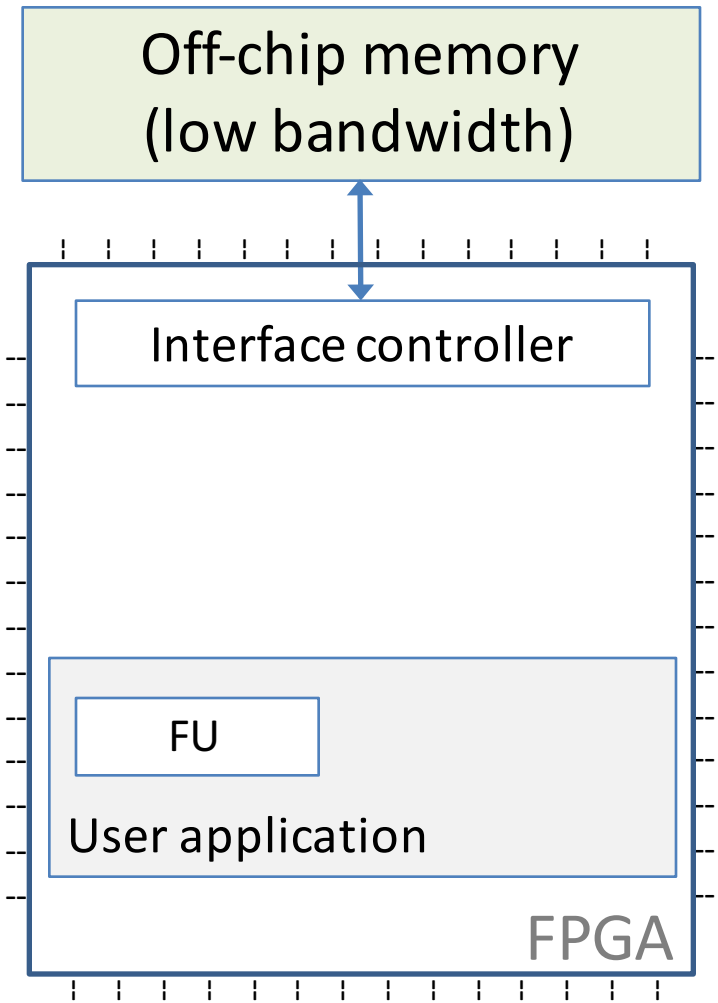
Imperial College London

Custom-Sized Caches in Application-Specific Memory Hierarchies

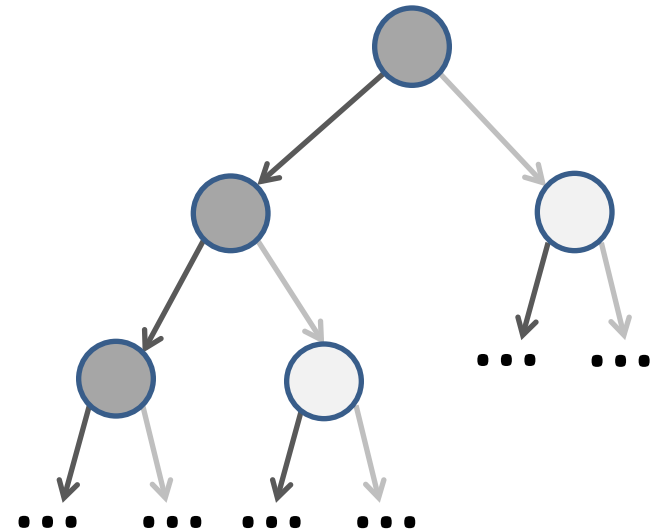
Felix Winterstein, Kermin Fleming, Hsin-Jung Yang, John Wickerson and George Constantinides

9 December 2015

Recap: FPGA'15

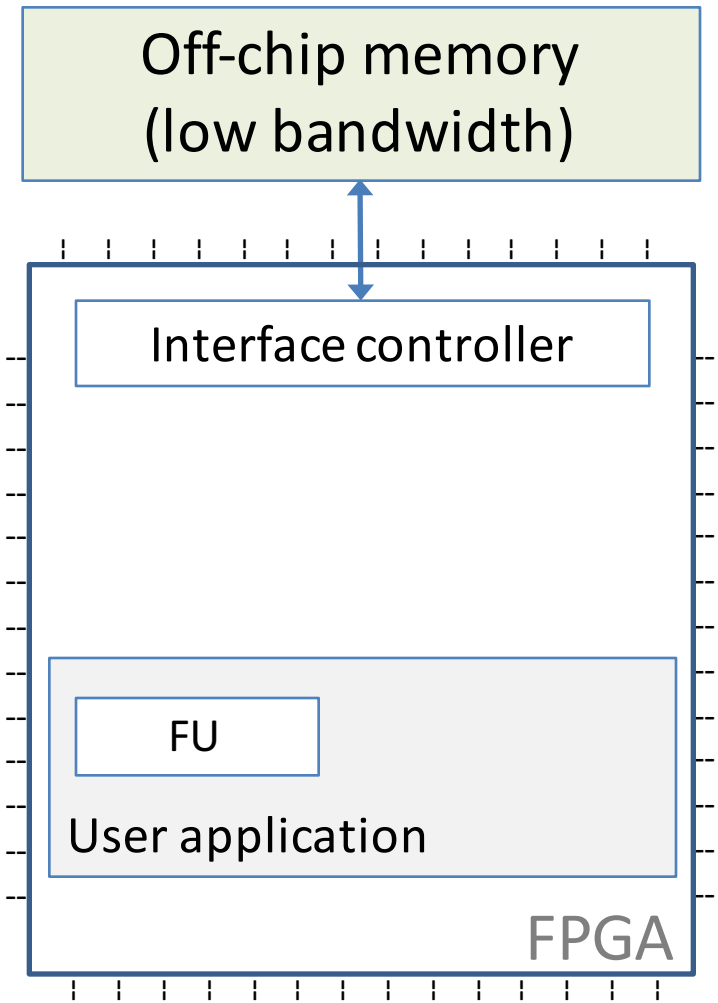


HLS

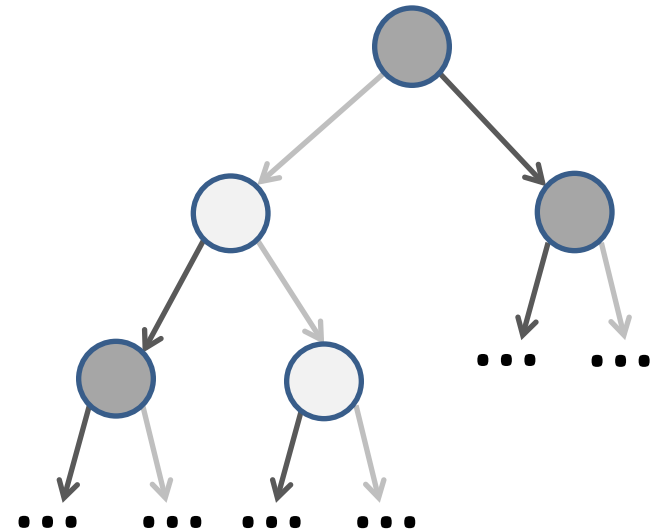


```
void reflectTree (typeB *root)
   typeA  *s = new typeA;
   s->u = root;
   s->n = 0;
   while s!=0 do
      typeB *u = s->u;
      …
```

Recap: FPGA'15

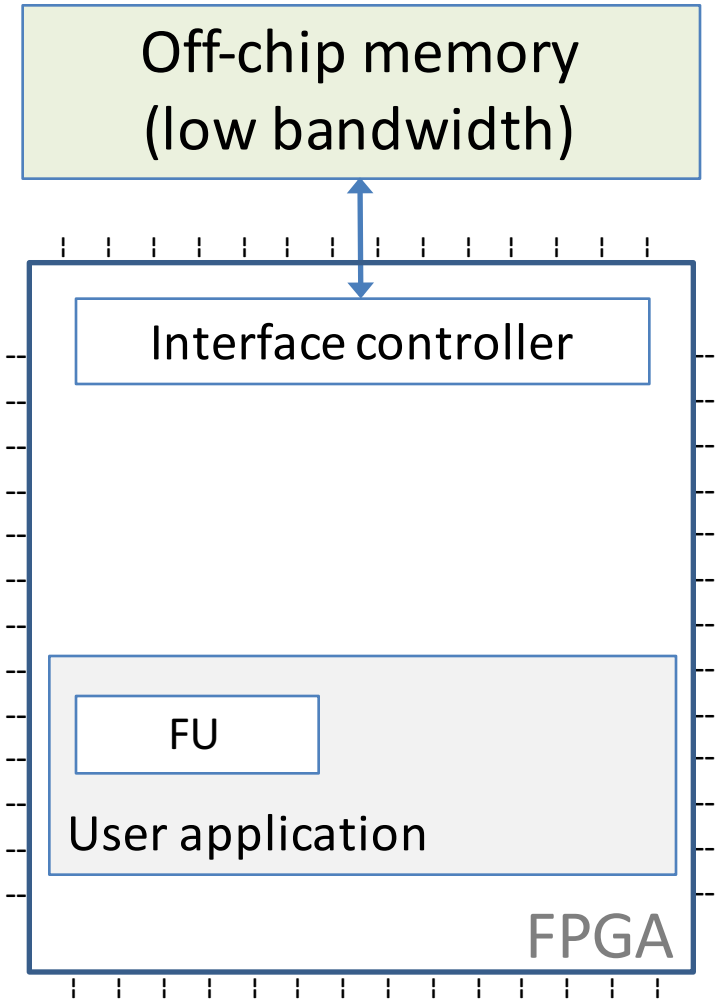


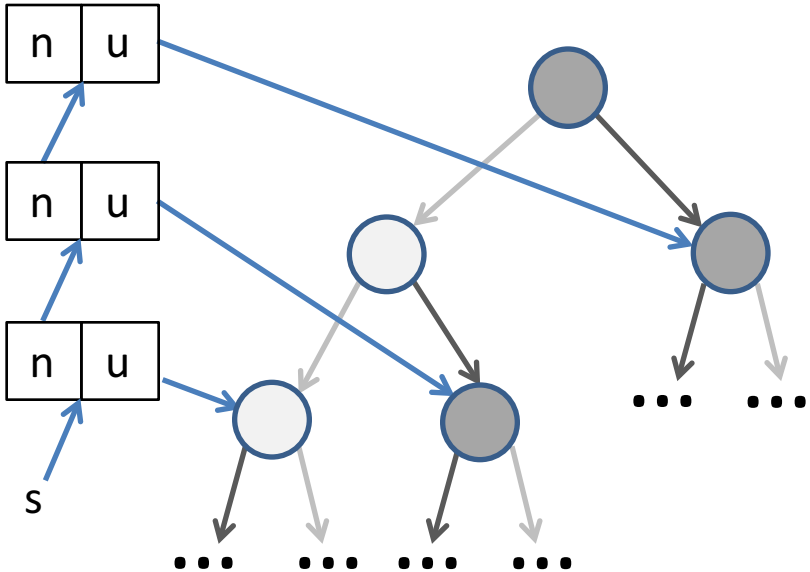
HLS



```
void reflectTree (typeB *root)
   typeA  *s = new typeA;
   s->u = root;
   s->n = 0;
   while s!=0 do
      typeB *u = s->u;
      …
```

Recap: FPGA'15

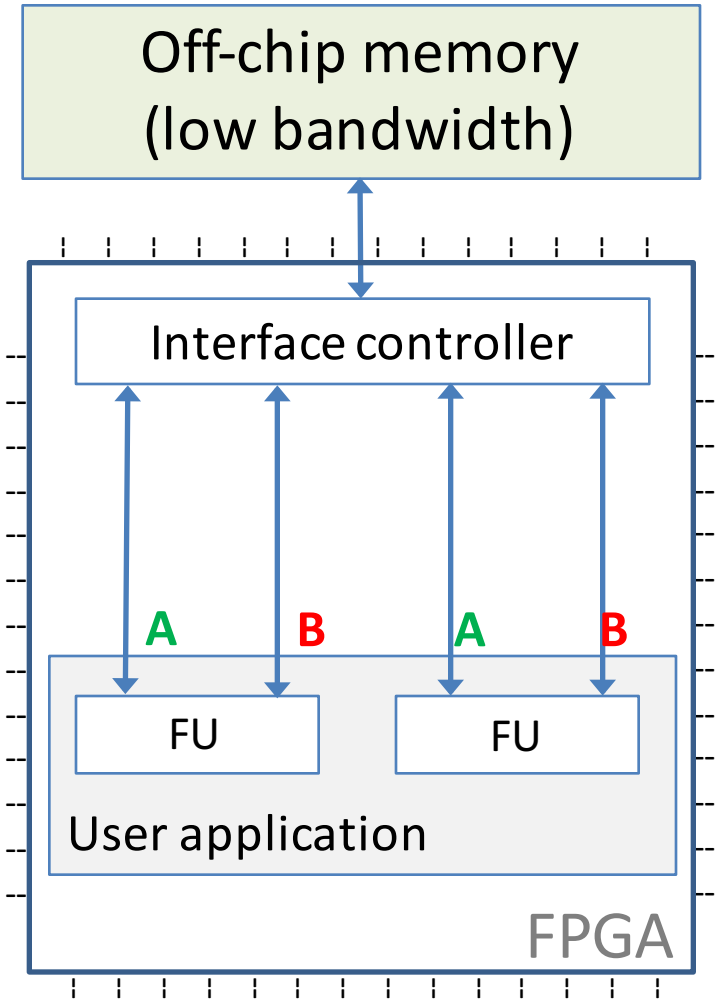


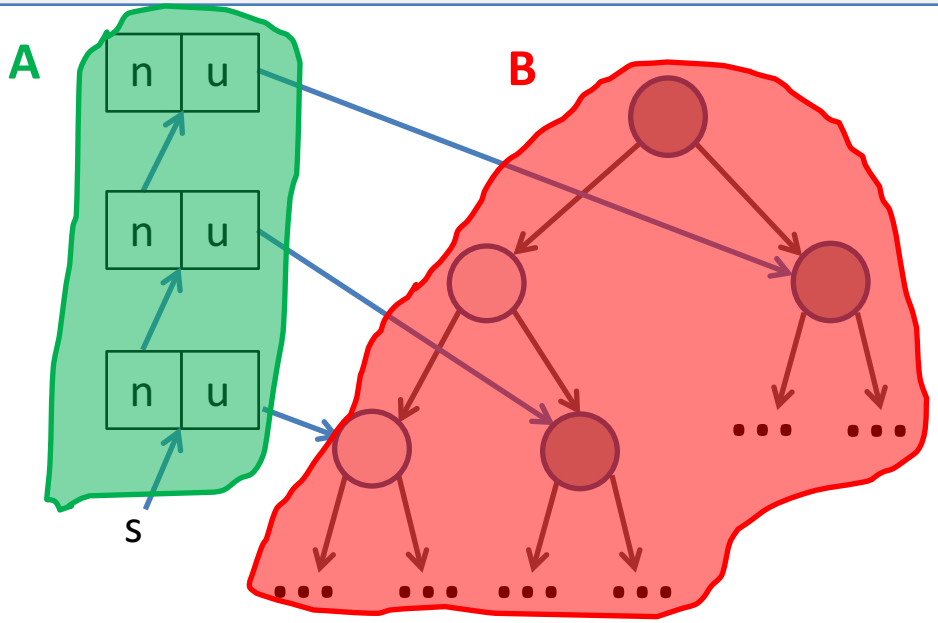


HLS

```
void reflectTree (typeB *root)
   typeA  *s = new typeA;
   s->u = root;
   s->n = 0;
   while s!=0 do
      typeB *u = s->u;
      …
```

Recap: FPGA'15

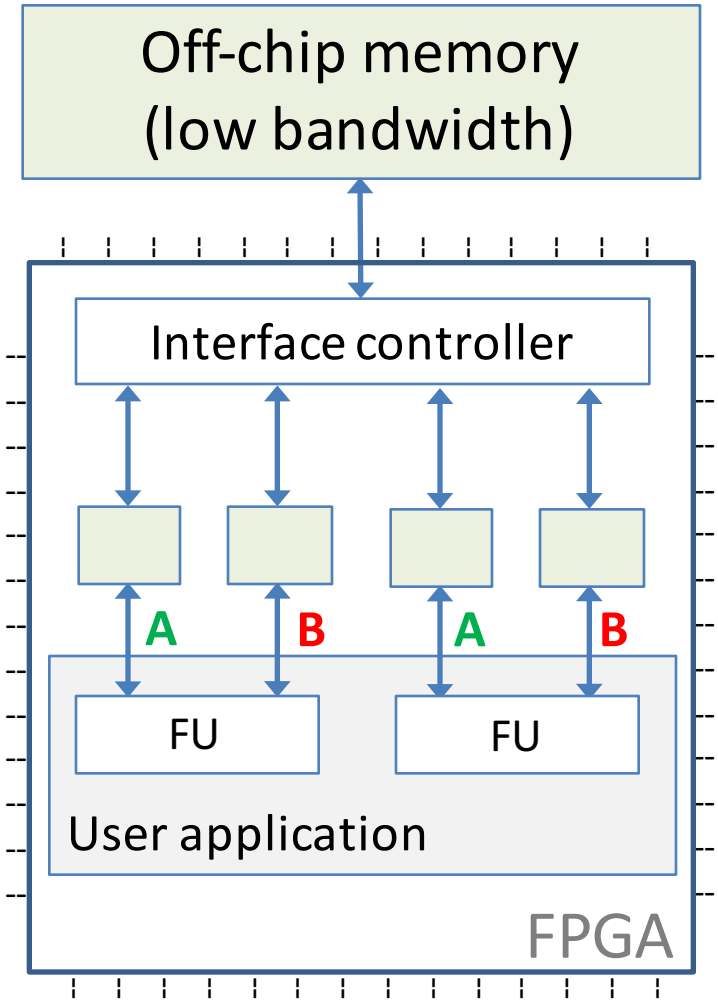


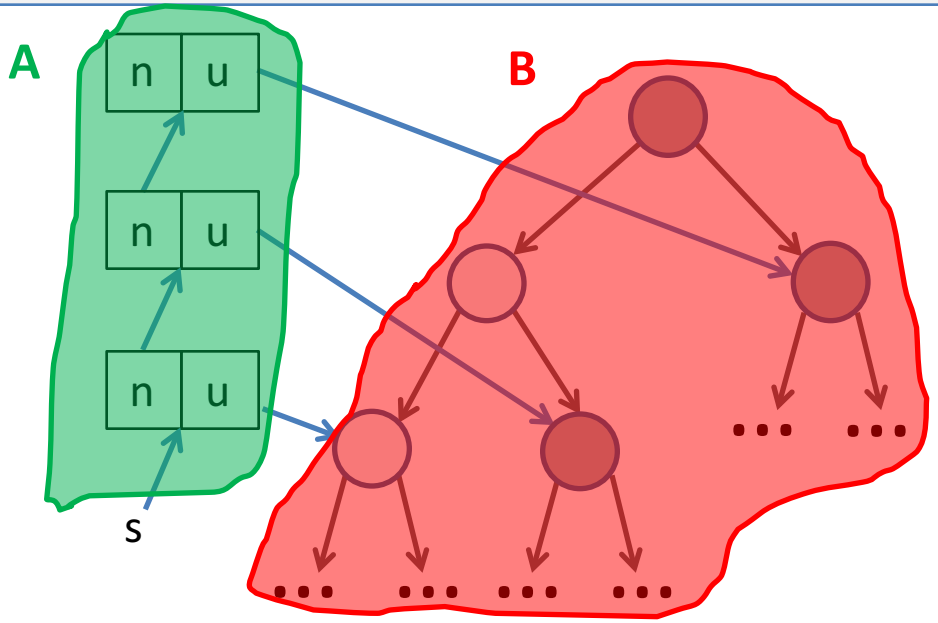


HLS

```
void reflectTree (typeB *root)
    typeA  *s = new typeA;
    s->u = root;
    s->n = 0;
    while s!=0 do
        typeB *u = s->u;
        …
```

Recap: FPGA'15

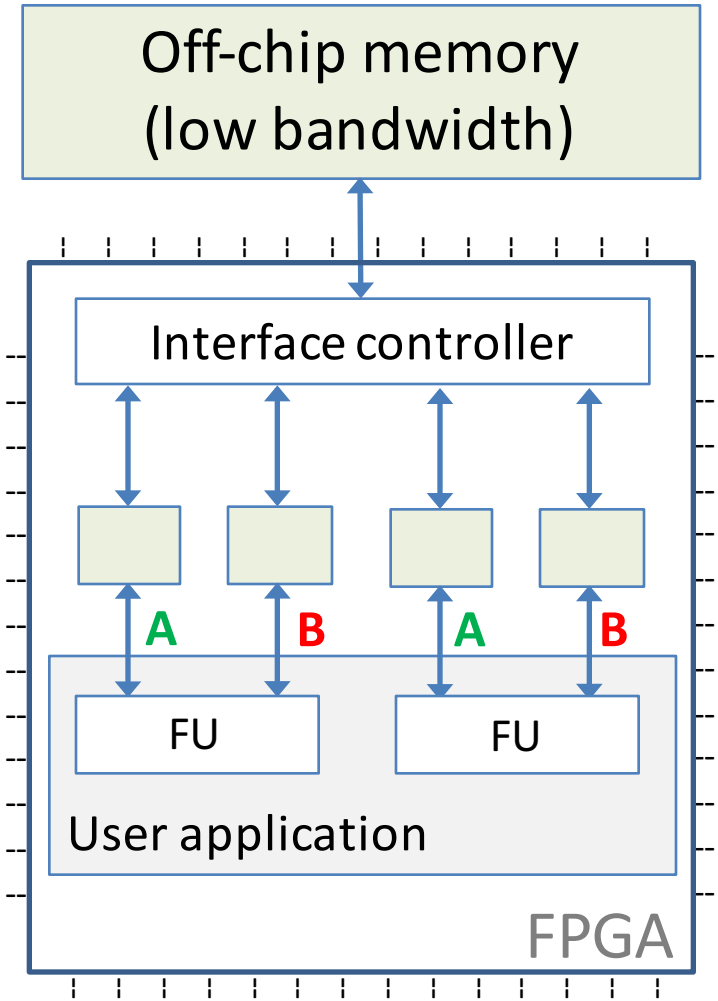


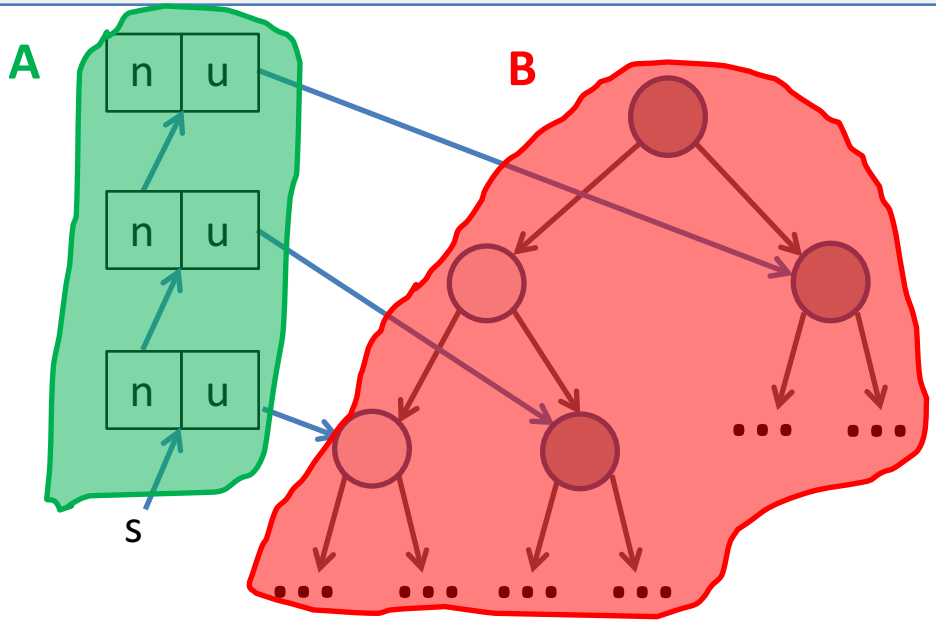


HLS

```
void reflectTree (typeB *root)
    typeA  *s = new typeA;
    s->u = root;
    s->n = 0;
    while s!=0 do
        typeB *u = s->u;
        …
```

Goal



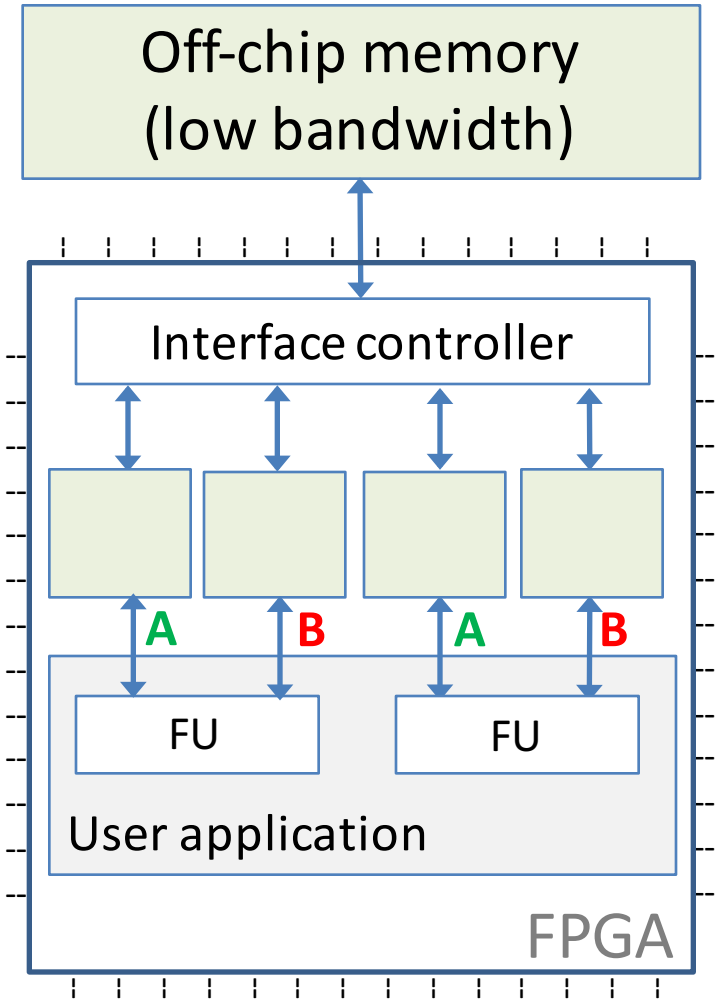


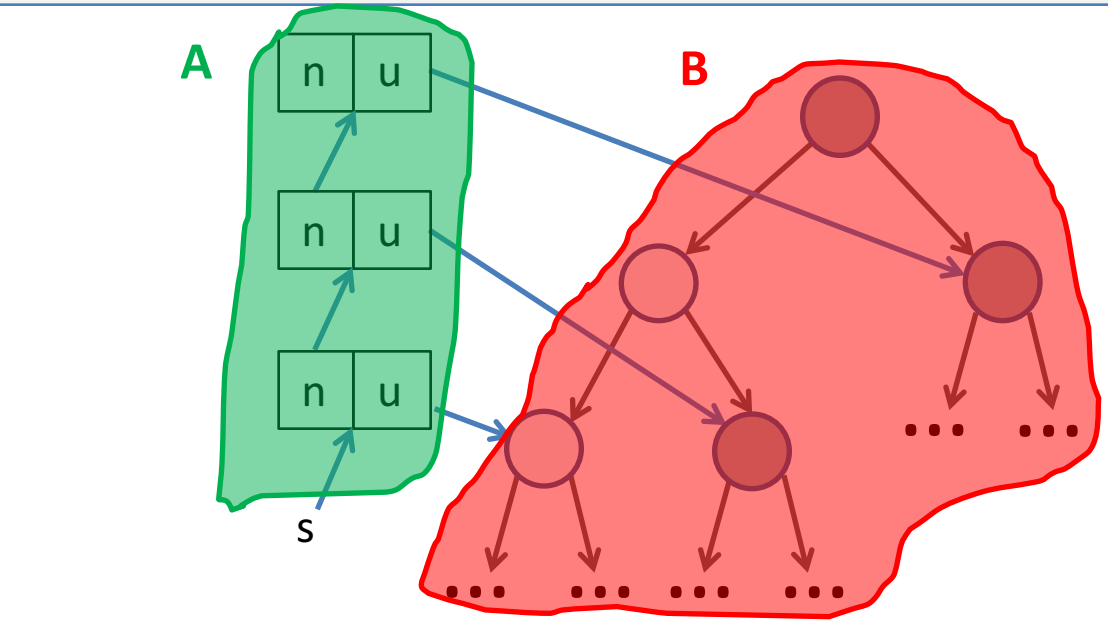
HLS

Custom cache sizing

```
void reflectTree (typeB *root)
    typeA  *s = new typeA;
    s->u = root;
    s->n = 0;
    while s!=0 do
        typeB *u = s->u;
        …
```

Goal



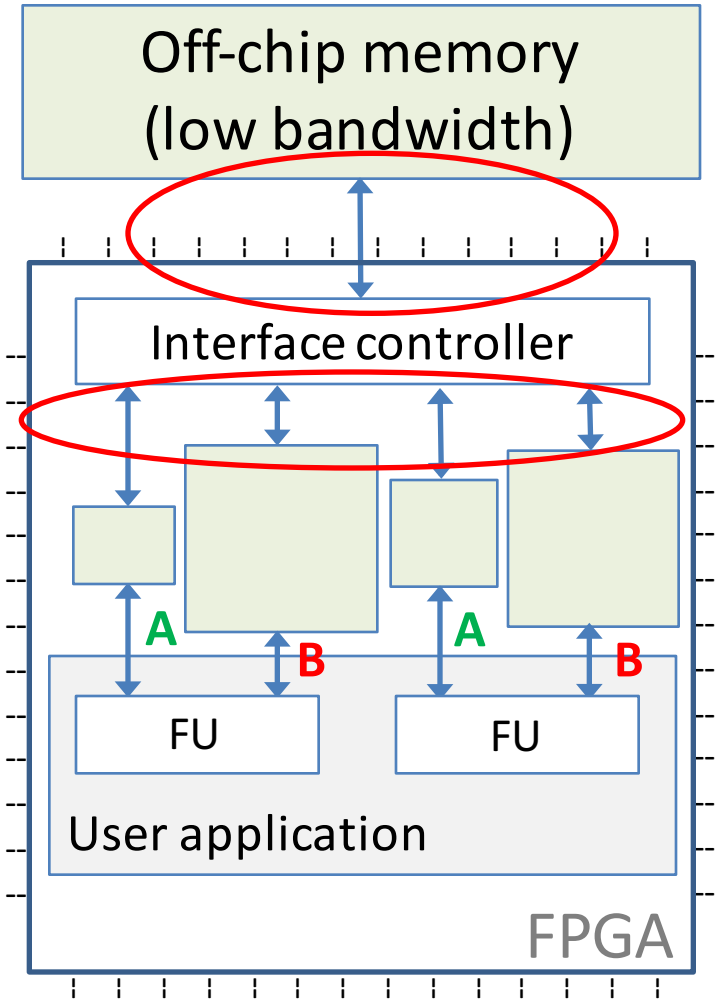


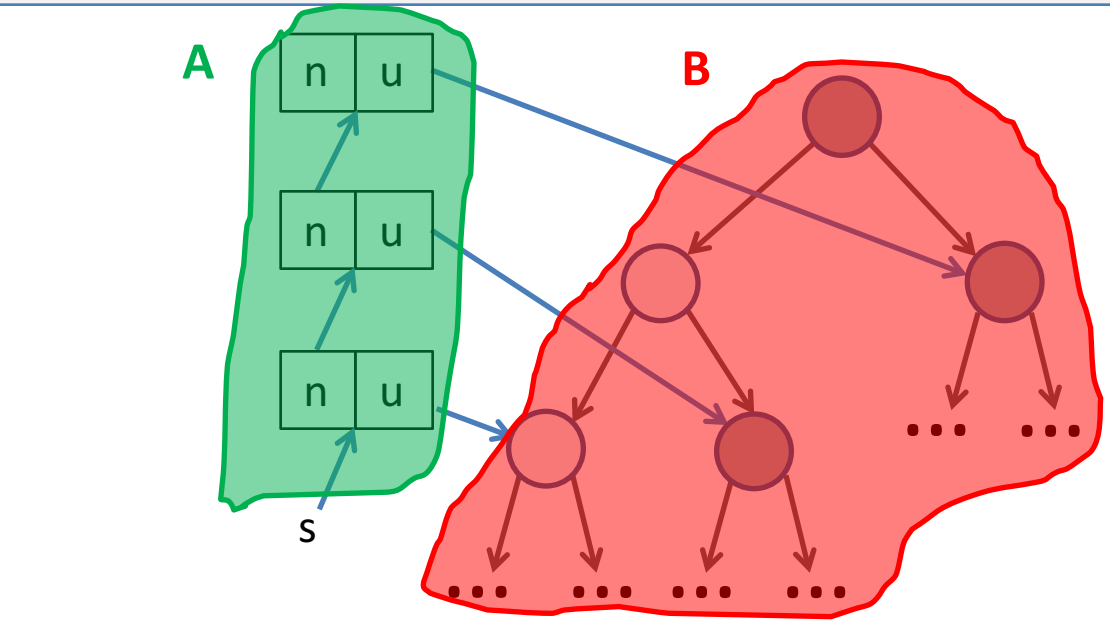
HLS

Custom cache sizing

```
void reflectTree (typeB *root)
    typeA  *s = new typeA;
    s->u = root;
    s->n = 0;
    while s!=0 do
        typeB *u = s->u;
        …
```

Goal





HLS

Custom cache sizing

```
void reflectTree (typeB *root)
    typeA  *s = new typeA;
    s->u = root;
    s->n = 0;
    while s!=0 do
        typeB *u = s->u;
        …
```

Executive summary

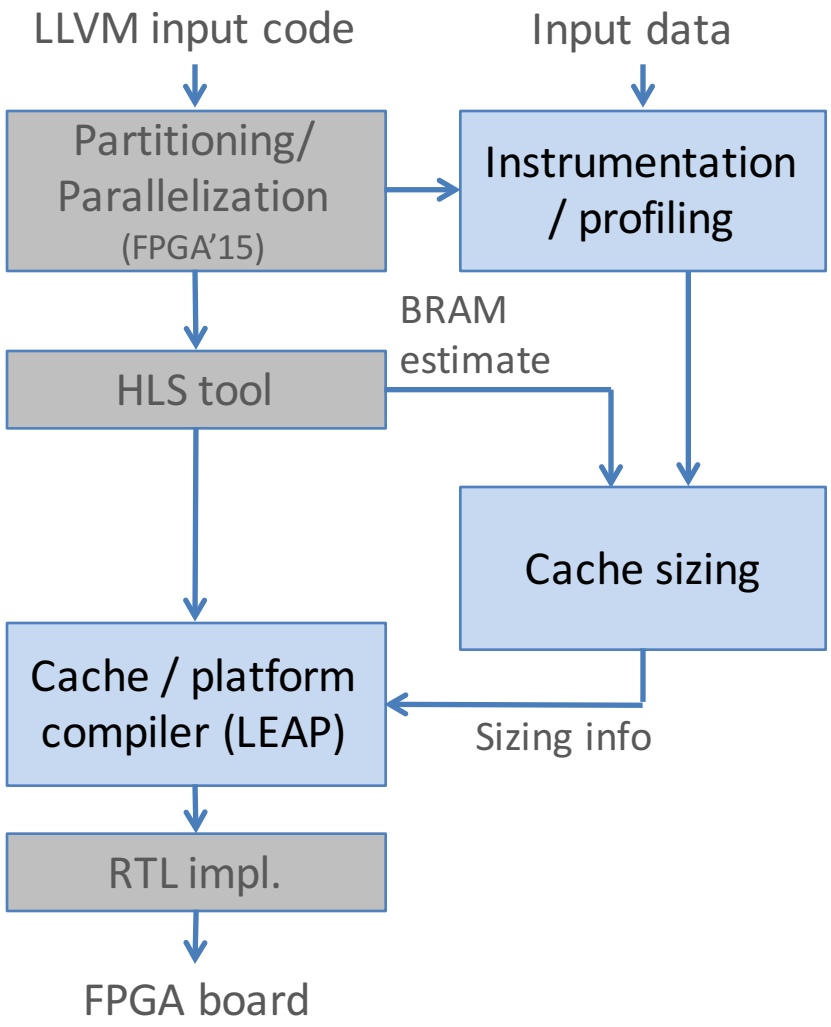
- Maximize aggregate hit rate of multi-cache system
- Scale up caches using "left-over" BRAM
- Assign individual size to each cache

Compared to uncached app:

- 3.6x latency reduction (average)
- 2.6x energy reduction (average)

Compared to default size:

- 1.4x latency savings from variable sizing (average)



Remainder of this talk

1. Estimate left-over BRAM
2. Variably-sized caches
3. Implementation
4. Evaluation

Estimate left-over BRAM

- BRAM estimate from HLS tool + known offsets
- ReflectTree example on a Virtex 7 FPGA:

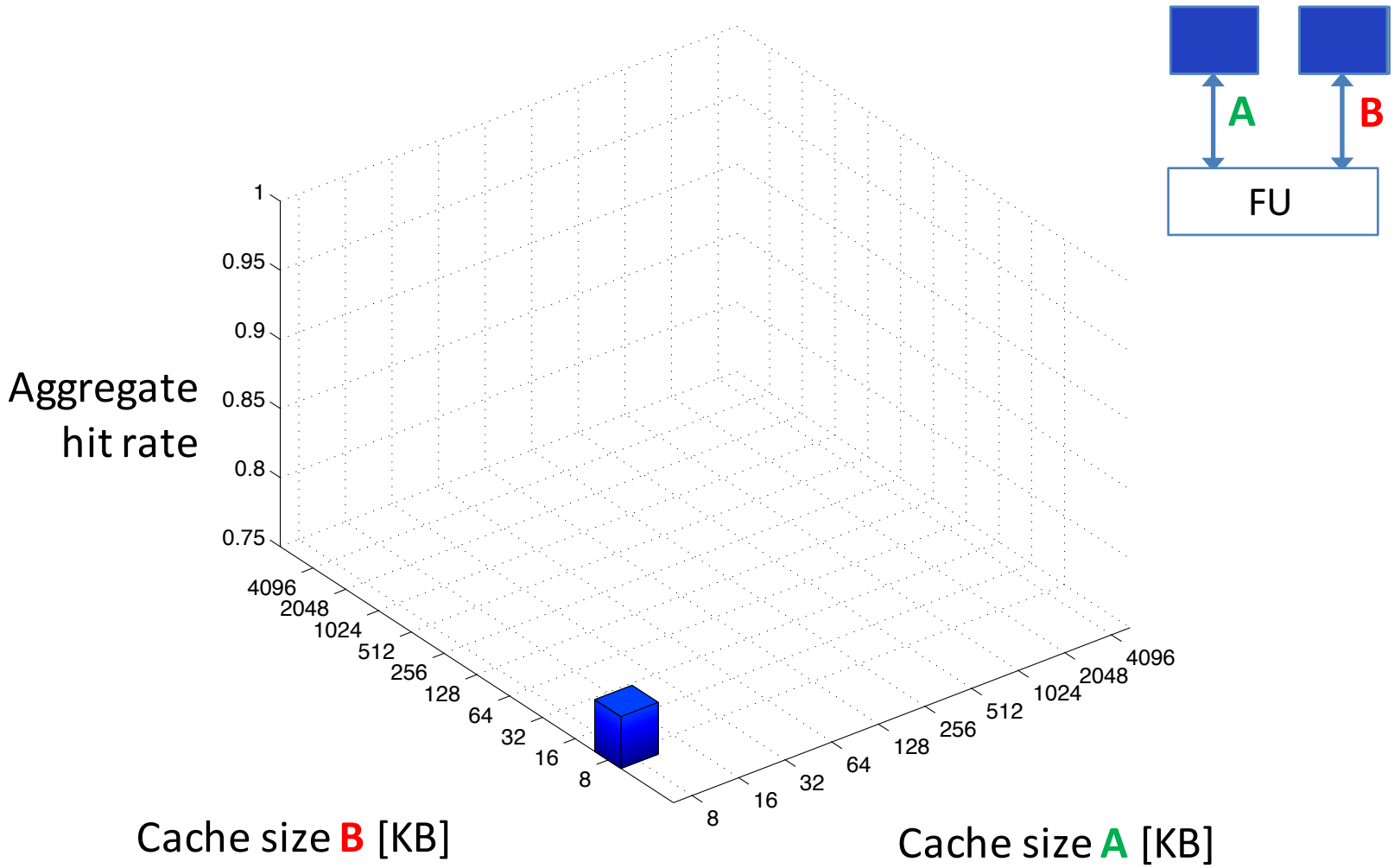
Component	Estimate [RAM blocks]	Post-PAR [RAM blocks]
HLS core	208	158
Memory interfaces	40	40
LEAP platform (without scratchpads, fixed offset)	50.5	50.5
Unused left-over BRAM (Virtex 7)	**628.5**	**678.5**

~2200 KB can be repurposed for cache construction

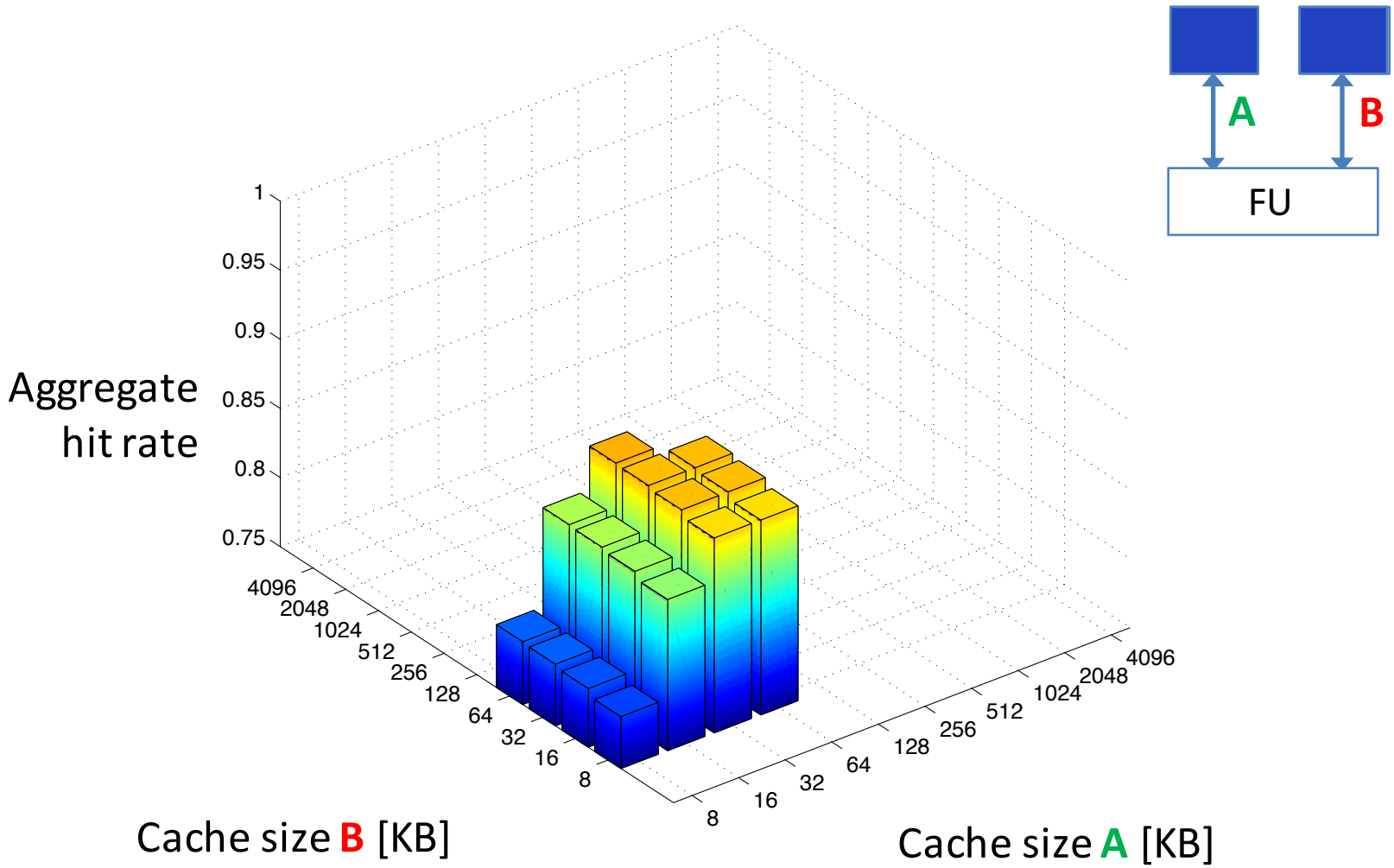
Outline

1. Estimate left-over BRAM
2. Variably-sized caches
3. Implementation
4. Evaluation

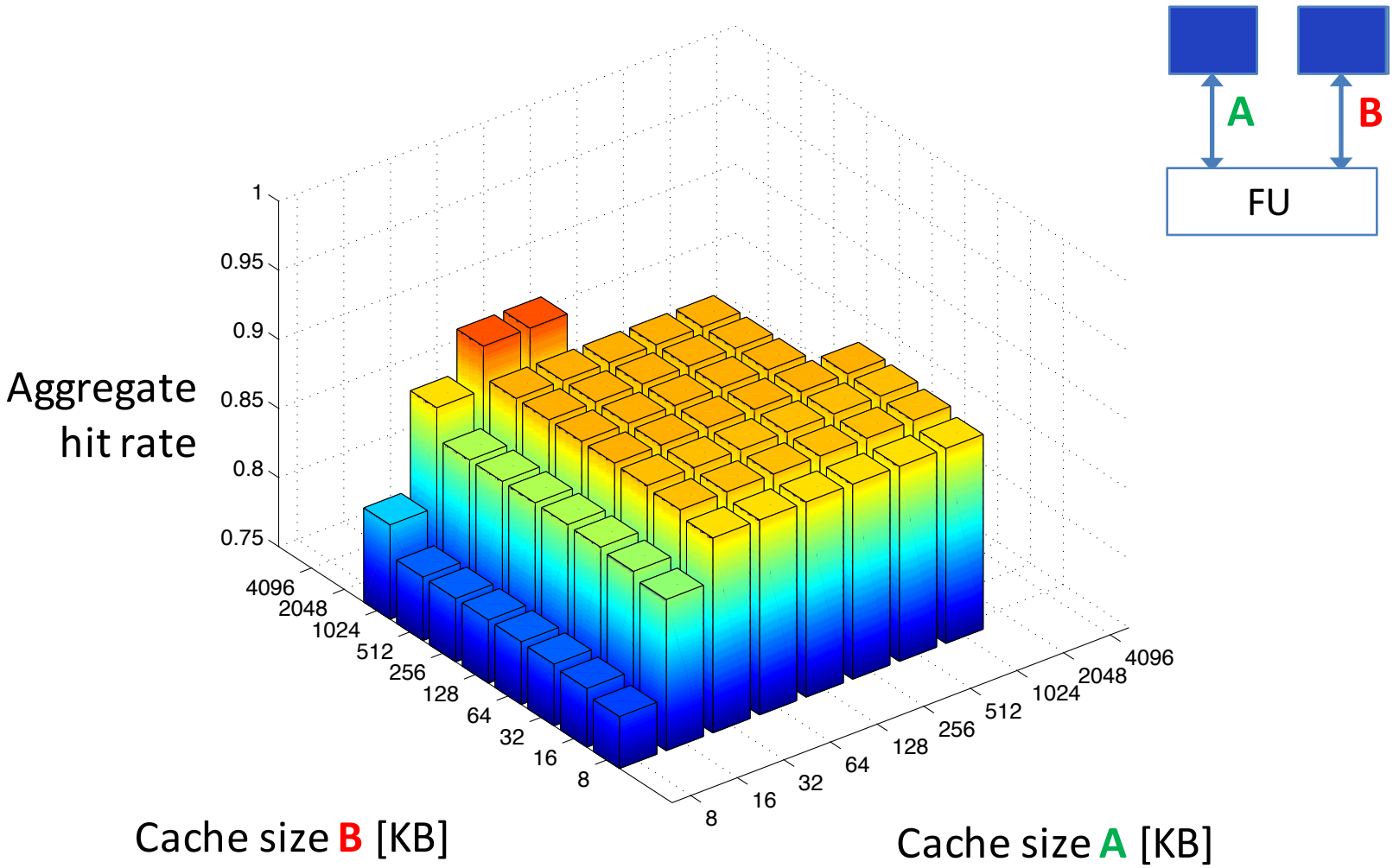
Variably-sized caches



Variably-sized caches

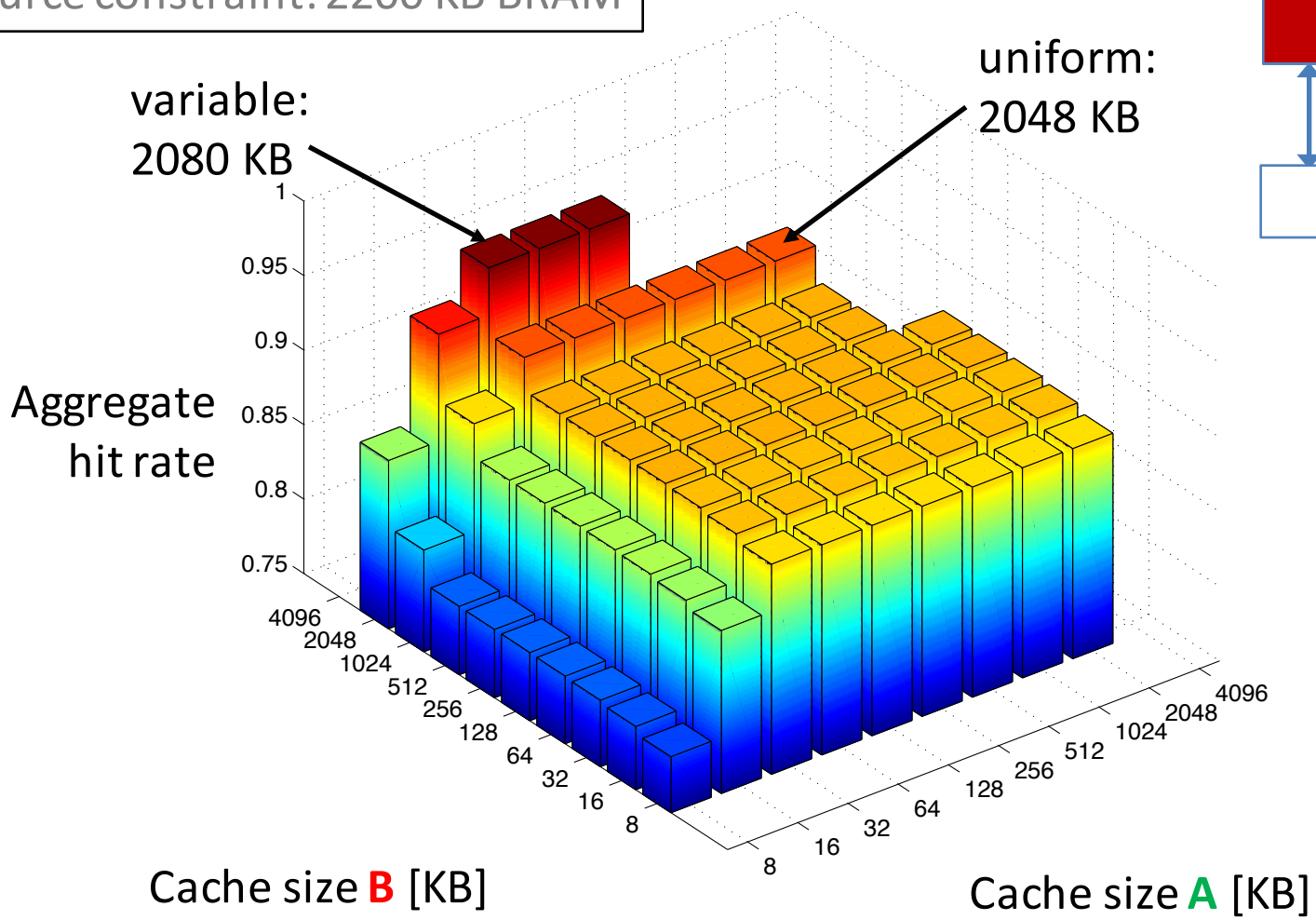


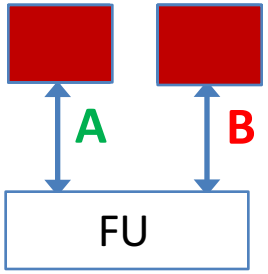
Variably-sized caches



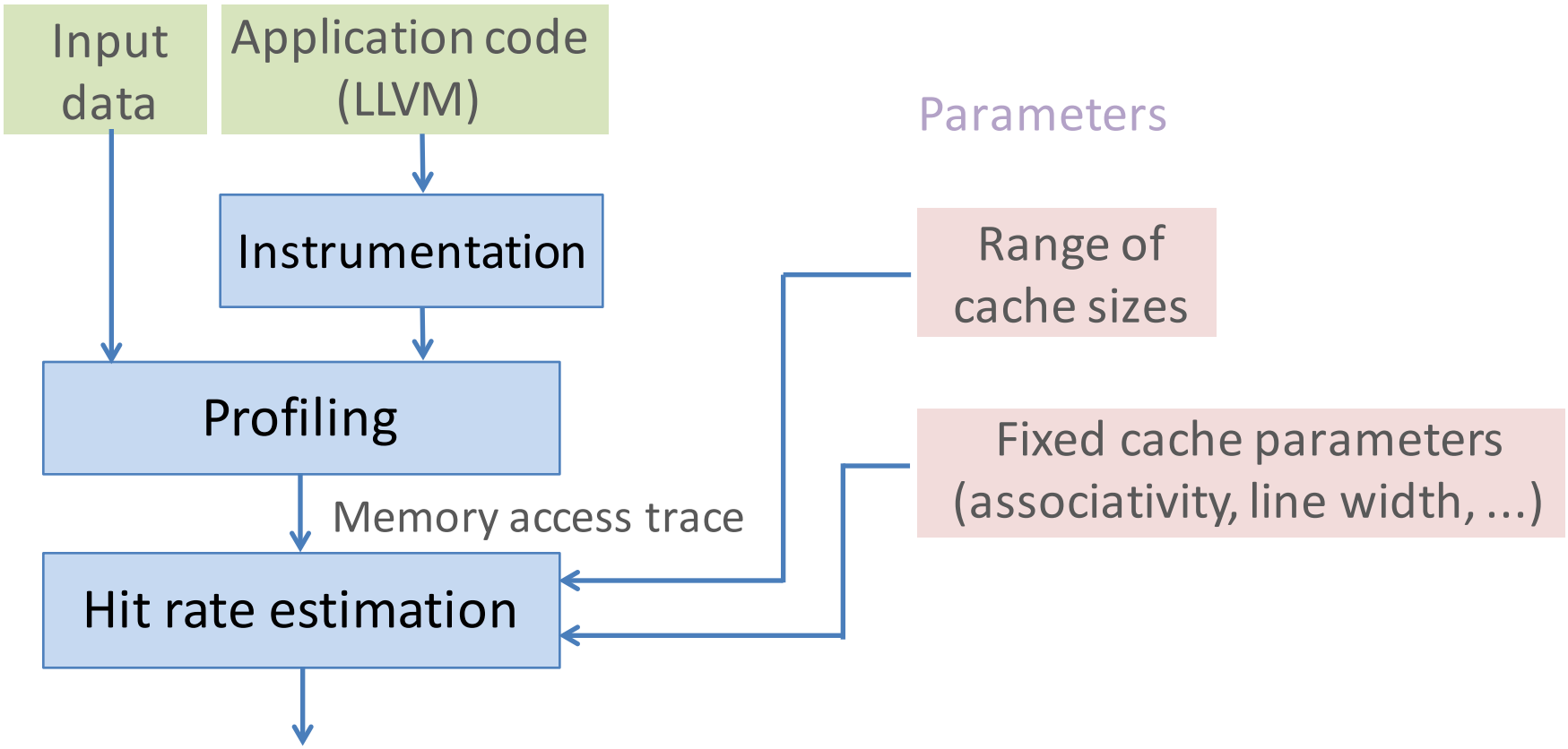
Variably-sized caches

Resource constraint: 2200 KB BRAM





Cache performance model



Output: Hit rate as function of cache size $h(B_j)$

Optimization strategy

Task:

- Find "optimal" size assignment for *K* caches
- Choose among *N* sizes for each cache

Cast into Multiple-Choice Knapsack Problem:

maximize $$\sum_{i=1}^{K}\sum_{j=1}^{N} h_i(B_j)x_{ij}$$

hit rate for cache size B_j

subject to $$\sum_{i=1}^{K}\sum_{j=1}^{N} BRAM_i(B_j)x_{ij} \leq C$$

BRAM usage for cache size B_j

Global BRAM constraint

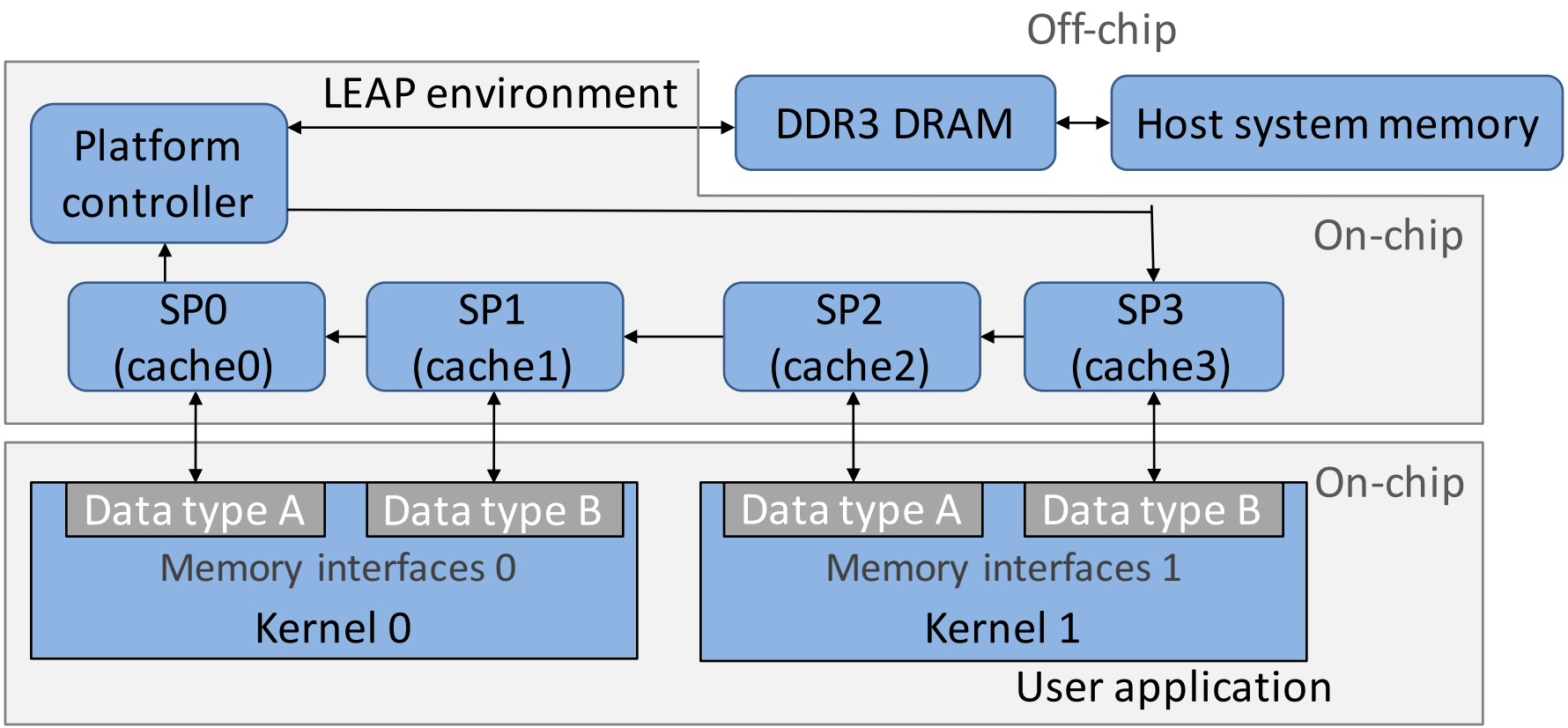
and $$\sum_{i=1}^{N} x_{ij} = 1, \qquad i = 1 \ldots K$$

Select each cache exactly once

Outline

1. Estimate left-over BRAM
2. Variably-sized caches
3. Implementation
4. Evaluation

Implementation



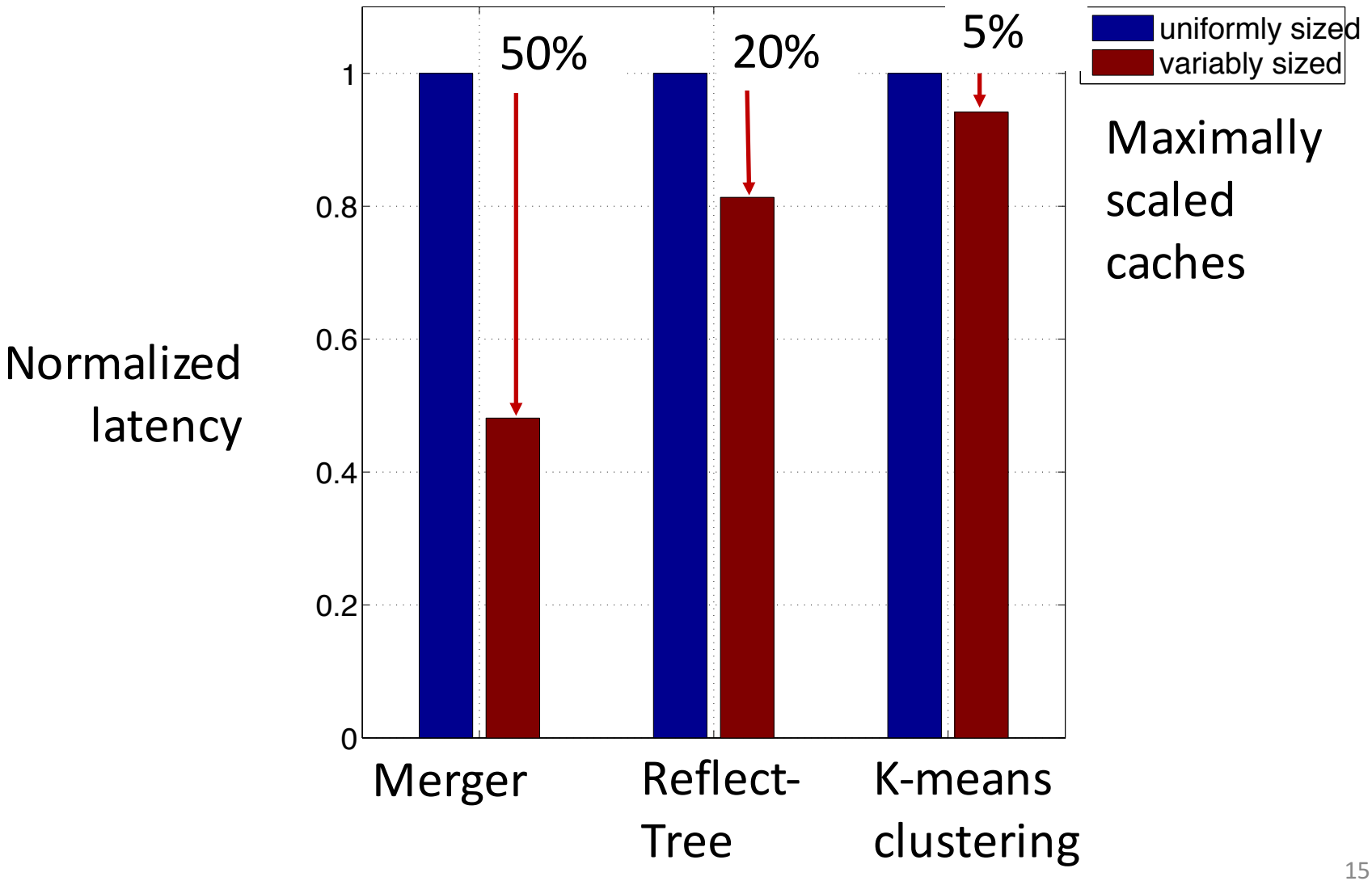
Outline

1. Estimate left-over BRAM
2. Variably-sized caches
3. Implementation
4. Evaluation

Results

		LUT	FF	BRAM	Latency	
1	**Merger (8 scratchpads)**					
	No caches	67k	67k	586.5	7.7 ms	x5.0
	Variably-sized, scaled caches	92k	88k	874.5	1.5 ms	
2	**ReflectTree (4 scratchpads)**					
	No caches	74k	77k	248.5	345.6 ms	x3.4
	Variably-sized, scaled caches	85k	83k	944.5	100.3 ms	
3	***K*-means clustering (6 scratchpads)**					
	No caches	91k	91k	347.5	598.7 ms	x2.7
	Variably-sized, scaled caches	107k	103k	829.5	221.8 ms	

Benefit of custom sizing



Energy

		Energy (FPGA+DRAM)	
1	**Merger (8 scratchpads)**		
	No caches	22.0 mJ	x3.9
	Variably-sized, scaled caches	5.6 mJ	
2	**ReflectTree (4 scratchpads)**		
	No caches	1060.5 mJ	x2.2
	Variably-sized, scaled caches	477.6 mJ	
3	***K*-means clustering (6 scratchpads)**		
	No caches	1976.9 mJ	x2.0
	Variably-sized, scaled caches	973.5 mJ	

Conclusion

- Automatic cache size scaling in a multi-cache system
  - Using spare BRAM resources
  - Profiling-based hit rate estimation
  - Find the resource allocation that maximizes aggregate hit rate
- Future work
  - Model of a coherency network
  - Optimal partitioning of the memory controller network
  - High-level energy estimation and explicit minimization of energy consumption

Questions?

Thank you.