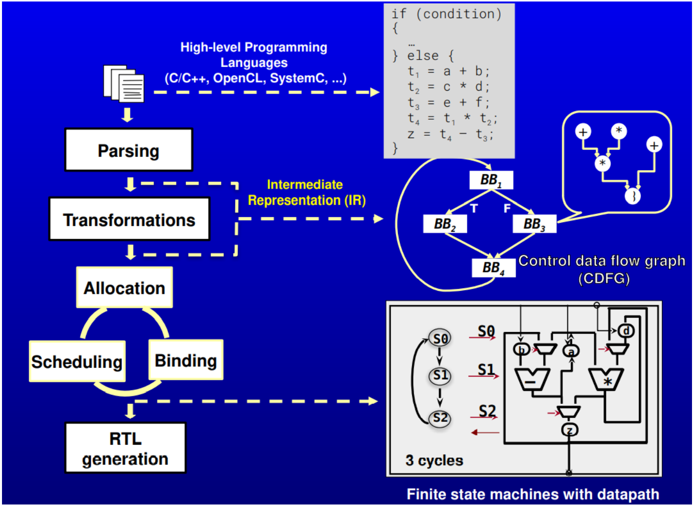
# Constraint based scheduling of Weakly Consistent C programs for Reconfigurable Hardware

Akshay Gopalakrishnan

February 24, 2022

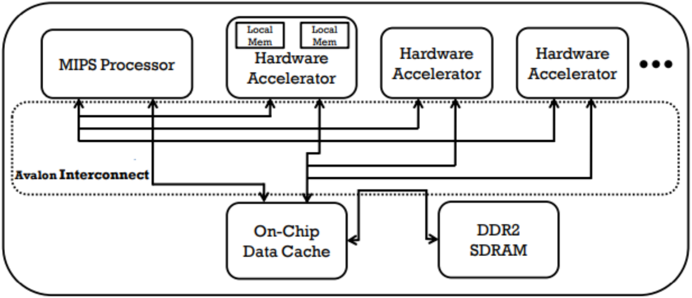
# Recap: HLS Design Flow


High-level Programming Languages (C/C++, OpenCL, SystemC, ...)
Parsing
Transformations
Intermediate Representation (IR)
Allocation
Scheduling
Binding
RTL generation
+
*
+
*
-
BB1
T
F
BB2
BB3
BB4
Control data flow graph (CDFG)
S0
S1
S2
S0
S1
S2
b
a
d
-
*
z
3 cycles
Finite state machines with datapath

```
if (condition)
{
 …
} else {
  t1 = a + b;
  t2 = c * d;
  t3 = e + f;
  t4 = t1 * t2;
  z = t4 − t3;
}
```

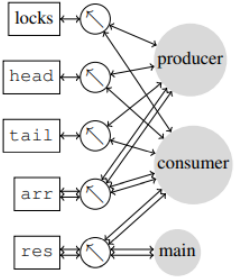
Focus on the scheduling and transformation phases.

# Mapping Threads to Reconfigurable Hardware



Each thread mapped to a unique hardware accelerator.

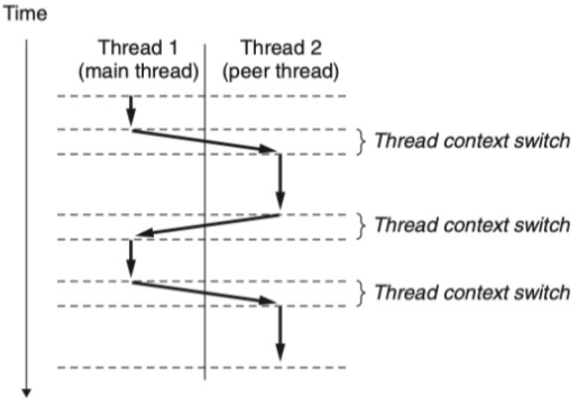
# Concrete Example: Producer Consumer System



(b) LegUp 5.1

Assumption is we have infinite supply of accelerators

# Resource Constraint: Limited Hardware Accelerator

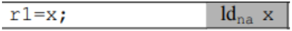


Schedule involves additional context switch cycles.

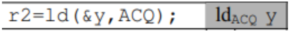
# Proposed Solution

Perform inlining at the transformation phase.
Saves context switch clock cycles.

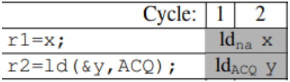
# Advantage in Weak Memory Setting



| | |
|---|---|
| `r1=x;` | $\mathrm{ld}_{\mathtt{na}}$ `x` |



| | |
|---|---|
| `r2=ld(&y,ACQ);` | $\mathrm{ld}_{\mathtt{ACQ}}$ `y` |



| Cycle: | 1 | 2 |
|---|---|---|
| `r1=x;` | $\mathrm{ld}_{\mathtt{na}}$ `x` | |
| `r2=ld(&y,ACQ);` | $\mathrm{ld}_{\mathtt{ACQ}}$ `y` | |

2 clock cycles only !

# Methodology

- Insert AST node for threads in assignment code base.
- Programs assumed to be given in the form of memory accesses ONLY.
- Encode dependencies for weak atomics from previous work.
- Add thread inlining transformation.
- Add analysis pre-inlining to identify which threads to inline.
- Compare schedules.

Testbench will be mainly custom-made examples in addition to Message Passing and Producer Consumer algorithms from previous work.

# Thank you!

Some paper references.

- Pthreads to Hardware
- Relaxed Memory C Programs to Hardware
- Global Analysis for Efficient Scheduling of Concurrent C programs for Hardware
- Scheduling Problem

Any feedback?