Canal: A Flexible Interconnect Generator for Coarse-Grained Reconfigurable Arrays

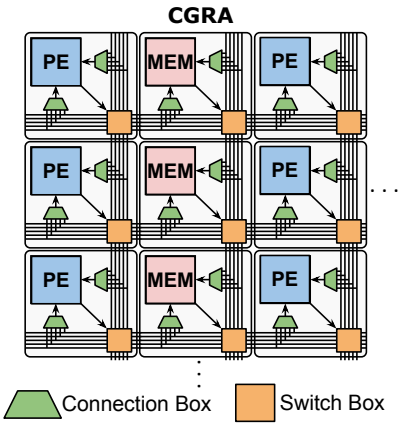
Jackson Melchert*, **Keyi Zhang***, Yuchen Mei, Mark Horowitz, Christopher Torng, Priyanka Raina

Stanford University

October 12, 2022

Challenges with evolving a reconfigurable hardware accelerator

- Configurable IP cores
  - Designed using different frameworks
  - Independent changes
  - Agile
- Register space allocation and bitstream generation
- Design space exploration



Interconnect-centric solution

- These challenges are system integration problems
- Interconnect integrates tiles together
- Introspection and automatic adaptation

Interconnect-centric solution

- These challenges are system integration problems
- Interconnect integrates tiles together
- Introspection and automatic adaptation

- Most prior work uses rigid architecture description files [1, 2, 3]
- The IP cores are frozen
- Agile?

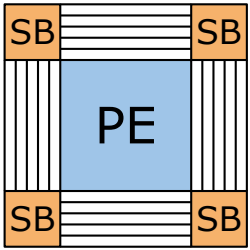
Introducing Canal: More than an interconnect generator

Canal is a Python-embedded domain-specific language and a system integration oriented CGRA interconnect generator:

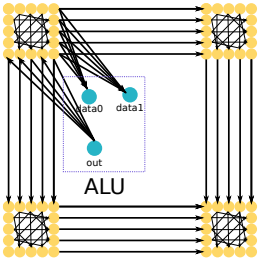
- A graph-based intermediate representation (IR) for CGRA interconnects
- Hardware "compiler" backends
- A configuration management tool that takes the IR and produces a bitstream
- A design-space exploration tool

Graph-based IR for CGRA interconnect

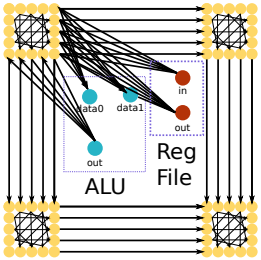
Canal IR models connectivity as a general and simple graph using nodes and edges, instead of wires and modules



(a) Traditional view of switchboxes with a PE core



(b) Canal IR representation of a PE with one ALU core



(c) Canal IR representation of a PE with ALU and reg file

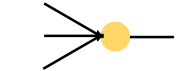
How core introspection works

All cores need to implement the following interface primitives

```
def inputs() -> List[Port]

def outputs() -> List[Port]

def add_config_reg(name: str, width: int) -> Reg

def get_core_configuration(instr) -> List[Reg, Value]
```

Adding information Canal IR primitives

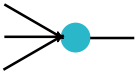
- Each node can have different types, attributes, and other metadata
- Edges can also contain metadata, such as wire delay, which is used for PnR
- Edges are ordered



SwitchBoxNode

Node

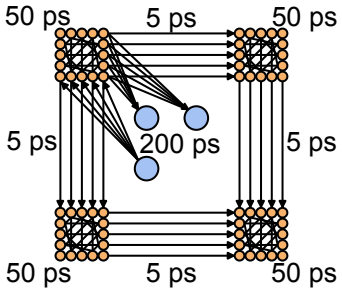
RegisterNode



PortNode

Adding information Canal IR primitives

- Each node can have different types, attributes, and other metadata
- Edges can also contain metadata, such as wire delay, which is used for PnR
- Edges are ordered



Frontend language design: IR construction

Example

```
node = Node(x = 1, y = 1, side = SwitchBoxSide.South, track=1)
for port_node in tile.pe.inputs():
    # automatic introspection
    node.add_edge(port)
```

Higher-level construction functions:

Example

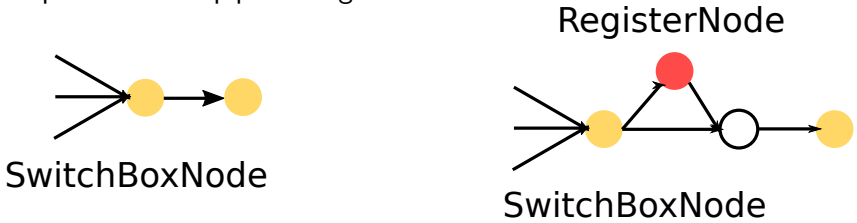
```
def create_uniform_interconnect(width: int, height: int,
                                track_width: int,
                                track_info: Dict[int, int],
                                sb_type: SwitchBoxType, ...)
```

Benefits of using an IR for Canal: transformation

A compiler pass can transform the interconnect connection.

Example

A compiler pass to insert pipeline registers into the switchbox



Pipeline register insertion by transforming the edge into a group of nodes and connections

Benefits of using an IR for Canal: different backend toolchains

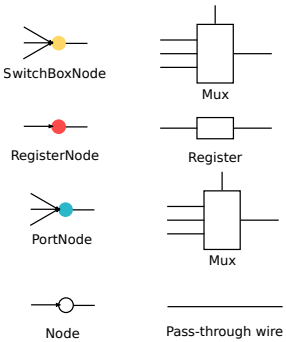
The IR offers different views to different tools:

- Hardware "compiler" backends
- PnR tools
- Bitstream generator
- Functional model

How to lower the IR into hardware

Static interconnect backend:

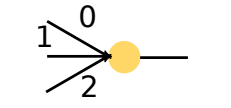
- Edges are lowered to wires and nodes are lowered to corresponding hardware modules
- Multiplexers can be inferred automatically because in Canal IR, all the connections are ordered.

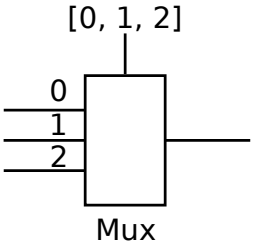


How to lower the IR into hardware

Static interconnect backend:

- Edges are lowered to wires and nodes are lowered to corresponding hardware modules
- Multiplexers can be inferred automatically because in Canal IR, all the connections are ordered.

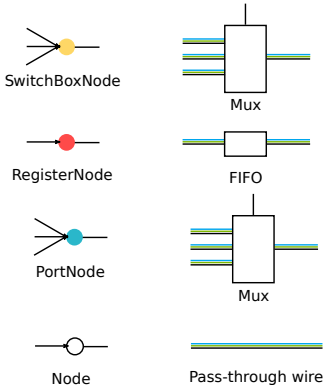




How to lower the IR into hardware

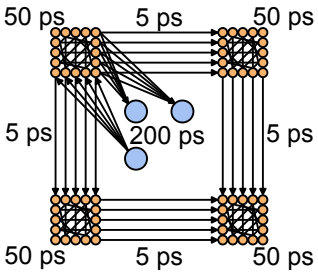
Ready-valid interconnect backend:

- Edges are lowered into three different nets - data, ready, and valid
- All hardware modules need to handle ready/valid. We can optimize the hardware independently of the IR
- Some optimization can be done on the FIFO to reduce FIFO area.



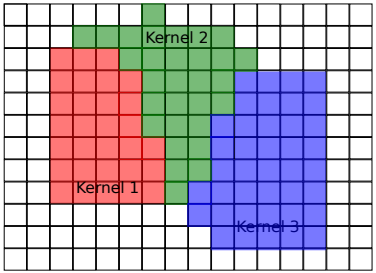
A PnR system built for Canal

- The IR can be lowered to PnR collateral easily with timing attributes
- Multi-stage placement and routing
- Bitstream generation is straightforward due to ordered edges



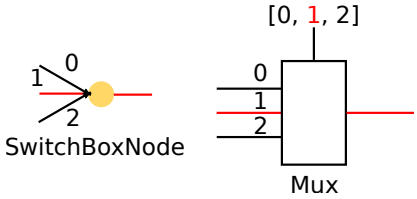
A PnR system built for Canal

- The IR can be lowered to PnR collateral easily with timing attributes
- Multi-stage placement and routing
- Bitstream generation is straightforward due to ordered edges



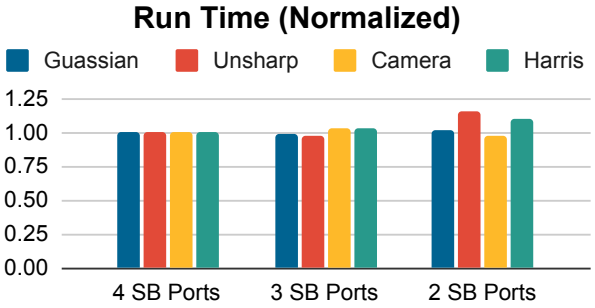
A PnR system built for Canal

- The IR can be lowered to PnR collateral easily with timing attributes
- Multi-stage placement and routing
- Bitstream generation is straightforward due to ordered edges



Design space exploration

Streamlined physical design flow to explore different micro architecture decisions.



Conclusions

- Canal is an interconnect generator for system integration
- It has an expressible eDSL language for integration and topology construction
- Using an IR can significantly improve the flexibility and allows a diverse set of tools to work together (single source of truth).
  - RTL generator
  - PnR tools
  - Bitstream generator
- Open-source tools from Stanford AHA! center.

References

[1] V. Betz and J. Rose, "Vpr: A new packing, placement and routing tool for fpga research," in *International Workshop on Field Programmable Logic and Applications*. Springer, 1997.

[2] S. A. Chin, N. Sakamoto, A. Rui, J. Zhao, J. H. Kim, Y. Hara-Azumi, and J. Anderson, "Cgra-me: A unified framework for cgra modelling and exploration," in *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2017.

[3] S. Zheng, K. Zhang, Y. Tian, W. Yin, L. Wang, and X. Zhou, "Fastcgra: A modeling, evaluation, and exploration platform for large-scale coarse-grained reconfigurable arrays," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, 2021.