

Lake Memory Generator and SMT for Automated Memory Configuration

MAXWELL STRANGE, KAVYA SREEDHAR, NESTAN TSISKARIDZE

AHA! Retreat

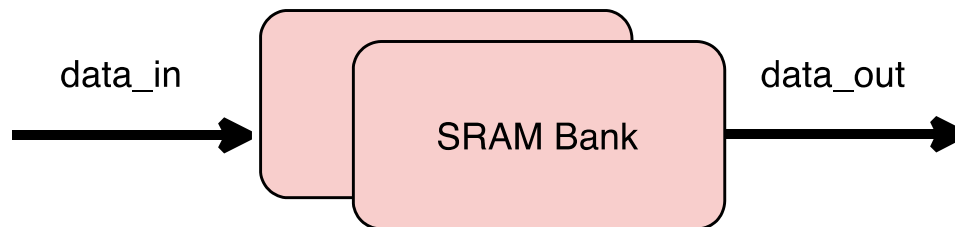
7/30/20

Evolution of Memory Tile

- Memory Tile
 - Line buffer
 - Line + Double Buffer
 - Unified Buffer
- Unified Buffer Implementation
 - Efficiency, cheaper ports, cleaner
 - Match the compiler
 - Lake

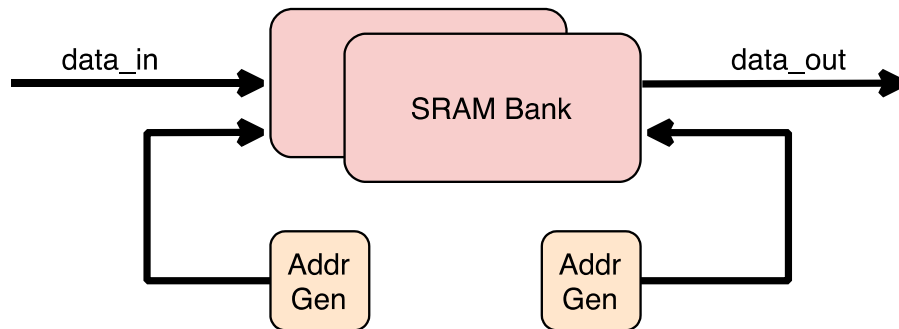
From Humble Beginnings... (Jade)

- For image applications
 - Only supports line buffer
 - Uses two banks of single-width memory
 - To provide simultaneous read and write for a FIFO



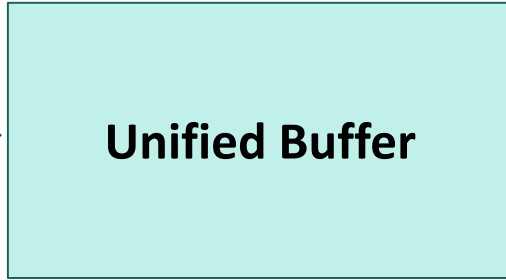
Need Double Buffering Too

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

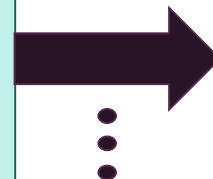


Unified Buffer

Data Streams In



Data Streams Out



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

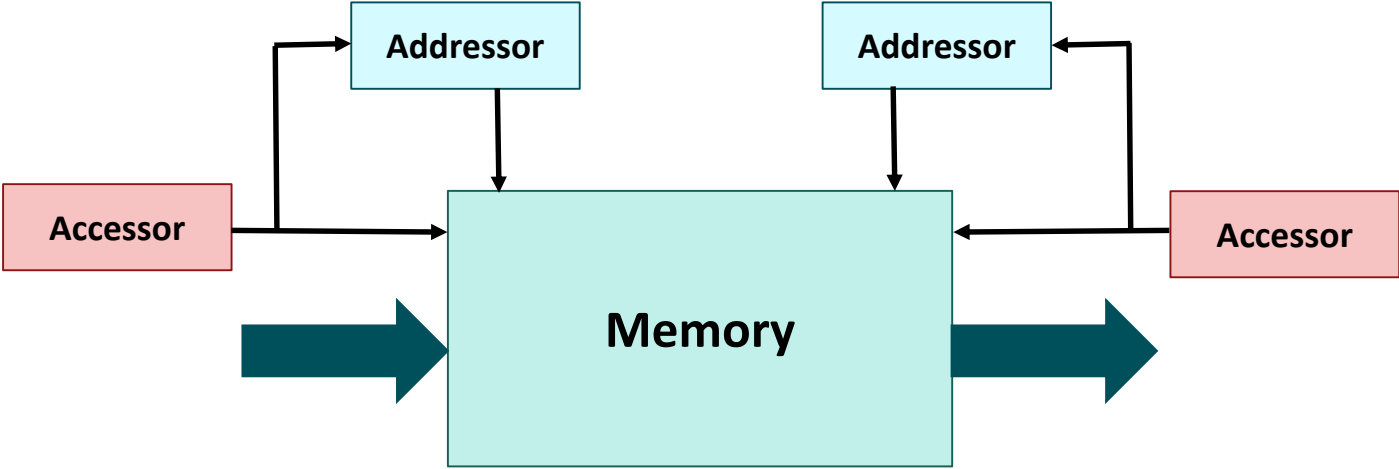
Clock Cycle

Data

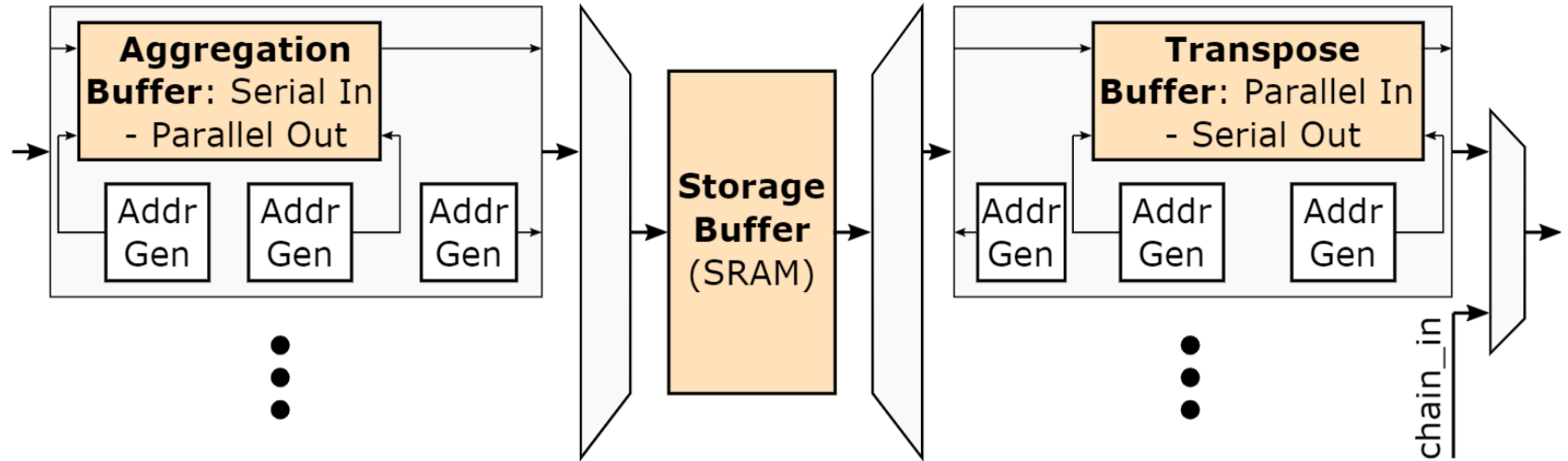
Valid

12	13	14	15	16
1	5	3	-	7
1	1	1	0	1

Unified Buffer



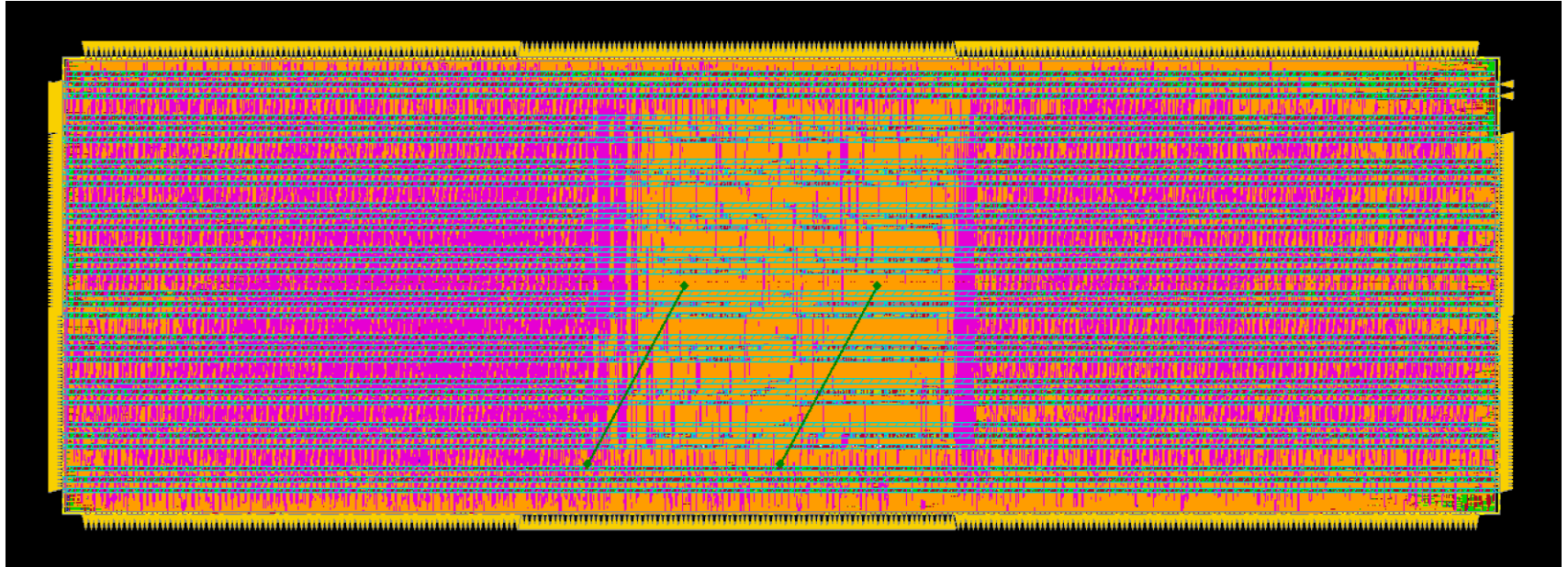
Wide Fetch Width



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%

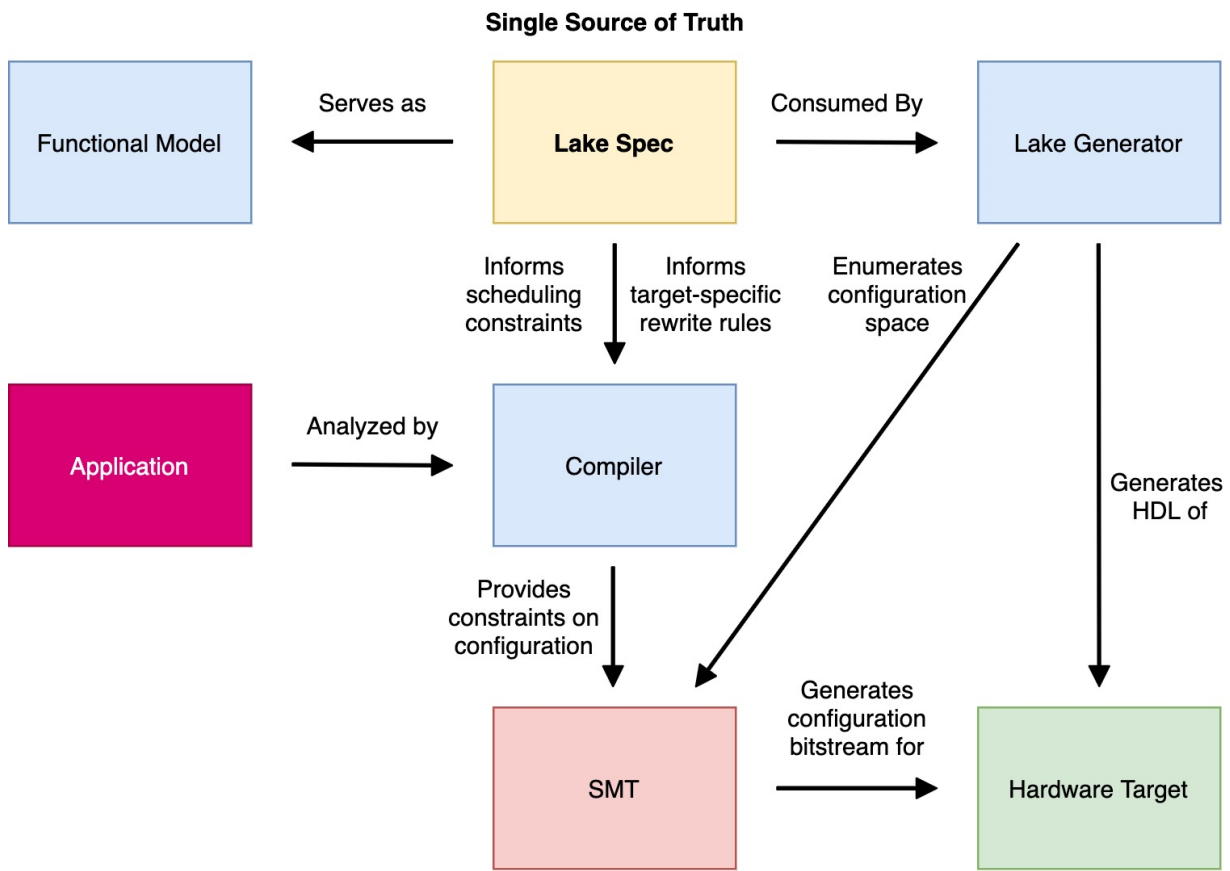
Are We Done?

No

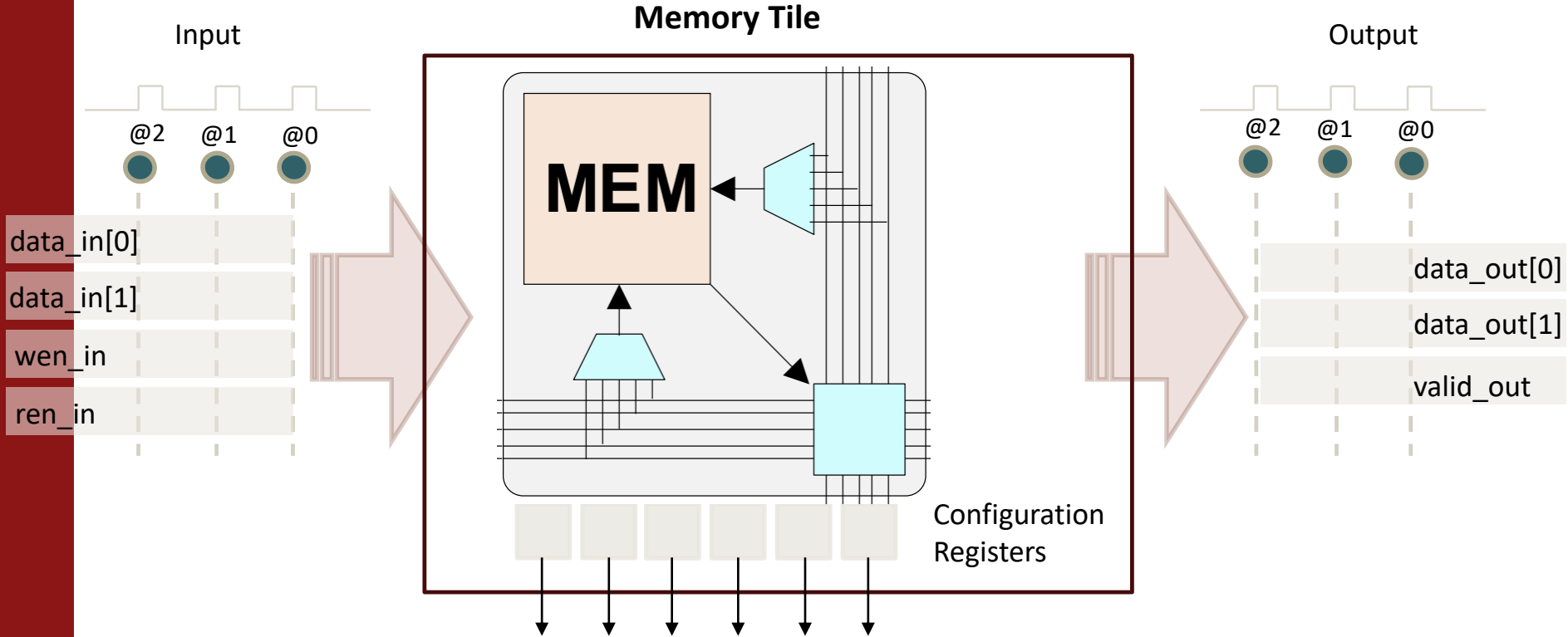
- Don't have a single source of truth
 - We talk with Qiaoyi(Joey) all the time about the hardware
 - Had to hand write the functional model (again, and again, and ...)
- Don't really have a DSL
 - More like a set of parameters

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



Tapeout Design: 2 input ports, 2 output ports.



Memory Tile



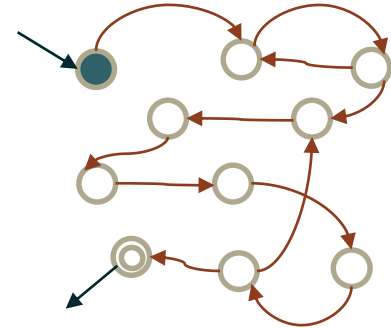
System Verilog

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

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

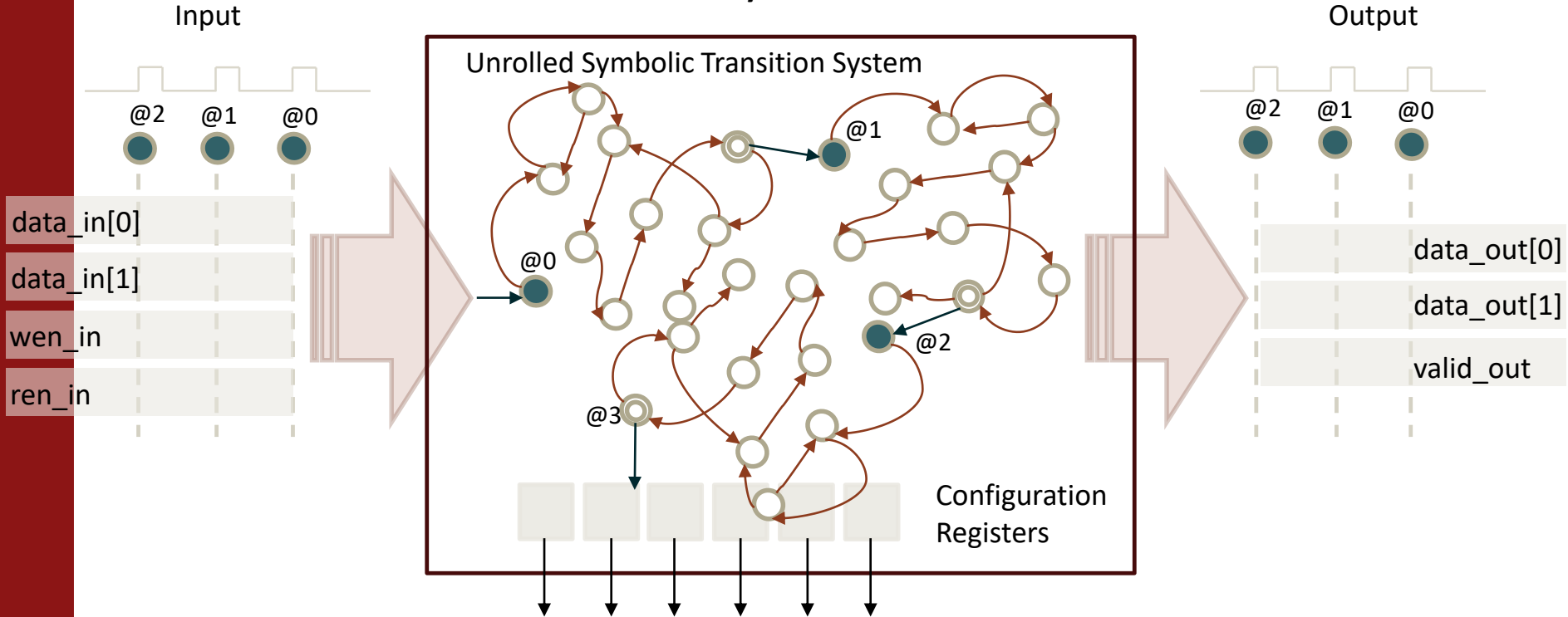


Btor2 format allows specifying word-level model checking problems.

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

Configuration Solver

Memory Tile in Pono



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

```

Configuration Constraints

```

// strg_ub_pre_fetch_0_input_latency = 4
// strg_ub_pre_fetch_1_input_latency = 4
// chain_idx_input = 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

```

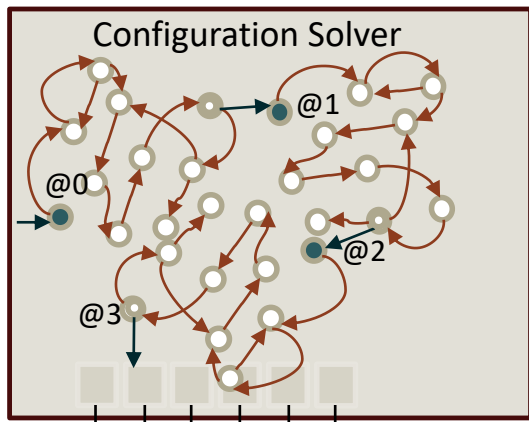
- Avoid non-determinism;
- Prune the search space;
- Data exchange only.

Tile Interface/ Configuration Register List

```

1 input logic [1:0] [0] addr_in, SET0
2 input logic [1:0] [1] chain_data_in, SET0
3 input logic [1:0] [2] chain_idx_input, SET0
4 input logic [1:0] [3] chain_idx_output, SET0
5 input logic [1:0] [4] chain_valid_in, SET0
6 output logic [1:0] [5] chain_valid_out, X
8 input logic clk, CLK
9 input logic clk_en, SET1
10 input logic [7:0] config_addr_in, SET0
11 input logic [15:0] config_data_in, SET0
12 output logic [1:0] [31:0] config_data_out, X
13 input logic [1:0] config_en, SET0
14 input logic config_read, SET0
15 input logic config_write, SET0
16 input logic [1:0] [15:0] data_in, SEQUENCE
17 output logic [1:0] [15:0] data_out, SEQUENCE
18 output logic empty, X
19 input logic enable_chain_input, SOLVE
20 input logic enable_chain_output, SOLVE
21 input logic [15:0] rifo_ctrl_rifo_depth, SET0
22 input logic flush, X
23 output logic full, X
24 input logic [1:0] mode, SET0
25 input logic [1:0] ren_in, SEQUENCE
26 input logic rst_n, RSTN
27 output logic stam_ready_out, X
28 input logic [4:0] strg_ub_agg_align_0_line_length, SOLVE
29 input logic [6:0] strg_ub_agg_align_1_line_length, SOLVE
30 input logic [3:0] strg_ub_agg_in_0_in_period, SOLVE
31 input logic [15:0] [1:0] strg_ub_agg_in_0_in_sched, SOLVE
32 input logic [3:0] strg_ub_agg_in_0_out_period, SOLVE
33 input logic [15:0] [1:0] strg_ub_agg_in_0_out_sched, SOLVE
34 input logic [3:0] strg_ub_agg_in_1_in_period, SOLVE

```



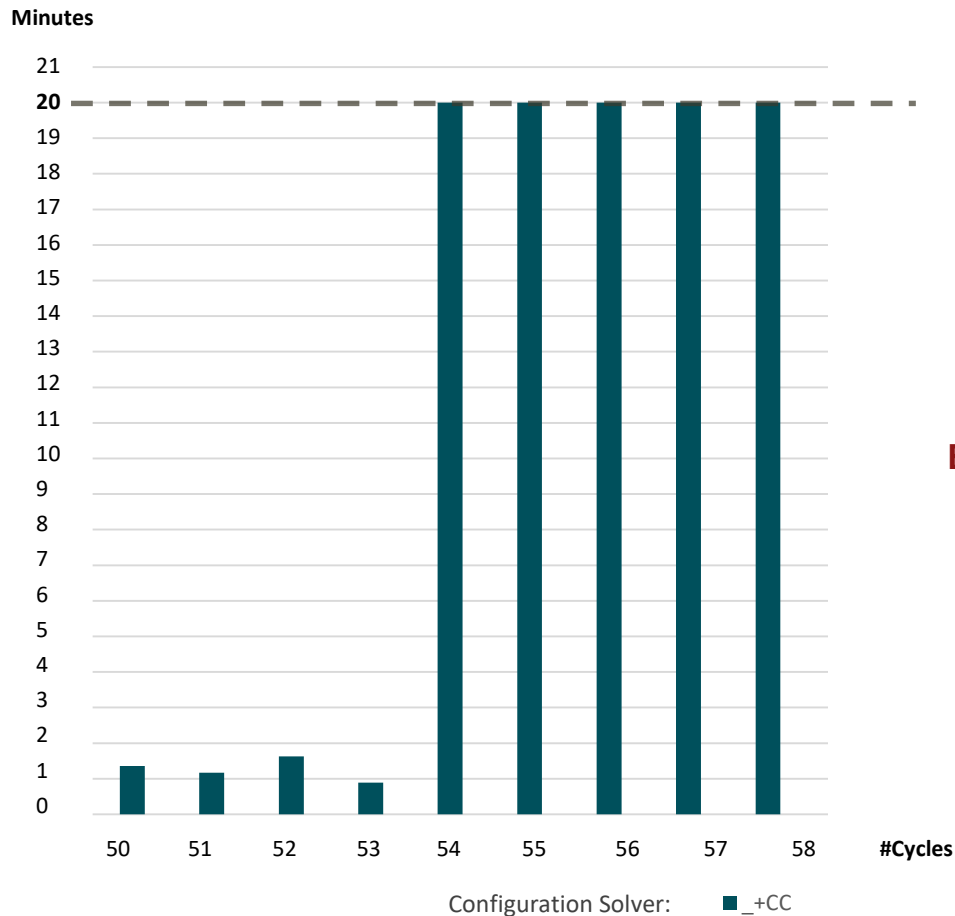
Memory Tile Sequence

```

1 data_in_wen_in_ren_in_data_out_valid_out
2 [[0], [0]], 1, 0, [[0], [0]], 0
3 [[1], [0]], 1, 0, [[0], [0]], 0
4 [[2], [0]], 1, 0, [[0], [0]], 0
5 [[3], [0]], 1, 0, [[0], [0]], 0
6 [[4], [0]], 1, 0, [[0], [0]], 0
7 [[5], [0]], 1, 0, [[0], [0]], 0
8 [[6], [0]], 1, 0, [[0], [0]], 0
9 [[7], [0]], 1, 0, [[0], [0]], 0
10 [[8], [0]], 1, 0, [[0], [0]], 0
11 [[9], [0]], 1, 0, [[0], [0]], 0
12 [[10], [0]], 1, 0, [[0], [0]], 0
13 [[11], [0]], 1, 0, [[0], [0]], 0
14 [[12], [0]], 1, 0, [[0], [0]], 0
15 [[13], [0]], 1, 0, [[0], [0]], 0
16 [[14], [0]], 1, 0, [[0], [0]], 0
17 [[15], [0]], 1, 0, [[0], [0]], 0
18 [[16], [0]], 1, 0, [[0], [0]], 0
19 [[17], [0]], 1, 0, [[0], [0]], 0
20 [[18], [0]], 1, 0, [[0], [0]], 0
21 [[19], [0]], 1, 0, [[0], [0]], 0
22 [[20], [0]], 1, 0, [[0], [0]], 0
23 [[21], [0]], 1, 0, [[0], [0]], 0
24 [[22], [0]], 1, 0, [[0], [0]], 0
25 [[23], [0]], 1, 0, [[0], [0]], 0
26 [[24], [0]], 1, 0, [[0], [0]], 0
27 [[25], [0]], 1, 0, [[0], [0]], 0
28 [[26], [0]], 1, 0, [[0], [0]], 0
29 [[27], [0]], 1, 0, [[0], [0]], 0
30 [[28], [0]], 1, 3, [[0], [0]], 0
31 [[29], [0]], 1, 3, [[0], [0]], 0
32 [[30], [0]], 1, 3, [[0], [0]], 0
33 [[31], [0]], 1, 3, [[0], [0]], 0
34 [[32], [0]], 1, 3, [[1], [16]], 3
35 [[33], [0]], 1, 3, [[1], [17]], 3
36 [[34], [0]], 1, 3, [[1], [18]], 3

```

How far can we scale?



Memory Tile reads start @54

Timeout = 20 min

Better...?

Conv33

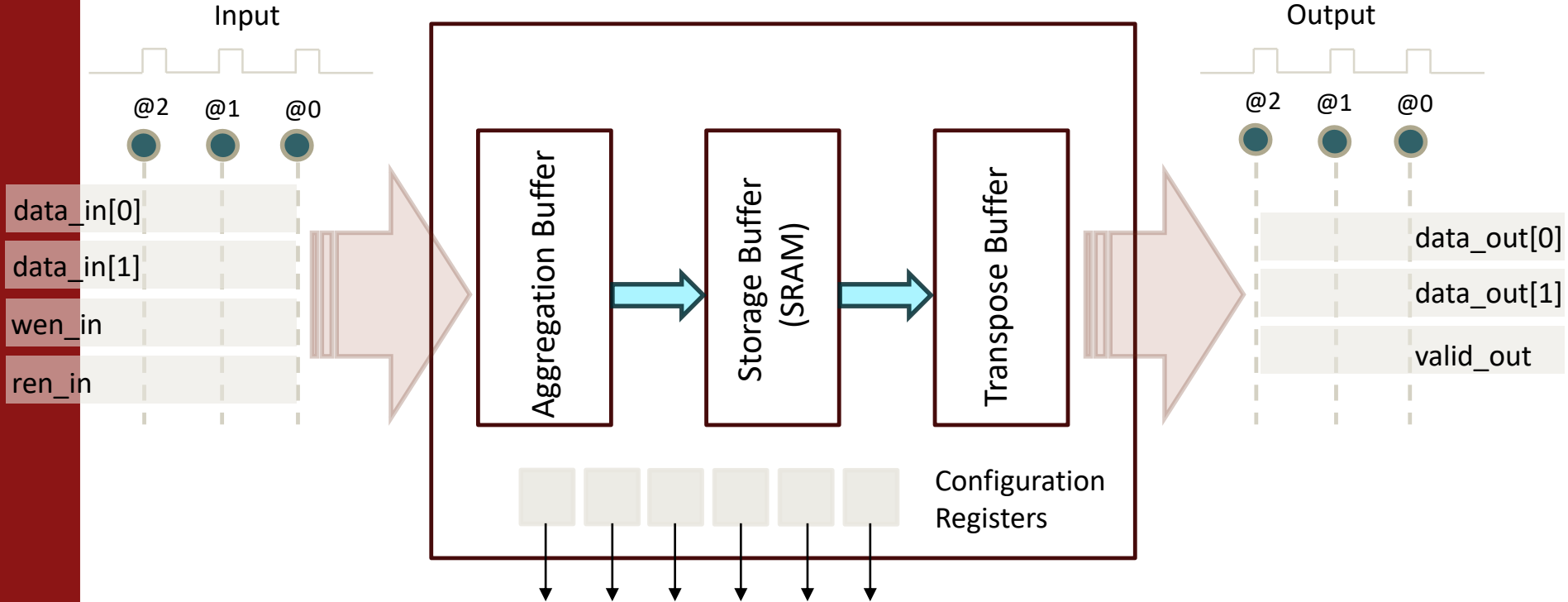
32x32 image

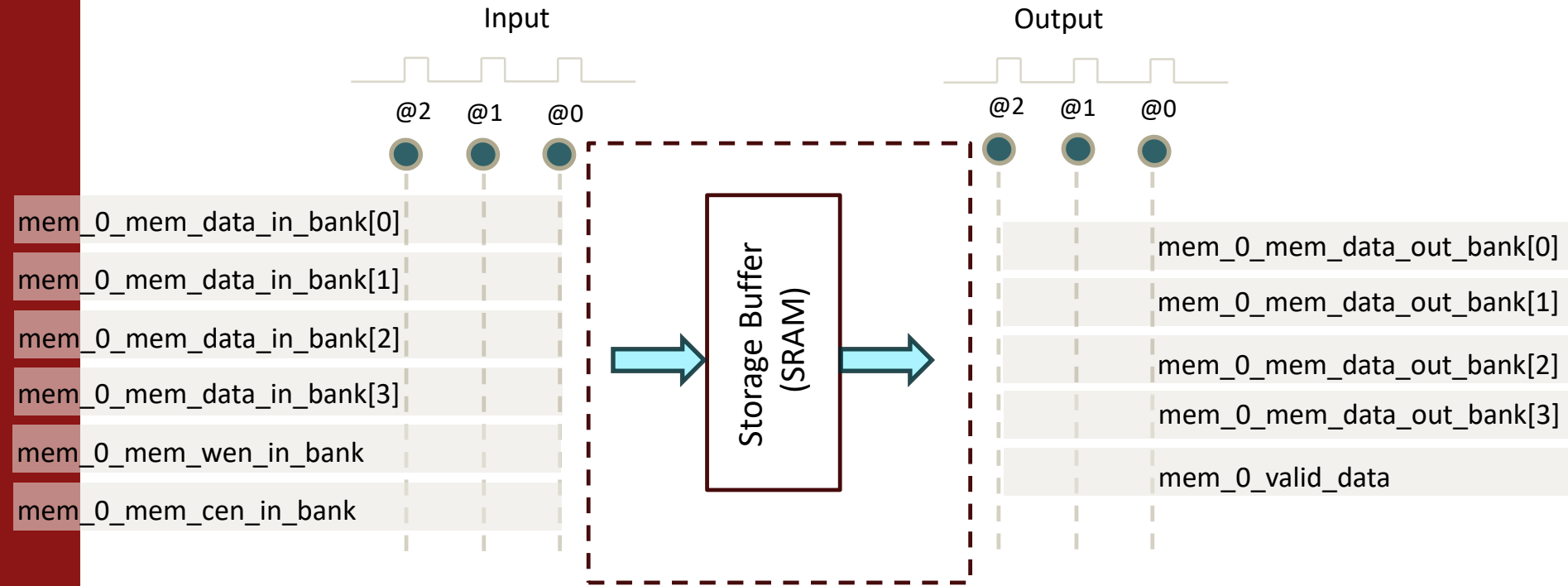
Intel Xeon E5-2620 v4,
CPU 1.6GHz, 16GB.

Stanford University

Can we utilize more information from users/designers?

Memory Tile in Pono





Internal sequence obtained from the polyhedral analysis.

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

```

Configuration Constraints

```

// strg_ub_pre_fetch_0_input_latency = 4
// strg_ub_pre_fetch_1_input_latency = 4
// chain_idx_input = 0
// enable_chain_input = 0
// enable_chain_output = 0
// 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@strg_ub_agg_in_0_in_period SOLVE
// strg_ub_agg_in_0_out_sched@strg_ub_agg_in_0_out_period SOLVE
// strg_ub_agg_in_1_in_sched@strg_ub_agg_in_1_in_period SOLVE

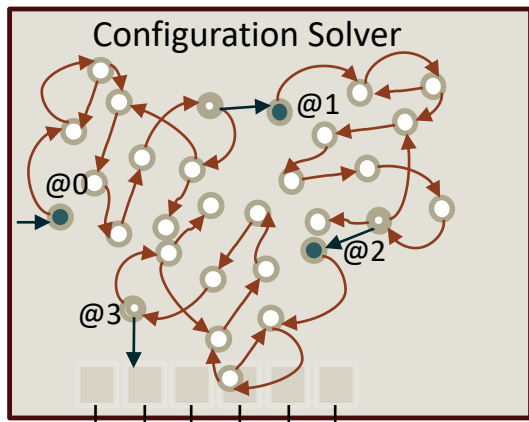
```

SRAM Sequence

```

1 mem_0_mem_data_in_bank mem_0_mem_en_in_bank mem_0_mem_data_out_bank mem_0_valid_data
2 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
3 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
4 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
5 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
6 [[0], [1], [1], [1], 1, 1, [0], [0], [0], [0], 0
7 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
8 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
9 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
10 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
11 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
12 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
13 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
14 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
15 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
16 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
17 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
18 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
19 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
20 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
21 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
22 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
23 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
24 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
25 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
26 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
27 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
28 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
29 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
30 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
31 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
32 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
33 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
34 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
35 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
36 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
37 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
38 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
39 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
40 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
41 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
42 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
43 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
44 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
45 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
46 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
47 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
48 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
49 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
50 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
51 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
52 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
53 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
54 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
55 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
56 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
57 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
58 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
59 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
60 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
61 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
62 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
63 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
64 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
65 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
66 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
67 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
68 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
69 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
70 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
71 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
72 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
73 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
74 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
75 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
76 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
77 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
78 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
79 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
80 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
81 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
82 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
83 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
84 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
85 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
86 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
87 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
88 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
89 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
90 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
91 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
92 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
93 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
94 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
95 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
96 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
97 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
98 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
99 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0
100 [[0], [0], [0], [0], 0, 0, [0], [0], [0], [0], 0

```



Memory Tile Sequence

```

1 data_in, wen_in, ren_in, data_out, valid_out
2 [[0], [0], 1, 0, [0], [0], 0
3 [[1], [0], 1, 0, [0], [0], 0
4 [[2], [0], 1, 0, [0], [0], 0
5 [[3], [0], 1, 0, [0], [0], 0
6 [[4], [0], 1, 0, [0], [0], 0
7 [[5], [0], 1, 0, [0], [0], 0
8 [[6], [0], 1, 0, [0], [0], 0
9 [[7], [0], 1, 0, [0], [0], 0
10 [[8], [0], 1, 0, [0], [0], 0
11 [[9], [0], 1, 0, [0], [0], 0
12 [[10], [0], 1, 0, [0], [0], 0
13 [[11], [0], 1, 0, [0], [0], 0
14 [[12], [0], 1, 0, [0], [0], 0
15 [[13], [0], 1, 0, [0], [0], 0
16 [[14], [0], 1, 0, [0], [0], 0
17 [[15], [0], 1, 0, [0], [0], 0
18 [[16], [0], 1, 0, [0], [0], 0
19 [[17], [0], 1, 0, [0], [0], 0
20 [[18], [0], 1, 0, [0], [0], 0
21 [[19], [0], 1, 0, [0], [0], 0
22 [[20], [0], 1, 0, [0], [0], 0
23 [[21], [0], 1, 0, [0], [0], 0
24 [[22], [0], 1, 0, [0], [0], 0
25 [[23], [0], 1, 0, [0], [0], 0
26 [[24], [0], 1, 0, [0], [0], 0
27 [[25], [0], 1, 0, [0], [0], 0
28 [[26], [0], 1, 0, [0], [0], 0
29 [[27], [0], 1, 0, [0], [0], 0
30 [[28], [0], 1, 3, [0], [0], 0
31 [[29], [0], 1, 3, [0], [0], 0
32 [[30], [0], 1, 3, [0], [0], 0
33 [[31], [0], 1, 3, [0], [0], 0
34 [[32], [0], 1, 3, [0], [0], 0
35 [[33], [0], 1, 3, [0], [0], 0
36 [[34], [0], 1, 3, [0], [0], 0
37 [[35], [0], 1, 3, [0], [0], 0
38 [[36], [0], 1, 3, [0], [0], 0
39 [[37], [0], 1, 3, [0], [0], 0
40 [[38], [0], 1, 3, [0], [0], 0
41 [[39], [0], 1, 3, [0], [0], 0
42 [[40], [0], 1, 3, [0], [0], 0
43 [[41], [0], 1, 3, [0], [0], 0
44 [[42], [0], 1, 3, [0], [0], 0
45 [[43], [0], 1, 3, [0], [0], 0
46 [[44], [0], 1, 3, [0], [0], 0
47 [[45], [0], 1, 3, [0], [0], 0
48 [[46], [0], 1, 3, [0], [0], 0
49 [[47], [0], 1, 3, [0], [0], 0
50 [[48], [0], 1, 3, [0], [0], 0
51 [[49], [0], 1, 3, [0], [0], 0
52 [[50], [0], 1, 3, [0], [0], 0
53 [[51], [0], 1, 3, [0], [0], 0
54 [[52], [0], 1, 3, [0], [0], 0
55 [[53], [0], 1, 3, [0], [0], 0
56 [[54], [0], 1, 3, [0], [0], 0
57 [[55], [0], 1, 3, [0], [0], 0
58 [[56], [0], 1, 3, [0], [0], 0
59 [[57], [0], 1, 3, [0], [0], 0
60 [[58], [0], 1, 3, [0], [0], 0
61 [[59], [0], 1, 3, [0], [0], 0
62 [[60], [0], 1, 3, [0], [0], 0
63 [[61], [0], 1, 3, [0], [0], 0
64 [[62], [0], 1, 3, [0], [0], 0
65 [[63], [0], 1, 3, [0], [0], 0
66 [[64], [0], 1, 3, [0], [0], 0
67 [[65], [0], 1, 3, [0], [0], 0
68 [[66], [0], 1, 3, [0], [0], 0
69 [[67], [0], 1, 3, [0], [0], 0
70 [[68], [0], 1, 3, [0], [0], 0
71 [[69], [0], 1, 3, [0], [0], 0
72 [[70], [0], 1, 3, [0], [0], 0
73 [[71], [0], 1, 3, [0], [0], 0
74 [[72], [0], 1, 3, [0], [0], 0
75 [[73], [0], 1, 3, [0], [0], 0
76 [[74], [0], 1, 3, [0], [0], 0
77 [[75], [0], 1, 3, [0], [0], 0
78 [[76], [0], 1, 3, [0], [0], 0
79 [[77], [0], 1, 3, [0], [0], 0
80 [[78], [0], 1, 3, [0], [0], 0
81 [[79], [0], 1, 3, [0], [0], 0
82 [[80], [0], 1, 3, [0], [0], 0
83 [[81], [0], 1, 3, [0], [0], 0
84 [[82], [0], 1, 3, [0], [0], 0
85 [[83], [0], 1, 3, [0], [0], 0
86 [[84], [0], 1, 3, [0], [0], 0
87 [[85], [0], 1, 3, [0], [0], 0
88 [[86], [0], 1, 3, [0], [0], 0
89 [[87], [0], 1, 3, [0], [0], 0
90 [[88], [0], 1, 3, [0], [0], 0
91 [[89], [0], 1, 3, [0], [0], 0
92 [[90], [0], 1, 3, [0], [0], 0
93 [[91], [0], 1, 3, [0], [0], 0
94 [[92], [0], 1, 3, [0], [0], 0
95 [[93], [0], 1, 3, [0], [0], 0
96 [[94], [0], 1, 3, [0], [0], 0
97 [[95], [0], 1, 3, [0], [0], 0
98 [[96], [0], 1, 3, [0], [0], 0
99 [[97], [0], 1, 3, [0], [0], 0
100 [[98], [0], 1, 3, [0], [0], 0

```

Tile Interface/ Configuration Register List

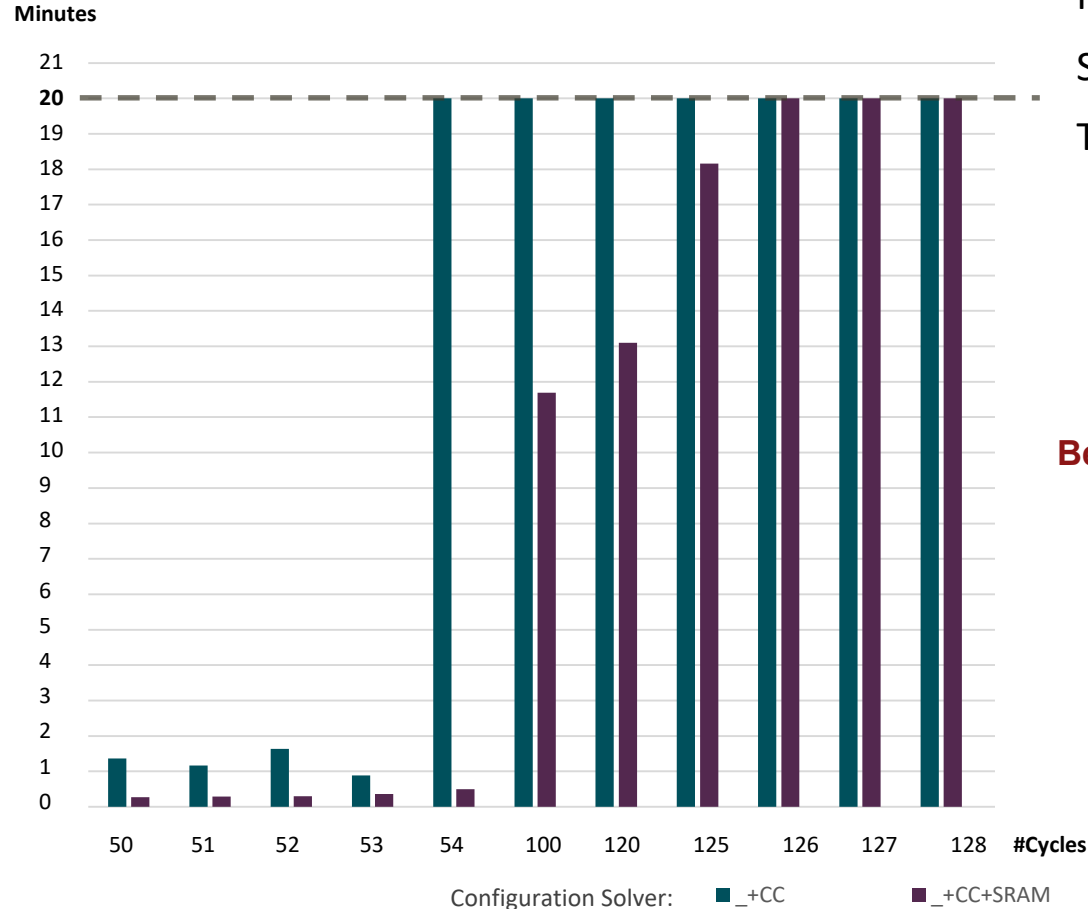
```

1 input logic [1:0] [0] addr_in, SET0
2 output logic [1:0] [0] chain_idx_input, SET0
3 output logic [1:0] [0] chain_idx_output, SET0
4 input logic [1:0] chain_data_in, SOLVE
5 output logic [1:0] chain_data_out, SOLVE
6 input logic [1:0] chain_valid_in, X
7 output logic [1:0] chain_valid_out, X
8 input logic clk, CLK
9 input logic clk_en, SET1
10 output logic [7:0] config_addr_in, SET0
11 input logic [15:0] config_data_in, X
12 output logic [1:0] [31:0] config_data_out, X
13 input logic [1:0] config_en, SET0
14 input logic config_read, SET0
15 input logic config_write, SET0
16 output logic [1:0] [15:0] data_in, SEQUENCE
17 output logic [1:0] [15:0] data_out, SEQUENCE
18 output logic empty, X
19 input logic enable_chain_input, SOLVE
20 input logic enable_chain_output, SOLVE
21 input logic [15:0] rifo_ctrl_rifo_depth, SET0
22 input logic flush, X
23 output logic full, X
24 input logic [1:0] mode, SEQUENCE
25 input logic [1:0] ren_in, SEQUENCE
26 input logic rst_n, RSTN
27 output logic sram_ready_out, X
28 input logic [4:0] strg_ub_agg_align_0_line_length, SOLVE
29 input logic [6:0] strg_ub_agg_align_1_line_length, SOLVE
30 input logic [3:0] strg_ub_agg_in_0_in_period, SOLVE
31 input logic [15:0] [1:0] strg_ub_agg_in_0_in_sched, SOLVE
32 input logic [3:0] strg_ub_agg_in_0_out_period, SOLVE
33 input logic [15:0] [1:0] strg_ub_agg_in_0_out_sched, SOLVE
34 input logic [3:0] strg_ub_agg_in_1_in_period, SOLVE

```



How far can we scale?



Memory Tile reads start @54

SRAM reads start @51

Timeout = 20 min

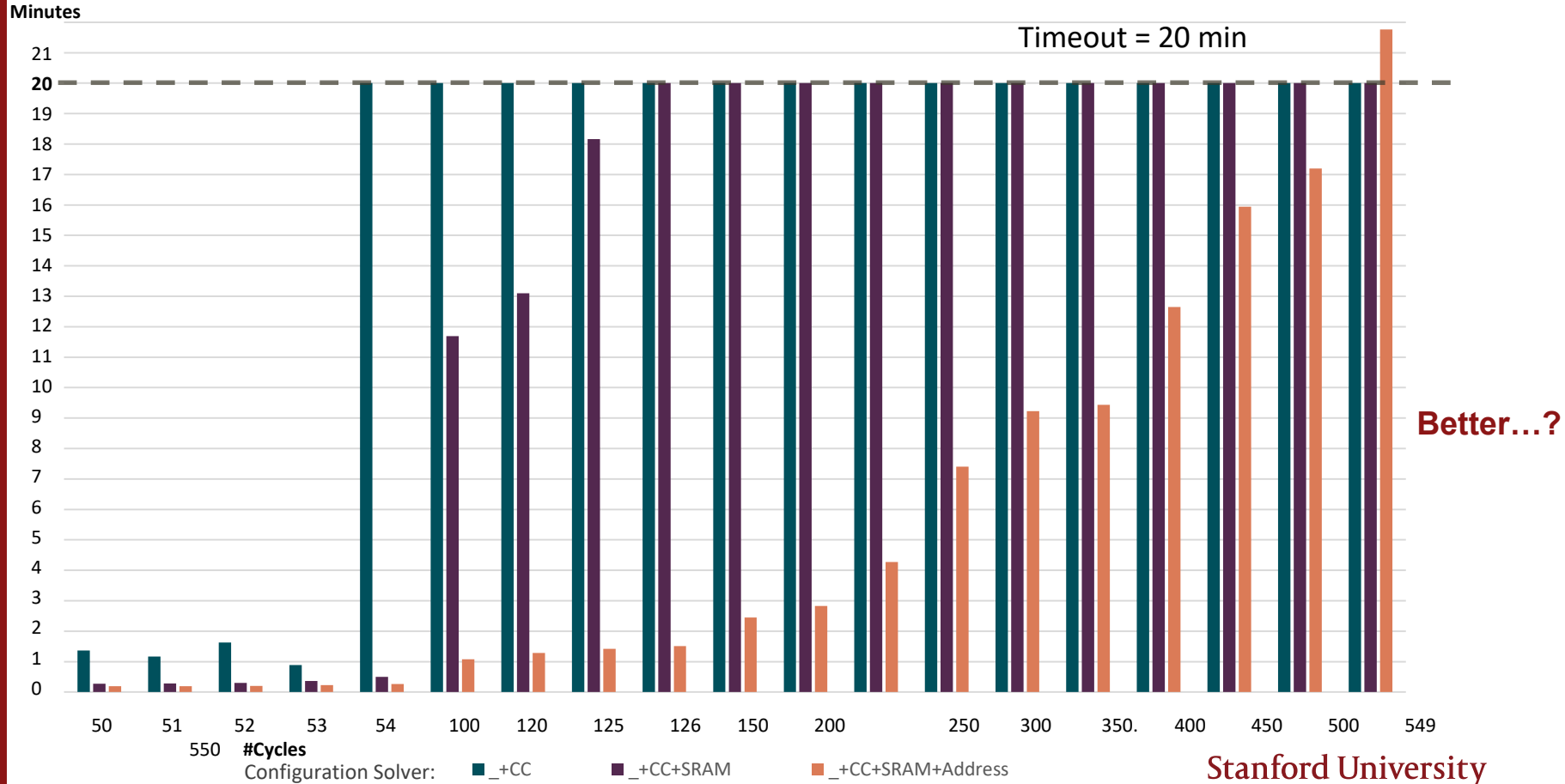
Better...?

How far can we scale?

Memory Tile reads start @54

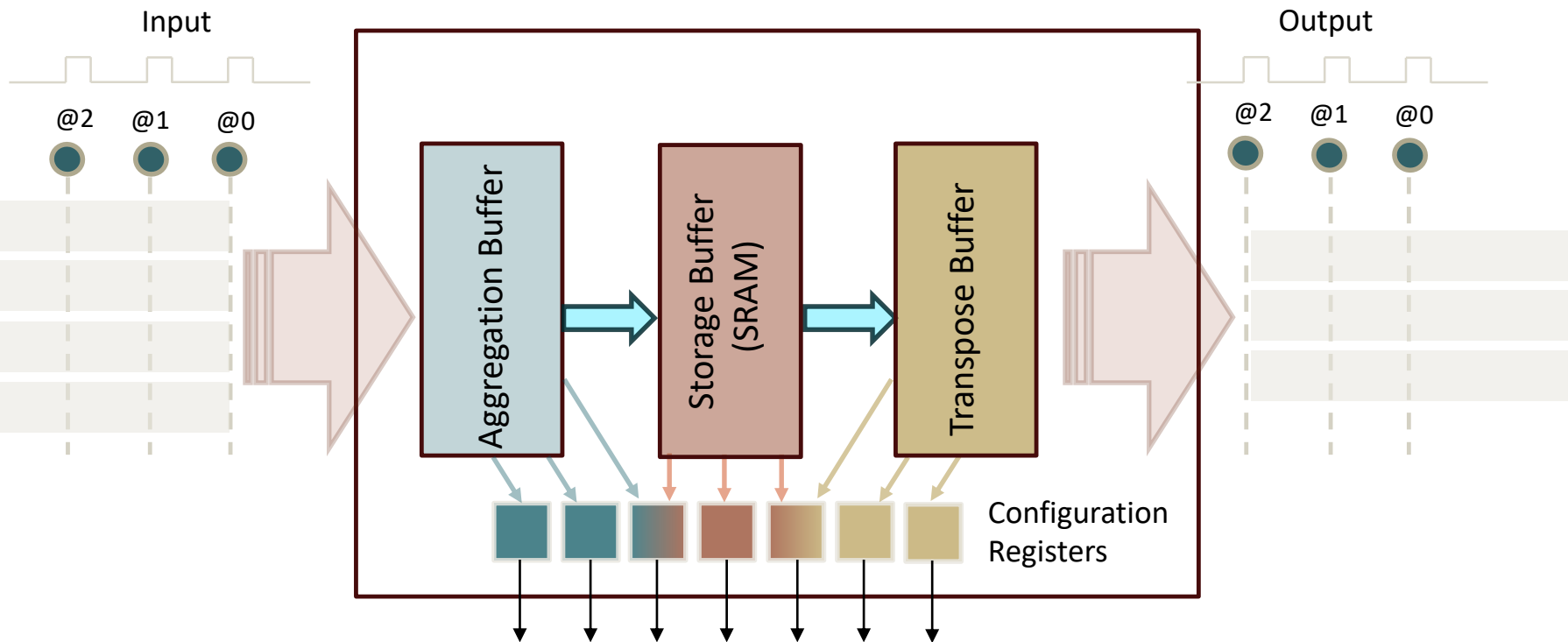
SRAM reads start @51

Timeout = 20 min



New Modular Design

Memory Tile in Pono



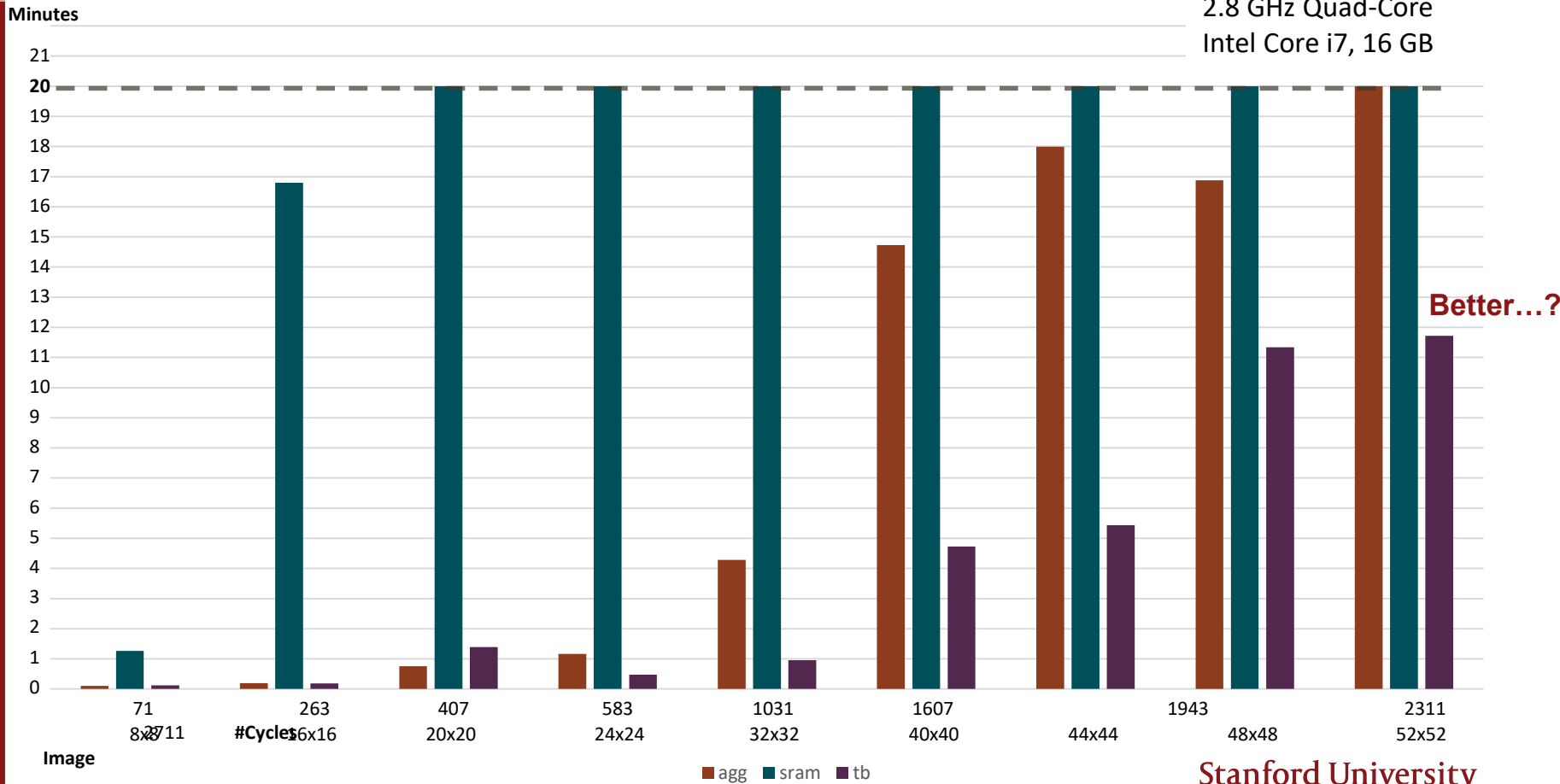
How far can we scale with a modular approach?

Identity stream

Memory Tile reads start @8

2.8 GHz Quad-Core

Intel Core i7, 16 GB



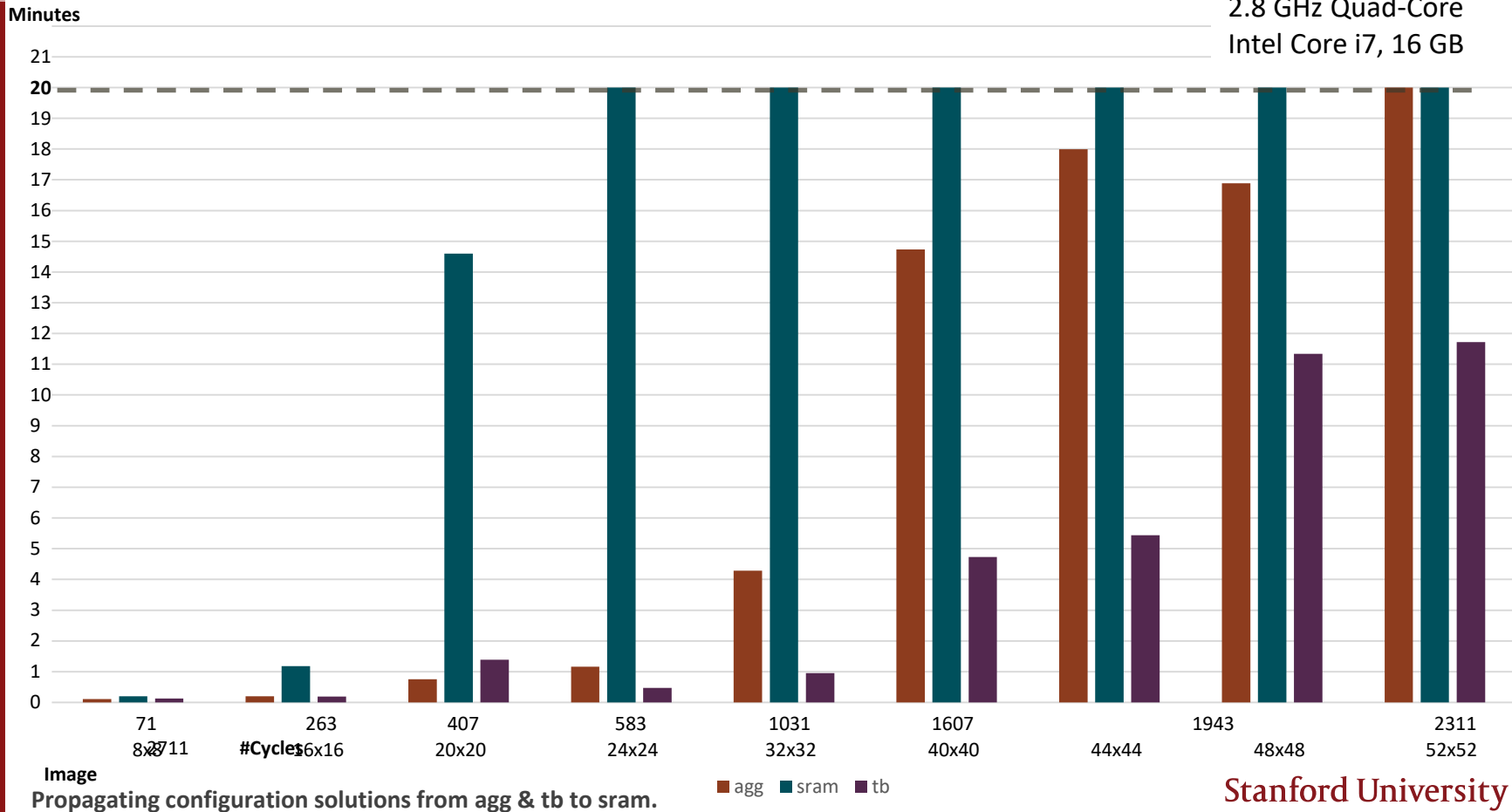
How far can we scale with a modular approach?

Identity stream

Memory Tile reads start @8

2.8 GHz Quad-Core

Intel Core i7, 16 GB



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.