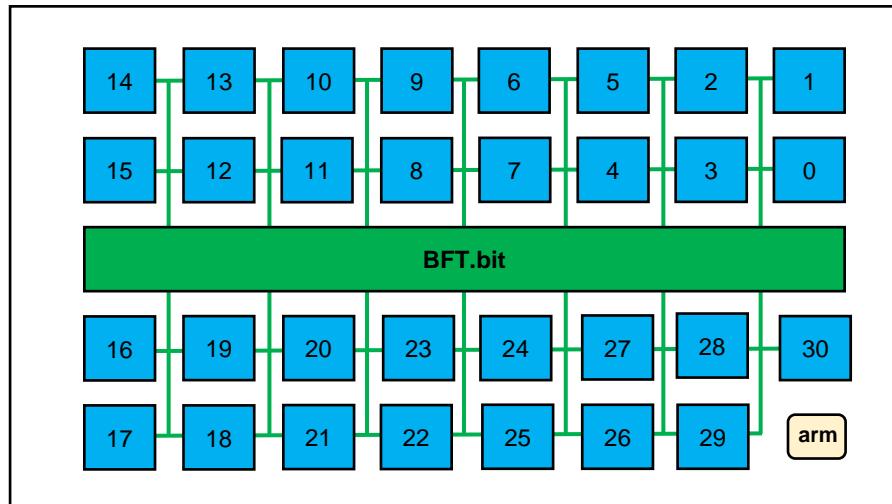


# Strategy



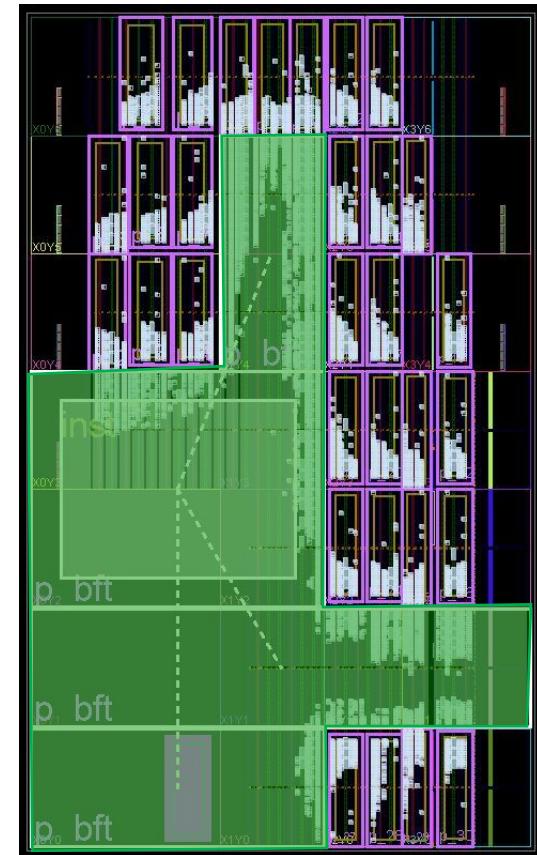
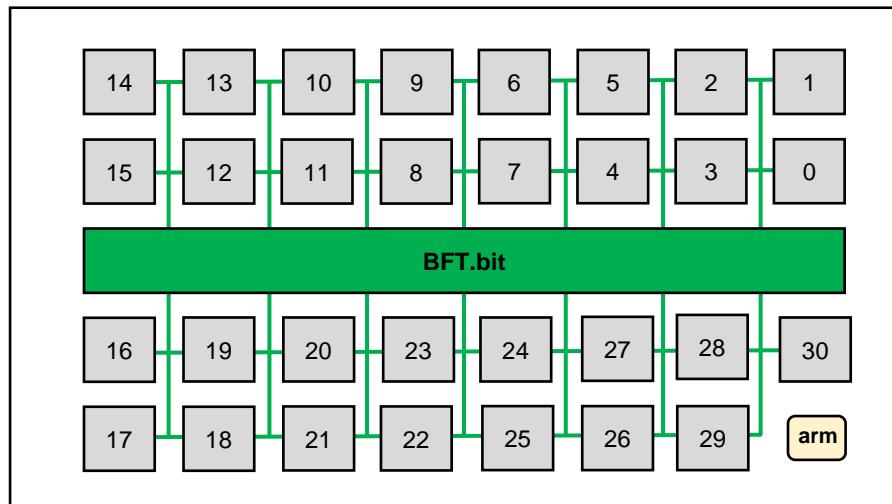
# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)



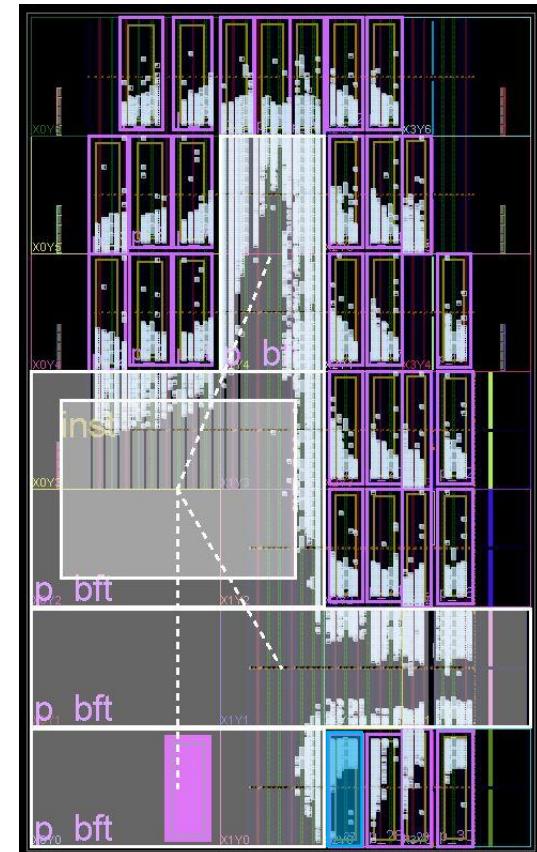
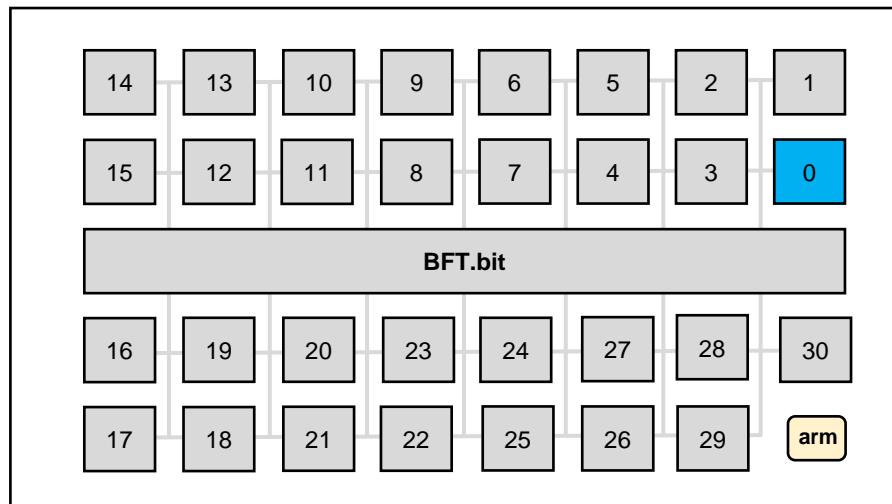
# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)



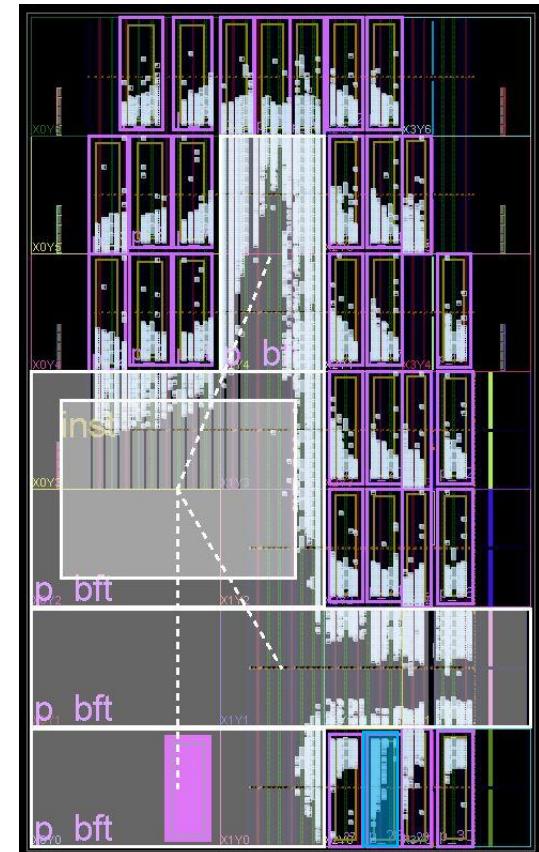
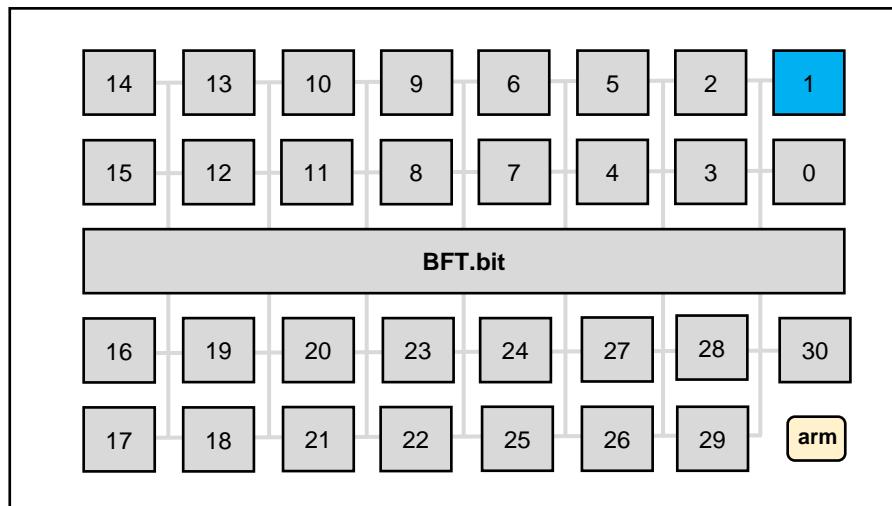
# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)



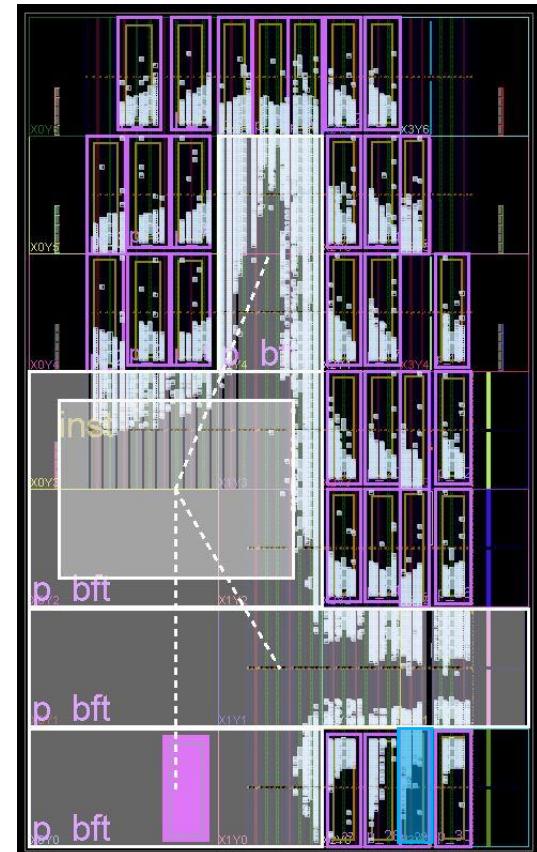
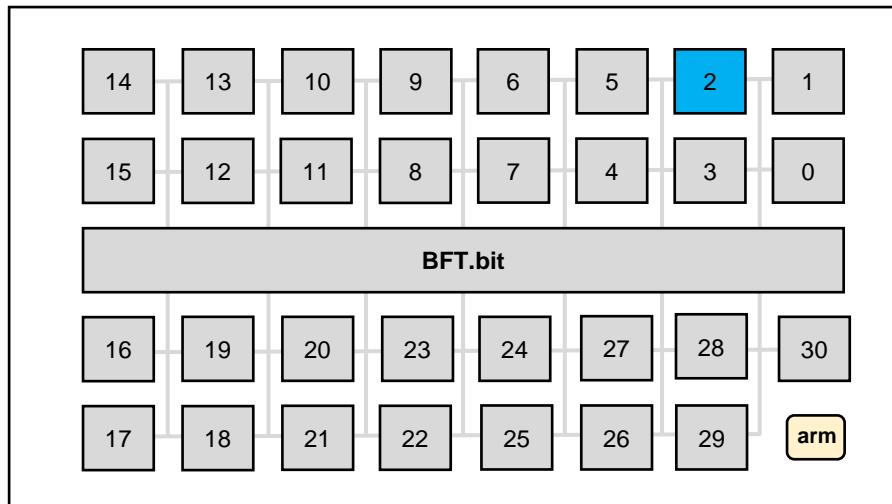
# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)



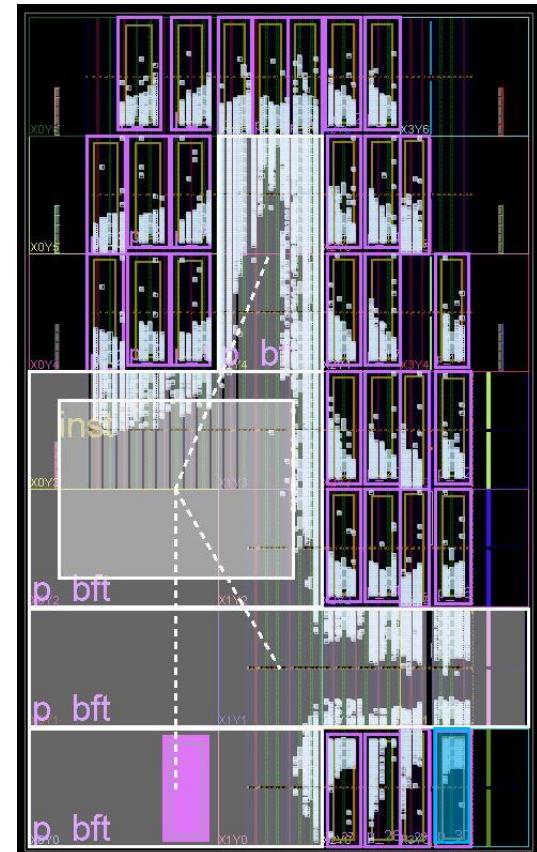
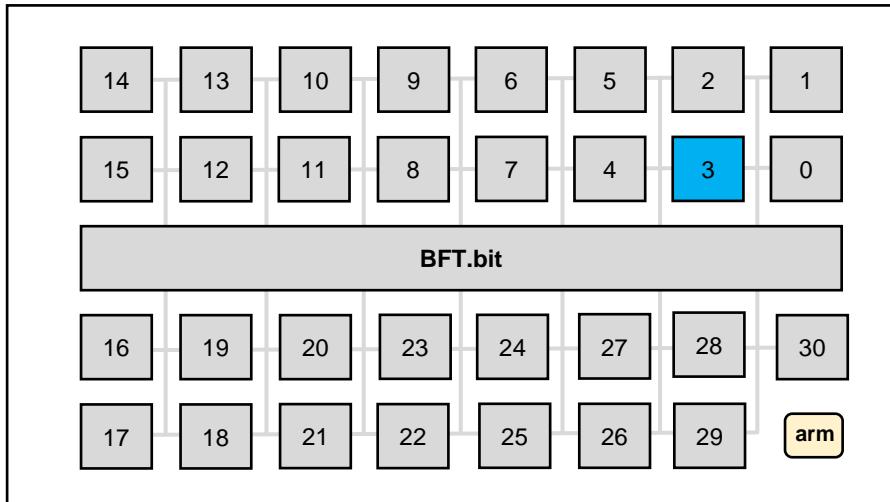
# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)

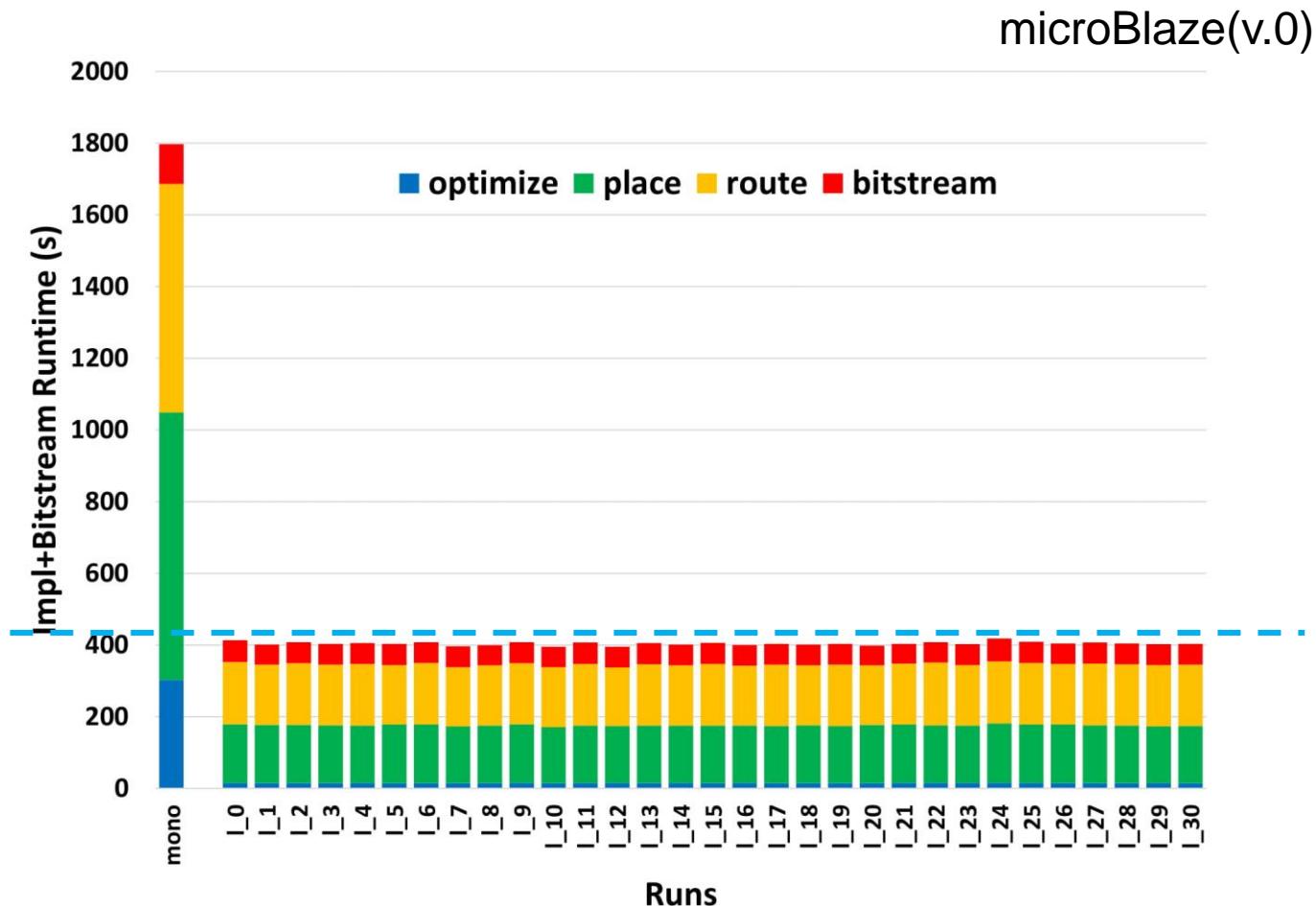


# Case study: multi-core design

- An array of soft-cores (MicroBlaze processors)



# Case study: multi-core design



# Case study: multi-core design

- Result

Runtime(secs)	microBlaze(v.0)		microBlaze(v.1)		microBlaze(v.2)	
	Mono	Parallel	Mono	Parallel	Mono	Parallel
Synthesis	3171	287	3118	283	2510	235
Impl+bitstream	<b>1797</b>	<b>418</b>	<b>1692</b>	<b>413</b>	<b>1283</b>	<b>398</b>

The diagram illustrates the performance gains across three versions of the microBlaze design. For each row, a curved arrow points from the v.0 value to the v.2 value, indicating the reduction in runtime. The labels below the arrows provide the exact倍速 (x) factor for each category.

- Synthesis:** The runtime goes from 3171 seconds (v.0) to 235 seconds (v.2), a reduction of  $4.30x$ .
- Impl+bitstream:** The runtime goes from 1797 seconds (v.0) to 398 seconds (v.2), a reduction of  $4.10x$ .
- Total:** The overall runtime goes from 1692 seconds (v.0) to 1283 seconds (v.2), a reduction of  $3.22x$ .

# Conclusion

- Showed **4x compile time speedup** using the current tool's existing facilities
- Work-in-progress
  - Tool(Xilinx Vivado) challenges
  - More automation in the flow and more benchmarks
  - Optimization in the overlay and PR architecture