Abridged Addressing: A Low Power Memory Addressing Strategy

Preeti Ranjan Panda

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Asia and South Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, Jan 24-27 2006

Previous Work

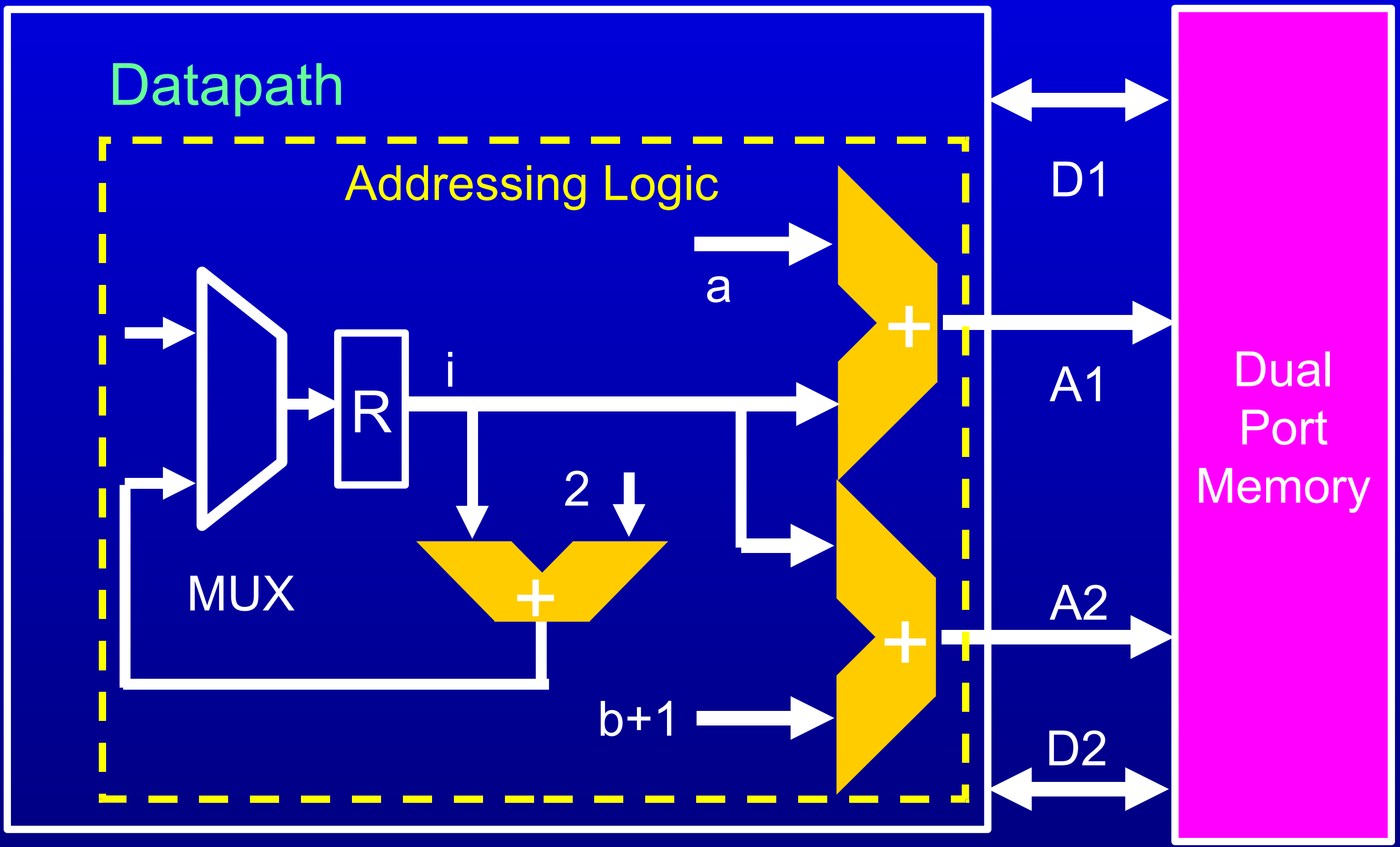
- Bus encoding
  - Bus-invert [Stan & Burleson ‘95]
  - Gray code [Su & Despain ‘95]
  - T0 [Benini ‘98], etc.
- Memory address bus
  - Data organisation [Panda & Dutt ‘96]
  - Working zone encoding [Musoll et al. ‘98]
  - Multiplexed address bus [Mamidipaka et al. ‘01]

Data Memory

- Addresses not sequential
- WZE handles to some extent
- CDMI
  - decoder is power-expensive
- Abridged Addressing
  - minimise use of registers
  - reduces energy in address bus + decoder

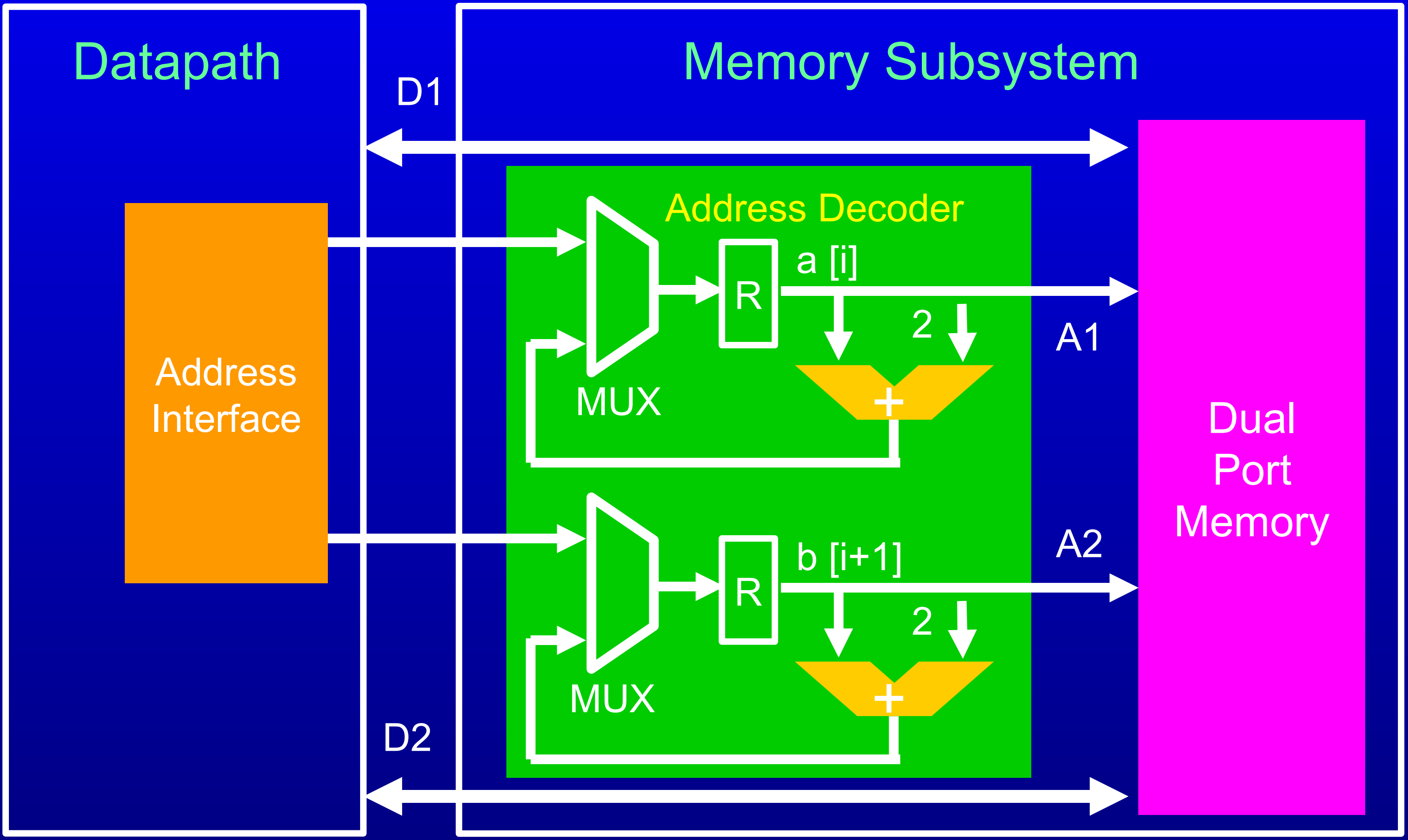
Memory Addressing in Synthesis

```
for (i = 0; i < n; i++)
   x = a [i] + b [i+1]; ...
```



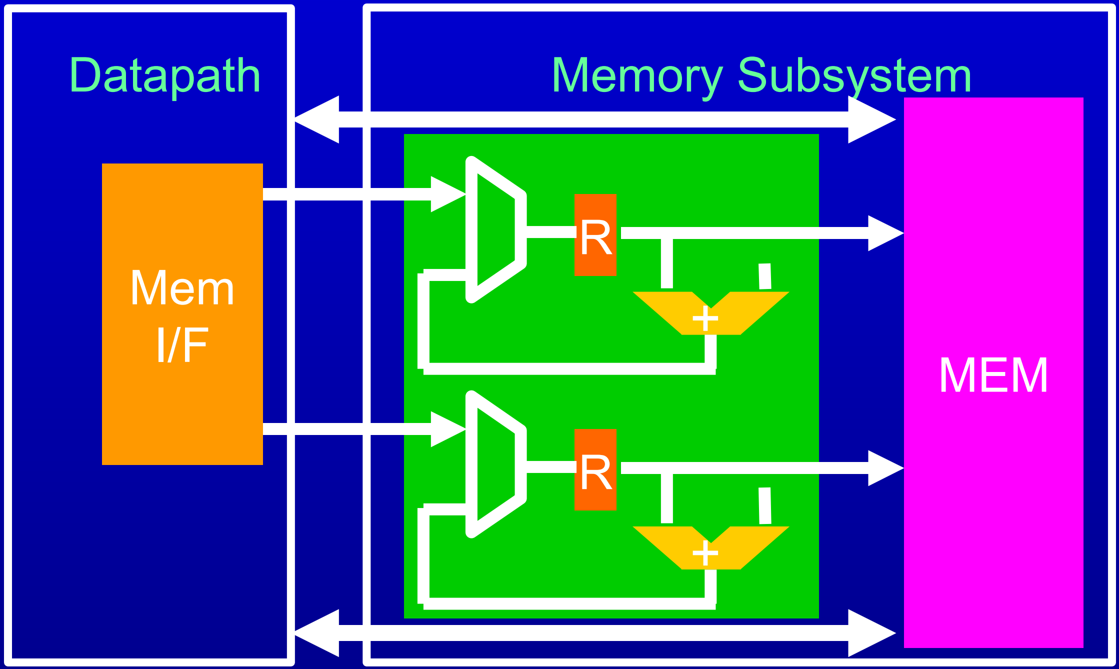
Optimising Memory Addressing

Address sent to memory only in first iteration [CDMI]



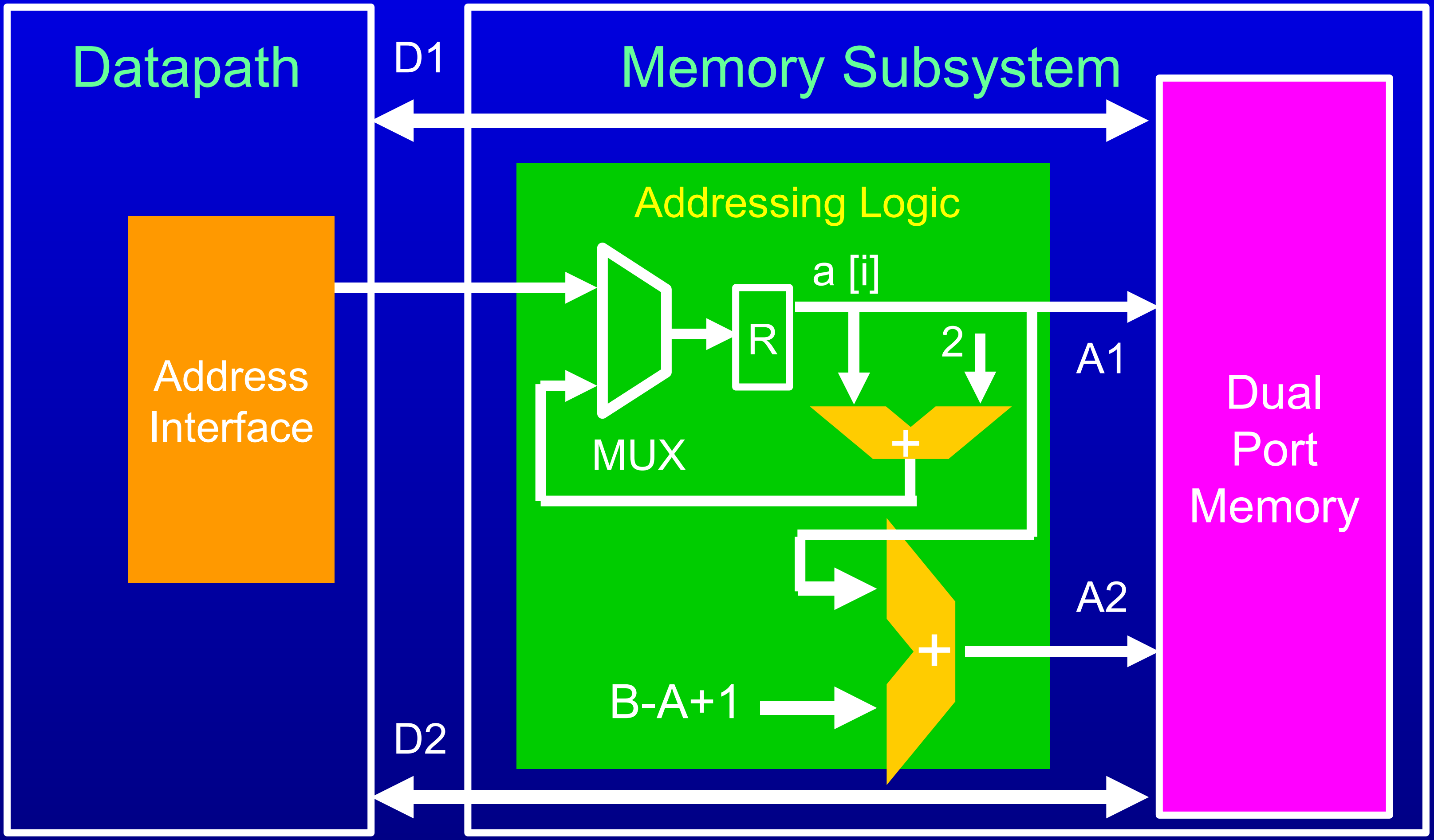
Optimised Addressing

- Good when buses are long wires
- Significant decoder overhead
  - one register for every memory access
    - Registers organised into RF for each port
    - Register File READ and WRITE for each memory access
  - may not be worthwhile for on-chip buses



Abridged Addressing

Transmit only one address to memory subsystem. Derive the rest.



Abridged Addressing: Comparison

- Only one address needs to be sent
  - others are internally generated in memory subsystem
    - using additions to constant offsets
  - avoids power-inefficient Register File READs and WRITEs
  - replaced by simple arithmetic operations
    - addition to constant
- 23% power reduction

Generalising to Multiple Accesses

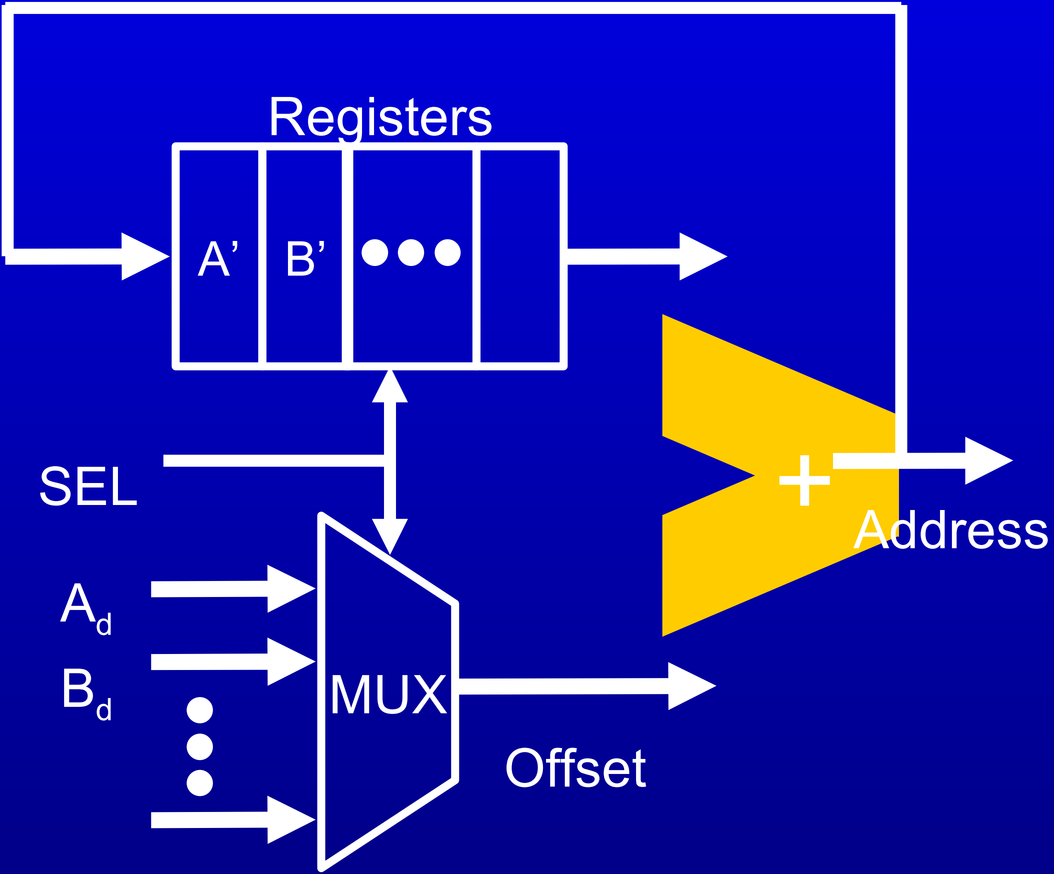
```
int A[$N_1$][$N_2$]...[$N_r$]

for ($i_1 = l_1$; $i_1 < h_1$; $i_1$ += $s_1$)
  for ($i_2 = l_2$; $i_2 < h_2$; $i_2$ += $s_2$)
  ...
     for ($i_n = l_n$; $i_n < h_n$; $i_n$ += $s_n$)
     READ A [$c_{11}i_1 + c_{12}i_2 + \ldots + c_{1n}i_n + k_1$]
            [$c_{21}i_1 + c_{22}i_2 + \ldots + c_{2n}i_n + k_2$]
            ...
            [$c_{r1}i_1 + c_{r2}i_2 + \ldots + c_{rn}i_n + k_r$]
```

Address different between two successive iterations:

$A_d = (N_2 N_3 \ldots N_r)c_{1n} + (N_3 \ldots N_r)c_{2n} + \ldots + N_r c_{r-1n} + c_{rn}$ $\leftarrow$ compile-time constant

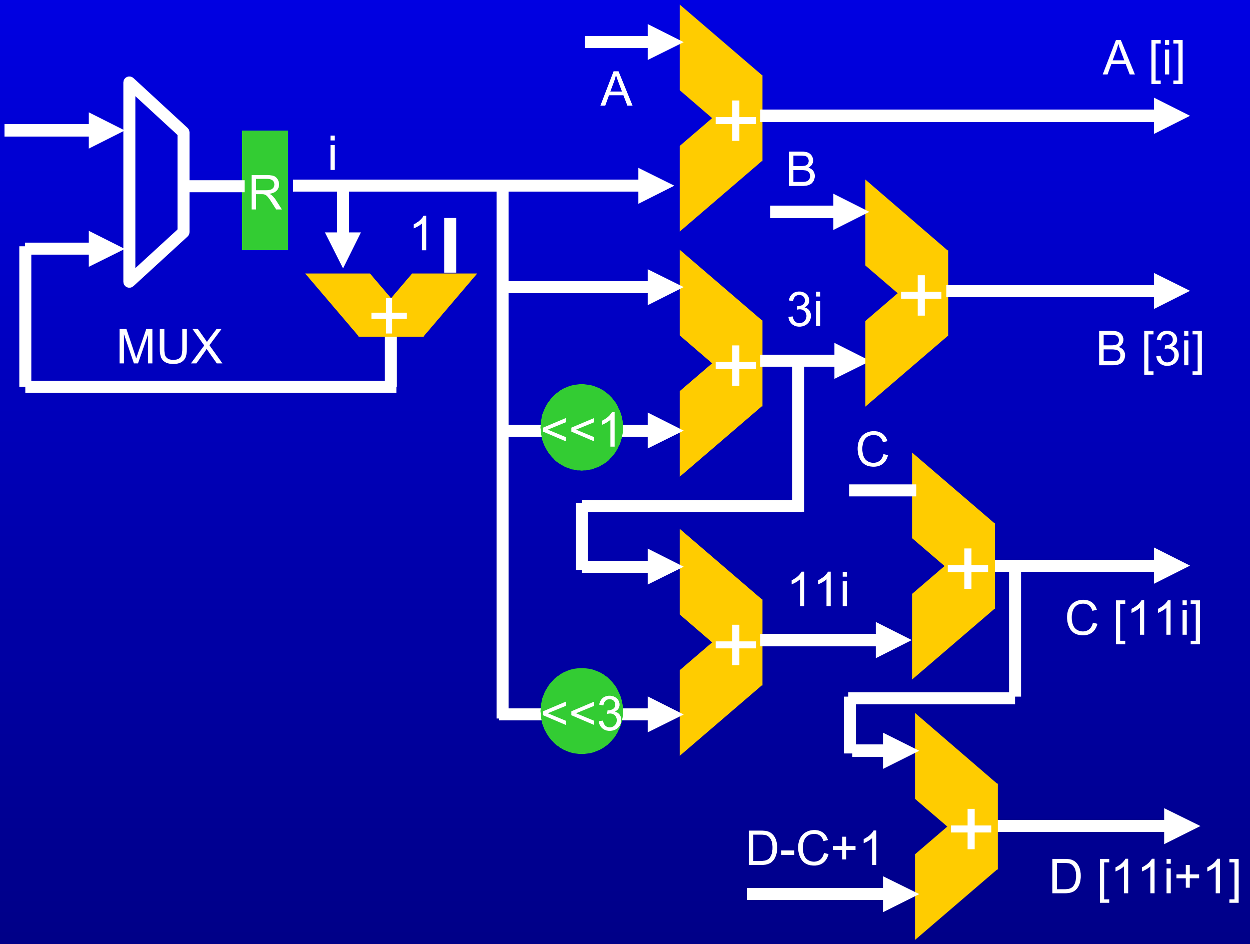
Addressing Circuit for Multiple Accesses

- SEL signal chooses memory access
  - generated by local FSM
- New address = $A'+A_d, B'+B_d$, etc.
- Register File initialised before loop begins



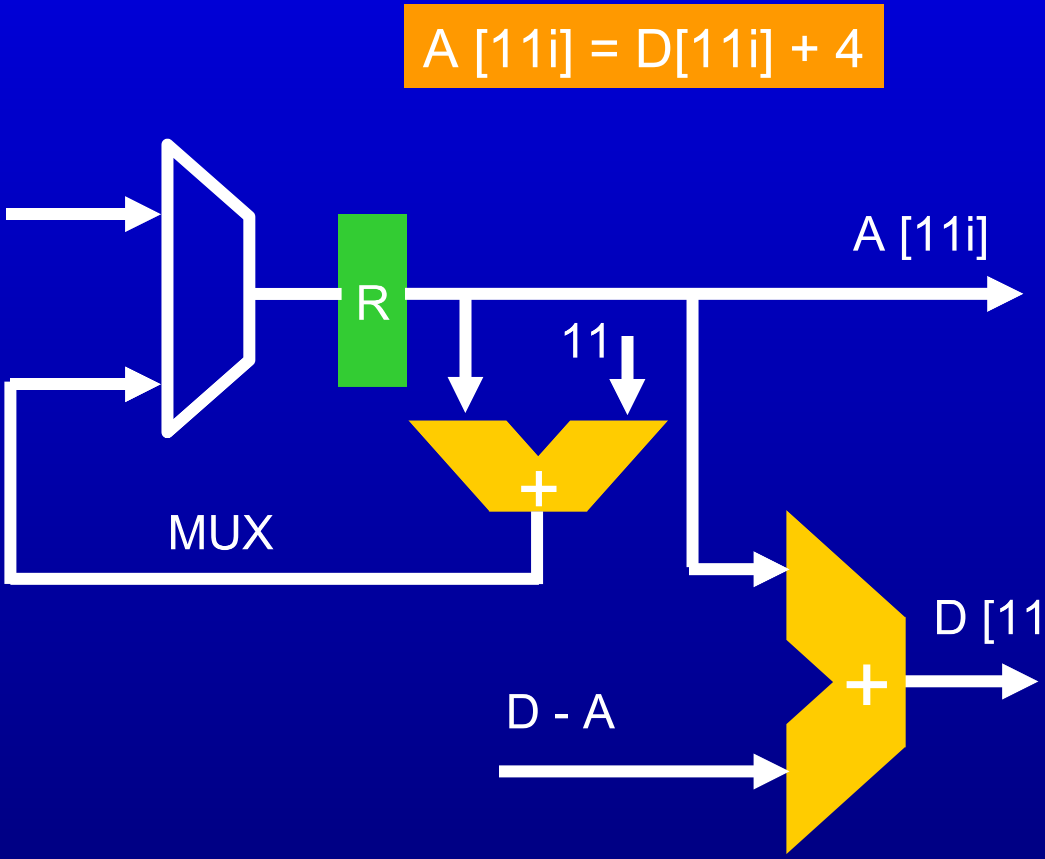
Abridged Addressing Logic Structure

- Only one address value stored in register
- Network of adders
  - no multiplier
  - no shifter
- Final level of MUX depending on ports



Need not choose loop induction variable for storage

- Choosing i leads to more adders
  - generating 11i from i requires 3 additions
- Instead, store A[11i]
  - generate D[11i] by adding D - A

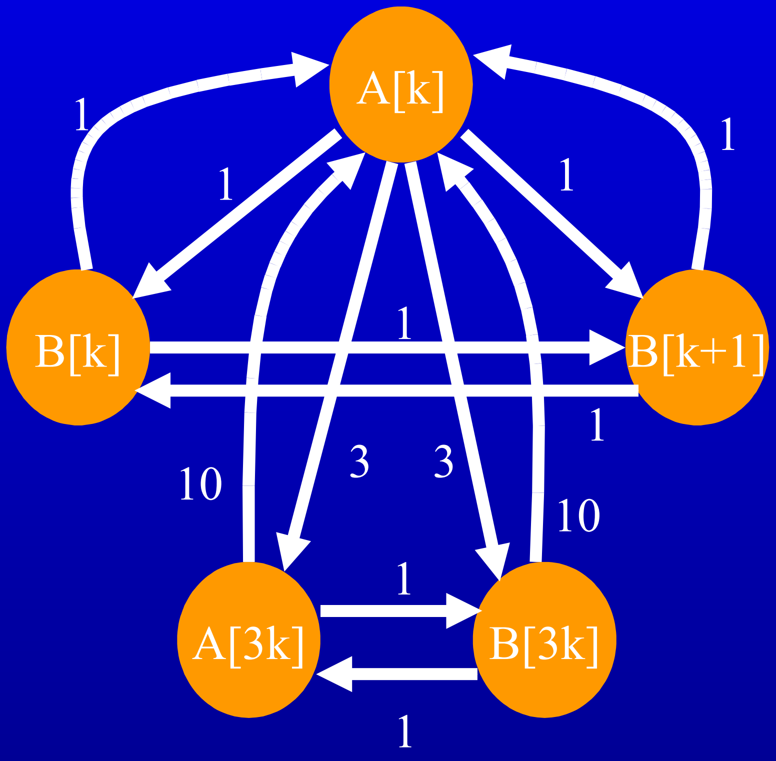


Problem Statement

- Given a set of memory references in a loop
  - each translates to an address output
- Determine a power-efficient addressing logic structure

Graph Formulation

- Construct Graph G(V,E)
- V = set of nodes
  - node = array access
- E = set of edges i→j
  - w(i,j) = cost of deriving j from i
  - cost = number of adders
    - some edge wts are high as derivation is expensive

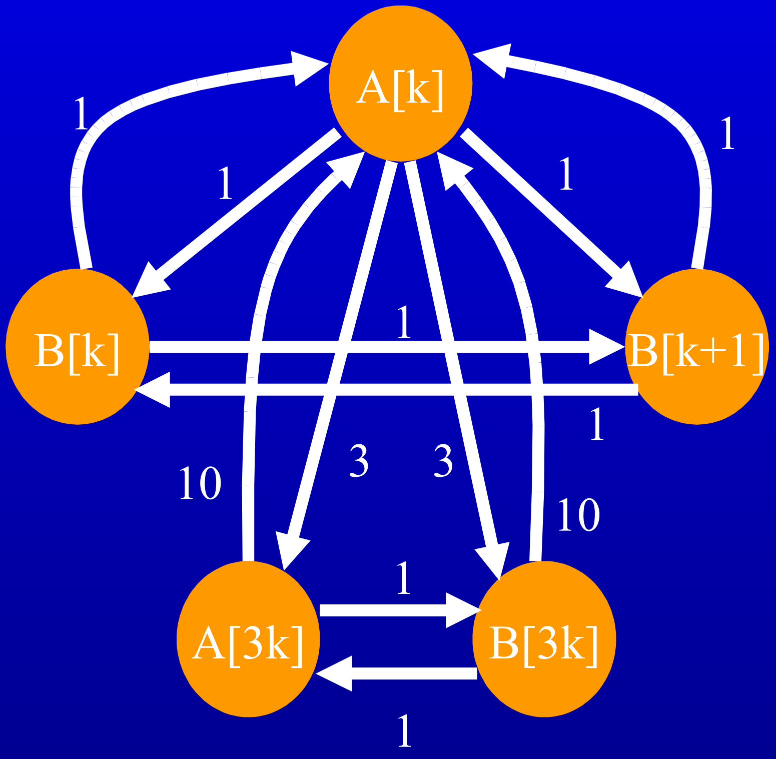


All edges not shown

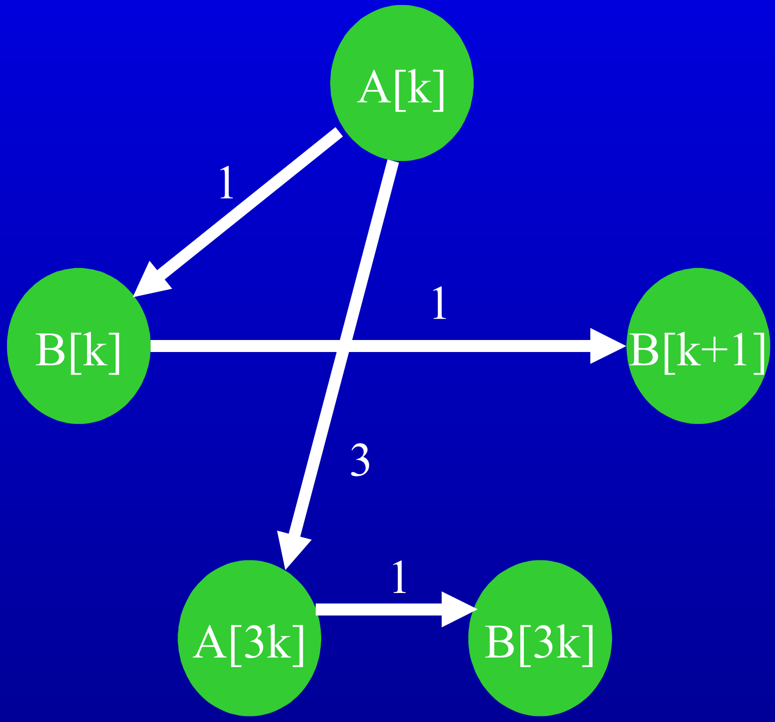
Power-Efficient Solution: Directed Minimum Spanning Tree

- Simplest Adder Network = Minimum Spanning Tree (MST)
- Standard MST solution does not work
  - directed graphs, asymmetry [$w(i,j) \neq w(j,i)$]
- Use Directed Minimum Spanning Tree (DMST)
  - MST on Directed Graph
  - Chu-Liu's algorithm
  - Polynomial time

DMST Example



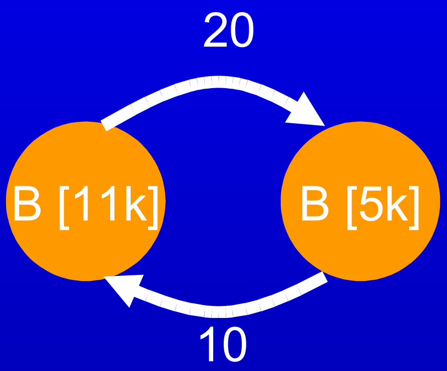
Original Graph



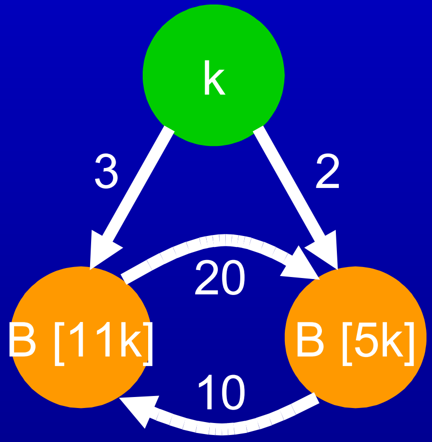
DMST

Augmenting the Graph

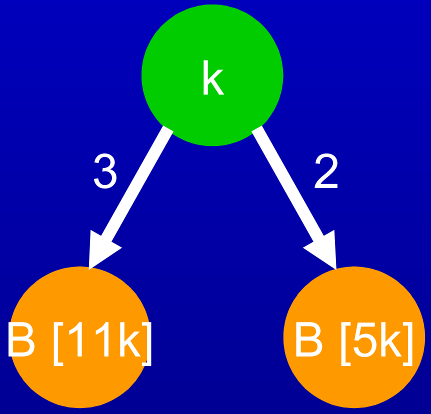
- Adding an auxiliary node may result in more efficient addressing logic
- Add node corresponding to loop induction variable k
  - implicit addition of nodes 2k, 4k, etc.
  - no extra hardware



Original Graph



Augmented Graph



DMST OF Augmented Graph

Overall Strategy

- Find DMST for both original graph G and augmented graph G'
- Choose solution with lower cost
- DMST leads directly to adder network

Analysis of Addressing Logic

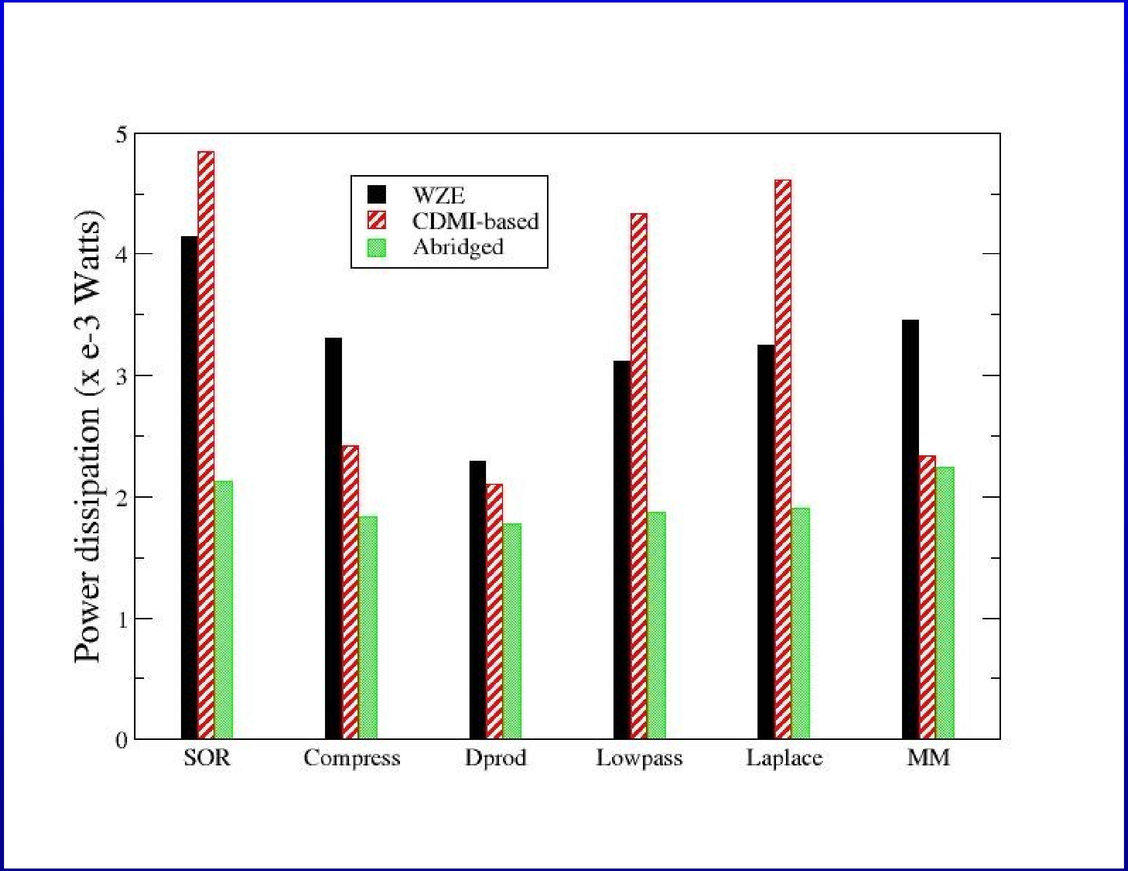
- Only one register Read/Write per loop iteration
  - instead of read/write on every memory access
  - minimise redundant storage
    - helps reduce power-expensive register file accesses
- More adders (but fewer multipliers)
  - each adder activated only once per iteration
  - NOT an overhead
    - computation shifted from Datapath to Memory subsystem
    - also worth performing in the Datapath
- Interface independent of memory ports
  - only one address bus
- Multiple loops
  - analyse independently
  - adder sharing decisions of common parts taken later
  - addressing logic NOT on the critical path

Experiments

- Dual port RAM, 0.18μ ASIC Library, 100 MHz clock
- Addressing logic synthesised and simulated
- Switching activity fed to power simulator (Synopsys Prime Power)
- Comparison with WZE, CDMI
  - these perform better than other proposed techniques

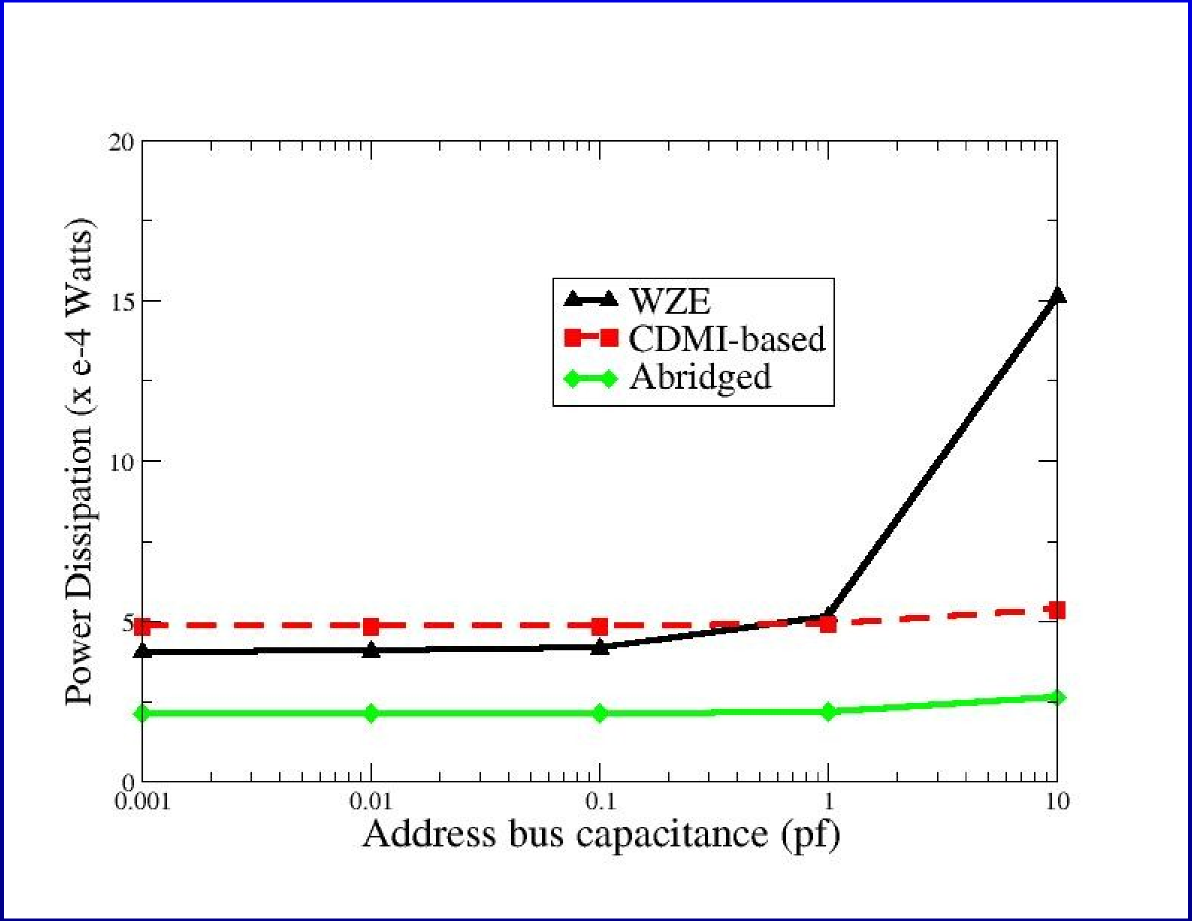
Results

- On-chip address buses
- Decoder overhead of CDMI is significant
- Abridged addressing is more power-efficient
  - 40% over WZE
  - 44% over CDMI
- Area
  - 65% less than CDMI
  - 10% more than WZE



Variation with Bus Capacitance

- Effect of longer wires
- Decoder overheads relatively smaller
- Addressing logic simpler in Abridged
  - suitable for entire range



Conclusion

- Abridged Addressing
  - power efficient memory addressing circuitry
  - targets both address bus and decoder logic
    - minimal information transmitted on address bus
    - omits redundant storage in decoder
      - replace by simple computation
  - suitable for both on-chip and off-chip memory