Lake Memory Generator and SMT for Automated Memory Configuration

Maxwell Strange, Kavya Sreedhar, Nestan Tsiskaridze

AHA! Retreat

7/30/20

Evolution of Memory Tile

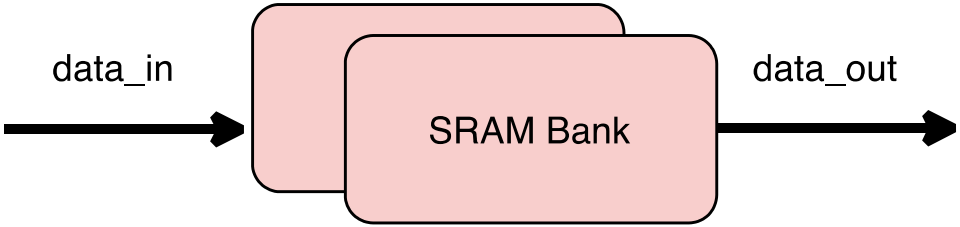
- Memory Tile
  - Line buffer
  - Line + Double Buffer
  - Unified Buffer

- Unified Buffer Implementation
  - Efficiency, cheaper ports, cleaner
  - Match the compiler
  - Lake

From Humble Beginnings… (Jade)

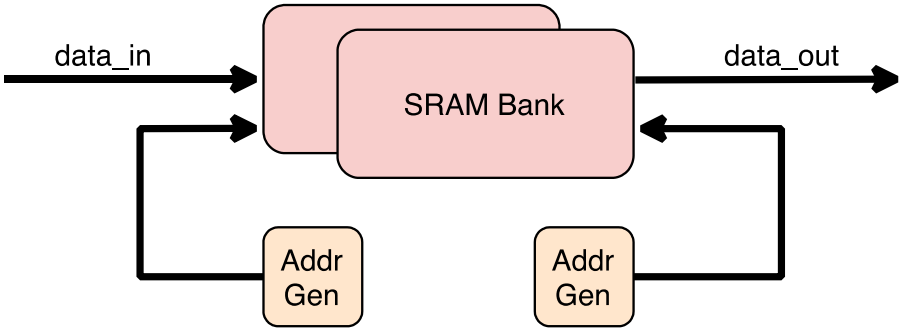
- For image applications
  - Only supports line buffer

  - Uses two banks of single-width memory
  - To provide simultaneous read and write for a FIFO

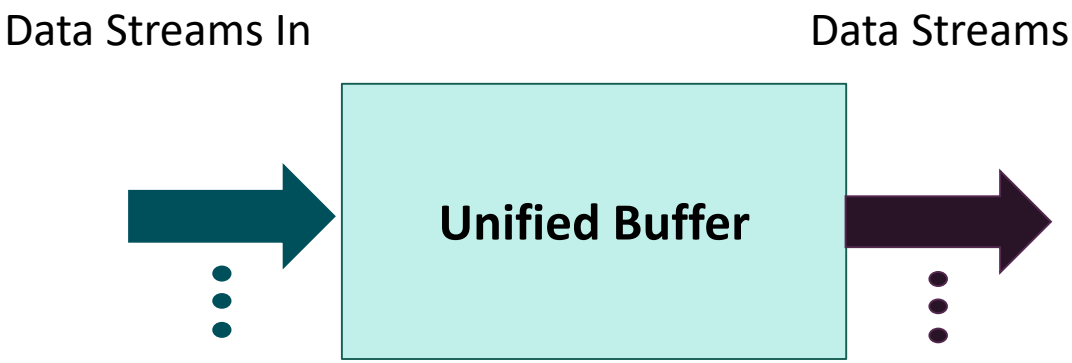


Need Double Buffering Too

- Add address generators to previous design
  - Use each bank as a buffer
  - Can map line buffer on this as well



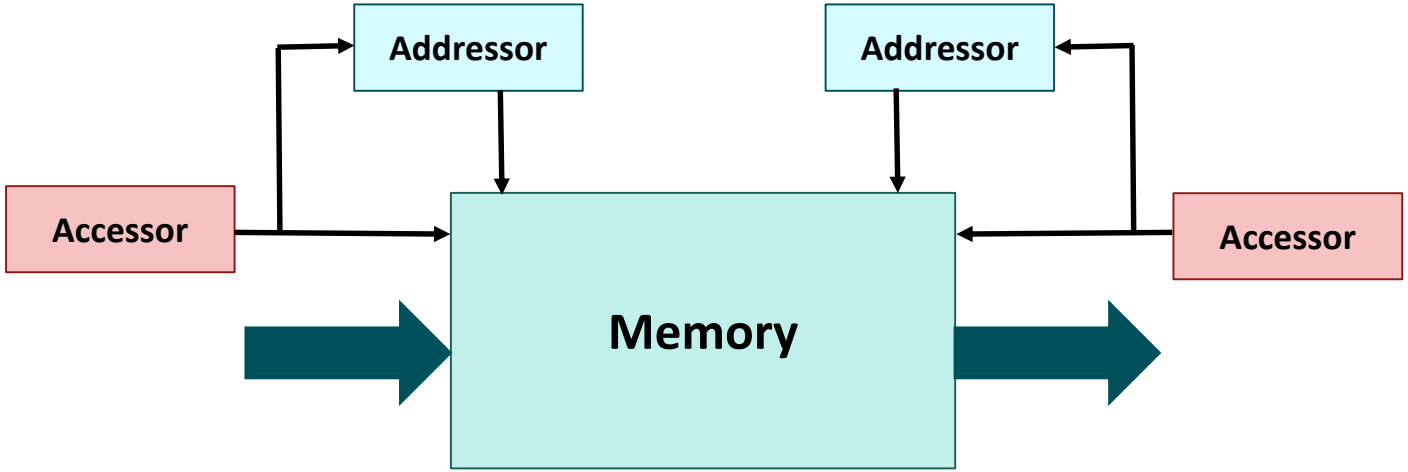
Unified Buffer



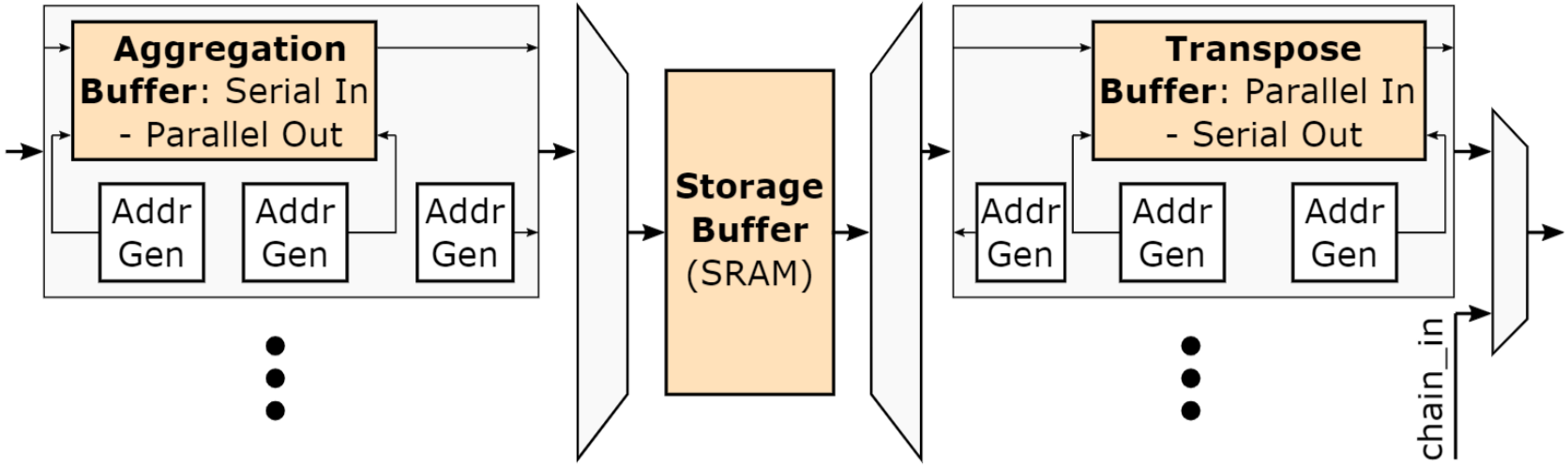
7	6	5	4	3	2	1	0
-	3	-	2	-	1	-	0
0	1	0	1	0	1	0	1

Clock Cycle	12	13	14	15	16
Data	1	5	3	-	7
Valid	1	1	1	0	1

Unified Buffer



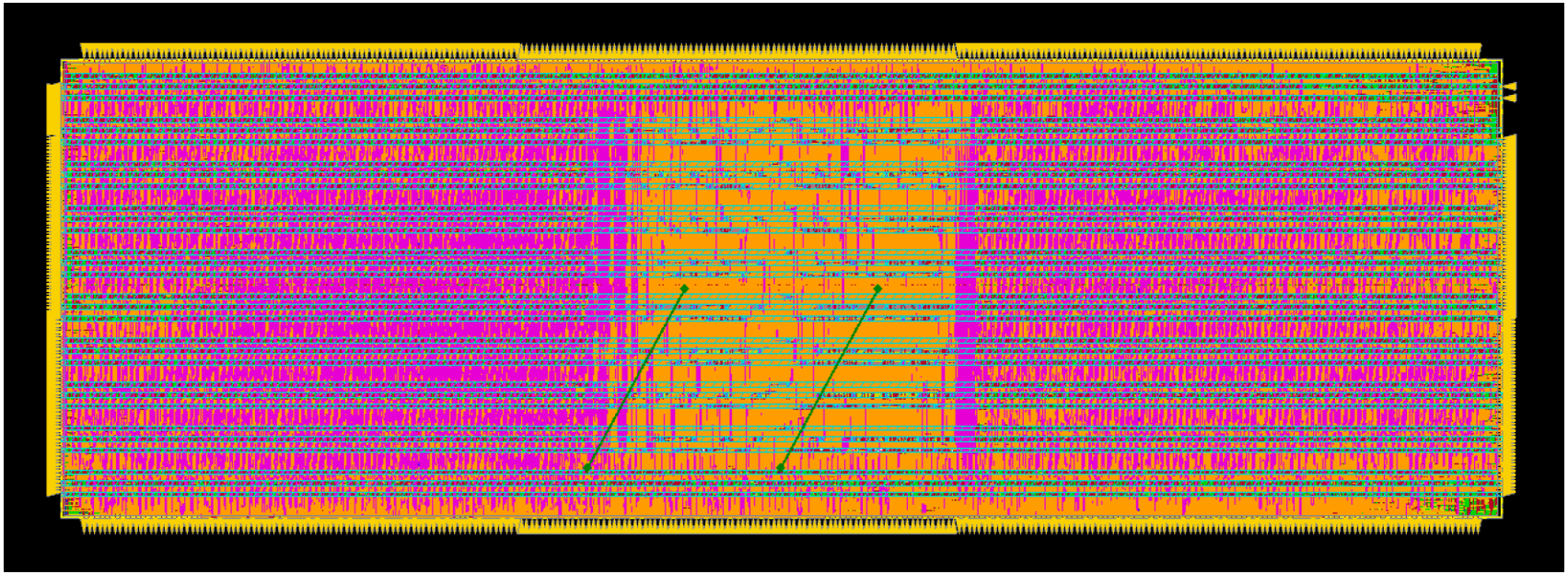
Wide Fetch Width



Lake – From Perl to Python (May 2020 tape-out)

- Memory Generator
  - Parameterizable
    - Ports, # banks, fetch width, etc.
  - Modes
    - RAM
    - FIFO
    - Unified Buffer
  - Chaining
    - Creating larger memories
    - Creating more ports

Garnet Memory Tile



Synthesis Breakdown

Module	General Idea	Area	% of Total
Interconnect	CB + SB	3324 µm^2	15.8%
Tile_MemCore/MemCore	Everything in the tile other than the interconnect	17676 µm^2	84.2%
… Logic	**All logic in the memory core**	**8304 µm^2**	**… 46.97%**
… Configuration Space	**Configuration registers and configuration bus demuxing**	**3856 µm^2**	**… 21.81%**
… SRAM Macros	2x 512x32 SRAM Macros	5516 µm^2	… 31.2%
Tile_MemCore	Entire memory tile – interconnect, core logic, SRAM Macros, configuration space	21000 µm^2	100%

Main Memory Core Logic Synthesis Breakdown

Module	General Idea	Area	% of Total MemCore Logic
FIFO Mode		675 μm^2	8.12%
SRAM Mode		95 μm^2	1.14%
Unified Buffer Mode		7268 μm^2	87.5%
… Accessors	**Drive read + write to memory components**	**1966** μm^2	**… 27%**
… Aggregation Buffers	Serial-in Parallel-out	800 (2 * 400) μm^2	… 11%
… SRAM address generators	**2x input + 2x output address generators (6 nested for loops, 16bit)**	**2620 (655 * 4)** μm^2	**… 36%**
… Transpose Buffers	Parallel-in Serial-out	800 (400 * 2) μm^2	… 11%
Tile_MemCore/Memcore Logic	All the logic in the memory core	8304 μm^2	100%

Diet Lake

Module	Area – Pre-Optimization	Area – Post Optimization	Savings
Configuration Space	3856 μm^2	3174 μm^2	17.6%
Unified Buffer Mode	7268 μm^2	2983 μm^2	58.9%
… Accessors	1966 μm^2	378 μm^2	**80.77%**
… SRAM address generators	2620 μm^2	900 μm^2	**65.648%**
Tile_MemCore/Memcore Logic	8304 μm^2	4019 μm^2	51.6%

Are We Done?

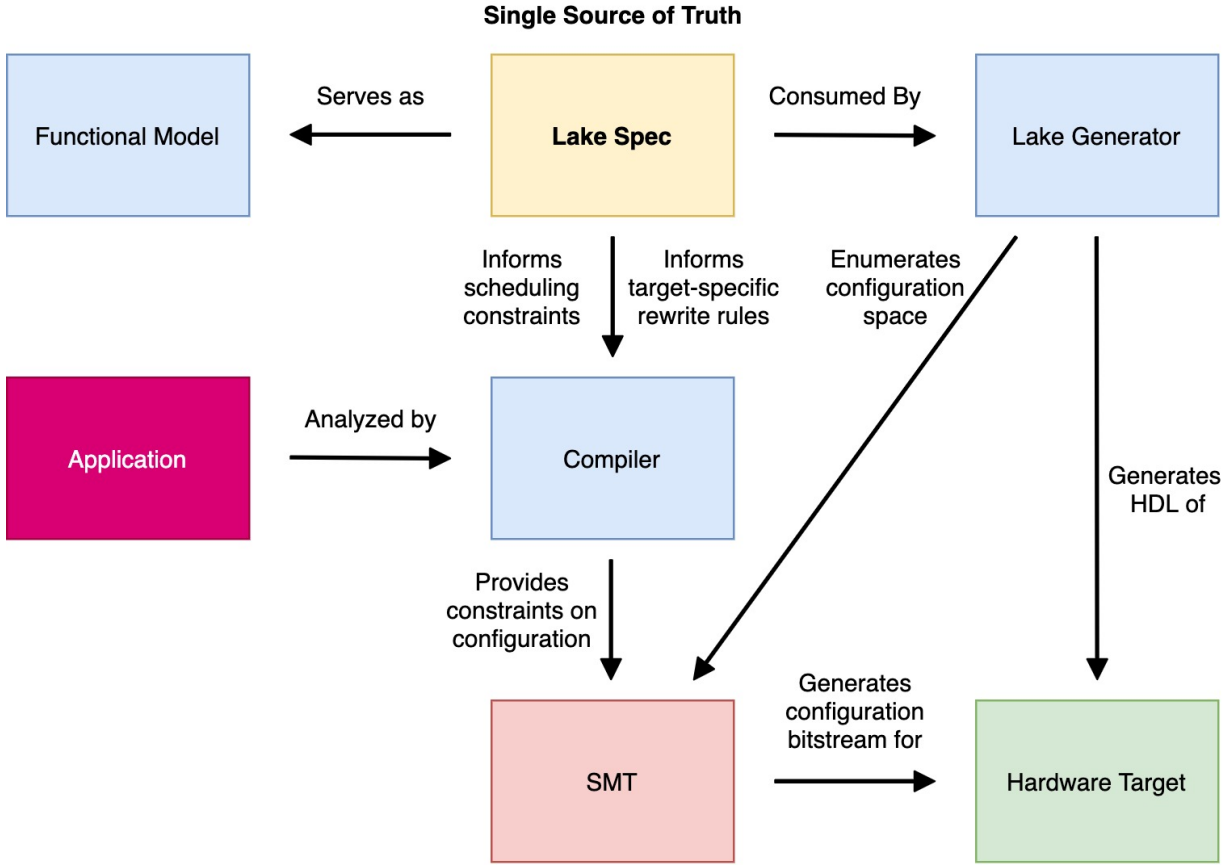
No

- Don't have a single source of truth
  - We talk with Qiaoyi(Joey) all the time about the hardware
  - Had to hand write the functional model (again, and again, and …)
- Don't really have a DSL
  - More like a set of parameters

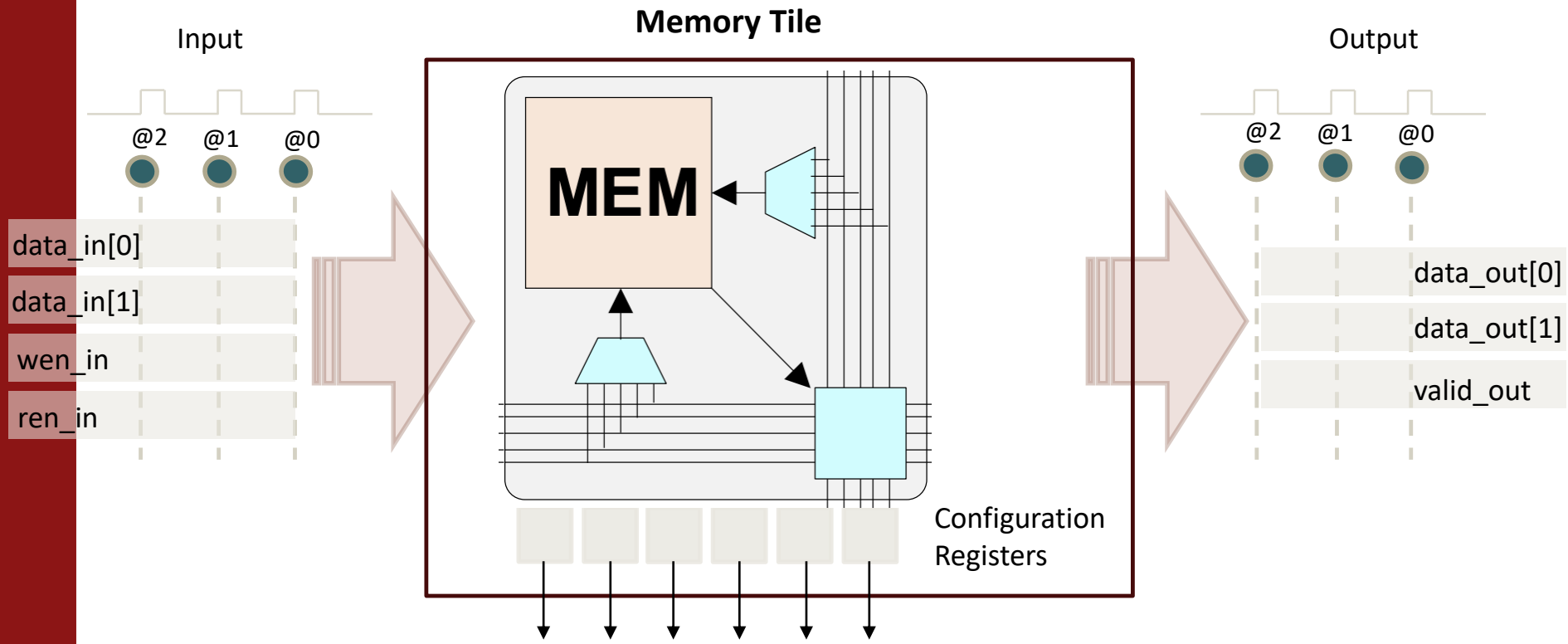
Lake, the Next Generation

1. Develop Lake primitives that can be connected in a user-specified graph (the new Lake specification)

2. Leverage hwtypes & Magma
   - For functional model and HW SST
   - For Lake specification with Graph, Node, Edge classes


Single Source of Truth
Functional Model
Serves as
Lake Spec
Consumed By
Lake Generator
Informs scheduling constraints
Informs target-specific rewrite rules
Enumerates configuration space
Application
Analyzed by
Compiler
Generates HDL of
Provides constraints on configuration
SMT
Generates configuration bitstream for
Hardware Target

Tapeout Design: 2 input ports, 2 output ports.



Memory Tile

Yosys

Pono

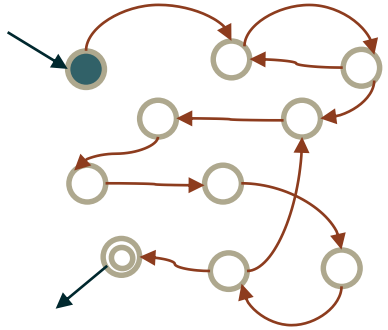
System Verilog

```
63
64 module LakeTop (
65   input logic [1:0] [15:0] addr_in,
66   input logic [1:0] [15:0] chain_data_in,
67   output logic [1:0] [15:0] chain_data_out,
68   input logic chain_idx_input,
69   input logic chain_idx_output,
70   input logic [1:0] chain_valid_in,
71   output logic [1:0] chain_valid_out,
72   input logic clk,
73   input logic clk_en,
74   input logic [7:0] config_addr_in,
75   input logic [31:0] config_data_in,
76   output logic [1:0] [31:0] config_data_out,
77   input logic [1:0] config_en,
```

Btor2

```
1  ; BTOR description generated by Yosys 0.9+1706
       for module LakeTop.
2  1 sort bitvec 16
3  2 input 1 addr_in[0]
4  3 input 1 addr_in[1]
5  4 input 1 chain_data_in[0]
6  5 input 1 chain_data_in[1]
7  6 sort bitvec 1
8  7 input 6 chain_idx_input
9  8 input 6 chain_idx_output
10 9 sort bitvec 2
11 10 input 9 chain_valid_in
12 11 input 6 clk
13 12 input 6 clk_en
14 13 sort bitvec 8
```

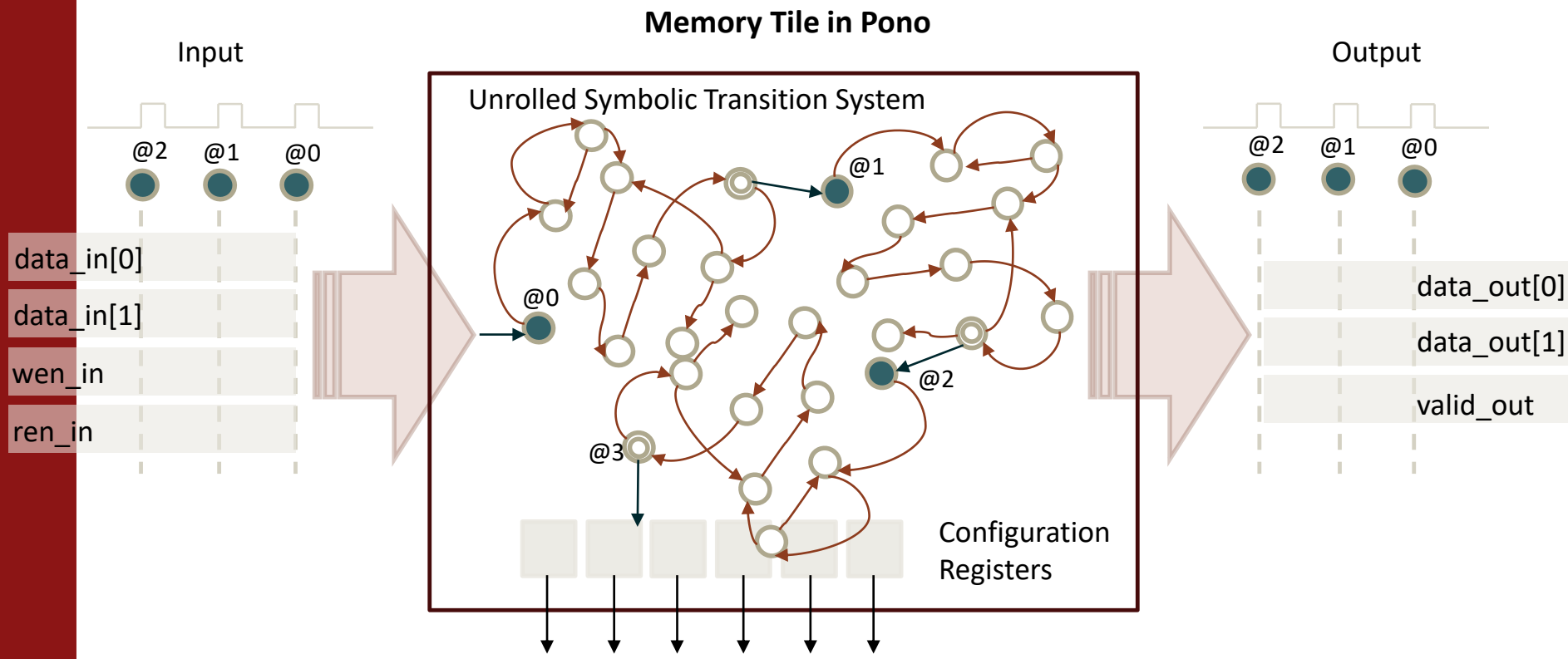
Symbolic Transition System



Btor2 format allows specifying word-level model checking problems.

Pono: our Performant, Adaptable, and Extensible SMT-based Model Checker (formerly Cosa2).

Configuration Solver



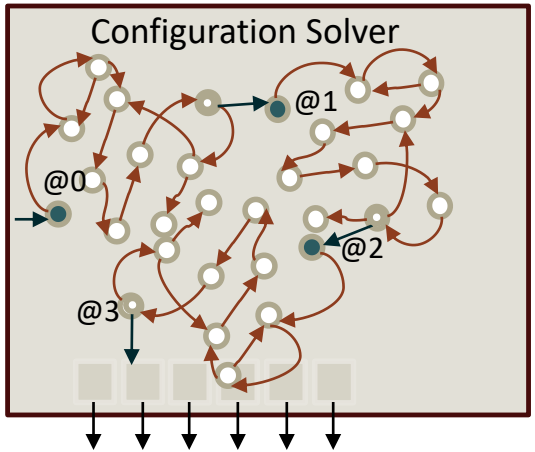
Btor2

```
; BTOR description generated by Yosys 0.9+1706
    for module LakeTop.
1 sort bitvec 16
2 input 1 addr_in[0]
3 input 1 addr_in[1]
4 input 1 chain_data_in[0]
5 input 1 chain_data_in[1]
6 sort bitvec 1
7 input 6 chain_idx_input
8 input 6 chain_idx_output
9 sort bitvec 2
10 input 9 chain_valid_in
11 input 6 clk
12 input 6 clk_en
13 sort bitvec 8
```

Configuration Constraints

```
strg_ub_pre_fetch_0_input_latency = 4
strg_ub_pre_fetch_1_input_latency = 4
chain_idx_input = 0
chain_idx_output = 0
enable_chain_input = 0
enable_chain_output = 0
tile_en = 1
mode = 0
strg_ub_sync_grp_sync_group[0] = 1
strg_ub_sync_grp_sync_group[1] = 1
var max_agg_schedule = 15
strg_ub_agg_in_0_in_period <= max_agg_schedule
strg_ub_agg_in_0_out_period <= max_agg_schedule
strg_ub_agg_in_1_in_period <= max_agg_schedule
strg_ub_agg_in_1_out_period <= max_agg_schedule
strg_ub_agg_in_0_in_sched[0]*strg_ub_agg_in_0_in_period] SOLVE
strg_ub_agg_in_0_out_sched[0]*strg_ub_agg_in_0_out_period] SOLVE
strg_ub_agg_in_1_in_sched[0]*strg_ub_agg_in_1_in_period] SOLVE
```

- Avoid non-determinism;
- Prune the search space;
- Data exchange only.



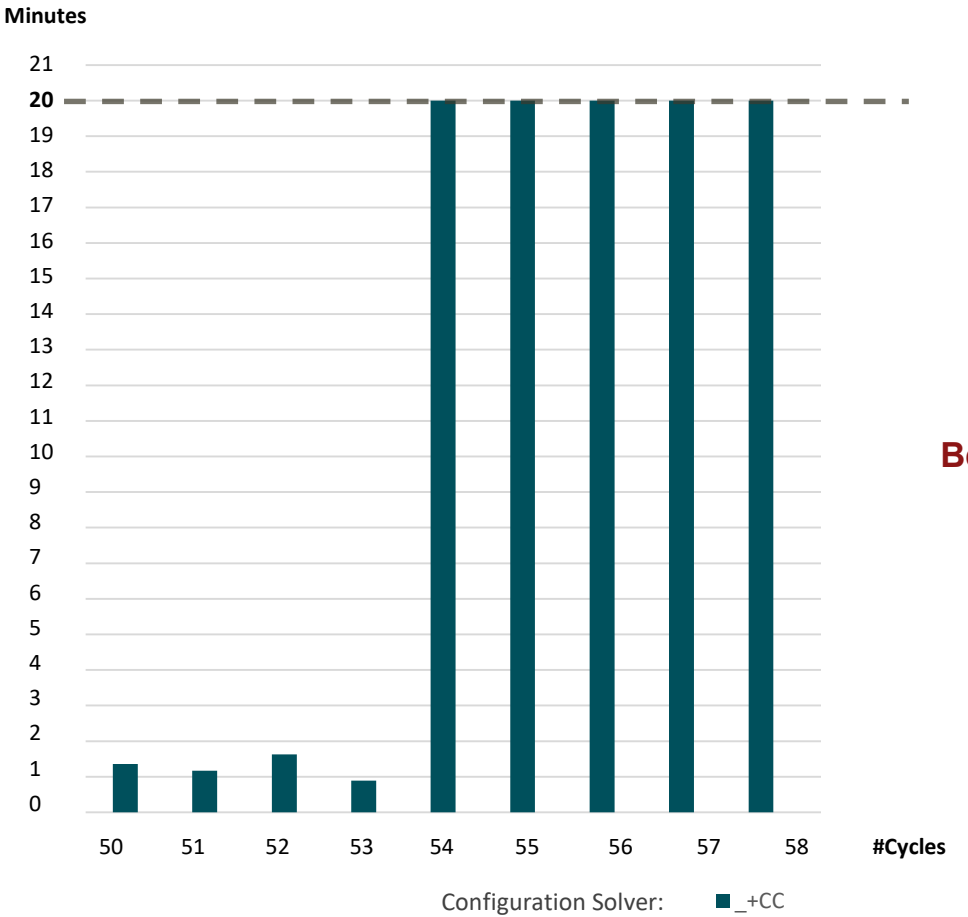
Tile Interface/ Configuration Register List

```
input logic [1:0] [15:0] addr_in,                    SET0
input logic clk_en,                                  SET1
output logic [7:0] config_addr_out,                  X
input logic chain_idx_output,                        SOLVE
input logic [1:0] chain_valid_in,                    X
output logic [1:0] chain_valid_out,                  X
input logic clk,                                     CLK
input logic clk_en,                                  SET1
input logic [7:0] config_addr_in,                    SET0
input logic [15:0] config_data_in,                   SET0
output logic [1:0] [31:0] config_data_out,           X
input logic [1:0] config_en,                         SET0
input logic config_read,                             SET0
input logic config_write,                            SET0
input logic [1:0] [15:0] data_in,                    SEQUENCE
output logic [1:0] [15:0] data_out,                  SEQUENCE
output logic empty,                                  X
input logic enable_chain_input,                      SOLVE
input logic enable_chain_output,                     SOLVE
input logic [15:0] fifo_ctrl_fifo_depth,             SOLVE
input logic flush,                                   SET0
output logic full,                                   X
input logic [1:0] mode,                              SOLVE
input logic [1:0] ren_in,                            SEQUENCE
input logic rst_n,                                   RSTN
output logic sram_ready_out,                         X
input logic [6:0] strg_ub_agg_align_0_line_length,   SOLVE
input logic [6:0] strg_ub_agg_align_1_line_length,   SOLVE
input logic [3:0] strg_ub_agg_in_0_in_period,        SOLVE
input logic [15:0] [1:0] strg_ub_agg_in_0_in_sched,  SOLVE
input logic [3:0] strg_ub_agg_in_0_out_period,       SOLVE
input logic [15:0] [1:0] strg_ub_agg_in_0_out_sched, SOLVE
input logic [3:0] strg_ub_agg_in_1_in_period,        SOLVE
```

Memory Tile Sequence

```
data_in, wen_in, ren_in, data_out, valid_out
[[0], [0]], 1, 0, [[0], [0]], 0
[[1], [0]], 1, 0, [[0], [0]], 0
[[2], [0]], 1, 0, [[0], [0]], 0
[[3], [0]], 1, 0, [[0], [0]], 0
[[4], [0]], 1, 0, [[0], [0]], 0
[[5], [0]], 1, 0, [[0], [0]], 0
[[6], [0]], 1, 0, [[0], [0]], 0
[[7], [0]], 1, 0, [[0], [0]], 0
[[8], [0]], 1, 0, [[0], [0]], 0
[[9], [0]], 1, 0, [[0], [0]], 0
[[10], [0]], 1, 0, [[0], [0]], 0
[[11], [0]], 1, 0, [[0], [0]], 0
[[12], [0]], 1, 0, [[0], [0]], 0
[[13], [0]], 1, 0, [[0], [0]], 0
[[14], [0]], 1, 0, [[0], [0]], 0
[[15], [0]], 1, 0, [[0], [0]], 0
[[16], [0]], 1, 0, [[0], [0]], 0
[[17], [0]], 1, 0, [[0], [0]], 0
[[18], [0]], 1, 0, [[0], [0]], 0
[[19], [0]], 1, 0, [[0], [0]], 0
[[20], [0]], 1, 0, [[0], [0]], 0
[[21], [0]], 1, 0, [[0], [0]], 0
[[22], [0]], 1, 0, [[0], [0]], 0
[[23], [0]], 1, 0, [[0], [0]], 0
[[24], [0]], 1, 0, [[0], [0]], 0
[[25], [0]], 1, 0, [[0], [0]], 0
[[26], [0]], 1, 0, [[0], [0]], 0
[[27], [0]], 1, 0, [[0], [0]], 0
[[28], [0]], 1, 3, [[0], [0]], 0
[[29], [0]], 1, 3, [[0], [0]], 0
[[30], [0]], 1, 3, [[0], [0]], 0
[[31], [0]], 1, 3, [[0], [0]], 0
[[32], [0]], 1, 3, [[0], [0]], 0
[[33], [0]], 1, 3, [[0], [16]], 3
[[34], [0]], 1, 3, [[1], [17]], 3
[[35], [0]], 1, 3, [[2], [18]], 3
```

How far can we scale?



Memory Tile reads start @54

Timeout = 20 min

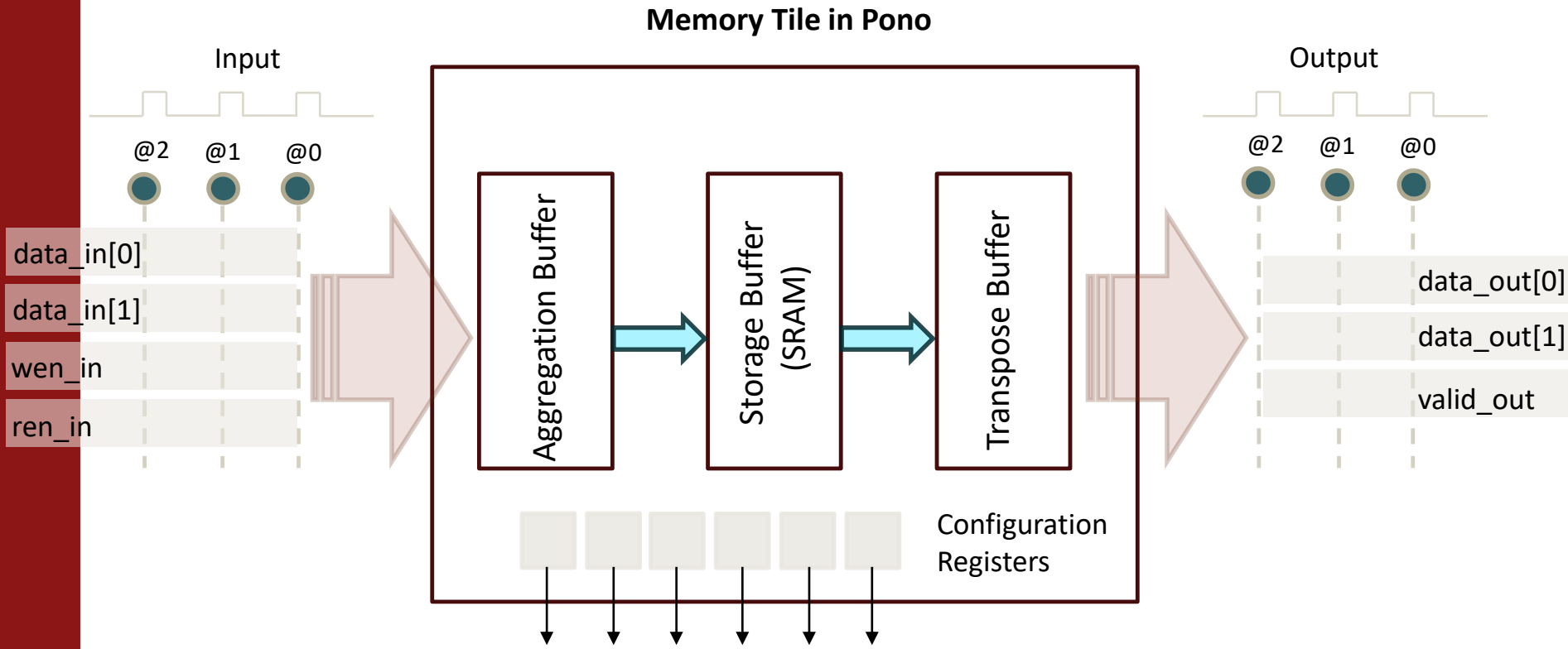
Better…?

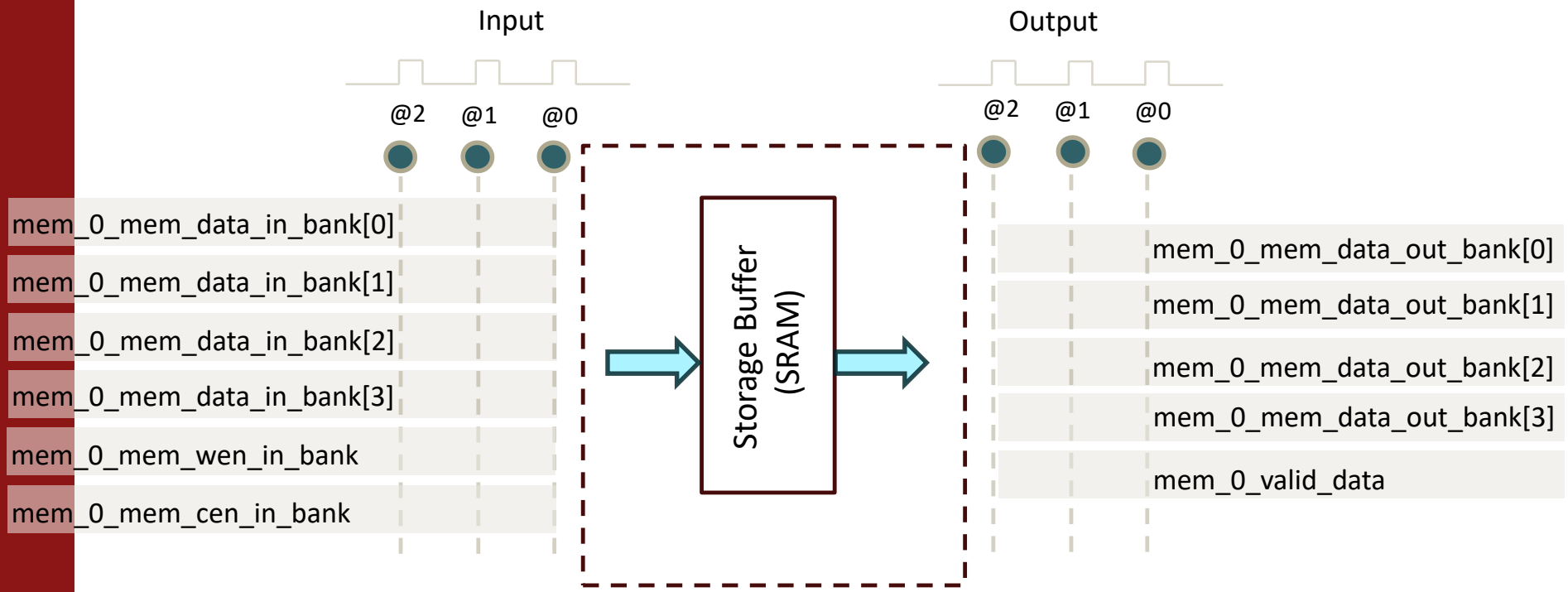
Conv33

32x32 image

Intel Xeon E5-2620 v4, CPU 1.6GHz, 16GB.

Can we utilize more information from users/designers?


Memory Tile in Pono
Input
@2
@1
@0
data_in[0]
data_in[1]
wen_in
ren_in
Aggregation Buffer
Storage Buffer (SRAM)
Transpose Buffer
Configuration Registers
Output
@2
@1
@0
data_out[0]
data_out[1]
valid_out



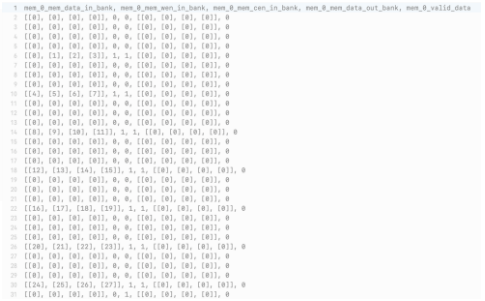
Internal sequence obtained from the polyhedral analysis.

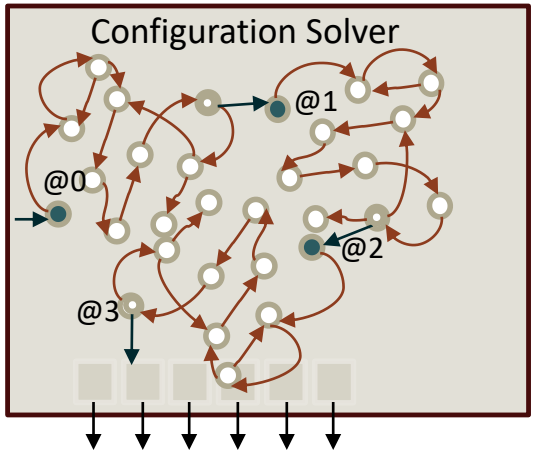
Btor2

```
; BTOR description generated by Yosys 0.9+1706
    for module LakeTop.
1 sort bitvec 16
2 input 1 addr_in[0]
3 input 1 addr_in[1]
4 input 1 chain_data_in[0]
5 input 1 chain_data_in[1]
6 sort bitvec 1
7 input 6 chain_idx_input
8 input 6 chain_idx_output
9 sort bitvec 2
10 input 9 chain_valid_in
11 input 6 clk
12 input 6 clk_en
13 sort bitvec 8
```

Configuration Constraints

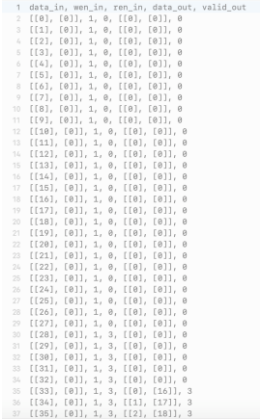
SRAM Sequence





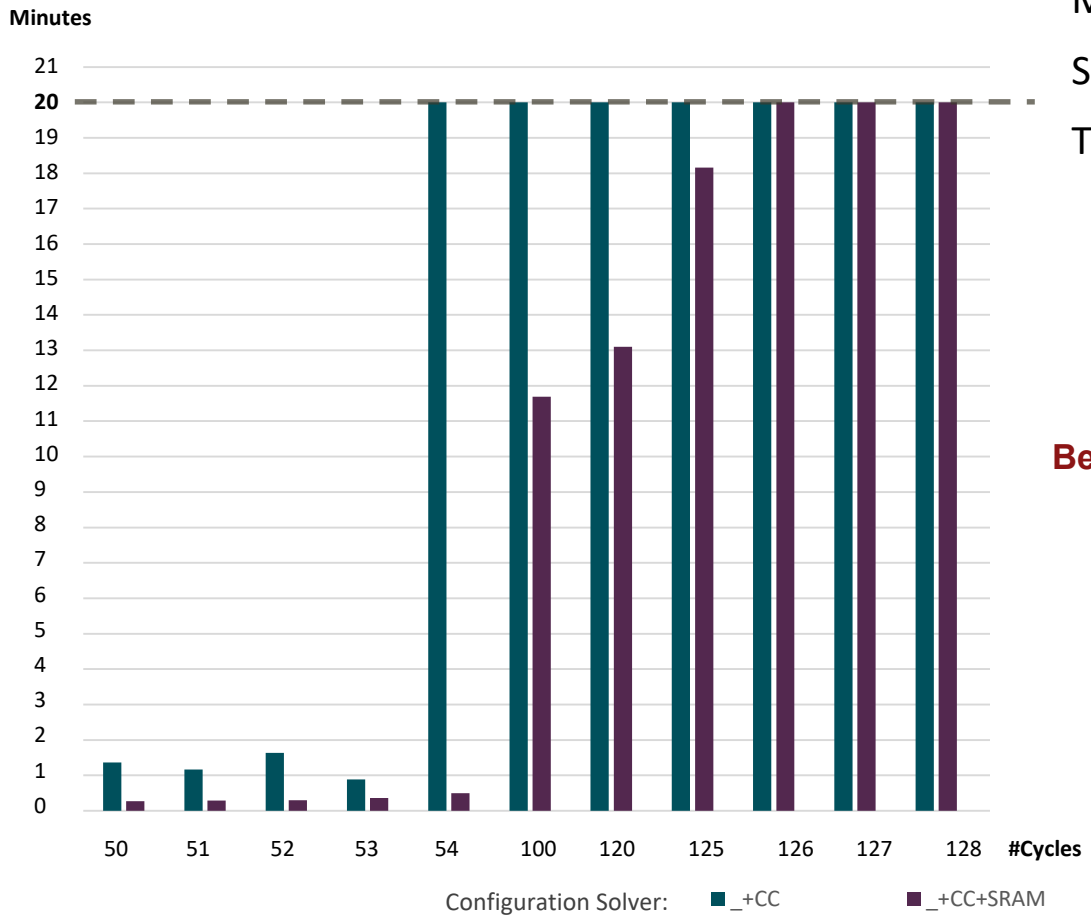
Tile Interface/ Configuration Register List

Memory Tile Sequence



Stanford University

How far can we scale?



Memory Tile reads start @54

SRAM reads start @51

Timeout = 20 min

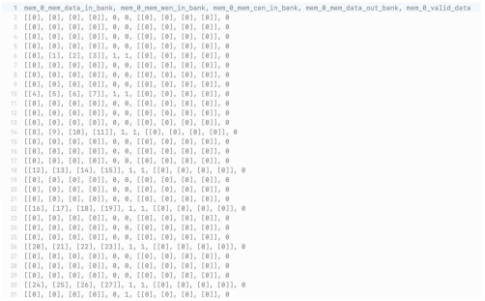
Better…?

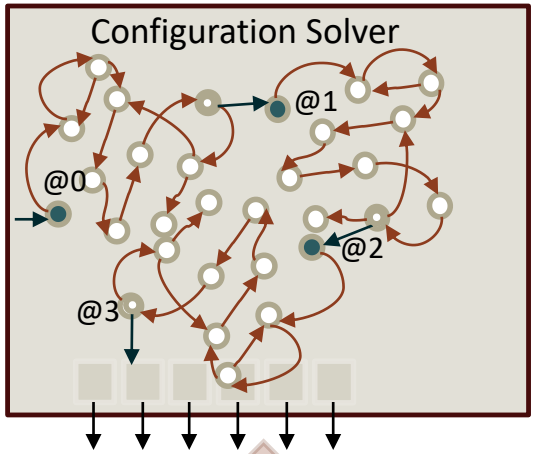
Btor2

```
; BTOR description generated by Yosys 0.9+1706
    for module LakeTop.
1 sort bitvec 16
2 input 1 addr_in[0]
3 input 1 addr_in[1]
4 input 1 chain_data_in[0]
5 input 1 chain_data_in[1]
6 sort bitvec 1
7 input 6 chain_idx_input
8 input 6 chain_idx_output
9 sort bitvec 2
10 input 9 chain_valid_in
11 input 6 clk
12 input 6 clk_en
13 sort bitvec 8
```

Configuration Constraints

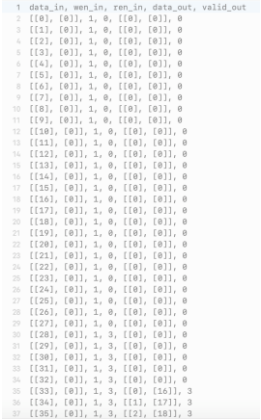
SRAM Sequence





Tile Interface/ Configuration Register List

Memory Tile Sequence



Address Configuration Registers

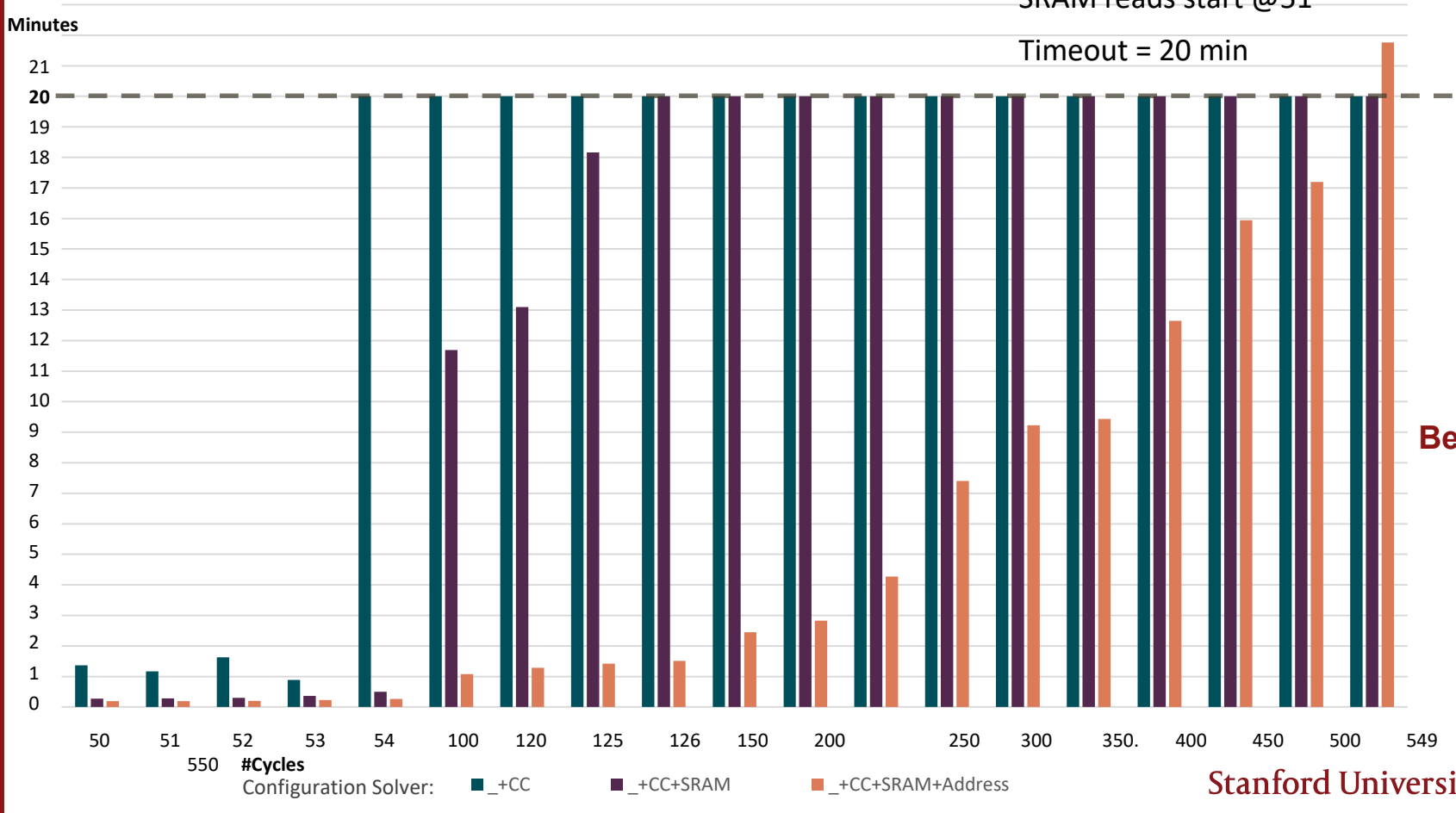
Stanford University

How far can we scale?

Memory Tile reads start @54

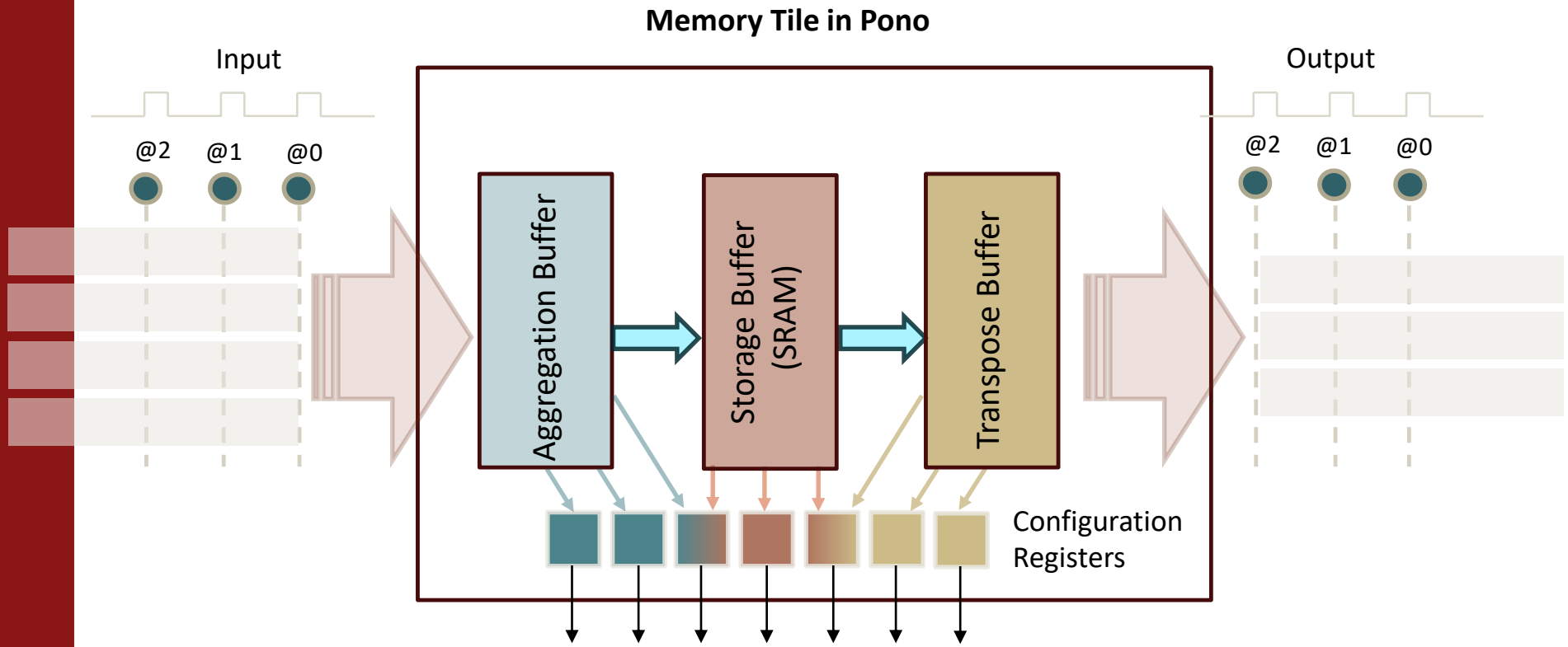
SRAM reads start @51

Timeout = 20 min



Better…?

New Modular Design

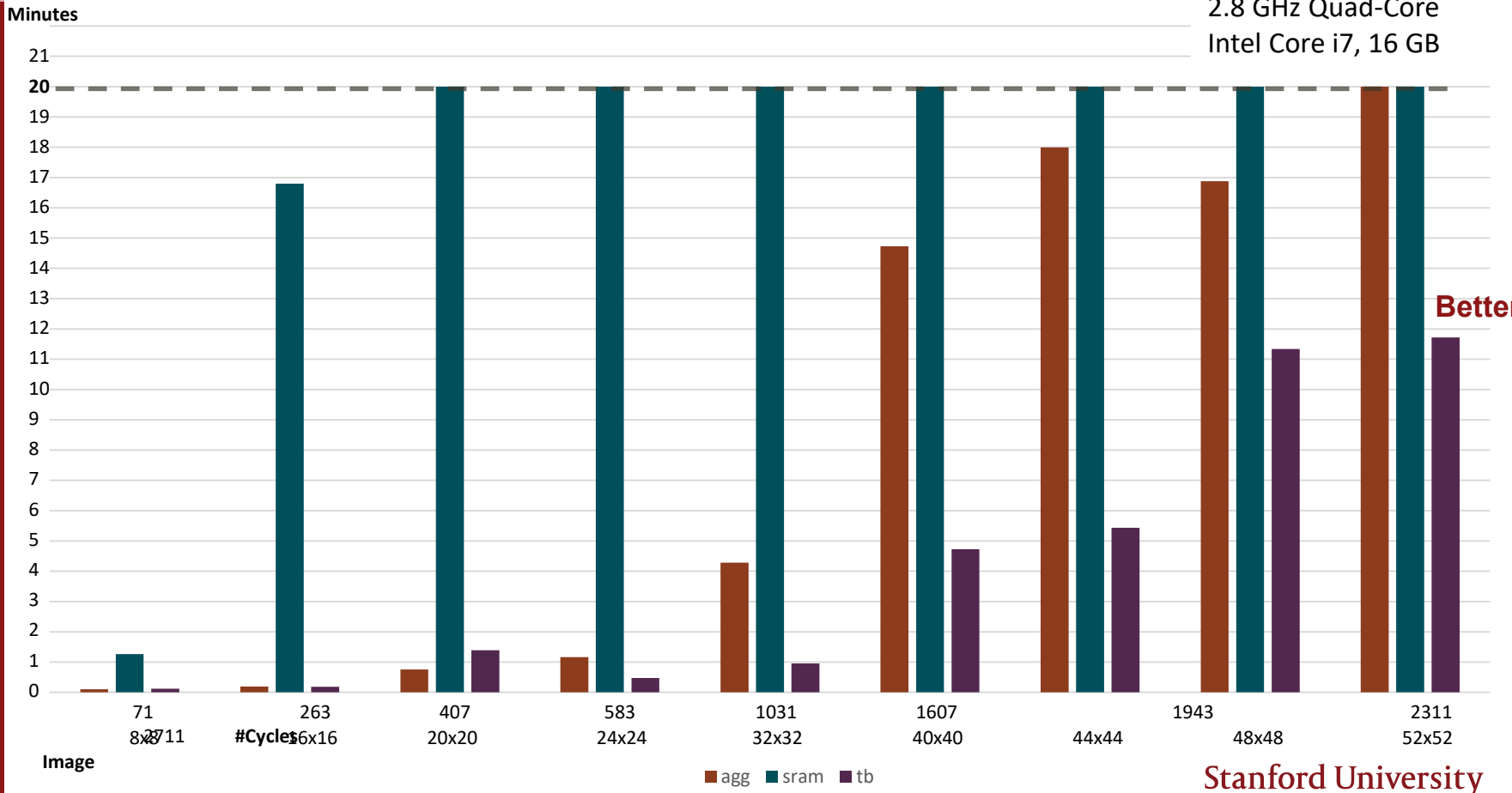


How far can we scale with a modular approach?

Identity stream

Memory Tile reads start @8

2.8 GHz Quad-Core
Intel Core i7, 16 GB



Better…?

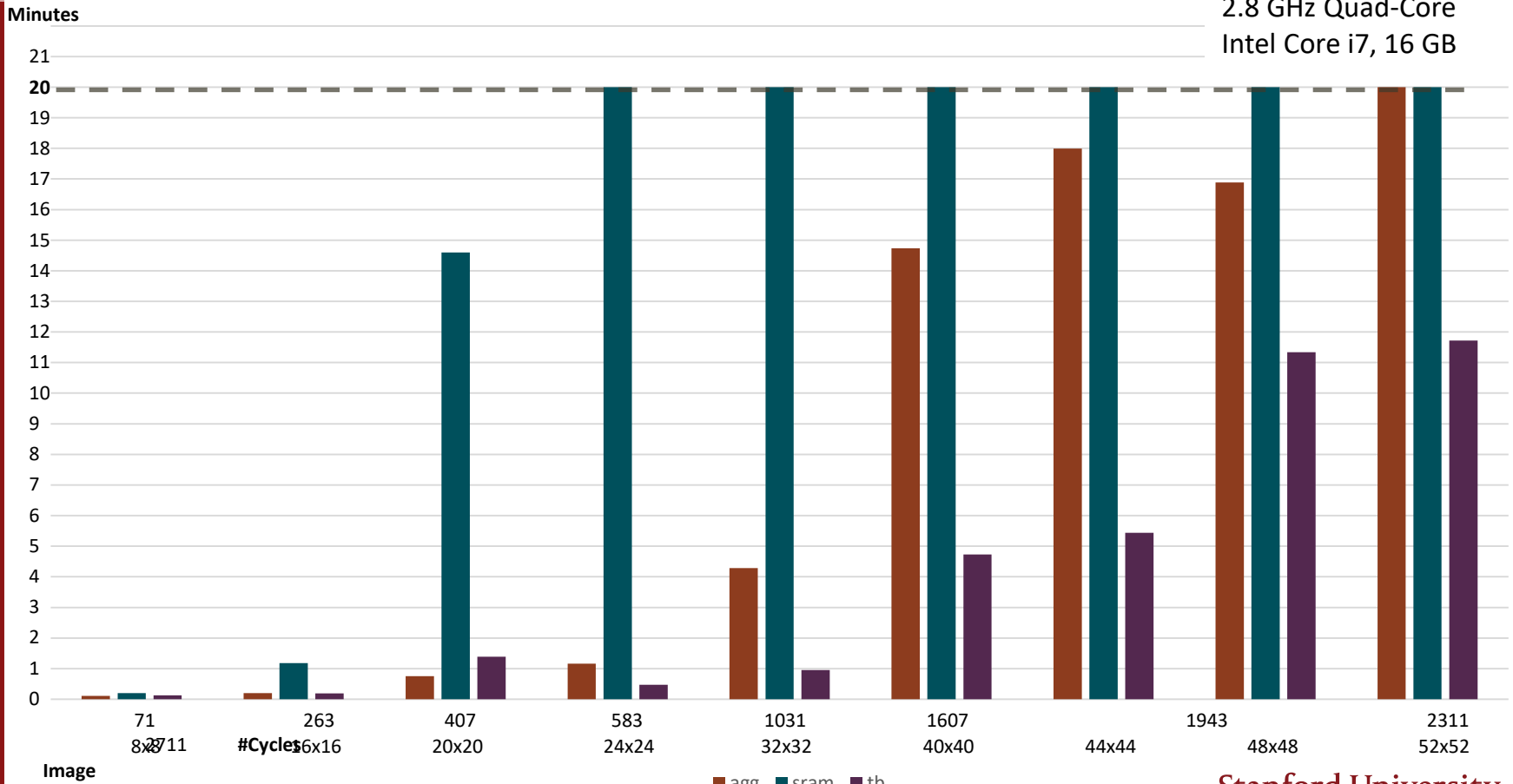
Stanford University

How far can we scale with a modular approach?

Identity stream

Memory Tile reads start @8

2.8 GHz Quad-Core
Intel Core i7, 16 GB



Image

Propagating configuration solutions from agg & tb to sram.

Tool Summary

- A **Configuration Solver** based on **Pono** and **SMT** that is *fully automated* and *flexible*.
- The approach works!

Towards More Agile Design

- More complex and larger designs;
- More information to utilize from inside the design;
- More modular control over different modules of the design – each module controlled through their configuration registers.