Center for Future Architectures Research
CFAR

Energy Efficiency of Full Pipelining:

A Case Study for Matrix Multiplication

Peipei Zhou, Hyunseok Park, Zhenman Fang, Jason Cong, Andre DeHon

UCLA

Center for Domain Specific Computing

Motivation and Contributions

Motivation

- The customized pipeline design has been one of the most important optimizations and widely used to improve the performance of FPGA accelerators.
- The impact of the pipeline II on the energy efficiency of accelerator designs remains unclear.

Contributions

- Provide a set of high-level yet accurate analytical models to investigate the impact of the pipeline II on the energy consumption of FPGA accelerators designed in high-level synthesis (HLS).
- Provide insight into Matrix-Multiply with II > 1 is optimal
- Identify Sources of inefficient mapping in commercial HLS flow

Computation & Memory

Computation Energy

$$E_{compute} \propto \frac{N}{II} \times (N^2 \times II) = N^3$$

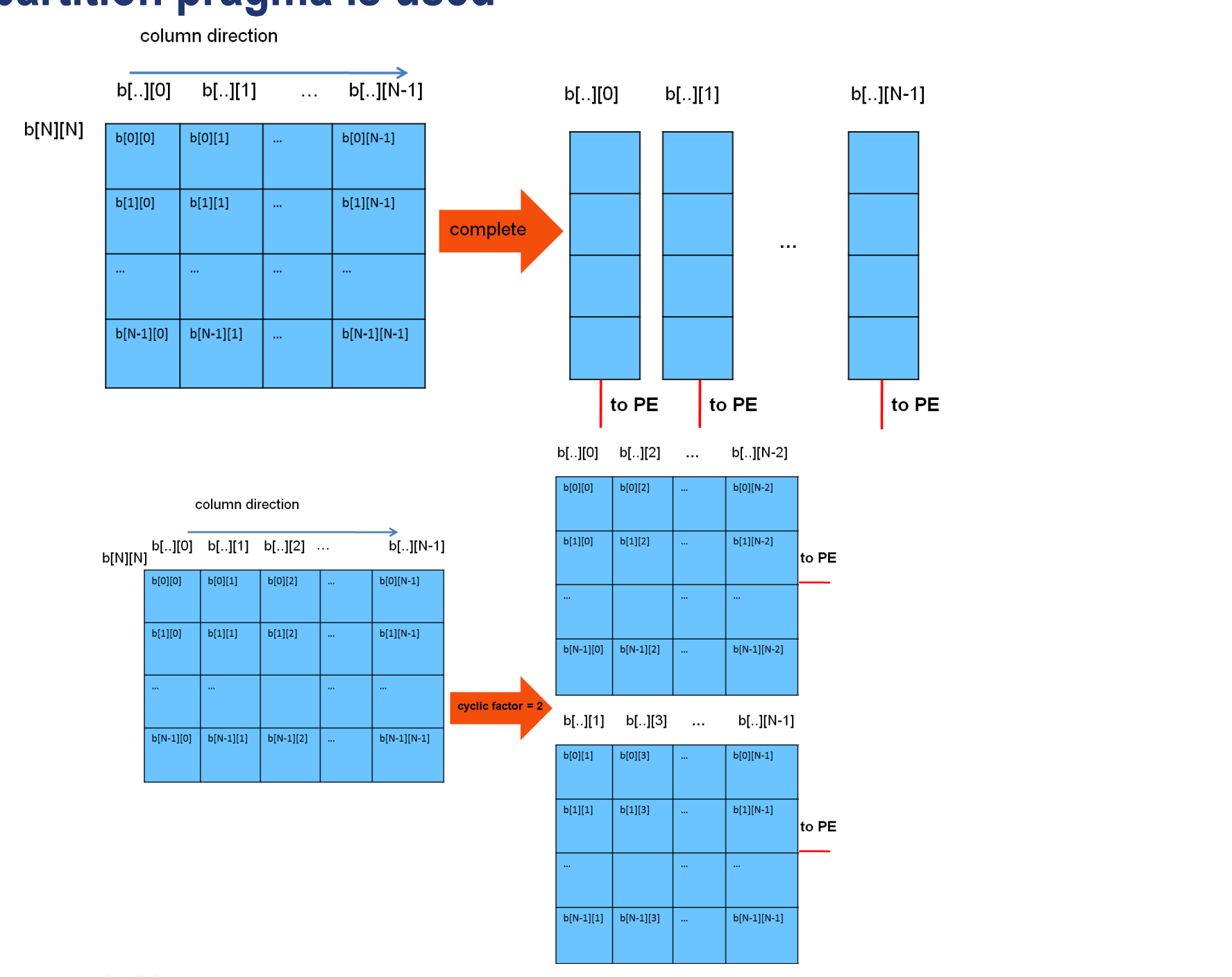
- Floating-point multiplier and adder usage

II	1	2	3	4	6	8	12	24
DSP	120	60	40	30	20	15	10	5
FF	8520	4260	2840	2130	1420	1065	710	355
LUT	8376	4188	2792	2094	1396	1047	698	349

Memory Energy

$$E_{mem} \propto \frac{N}{II} \times (N^2 \times II) = N^3$$

- Cyclic partition pragma is used



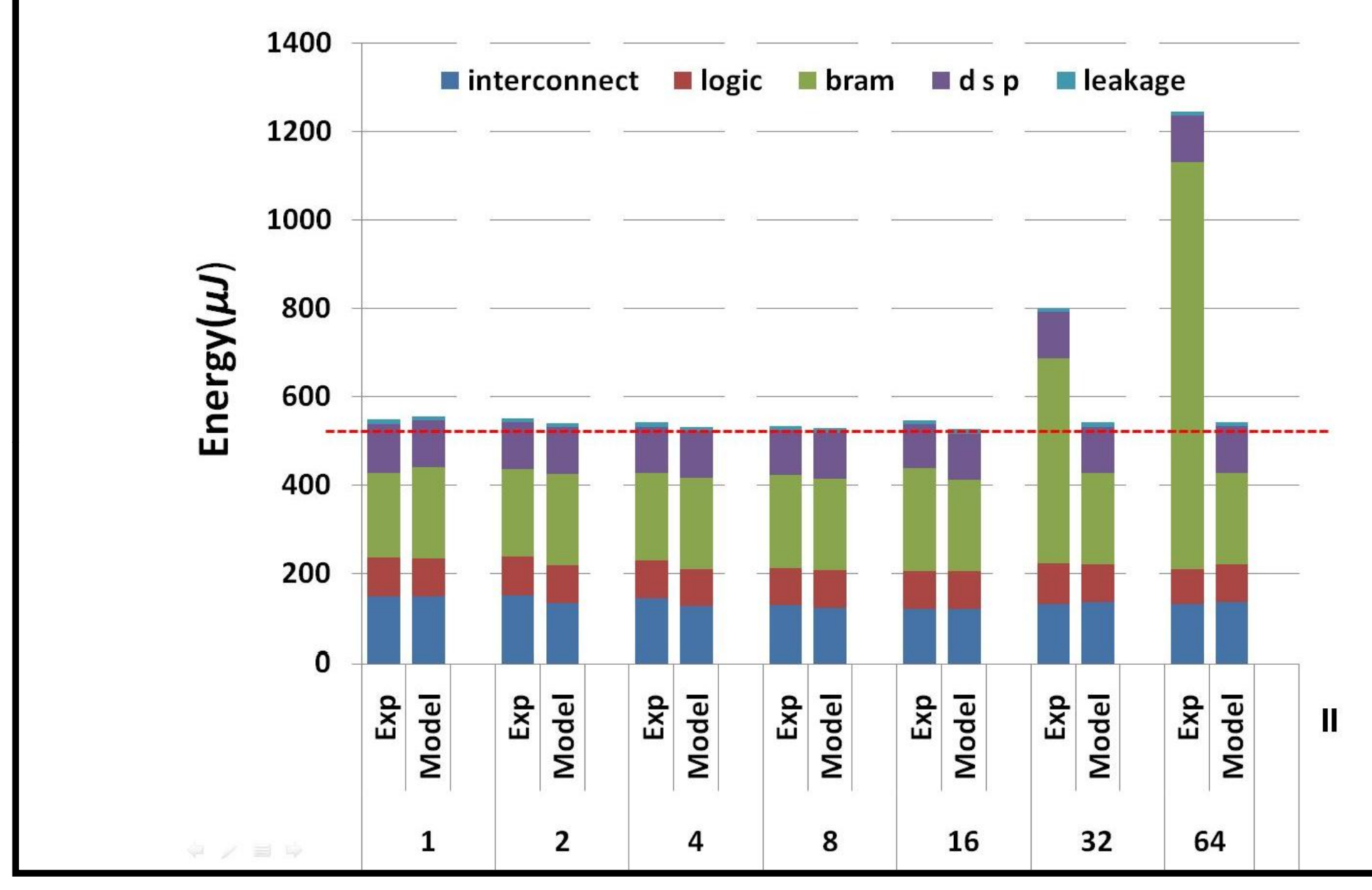
Results

Experimental Setup

- Virtex Virtix-7 XC7VX485T chip using Vivado 2015.1.5
- Simulated each mapped design in Vivado with random a and b matricies
- Switching Activity Interchange format (SAIF) file generated from post-implementation simulation to estimate the energy

N = 64, II =8 is optimal,

N = 64, II = 1 is within 5% of II =8



Matrix-Multiplication Kernel

```
void matrix_multiply(float a[N][N],
                     float b[N][N], float c[N][N]) {
  int i, j, k;
  k_loop:  for(k = 0; k < N; k++) {
    i_loop:  for(i = 0; i < N; i++) {
      #pragma HLS PIPELINE II = II_i
      j_loop:  for(j = 0; j < N; j++) {
        #pragma HLS UNROLL
        c[i][j] += a[i][k] * b[k][j];
} } } }
```

II determines performance, resource usage and energy

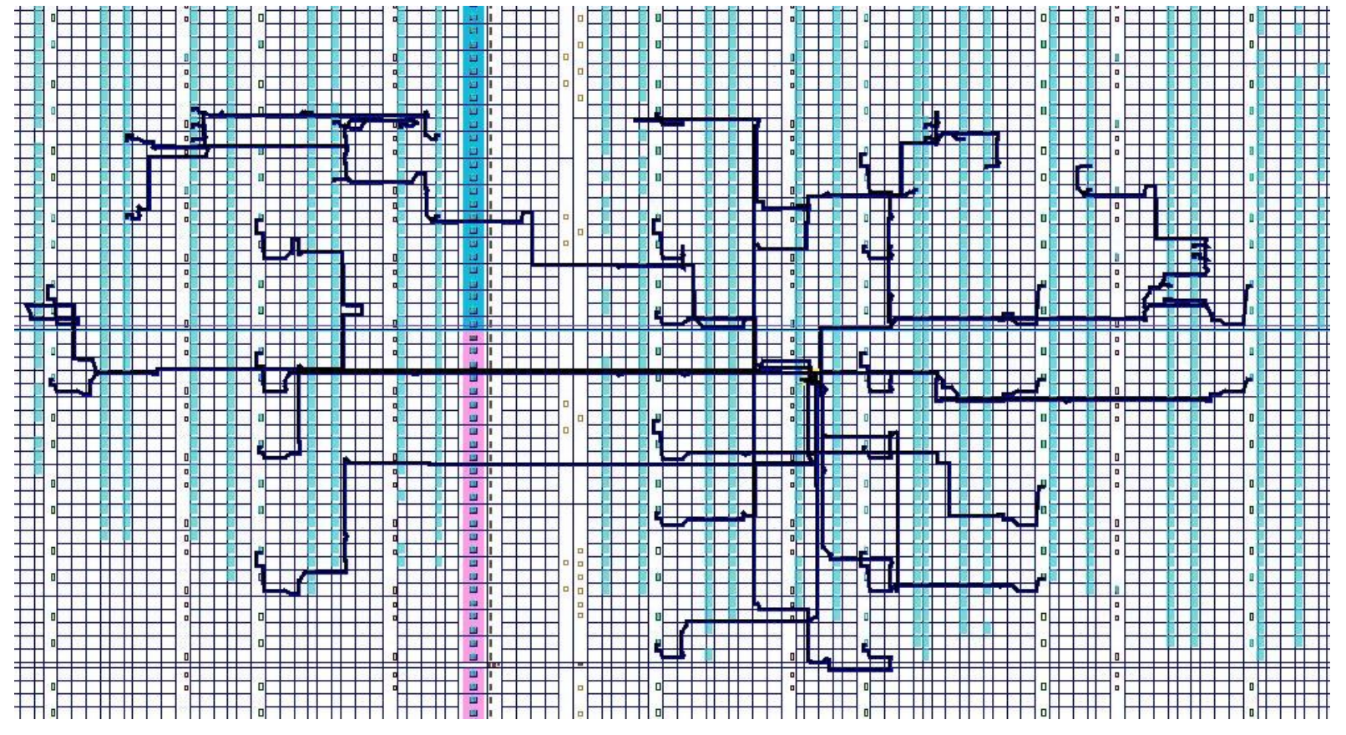
- Some useful code re-write is needed to achieve efficient architecture
- There are (N/II) multipliers and (N/II) adders.
- N/II independent memory bank for b matrix, each with II columns.
- Number of cycles: N^2 x II

```
void matrix_multiply(float a[N][N],
                     float b[N][N], float c[N][N]) {
  int i, j, k, p;
  k_loop:  for(k = 0; k < N; k++) {
    i_loop:  for(i = 0; i < N; i++) {
    //i_loop PIPELINE II = II_i
      p_loop:  for(p = 0; p < N; p += N/II_i) {
      #pramga HLS PIPELINE II = 1
        j_loop:  for(j = 0; j < N/II_i; j++) {
        #pragma HLS UNROLL
        c[i][p+j] += a[i][k] * b[k][p+j];
} } } } }
```

Interconnect

Wire transferring broadcast data

- N x II < BRAM 18 K, $E_{wire.share.A} \propto \frac{N}{II} \times N^2 = \frac{N^3}{II}$
  - DSP area dominated, II increases, PE area scales as N/II



- N x II > BRAM 18 K, $E_{wire.share.A} \propto N^2 \times N^2 = N^4$
  - BRAM area dominated, II further increases, PE area does not further decrease

Wire transferring private data

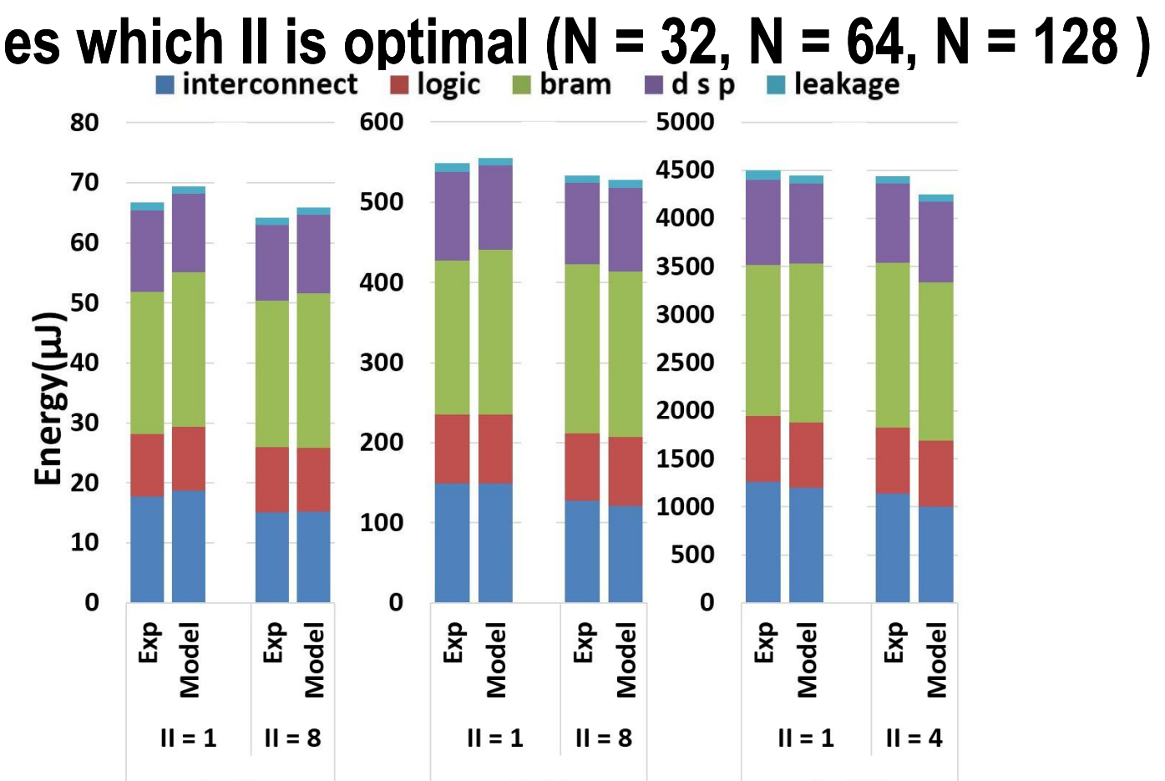
- N x II < BRAM 18K, $E_{wire.priv.B,C} \propto \frac{N}{II} \times (N^2 \times II) = N^3$
  - Constant distance between private b and c memory banks and PE logic
- N x II > BRAM 18K,

$$\begin{aligned} E_{wire.priv.B,C} &\propto \frac{N}{II} \times (N^2 \times II) \times \sqrt{N \times II} \\ &\propto N^{3.5} II^{0.5} \end{aligned}$$

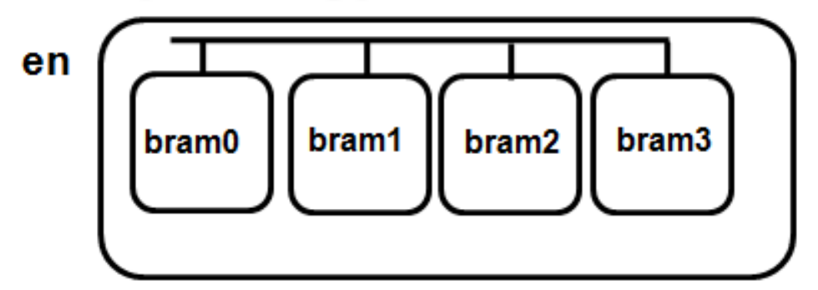
  - the wiring between the private b and c memory banks and the PE logic also grows as the square root of the memory capacity

Effect of Design size & current HLS limitation

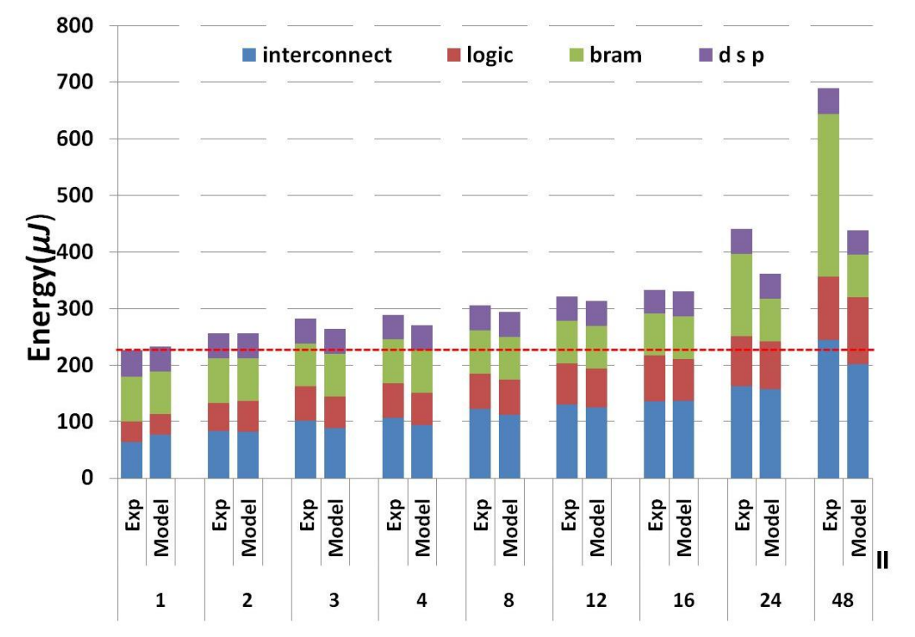
N determines which II is optimal (N = 32, N = 64, N = 128)



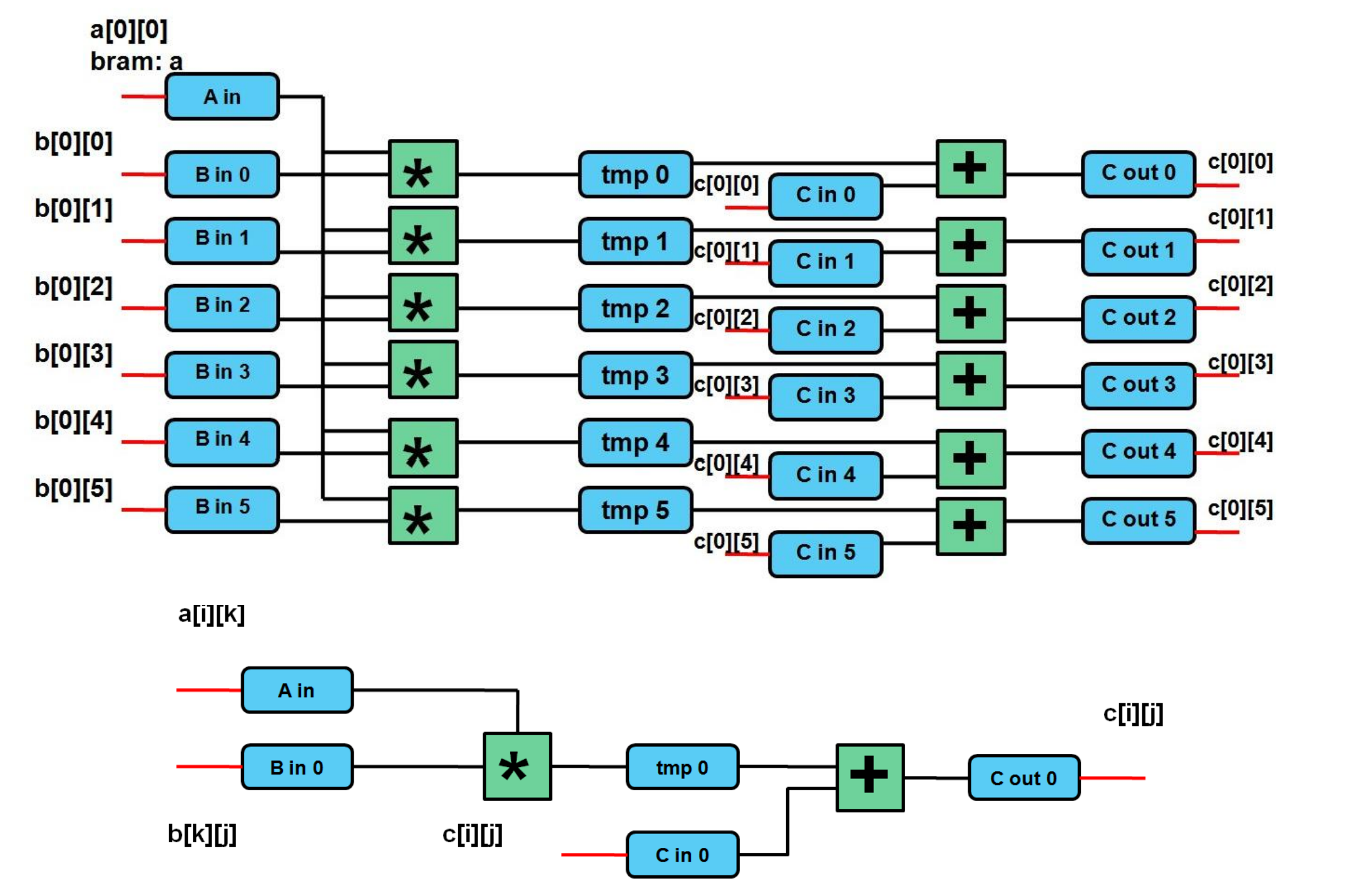
Memory Energy mismatch due to inefficient enabling signal



HLS code transformation is needed



Architecture for II = 1, II = N



II = 1

- Long wires for shared data a[][]
- Short wires for private data b[][] and c[][]

II = N

- Shorter wires for shared data a[][]
- Long wires for private data b[][] and c[][]

Leakage & Total Energy

Leakage

- Assume: If we put nothing else on the FPGA and use a fixed size FPGA that does not offer any power gating for unused components, leakage increases with runtime and hence II

$$E_{leak} \propto N^2 \times II \times P_{FPGA\_leak}$$

- Assume: use a design with perfect power gating of unused components

$$E_{leak} \propto \frac{N}{II} \times N^2 \times II = N^3$$

Total

$$\begin{aligned} E_{total} &= E_{compute} + E_{memory} + E_{wire} + E_{leak} \\ &= \begin{cases} N^3\left(c1 + \frac{c2}{II}\right), \\ \text{if } N \times II \le \text{BRAM18K} \\ N^3\left(c3 + c4 \times N + c5 \times II^{0.5}\right), \\ \text{if } N \times II > \text{BRAM18K} \end{cases} \end{aligned}$$

- Energy decreases before N x II < BRAM18K due to interconnect saving
- Energy increases after N x II > BRAM18K

Conclusion & Future Work

Conclusion

- Interconnect energy within our matrix-multiply kernel is minimized for an II > 1
- With efficient power gating or alternate use of chip resources = > minimum total energy at a point other than the fully pipelined, II = 1 point.

Future Work

- The energy modeling framework illustrated here can be adaptable to other kernels.
- Characterize how these components scale for other tasks & develop a suitably parameterized energy model.
- Model generation can be automated and provide high-level guidance for designers.
- Help to identify inefficiencies in current mapping tools that should be addressed to achieve energy efficient designs.