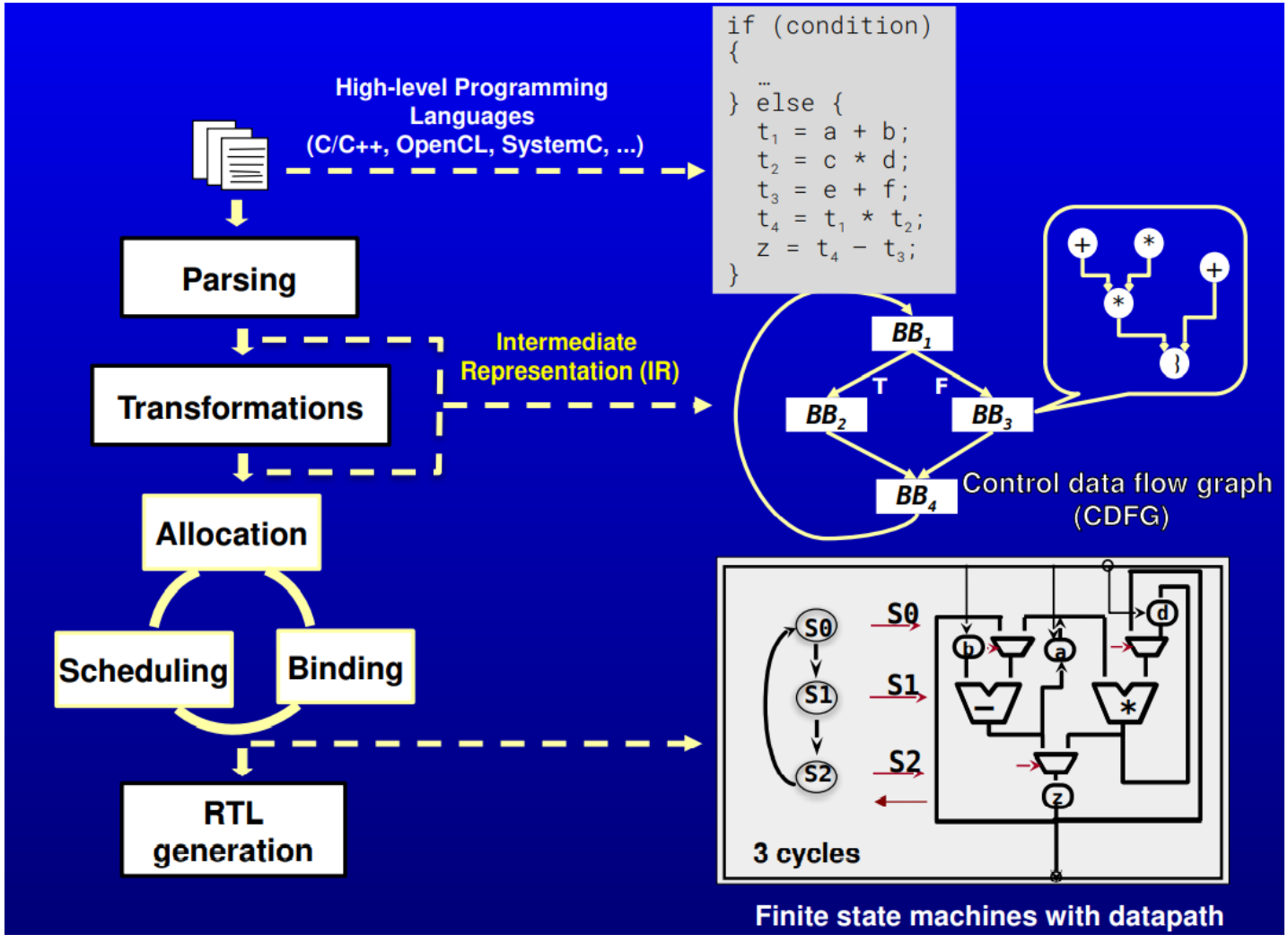
# Constraint based scheduling of Weakly Consistent C programs for Reconfigurable Hardware

Akshay Gopalakrishnan

February 24, 2022

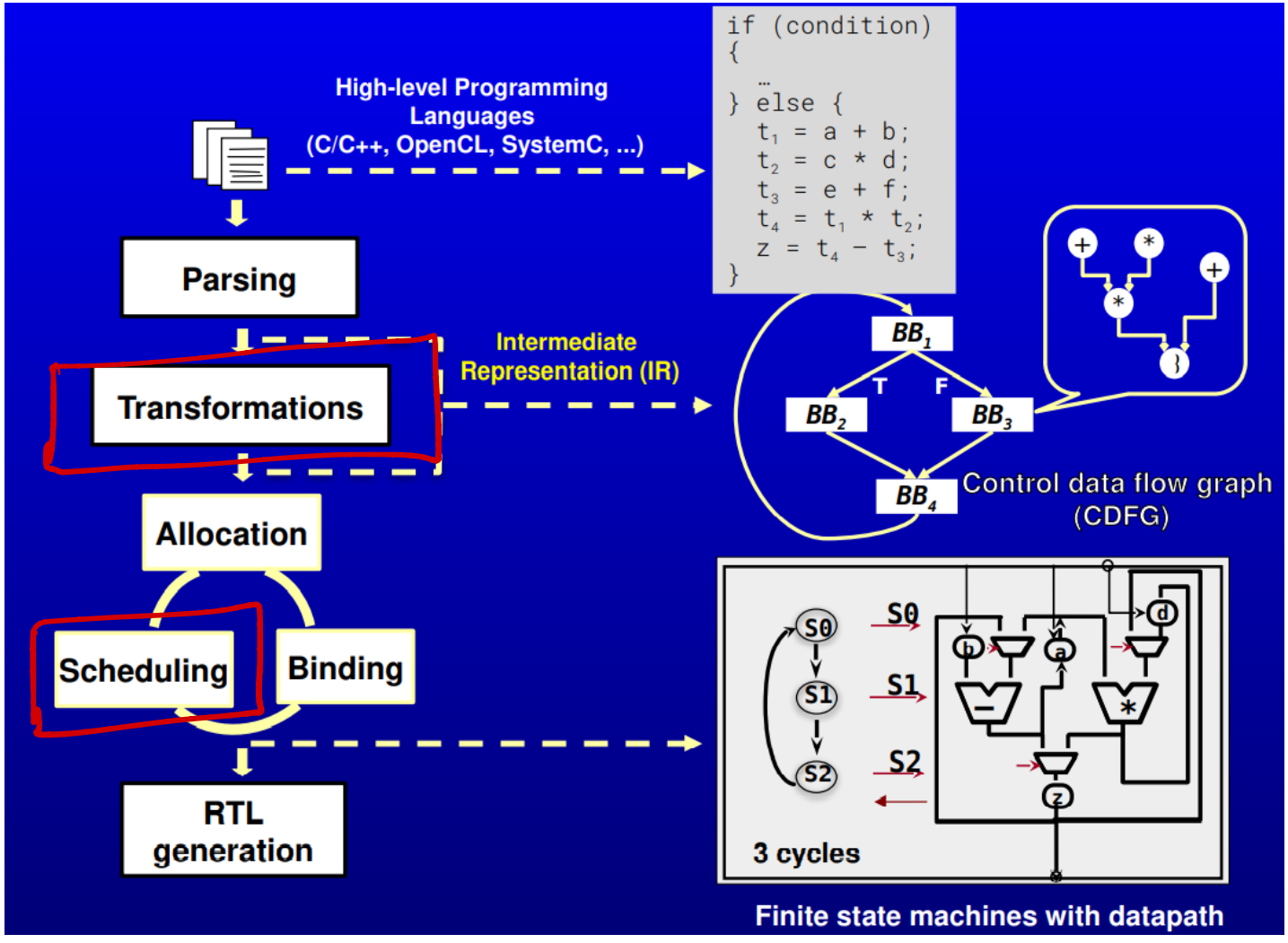
# Recap: HLS Design Flow





# Recap: HLS Design Flow

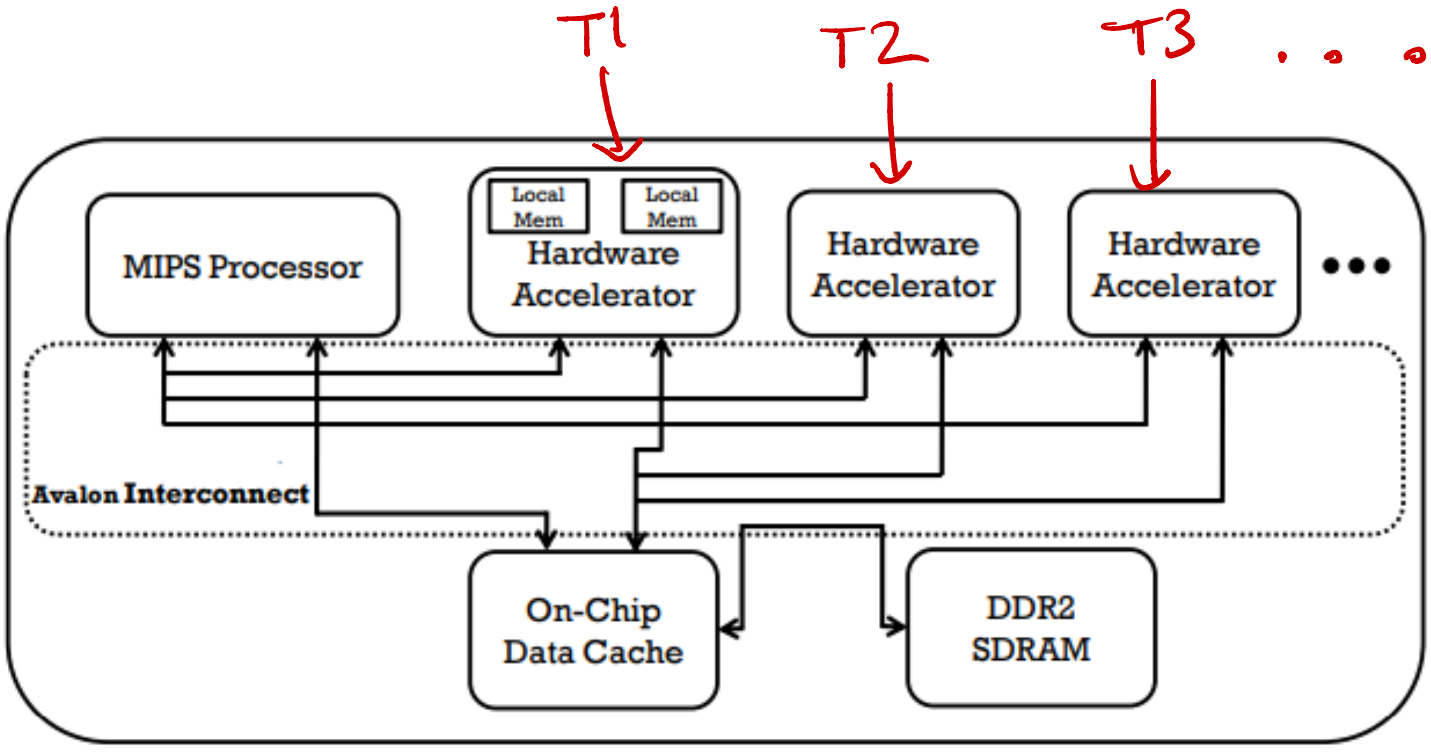





Focus on the scheduling and transformation phases.

# Mapping Threads to Reconfigurable Hardware

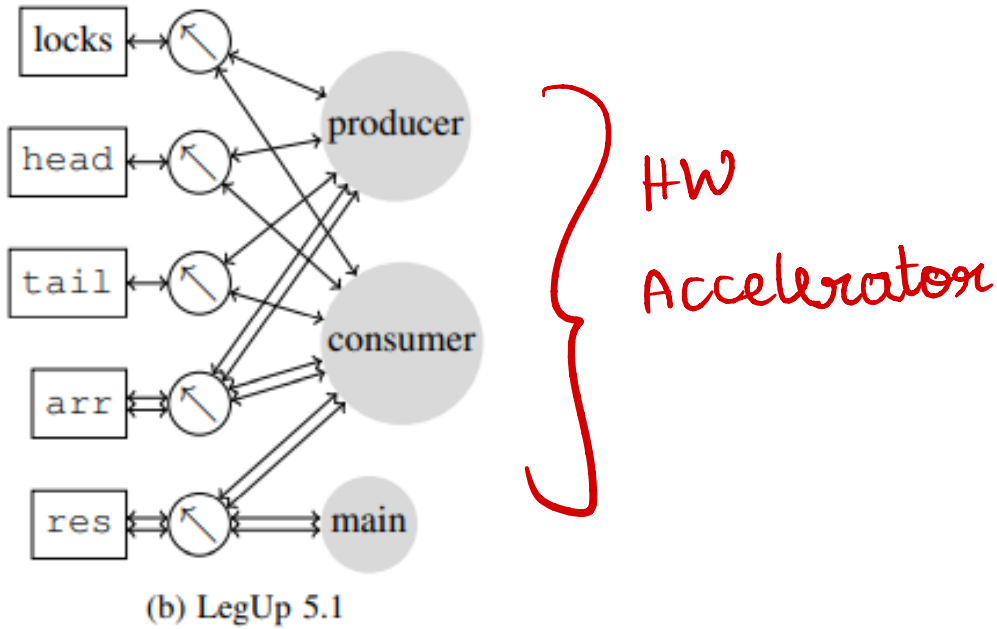





Each thread mapped to a unique hardware accelerator.

# Concrete Example: Producer Consumer System



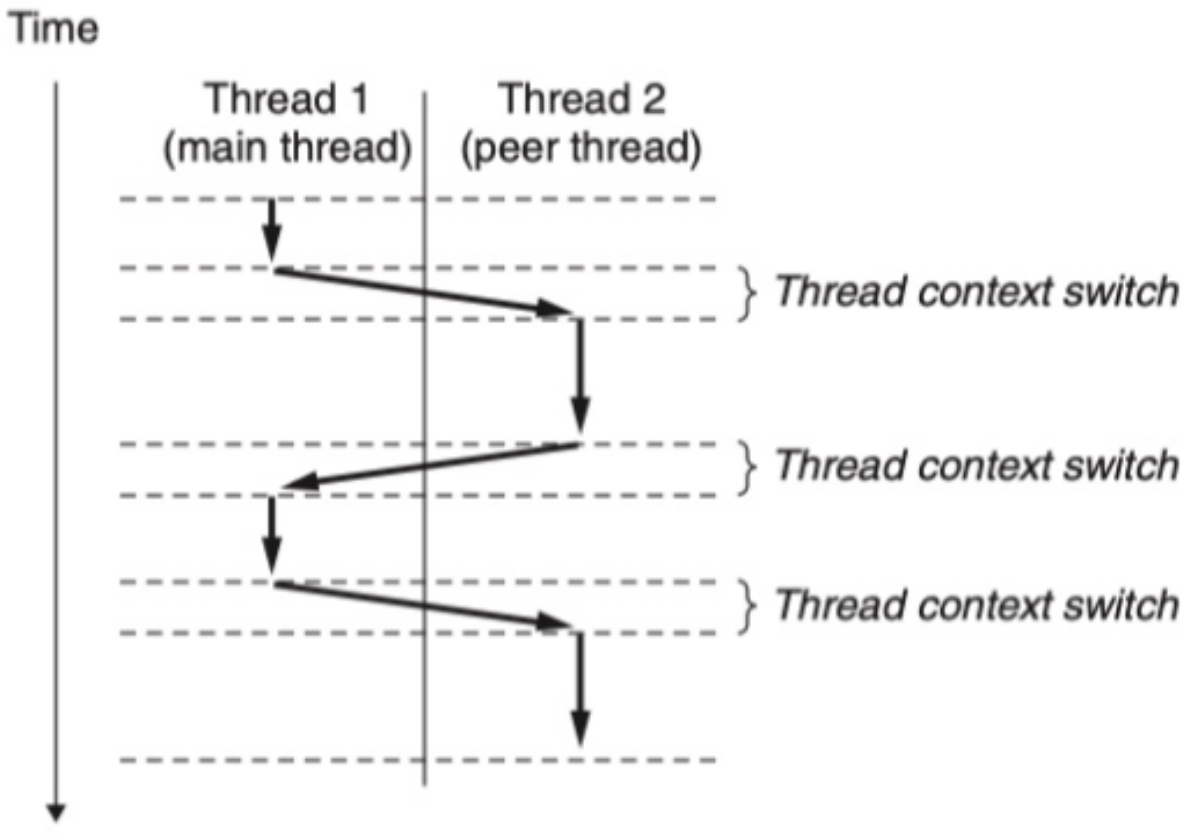



(b) LegUp 5.1

Assumption is we have infinite supply of accelerators

# Resource Constraint: Limited Hardware Accelerator

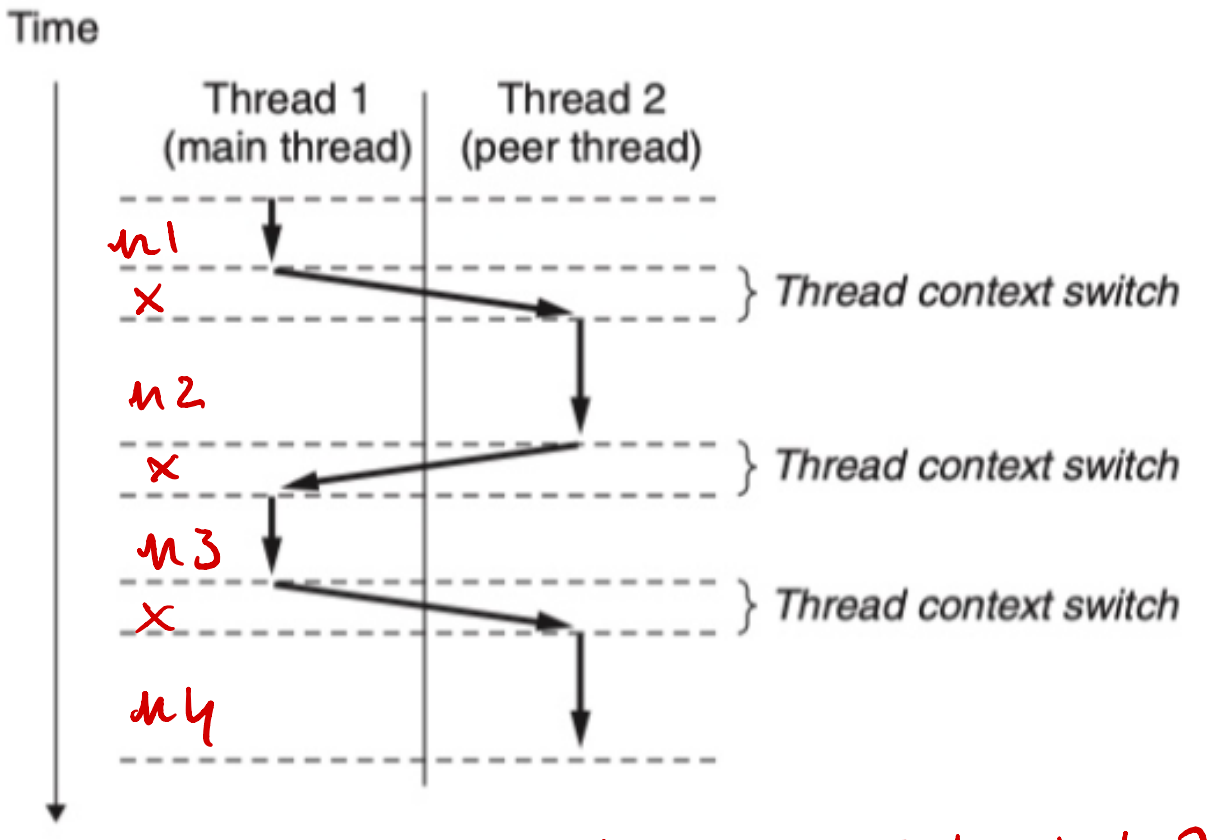





Schedule involves additional context switch cycles.

# Resource Constraint: Limited Hardware Accelerator






Total clock cycles = $n_1 + n_2 + n_3 + n_4 + 3x$.

# Proposed Solution



Perform inlining at the transformation phase.
Saves context switch clock cycles. (-3x)

T1 // T2 ⟶ T1 ; T2

# Advantage in Weak Memory Setting



T1 2clk

| | |
|---|---|
| r1=x; | $ld_{na}$ x |

T2 2clk

| | |
|---|---|
| r2=ld(&y,ACQ); | $ld_{ACQ}$ y |

T1;T2

| Cycle: | 1 | 2 |
|---|---|---|
| r1=x; | $ld_{na}$ x | |
| r2=ld(&y,ACQ); | $ld_{ACQ}$ 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?