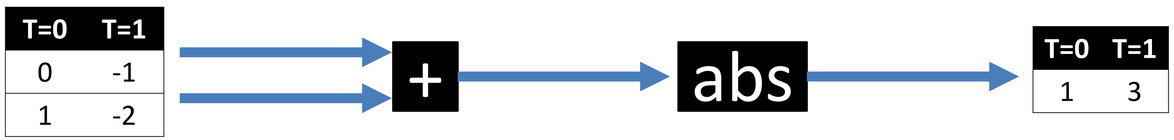
Aetherling: Resource-Aware, Space-Time Scheduling

David Durst, David Akeley, Jeff Setter, Teguh Hofstee, James Hegarty, Ross Daly, Kayvon Fatahalian, Pat Hanrahan

1. Introduction



Add >>> Abs

An IR for compiling dataflow pipelines to FPGAs and CGRAs with:

1. A type system that ensures pipelines compile to efficient implementations in hardware
2. The ability to automatically change a pipeline's throughput

2. Throughput Types Ensure Efficiency

- Throughput mismatches between adjacent modules in a pipeline lead to inefficiencies including underutilization and extra memory buffers
- Ensure efficiency by verifying adjacent module throughputs match
- Each module's type includes throughput

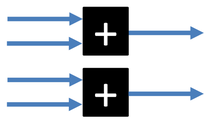
```
Add :: 1 (Int, Int) Per Clock -> 1 Int Per Clock
```

- Composition of modules in sequence verifies throughputs match

```
Add >>> Abs :: 1 (Int, Int) Per Clock -> 1 Int Per Clock
```

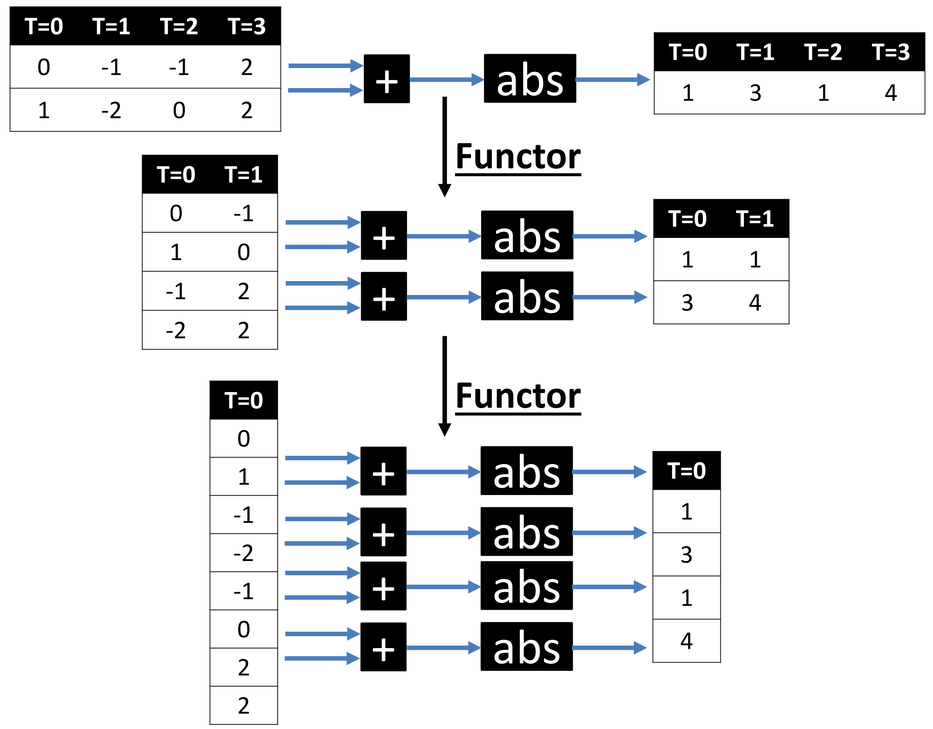
- Composition of modules in parallel derives throughput

```
Map 2 Add :: 2 (Int, Int) Per Clock -> 2 Int Per Clock
```



3. Changing Pipeline Throughput

- Types are implemented using Haskell Functors
- Functors provide tools for lifting a pipeline to a more parallel implementation
- Isomorphisms provide tools for showing equivalence between a pipeline's inputs and outputs at different throughputs



4. Future Work: Auto-Scheduler

- Model resource consumption of pipelines
- Try a pipeline with different throughputs to find the most parallel version that fits on target FPGA or CGRA

ISTC Agile