Accelerating Multi-headed Attention Layer in Transformer Model

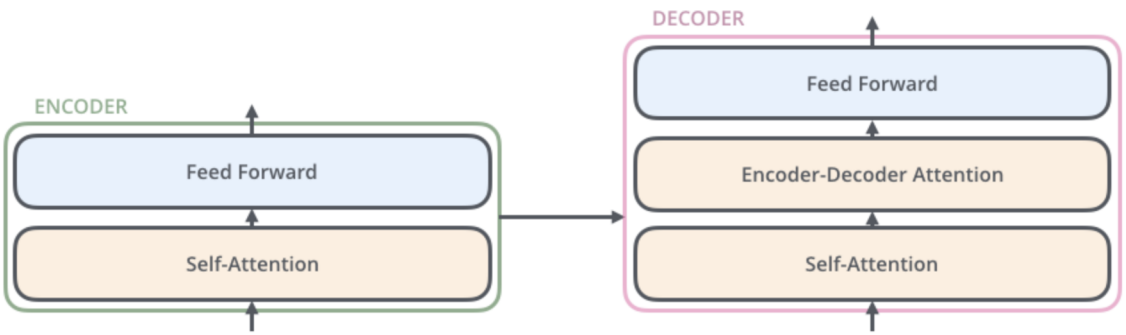
Zeyu Chen, Jonathan Nativ, Kevin Hutto

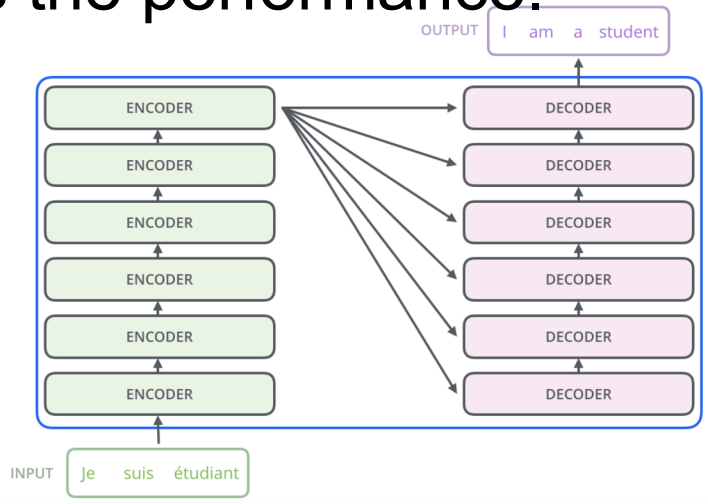
Date: 05/03/2022

Georgia Tech

Problem Statement

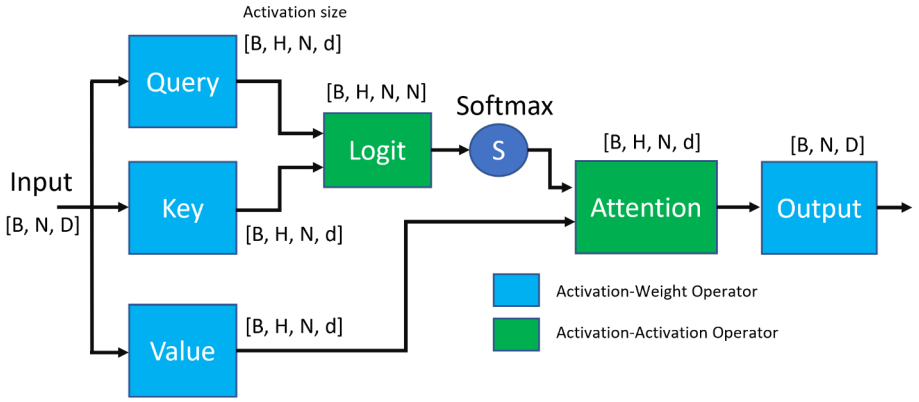
- Multi-Headed Attention Mechanism is widely adopted in most Transformer models, as it leverages weights for surrounding tokens based on the distance from the current word, which proves to be effective to NLP work.
- On the other hand, Attention Mechanism has high computation complexity and is heavily restricted by the memory bandwidth.
- Attention Layer is used by all Encoder/Decoder in Transformers and always determines the performance.

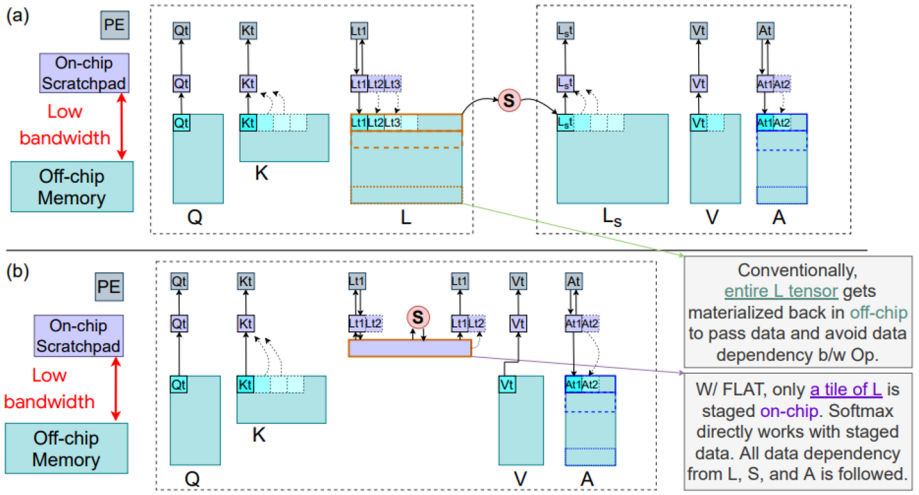




Proposed Design

- The software algorithm we accelerated is Fused Logit Attention Tiling (FLAT).
- FLAT is a tiled version of the Transformer model and fuses three operators: Logit, Softmax and Attention in original design.





- FLAT leverages on-chip memory and keeps all operations and data on-chip so they all enjoy the benefits of high memory bandwidth and low latency.

Georgia Tech

Attempted Optimization Techniques

- Simple Pipelining and Loop Unrolling
- Systolic Array Logit and Attention Operators
- Double buffering and DRAM Bursting
- Data quantization
  - Utilized 16-bit fixed-point (ap_fixed<16,3>)

Results for Basic Pipelining

	Latency	BRAM	DSP	FF	LUT
(a)	2.735e+11 ns	457(163%)	28(12%)	6353(5%)	10545(19%)
Data Loading Optimizations					
(b)	2.696e+11 ns	457(163%)	28(12%)	14398(13%)	177192(333%)
Multiplication Optimizations					
(c)	1.519e+11 ns	457(163%)	91 (41%)	8005 (7%)	14207 (26%)
(d)	1.519e+11 ns	457(163%)	28 (12%)	8347 (7%)	11713 (22%)
(e)	1.523e+11 ns	457(163%)	91 (41%)	7969 (7%)	14729 (27%)
Combined Pipeline Optimizations					
(f)	3.081e+9 ns	457(163%)	154 (70%)	10148 (9%)	45304 (85%)

- Optimizations from Simple Pipelining and Loop Unrolling
  - (a) Baseline hardware results.
  - (b) Data loading optimizations
  - (c) Fused Logit Operator optimizations
  - (d) Softmax Operator optimizations
  - (e) Fused Attention Operator optimizations
  - (f) Combined optimizations

Georgia Tech

Results for Basic Pipelining

	Latency	BRAM	DSP	FF	LUT
(a)	2.735e+11 ns	457(163%)	28(12%)	6353(5%)	10545(19%)
Data Loading Optimizations					
(b)	2.696e+11 ns	457(163%)	28(12%)	14398(13%)	177192(333%)
Multiplication Optimizations					
(c)	1.519e+11 ns	457(163%)	91 (41%)	8005 (7%)	14207 (26%)
(d)	1.519e+11 ns	457(163%)	28 (12%)	8347 (7%)	11713 (22%)
(e)	1.523e+11 ns	457(163%)	91 (41%)	7969 (7%)	14729 (27%)
Combined Pipeline Optimizations					
(f)	3.081e+9 ns	457(163%)	154 (70%)	10148 (9%)	45304 (85%)

BRAM usage currently exceeds 100% of the Pynq-Z2 board resources. Further work is needed to tile the multiplication to a smaller granularity than FLAT to fully implement the architecture on the Pynq-Z2.

Georgia Tech

Proposed Design

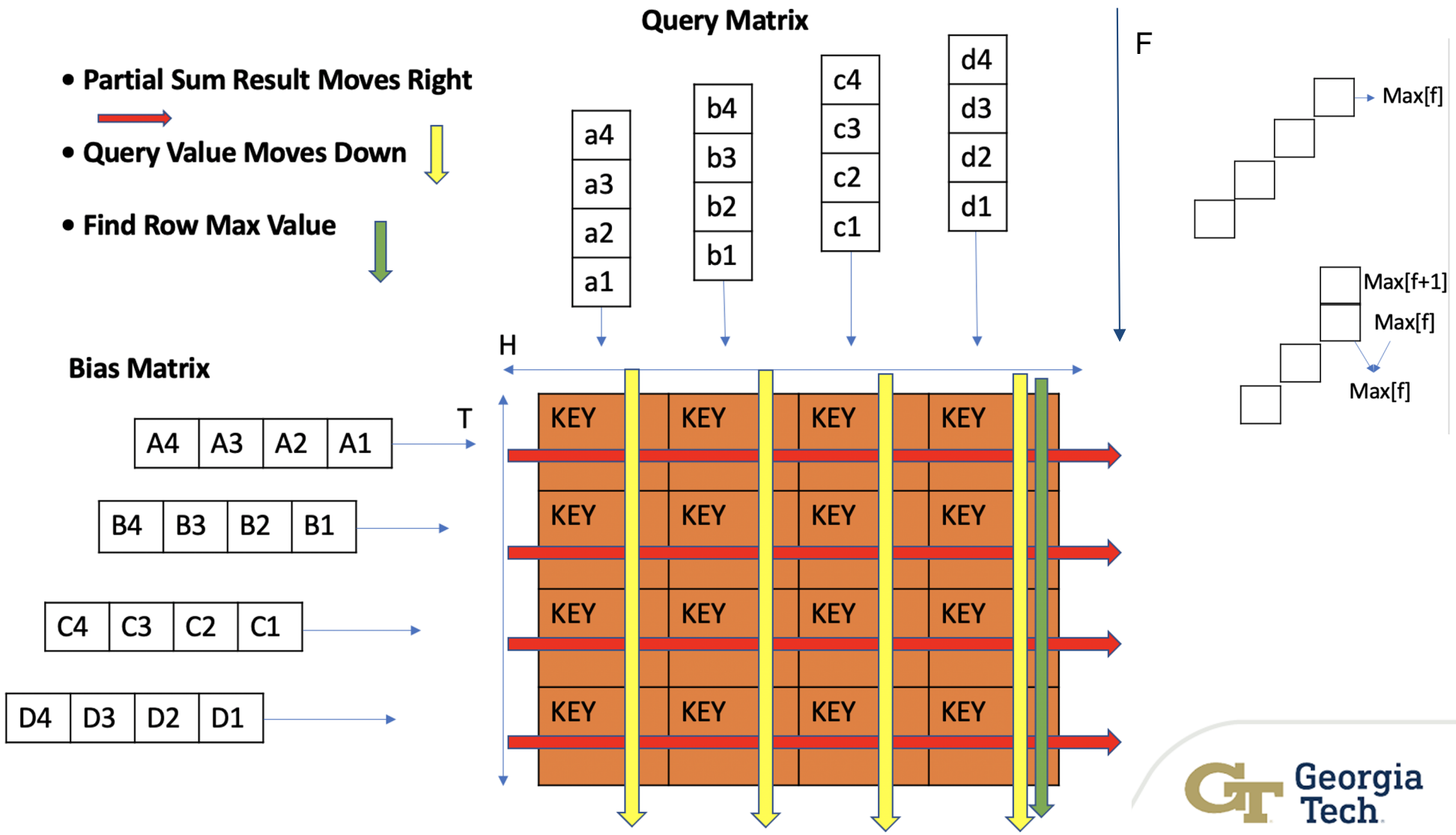
- Our main design is a Back-to-Back Systolic Structure for FLAT algorithm.
- The first systolic array is for the Logit operator, with weight stationary structure.
- The second systolic array is for the Attend operator, with output stationary structure as the initial design, but we then realize the fan-out of mapping data from each PE back to the output buffer is too much, so we change it to weight stationary to improve data movement.
- Softmax layer is in-between two systolic arrays

Proposed Design-SoftMax

- For Softmax Function, we use stable softmax version, which requires the normalization step before doing exponential.
- Original design has three separate loops, one for finding the maximum element per row, one for exponential and one for division.
- We try multiple optimization methods here.
  - We merge the loop to the first Logit operator so we don’t need additional loop.
  - Re-order loop to get more parallelism

Proposed Design-Logit

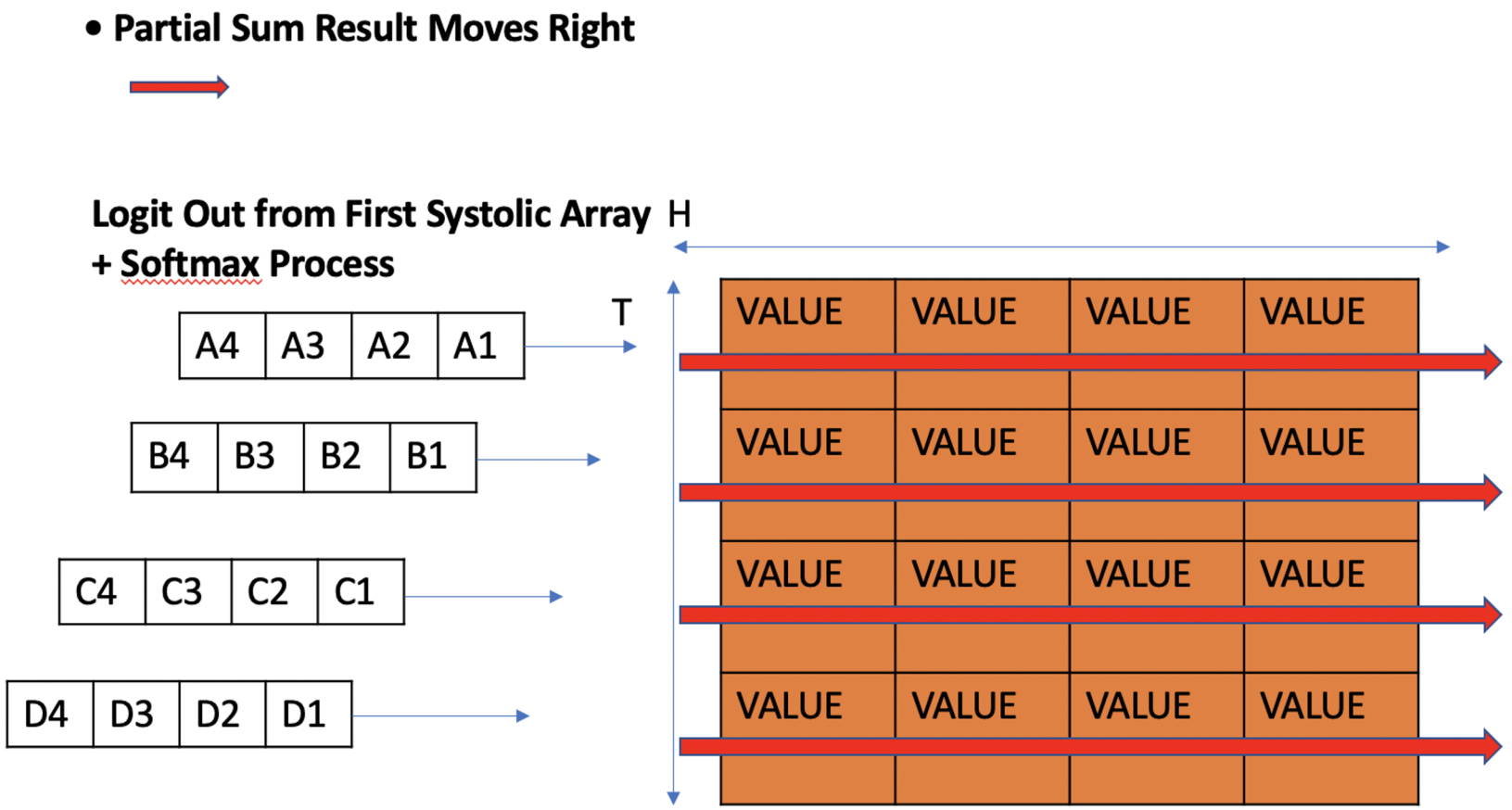
- Matrix 1: TH, Matrix 2: FH, Target: FT(TH, FH – FT) + bias



Georgia Tech

Proposed Design-Attention

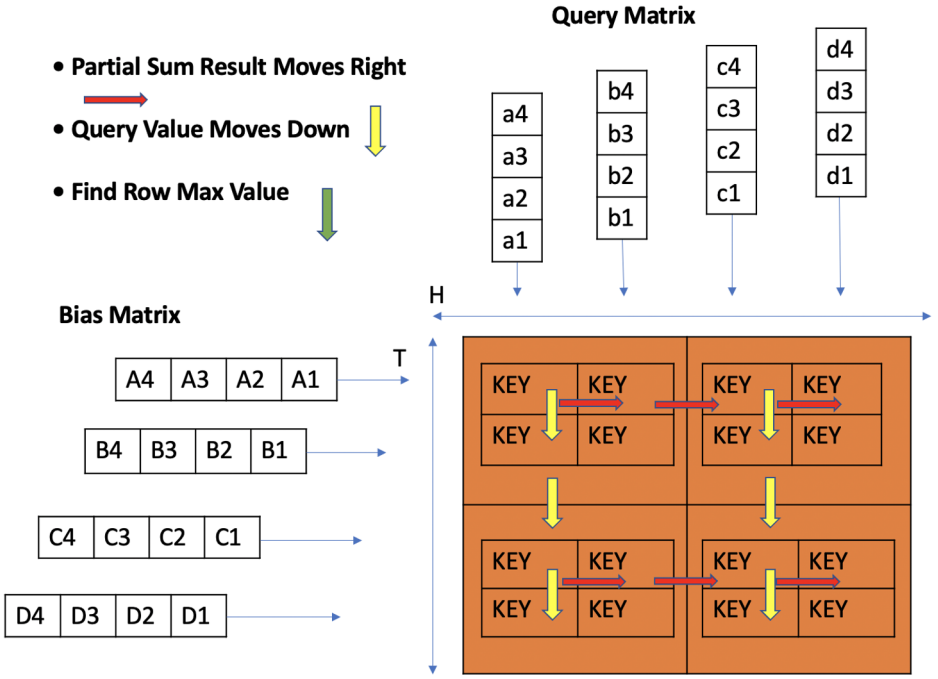
- Matrix 1: FT, Matrix 2: TH, Target: FH (FT, TH->FH)



Georgia Tech

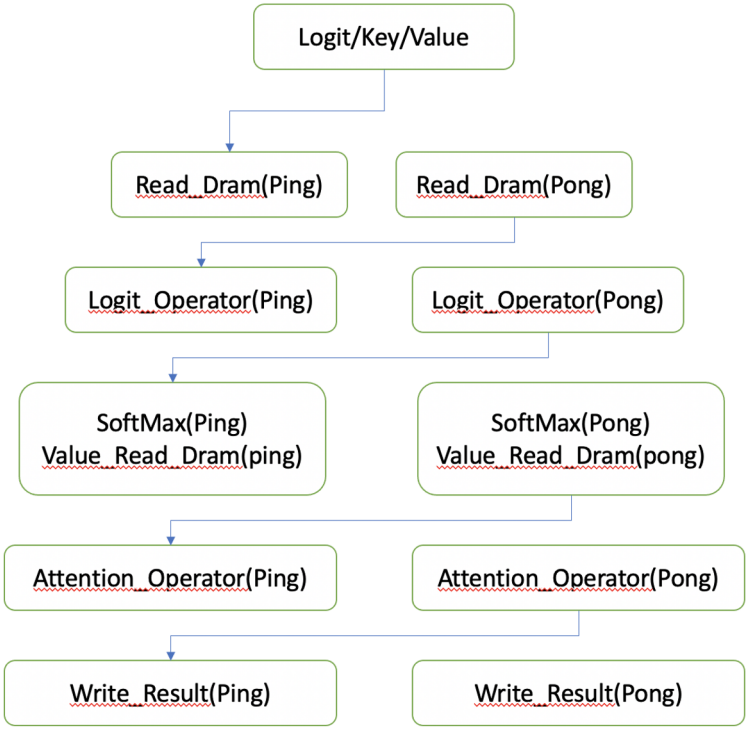
Systolic Array Design Considerations

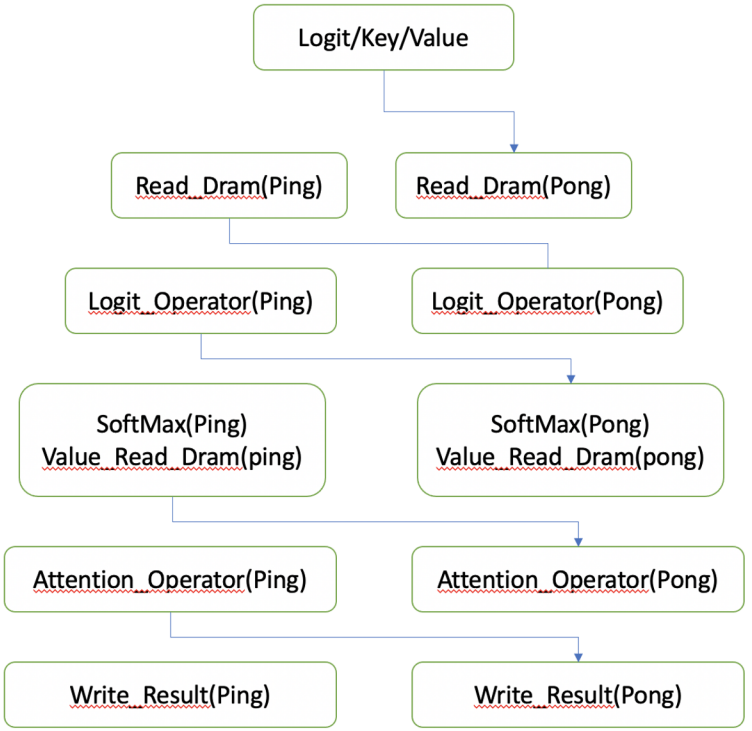
- Break 64x64 SA into two levels for greater control over resource utilization
  - Outer tiles are actual PEs -> UNROLL
  - Inner tiles are “pseudo” PEs -> PIPELINE
  - Found best resource utilization and latency with 4x4 outer tiles and 16x16 inner tiles
- Reverse loop iteration to avoid double buffering local outputs

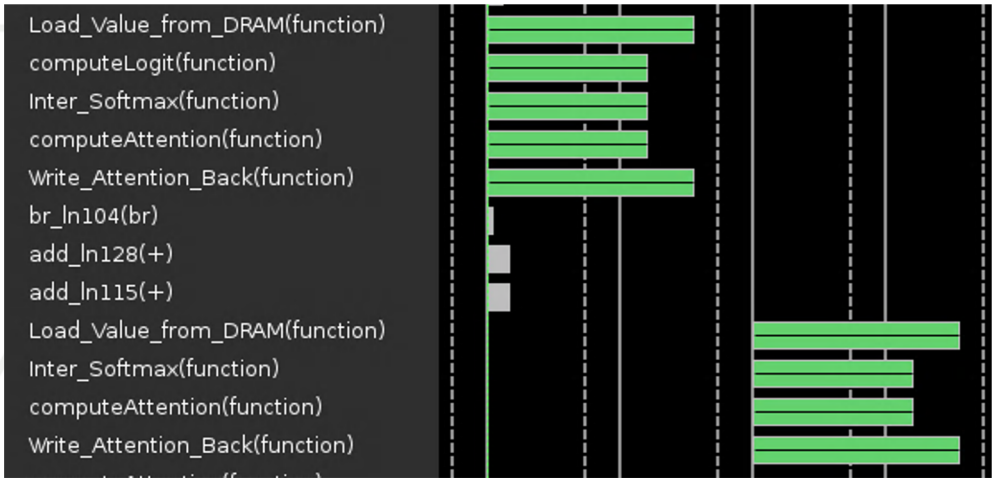


Georgia Tech

Double Buffering Design








```
void Load_Value_from_DRAM(int b, int n, data_t value_buffer[KEY_LENGTH_T][HEAD_DIM_H], MEM_TYPE value[576][64][16])
{

    for (int t = 0; t < 64; ++t)
    {
#pragma HLS pipeline
        MEM_TYPE data = value[b][t][n];
        for (int h = 0; h < 64; ++h)
        {
            value_buffer[t][h] = data.range(0 + (h*16), 15 + (h*16));
        }
    }
}
```

Georgia Tech

Results for Systolic Array, DRAM Bursting, and Double Buffering

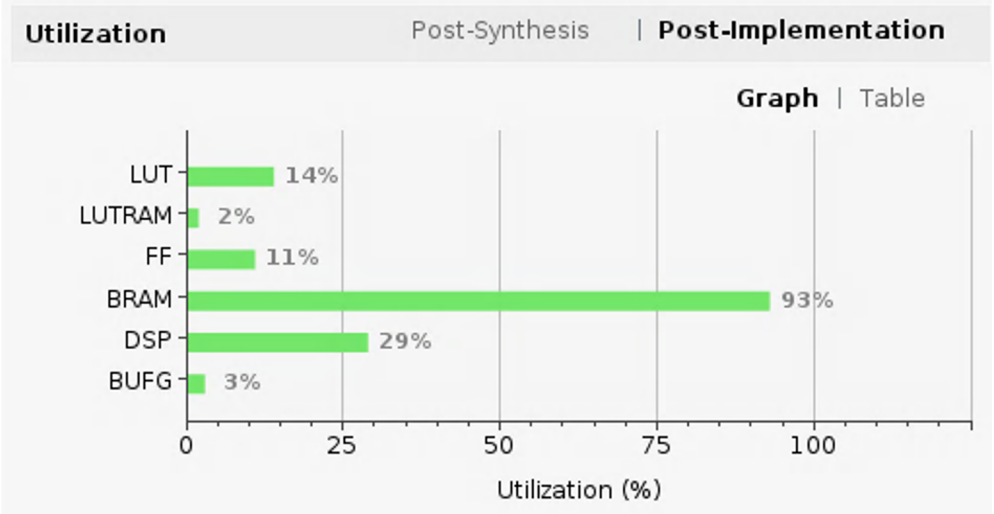
Latency	BRAM	DSP	FF	LUT
Baseline Hardware results				
273 s	457 (163%)	28 (12%)	6353 (5%)	10545 (19%)
Compute Logic Function				
1.18e+7 ns	4	16	6934	8049
Compute Attention Function				
8.02e+6 ns	4	16	6840	8249
Compute Softmax Function				
2.14e+4 ns	0	52	164651	203838
Systolic Array with Double Buffering				
0.190 s - 257.76 s	242 (86%)	84 (38%)	189423 (178%)	240349 (451%)

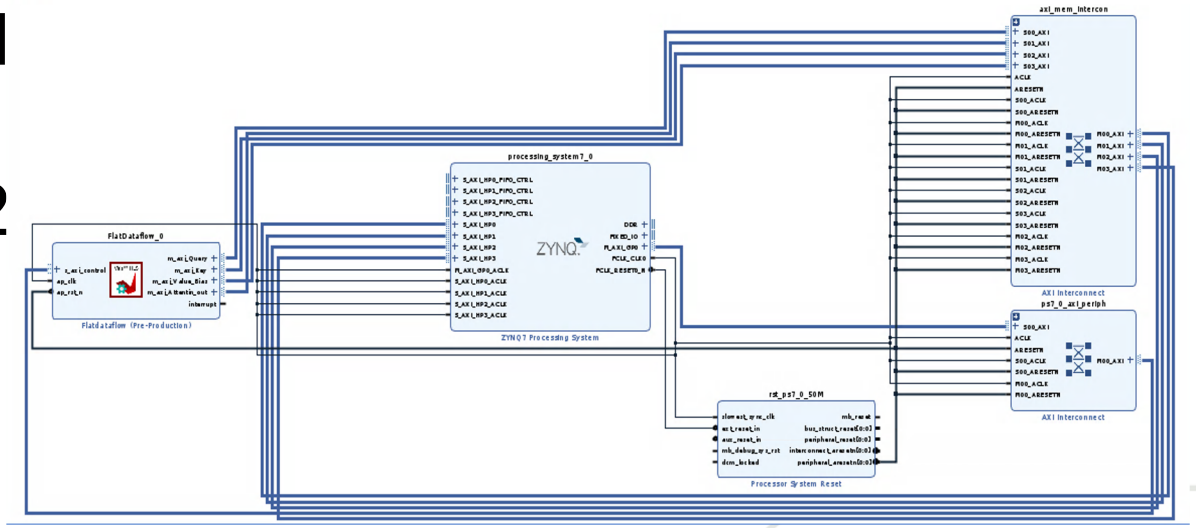
- DRAM bursting improved latency of reads by 50%
- Minimally partitioned arrays for synthesis time
- DSP usage may be inaccurate (lower than anticipated, possibly linked to above)
- Vitis timing for double buffering is imprecise

Georgia Tech

Pynq 2 Testing

- Due to the size of the pynq-z2 resources, only a subset of the full FLAT algorithm was able to be tested
- Successfully performed one batch of the logit function on the pynq-z2 board





Georgia Tech

Future Work

- Attempt Data Quantization
- Attempt to lower resource utilization for the systolic array implementation
  - Logit is the bottleneck of our current design
  - If we partition inputs, outputs and weights more, we expect both systolic arrays to run faster, but at the cost of higher BRAM and LUT utilization
  - Could also change systolic array dimensions to reduce LUT utilization
- Dataflow streaming instead of double buffering
  - Lower BRAM utilization and faster