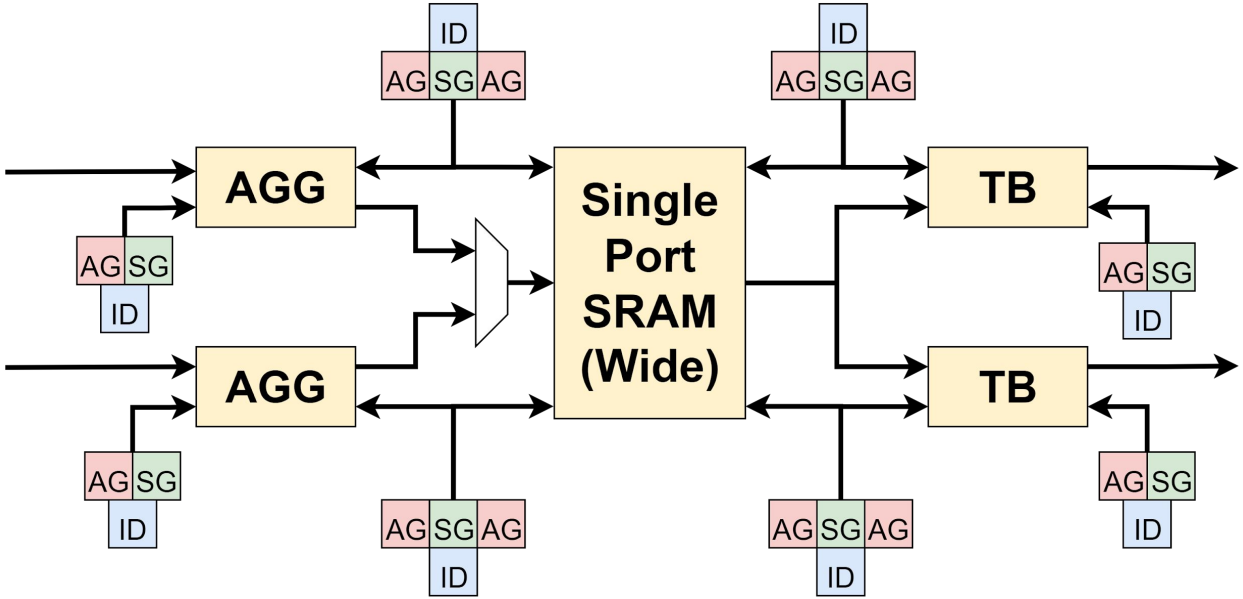
Onyx Memory Tile Area Optimizations and Ready-valid Discussion

Jake Ke
Jun. 8, 2022

Agenda

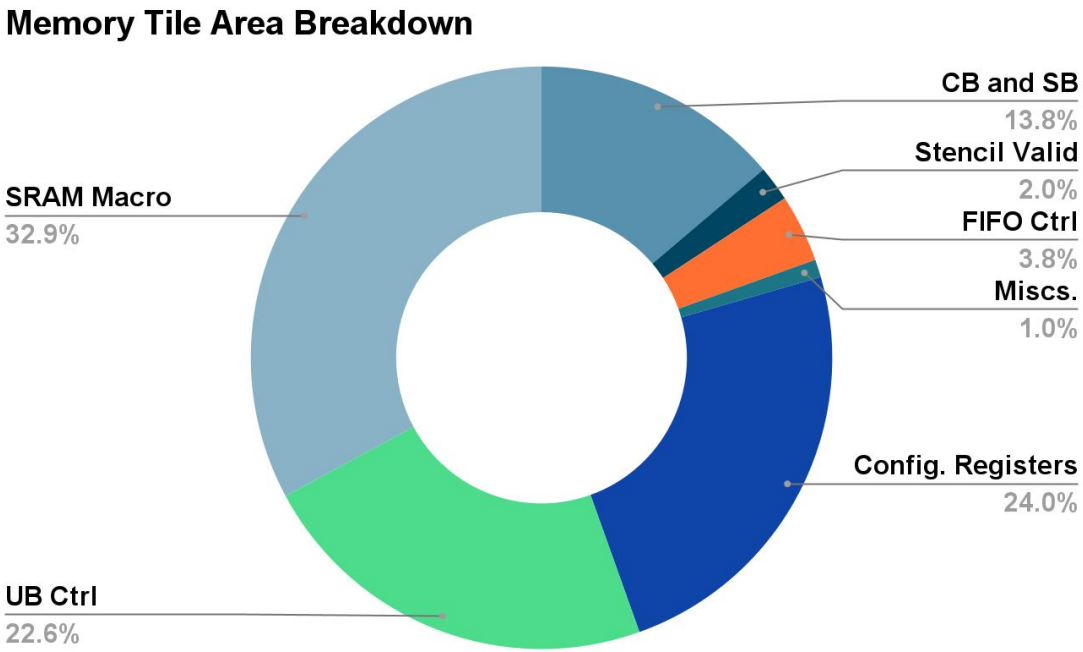
- Memory Tile Area optimizations (completed)
- Ready-valid discussion (on-going)

Amber Memory Tile Unified Buffer



- 2 read and 2 write ports with a single port SRAM
- Independent top and bottom datapath
- Controllers:
  - In2Agg
  - Agg2Sram
  - Sram2Tb
  - Tb2Out

Baseline Area Breakdown



- GF14 synthesis
- SRAM macro: **32.9%**
- Unified buffer (UB) controller and configuration registers: **46.6%**

Area Optimizations

- Shared controller for update operations
  - In2Agg
  - Agg2Sram
- Aggregator depth
- Configuration register bit width
- ~~Schedule generator counting relative strides~~
- ~~SRAM and TB outer loop factorization~~

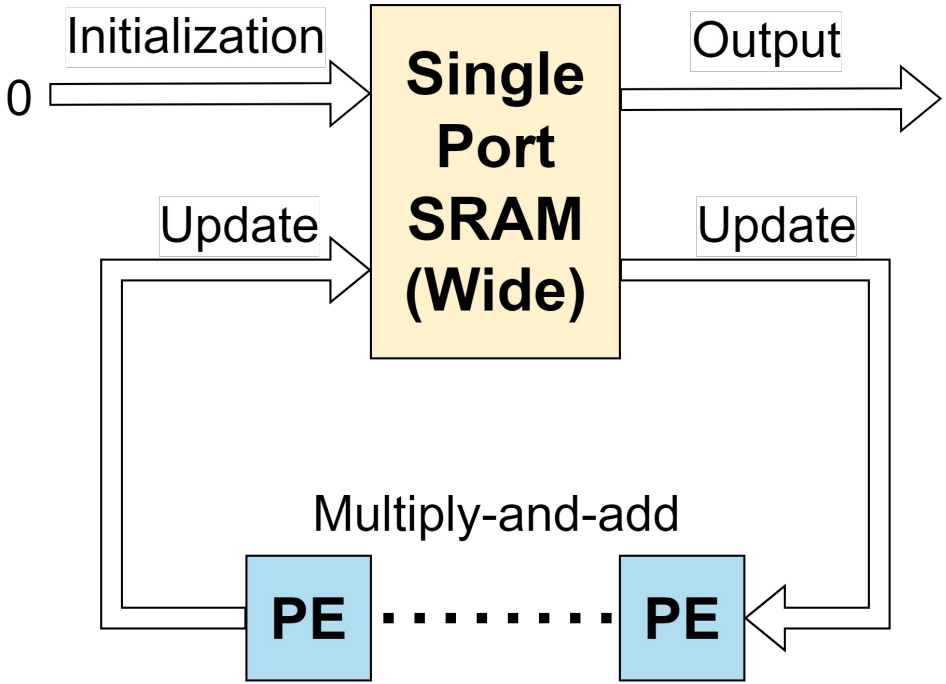
Area Optimizations

- **Shared controller for update operations**
  - **In2Agg**
  - **Agg2Sram**
- Aggregator depth
- Configuration register bit width
- ~~Schedule generator counting relative strides~~
- ~~SRAM and TB outer loop factorization~~

Why Do We Have Aggregators

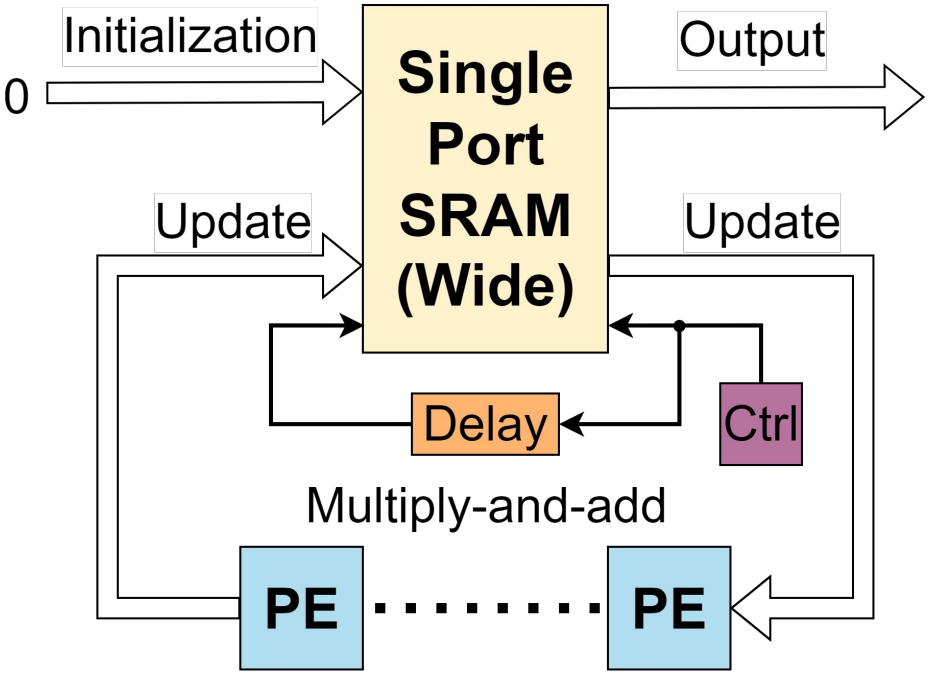
- The reason for AGG is that we have a wide SRAM and we group inputs into wide words
- The AGG read SG could be as simple as a counter which counts the number of inputs to 4
    - Whenever there are 4 words in the AGG, we send that as a single wide-word to the SRAM
- The AG can also be a counter as the input data are stored contiguously
- But there are discontiguous data accesses in the Update Operations in ResNet

Update Operations in ResNet



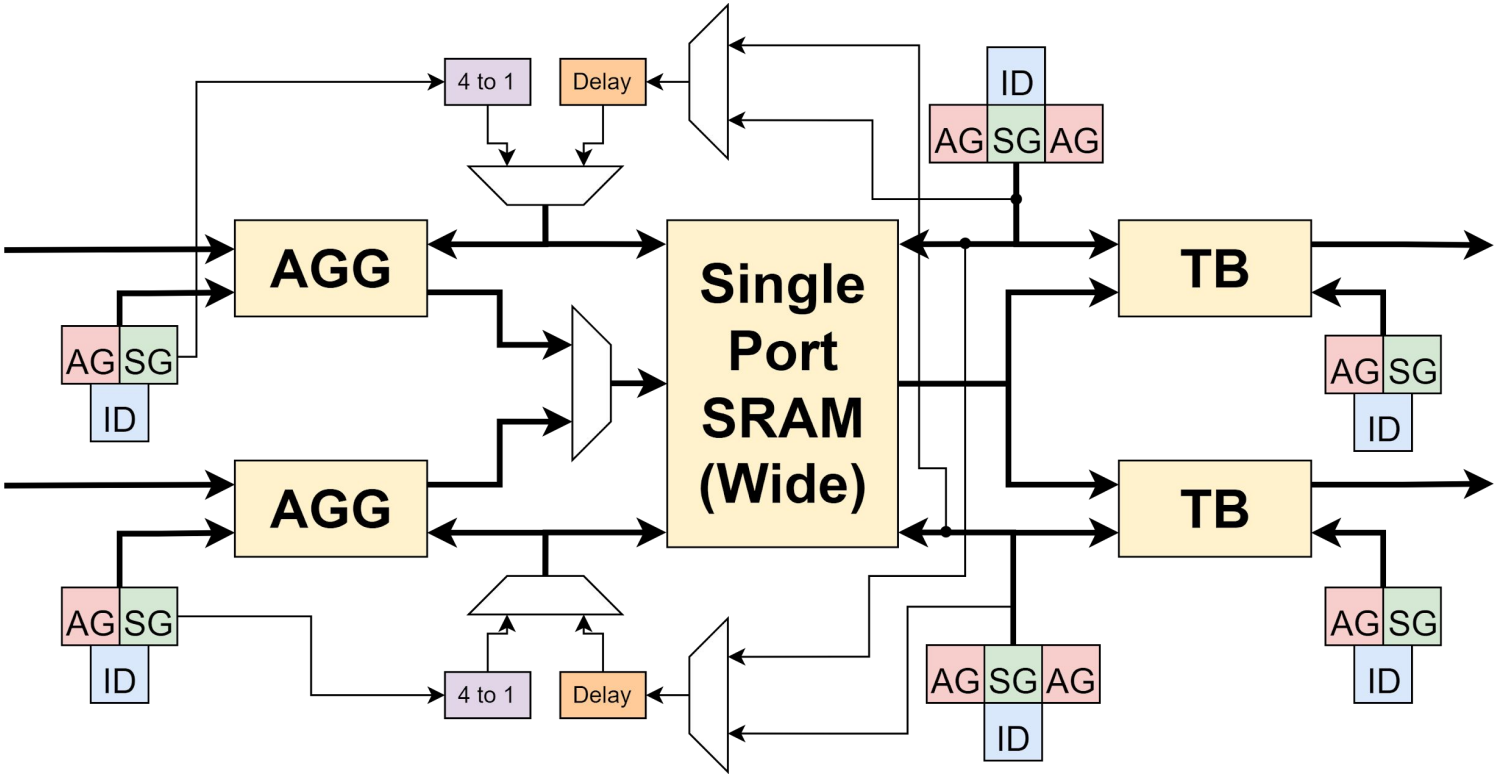
- Initialization have contiguous data access
- But Update have discontiguous data access as we will iteratively accumulate the products of ifmap and weights

Shared Controller for Update Operations

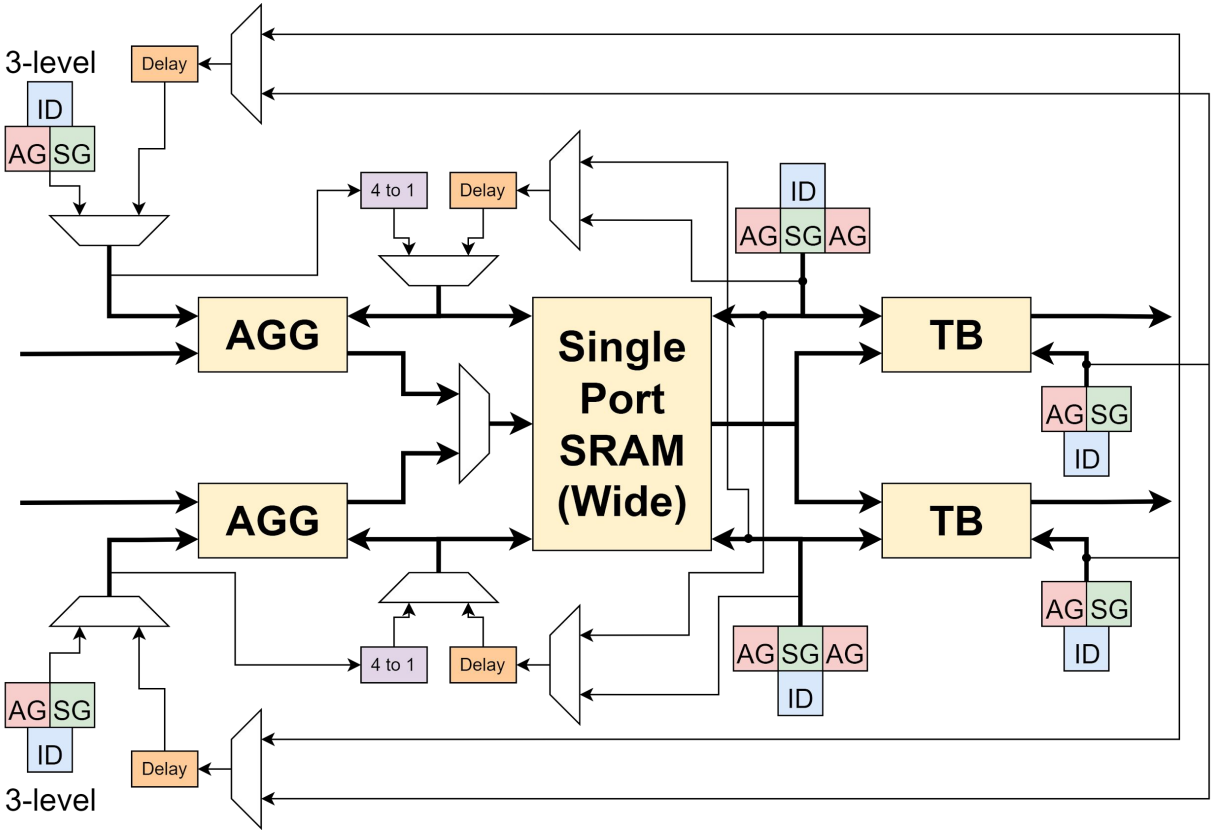


- Although the write back address are discontiguous, it is the same as the read address but with a delay
- We can just use a small FIFO to store and delay the read address and enable

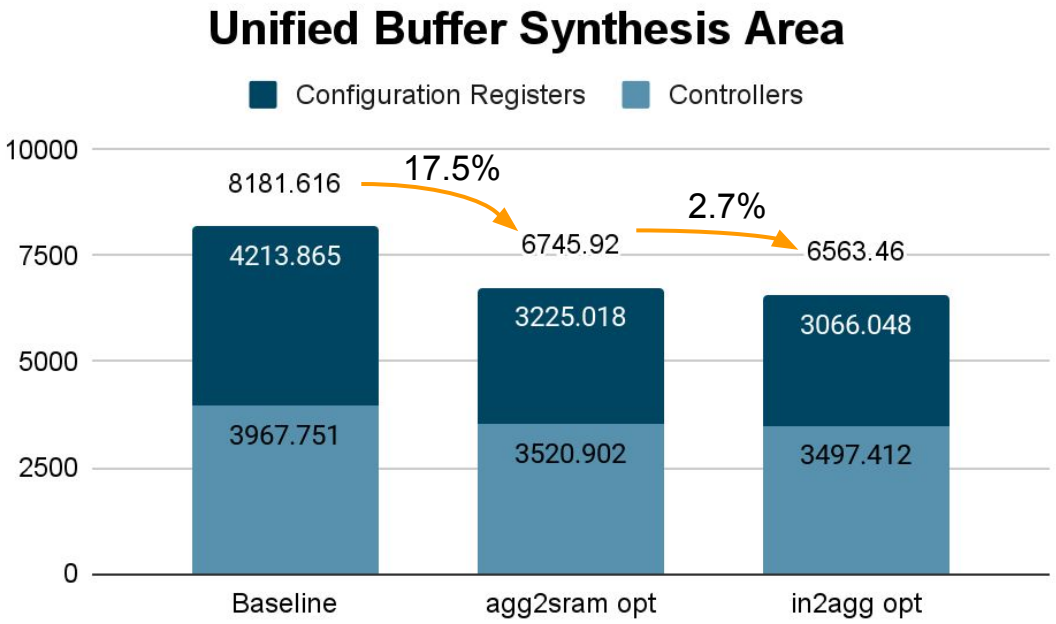
Agg2Sram


4 to 1
Delay
ID
AG SG AG
AGG
Single Port SRAM (Wide)
TB
AG SG
ID
AGG
TB
AG SG
ID
4 to 1
Delay
AG SG AG
ID

In2Agg


3-level
Delay
ID
AG
SG
4 to 1
Delay
ID
AG
SG
AG
AGG
Single Port SRAM (Wide)
TB
AG
SG
ID
AGG
TB
AG
SG
ID
4 to 1
Delay
AG
SG
AG
ID
AG
SG
ID
Delay
3-level

Shared Controller for Update Operations Area Change



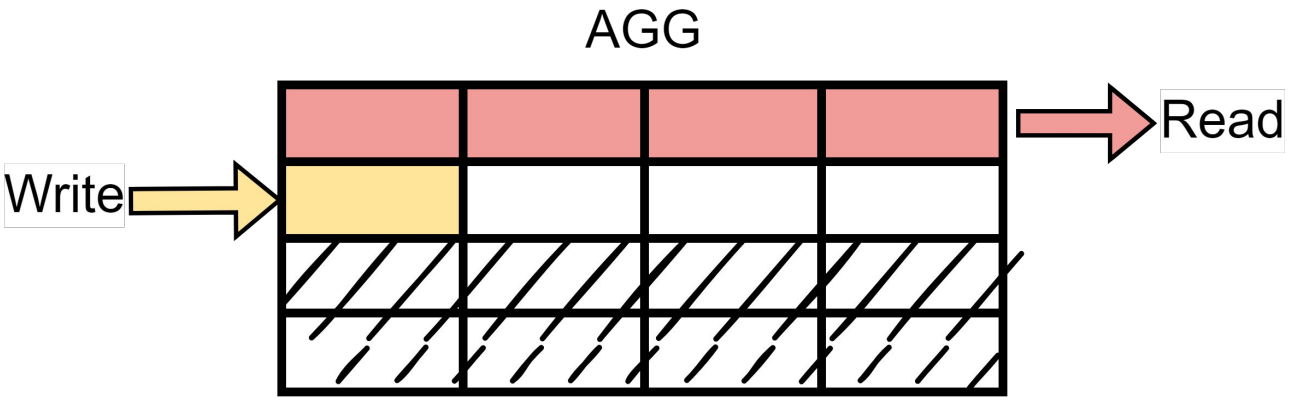
- Unified buffer (controller and config registers) total synthesis area reduction: **19.8%**
- Memory tile (including the SRAM, other controllers, CB, SB, etc.) total synthesis area reduction: **9.1%**

Area Optimizations

- Shared controller for update operations
  - In2Agg
  - Agg2Sram
- **Aggregator depth**
- Configuration register bit width

Aggregator Depth

- The AGG groups the inputs into wide words
- In theory, we could just use 1x4x16b registers
- Reduced from 4x4x16b registers to 2x4x16b registers



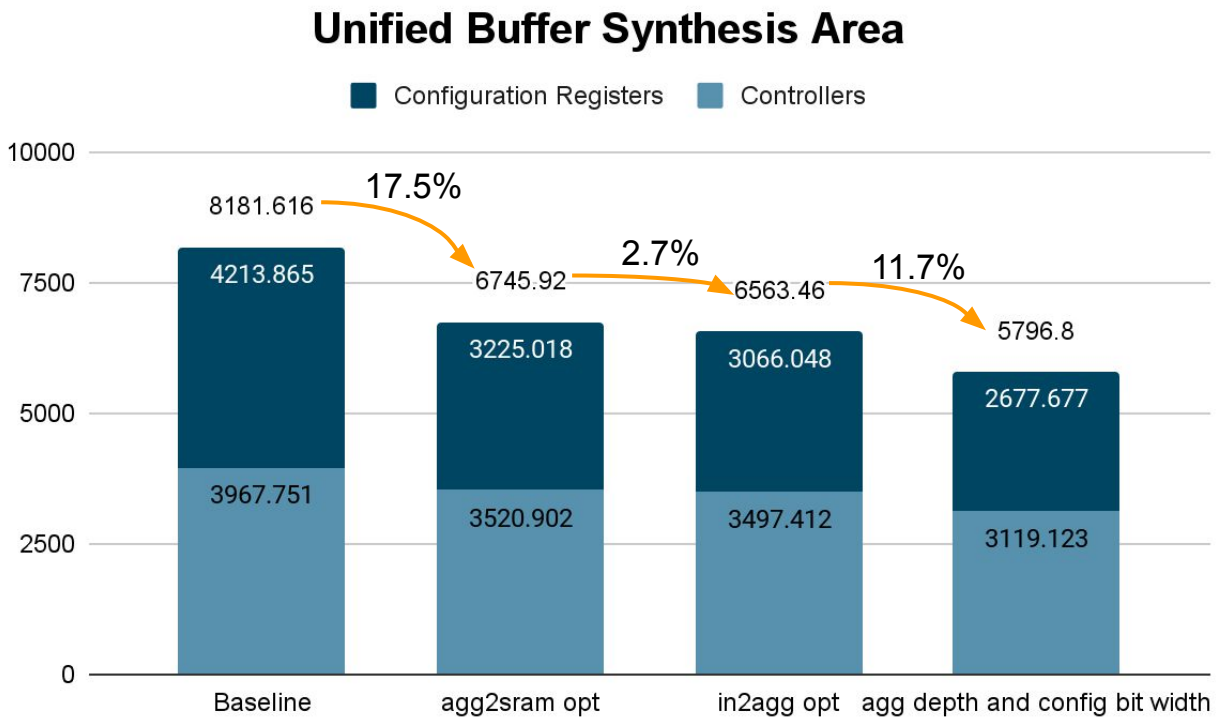
Area Optimizations

- Shared controller for update operations
  - In2Agg
  - Agg2Sram
- Aggregator depth
- **Configuration register bit width**

Configuration Register Bit Width

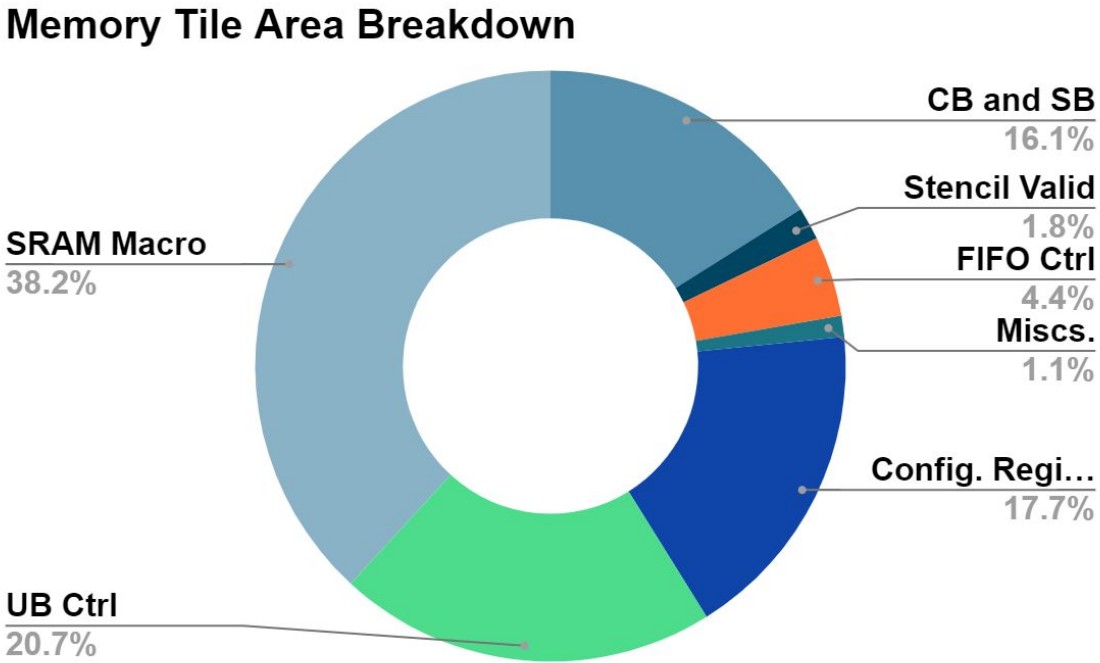
- 24% of the memory tile
- Extent
  - A survey of all regression and clockwork apps: 16 bits -> 10 bits (with some margin)
- ~~Data starting address~~
- Data strides
  - Possible but limited
- ~~Cycle starting address~~
- ~~Cycle strides~~
- ~~Dim~~

Incremental Area Changes of all Optimizations



- Total unified buffer synthesis area reduction: **29.1%**
- Total memory tile synthesis area reduction: **13.8%**
- Total memory tile signoff area reduction: **12.0%**

Area Breakdown After



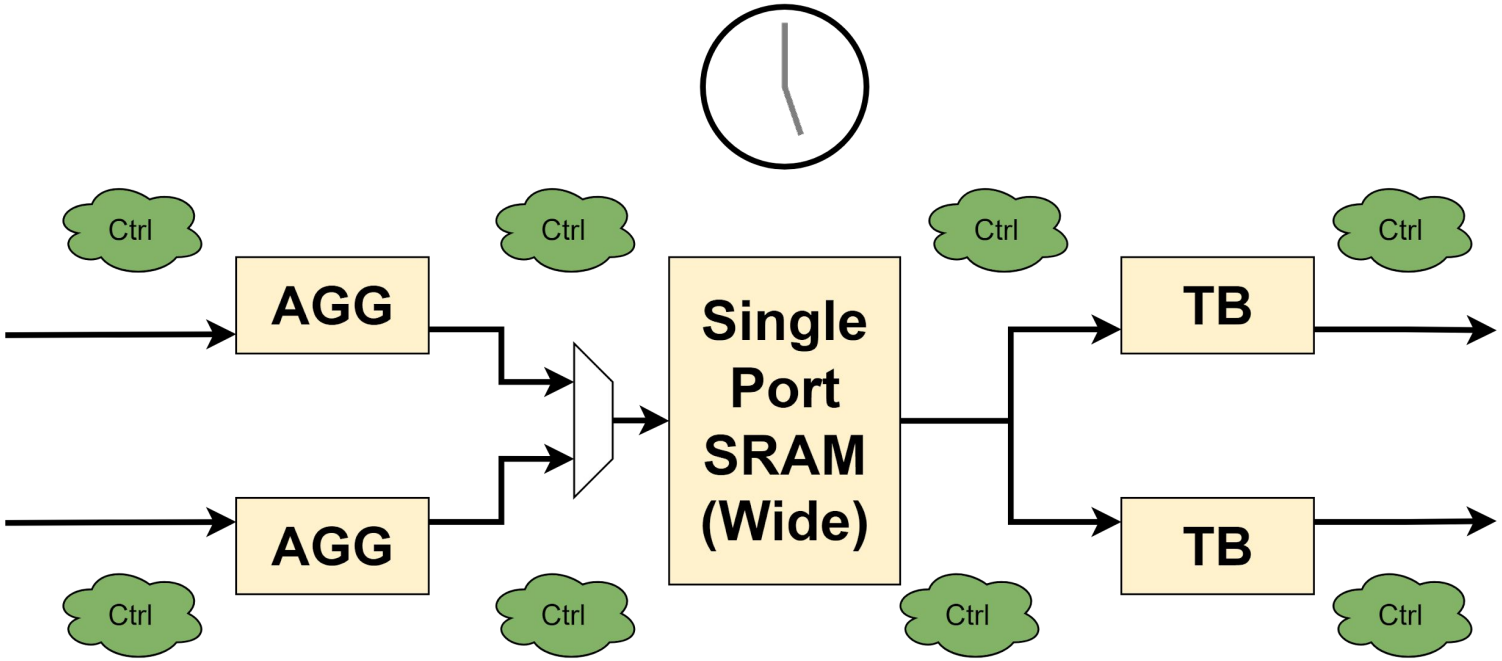
- SRAM macro: **38.2%**
- Unified buffer (UB) controller and configuration registers: **38.4%**
- New macro to UB ratio is **1:1**
- The macro to UB ratio was **1:1.4** (32.9:46.6)

Ready-valid Memory Tile

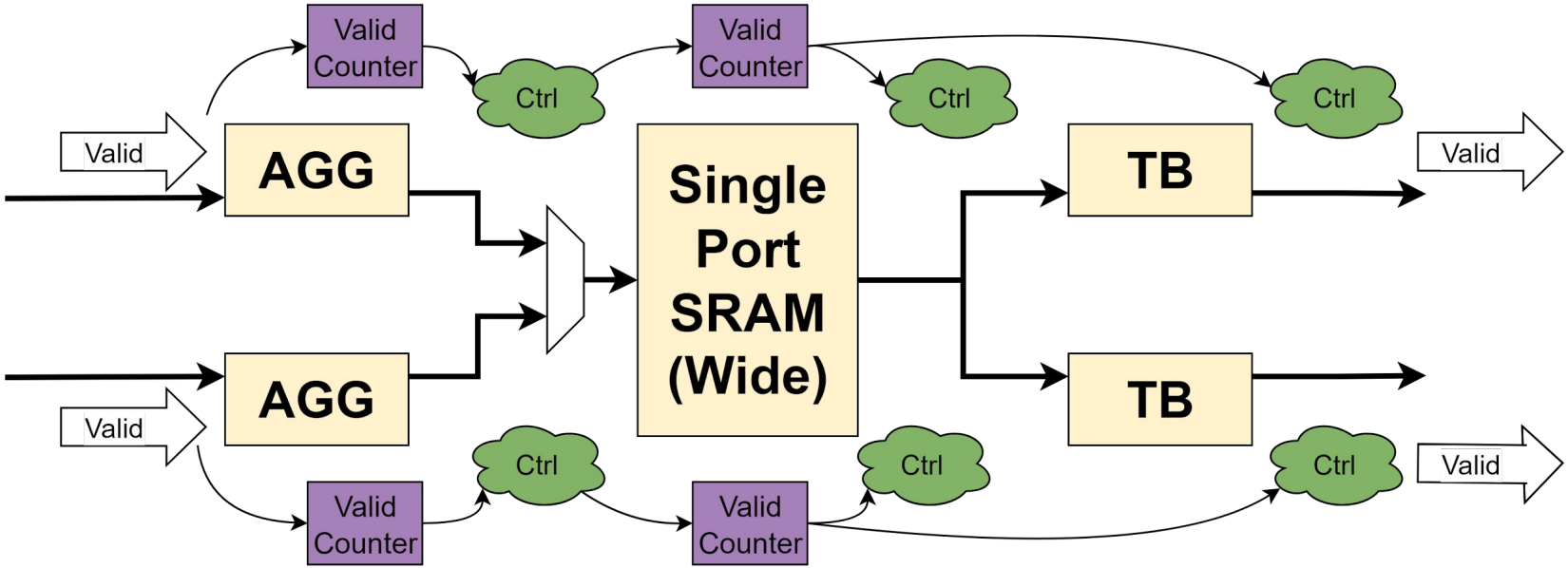
Motivation

- Have a consistent interface as the Sparsity Tiles and the new Pond
- Easier to integrate new tiles that use ready valid
- May improve power as we could have less activities during idle cycles
- Area change is likely unnoticable because the unified buffer occupies a smaller percentage of the memory tile
- No longer need the flush signal

Static Schedule Abstraction

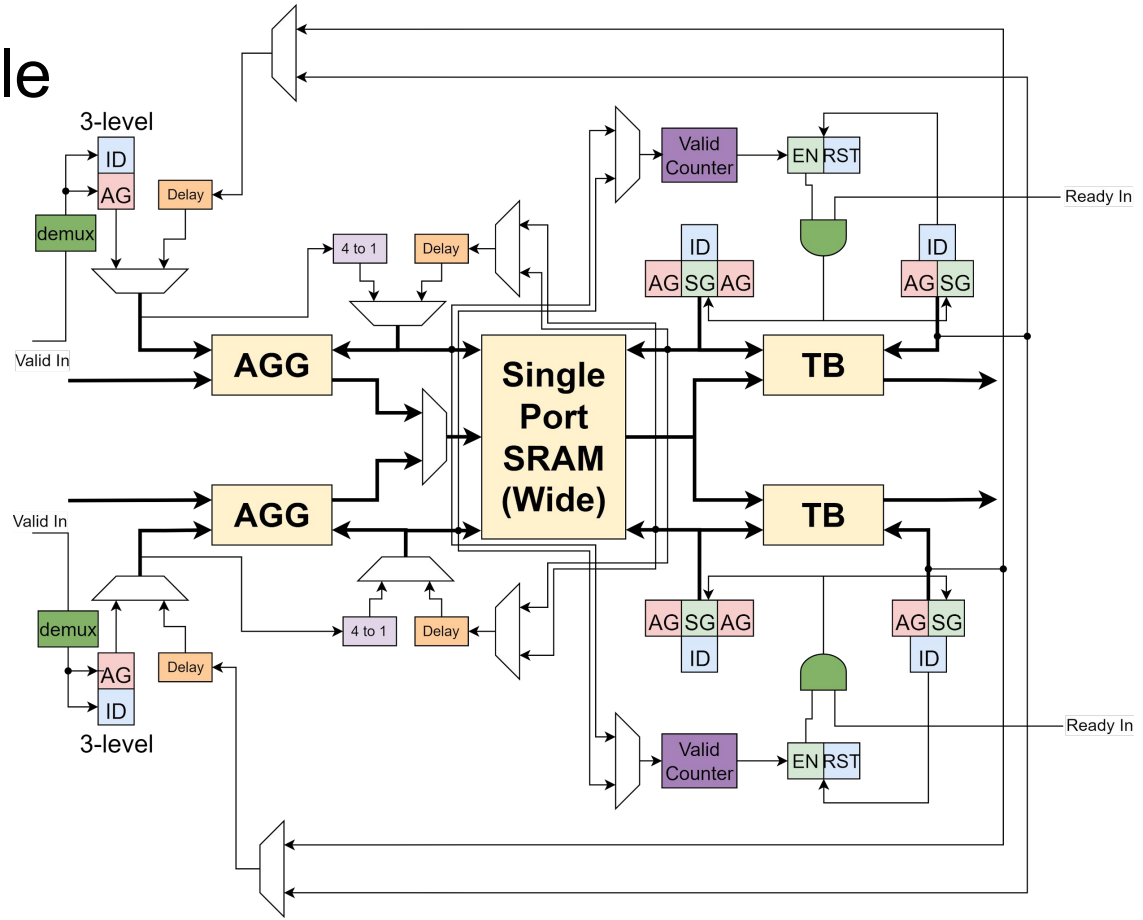


Using Ready Valid

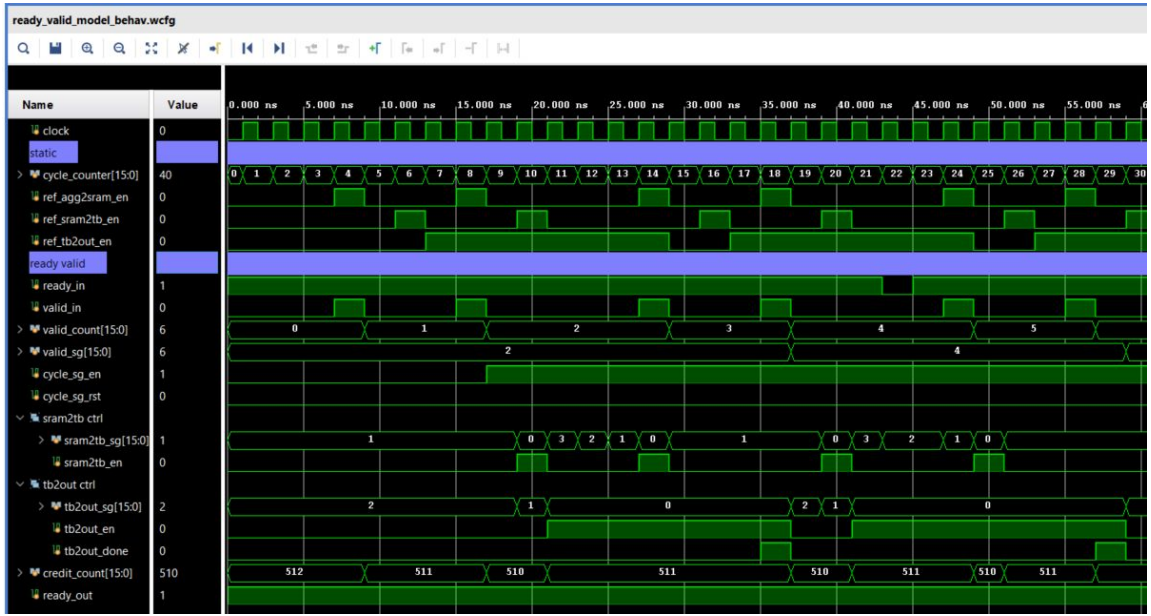


Ready-Valid Memory Tile

- Need a demux ctrl when the source is sending to multiple destinations
- Sram2tb and Tb2out ctrl are internally static, triggered by the valid counter
- Valid out is generated by Tb2Out valid-cycle ctrl
- Ready in enables the Sram2tb and Tb2out ctrl
- Ready out is asserted when there is space in the SRAM



App Example (Line Buffer)

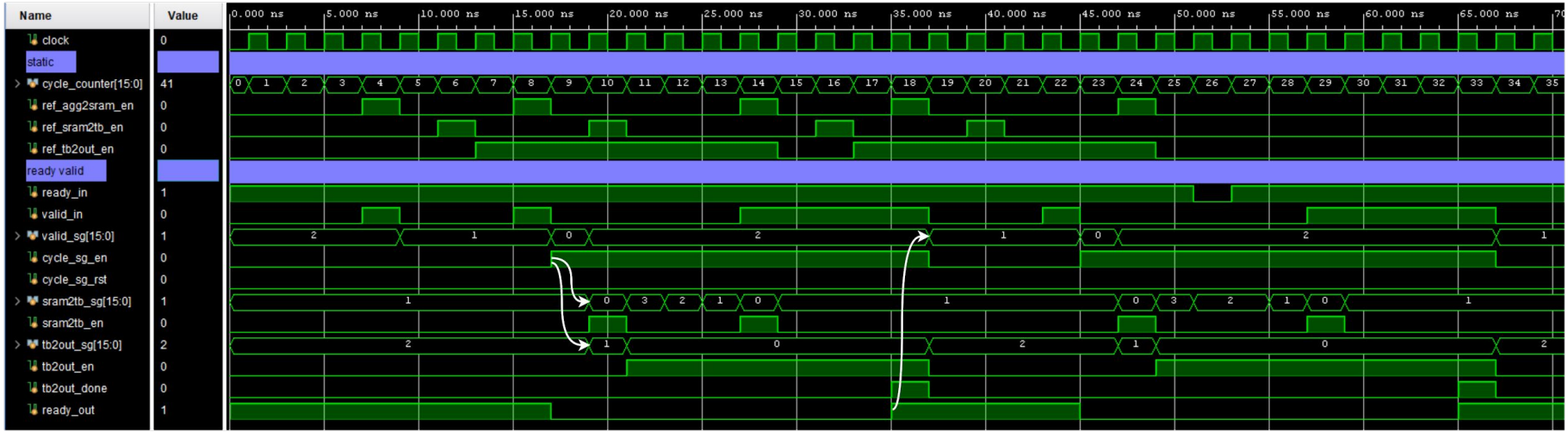


Static	Cycle start	Cycle stride	extent
agg2sram	4	4, 10	2, 60
sram2tb	6	4, 10	2, 60
tb2out	7	1, 10	8, 60

Ready-valid	Valid start (#v)	Valid strides (#v)	Valid extent (#v)	Cycle start (#c)	Cycle strides (#c)	Cycle extent (#c)
sram2tb	2	2	60	1	3	2
tb2out				2	0	8

Back-up

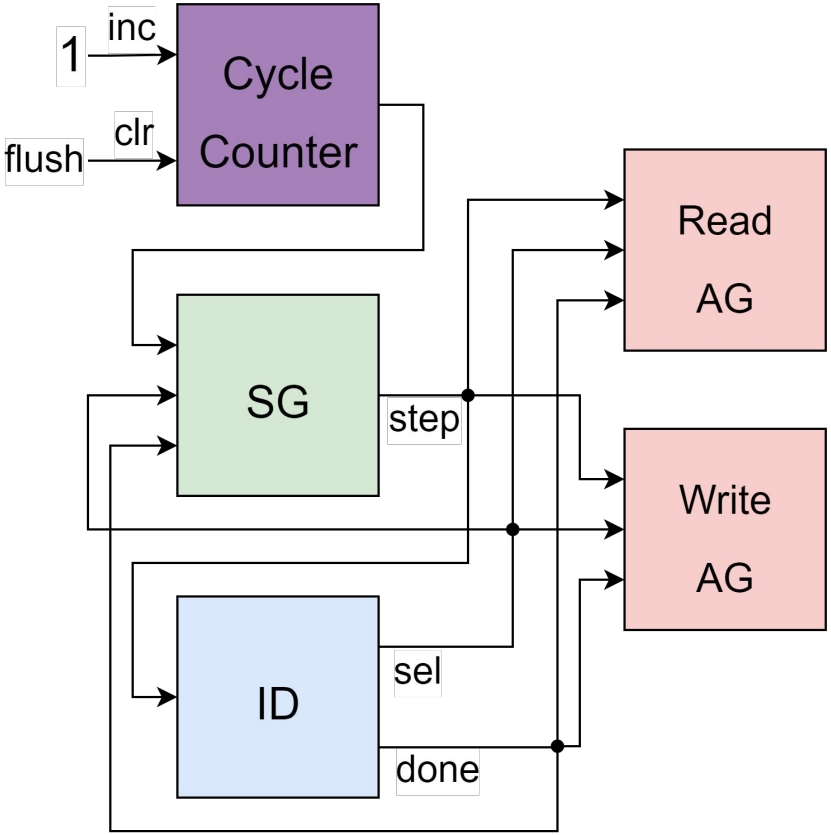
App Example (Line Buffer)



Static	Cycle start	Cycle stride	extent
in2agg	0	1, 10	8, 60
agg2sram	4	4, 10	2, 60
sram2tb	6	4, 10	2, 60
tb2out	7	1, 10	8, 60

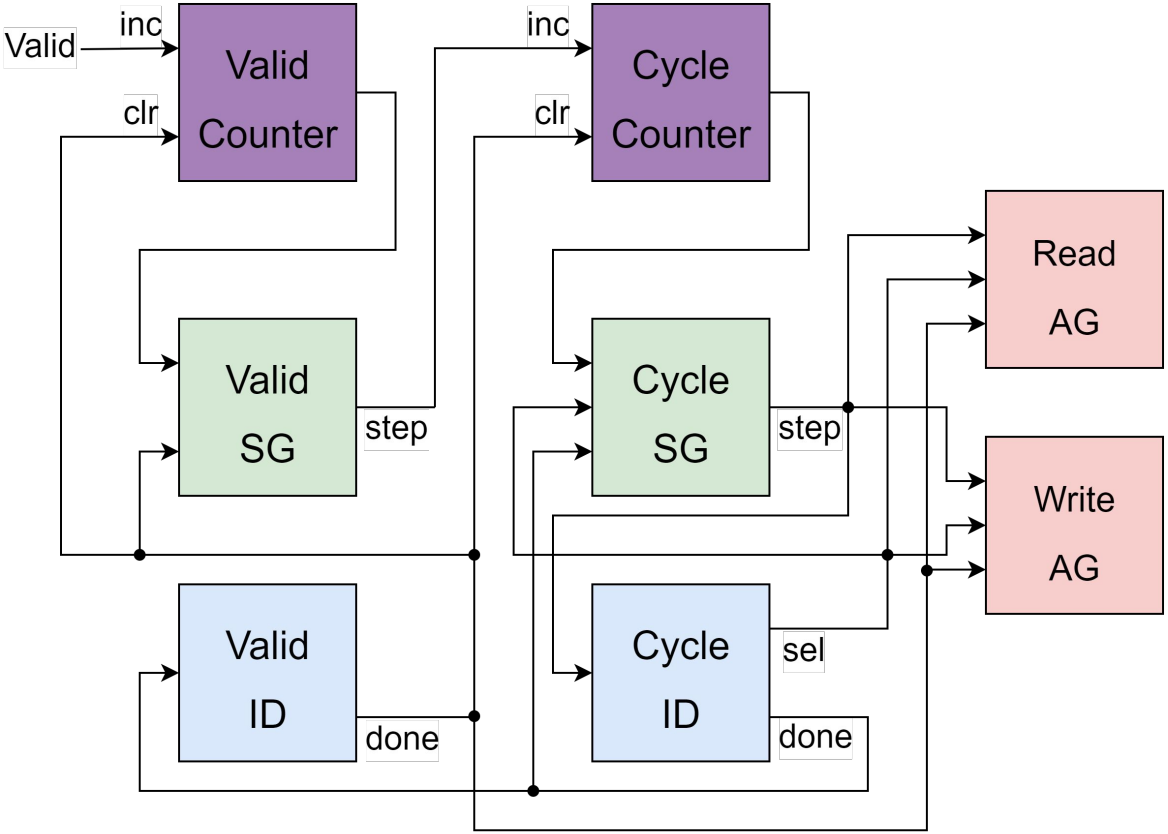
Ready-valid	Valid start (#v)	Valid strides (#v)	Valid extent (#v)	Cycle start (#c)	Cycle strides (#c)	Cycle extent (#c)
sram2tb	2	2	60	1	3	2
tb2out				2	0	8

Static Cycle Controller



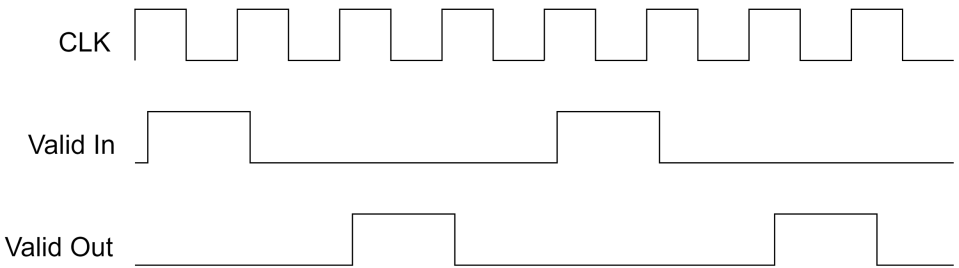
- Step: read/write enable
- Sel: selects the stride of the current iteration domain
- Done: resets SG and AG
- Flush: resets the cycle counter

Valid-cycle Controller



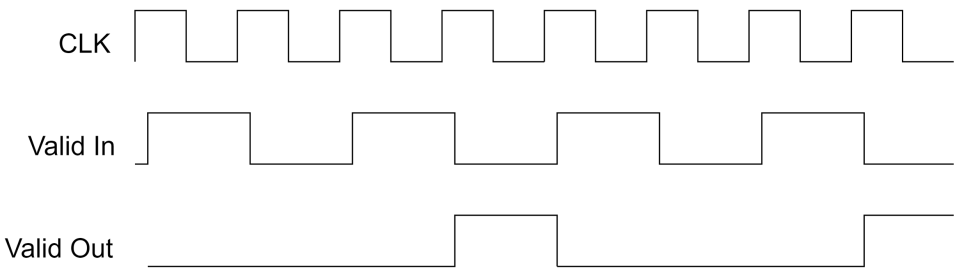
- Valid Counter and SG controls when to enable cycle SG and ID by incrementing the cycle counter
- Valid ID resets AG
- Other connections of cycle SG and ID are the same

App Example: Delay



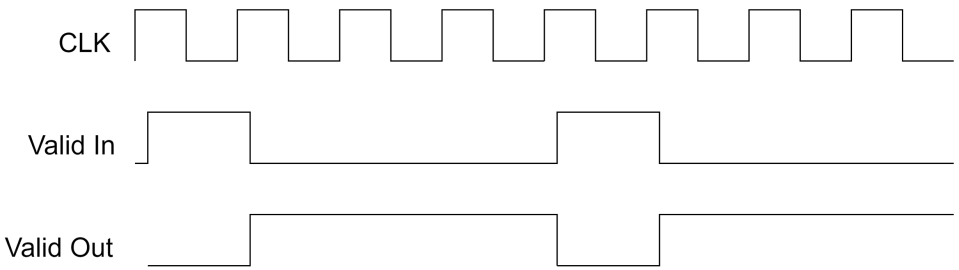
- Valid start: 1
- Valid strides: 1
- Cycle start: 1
- Cycle strides: not relevant
- Cycle extent: 1

App Example: Downsample



- Valid start: 2
- Valid strides: 2
- Cycle start: 0
- Cycle strides: not relevant
- Cycle extent: 1

App Example: Upsample



- Valid start: 1
- Valid strides: 1
- Cycle start: 0
- Cycle strides: 1
- Cycle extent: 3