Using Internal Redundant Representations and Limited Bypass to Support Pipelined Adders and Register Files

Mary D. Brown Yale N. Patt

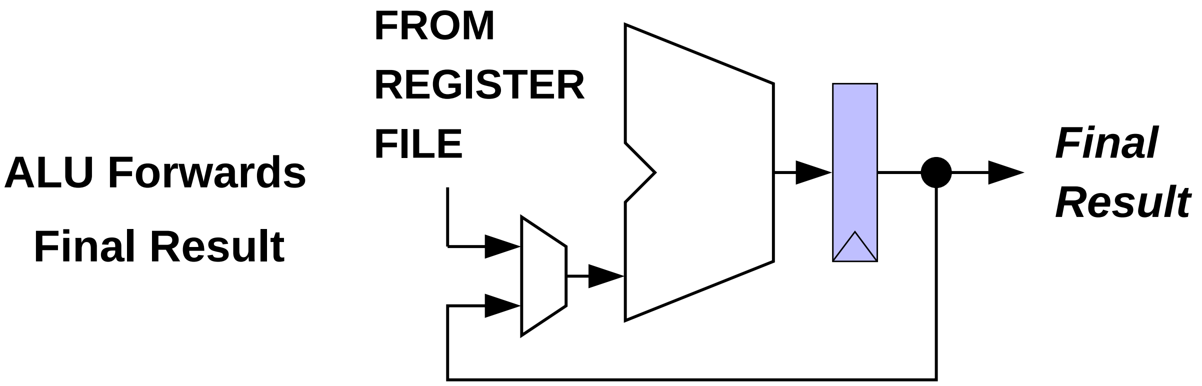
Dept. of Electrical and Computer Engineering
The University of Texas at Austin
{mbrown,patt}@ece.utexas.edu

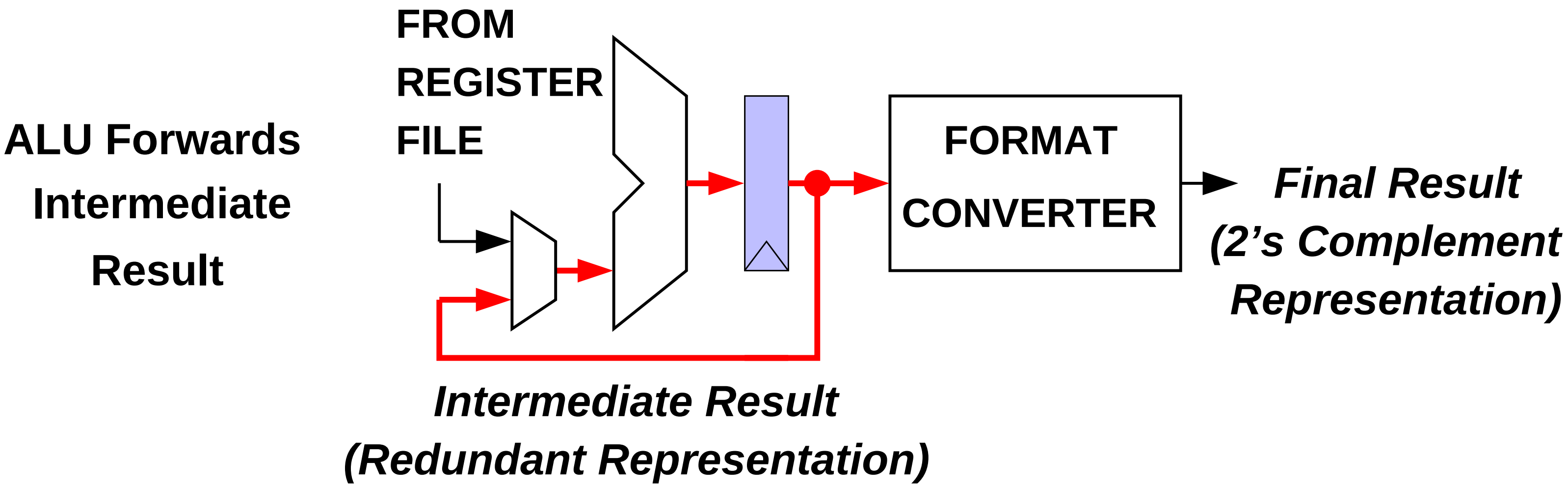
Increasing Execution Bandwidth

1. Faster ALUs
   - better circuit techniques, faster transistors
   - shorter execution latency
   - possibly higher power consumption
2. More ALUs
   - higher IPC
   - larger bypass muxes, longer bypass wires
   - longer execution latency
3. Pipelined ALUs
   - higher frequency
   - more latches means more overhead time
   - longer execution latency

How do we get high bandwidth *and* low latency?

Forward an intermediate result.



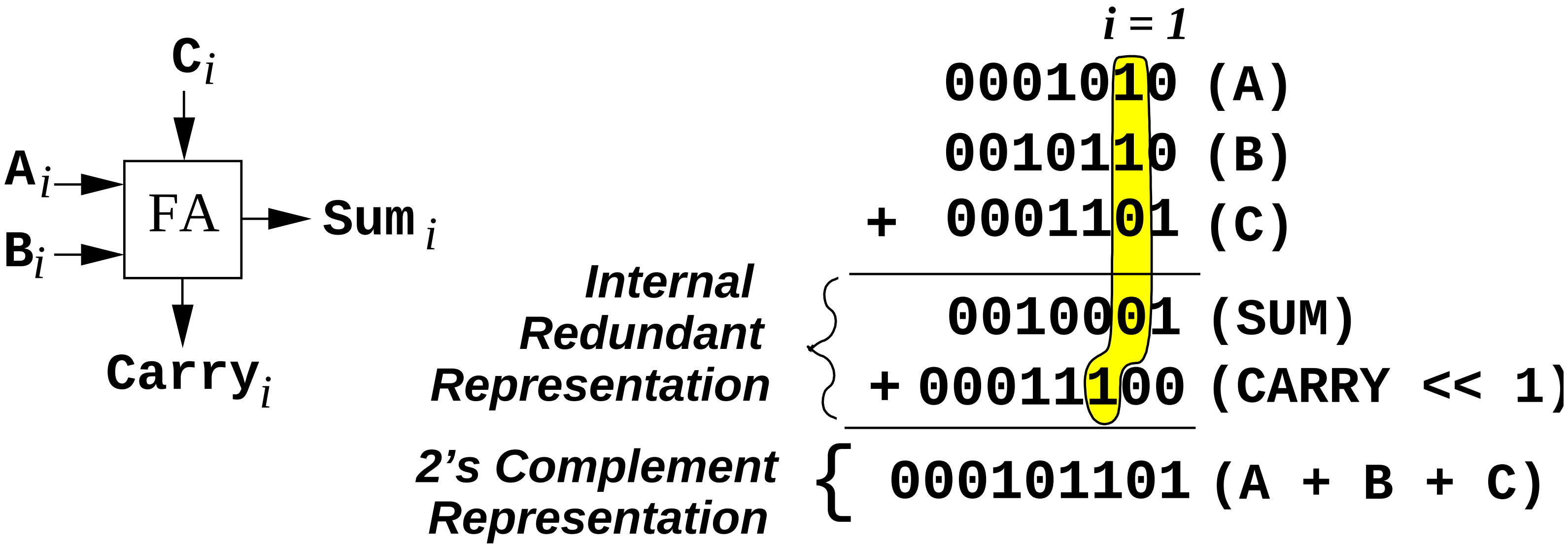


Outline

1. Tutorial of redundant binary representation
   - Two examples: Carry-Save and Signed-Digit
2. Using redundant binary functional units for a 2's complement ISA
3. Execution Core Configurations
4. Experiments

Redundant Number Representations

- A value can have more than one encoding.
- Result of carry-free addition
- Example: Carry-Save (3-to-2) Adder



- Many numbers can be added quickly before doing one addition with full carry propagation at the end.

Limitations of Redundant Number Representations

- Conversions from Redundant to 2's Complement require a carry-propagating add or subtract.
- Require more bits to encode numbers.
- Can't test for equality.
- Can't do logical operations (AND, XOR, etc.).
- Test for positive / negative may be difficult.
- Test for zero may be difficult (depends on representation).

Example: Redundant Binary Numbers

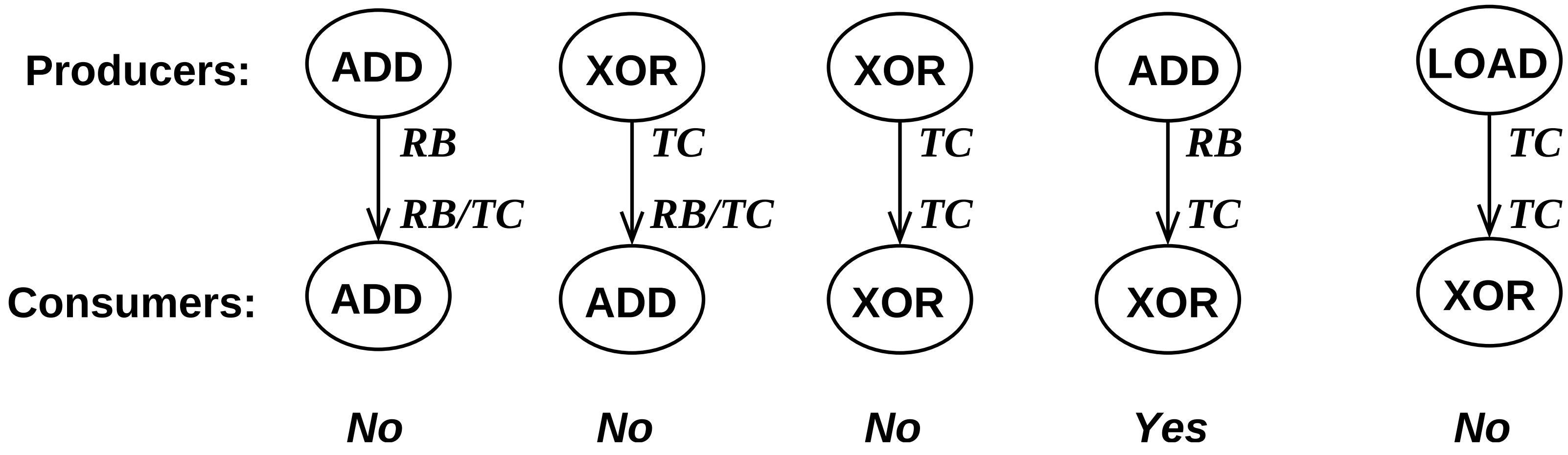
- *Signed Digit* – Each digit takes on a value in {-1, 0, 1} [Avizienis, 1951].
- Each digit represents a power of 2 multiplied by -1, 0, or 1.
  - $\langle 0, 1, 0, -1 \rangle$ represents $1 \cdot 2^2 - 1 \cdot 2^0 = 3$.
  - $\langle 0, 0, 1, 1 \rangle$ also represents 3.
- Encoding: 2 bits for each digit (or 2 bit vectors: positive and negative powers of 2).
  - Example: For $\langle 0, 1, 0, -1 \rangle$, positive bit vector is 0100, negative bit vector is 0001.
- One representation for zero: $\langle 0, 0, 0, 0 \rangle$.
- Addition without carry propagation.

Format Conversions

- Redundant to 2's Complement: addition or subtraction with full carry propagation.
  - Example: $\langle 0, 1, 0, -1 \rangle$ is $0100 - 0001 = 0011$.
- 2's Complement to Redundant: no conversion necessary.
  - Example with Redundant Binary:
    positive bit vector[MSB-1:0] := 2's Comp[MSB-1:0]
    negative bit vector[MSB-1:0] := 0
    positive bit vector[MSB] := 0
    negative bit vector[MSB] := 2's Complement[MSB]

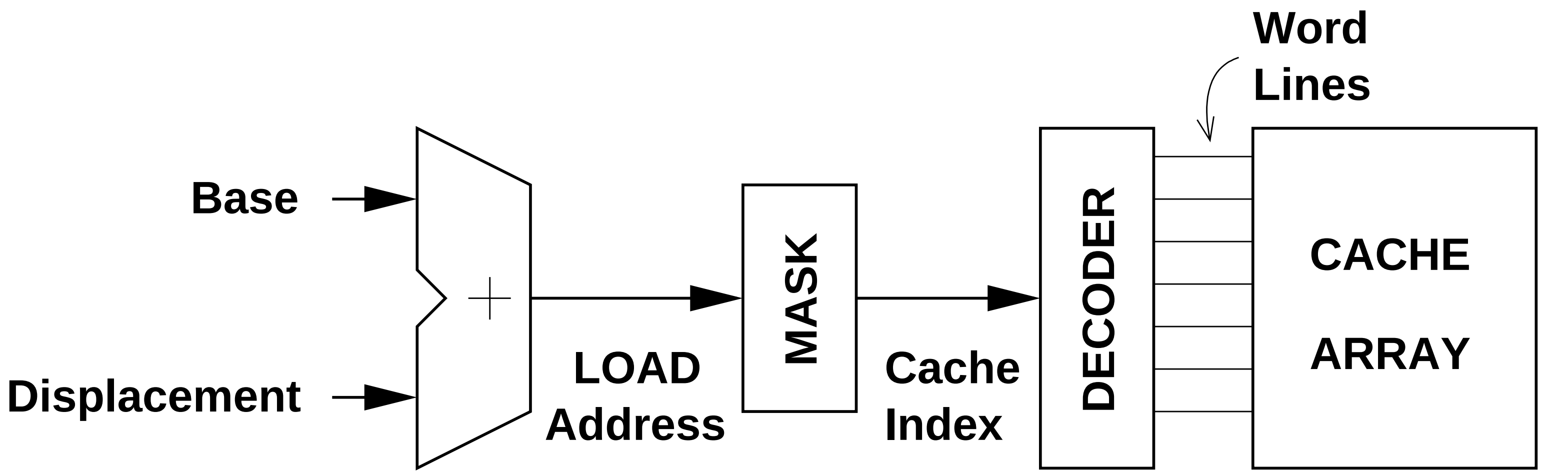
```
0011(3)                          1101(-3)
 ↓↓↓                              ↓↓↓
0011  positive bit vector        0101
0000  negative bit vector        1000
```

When are format conversions needed?



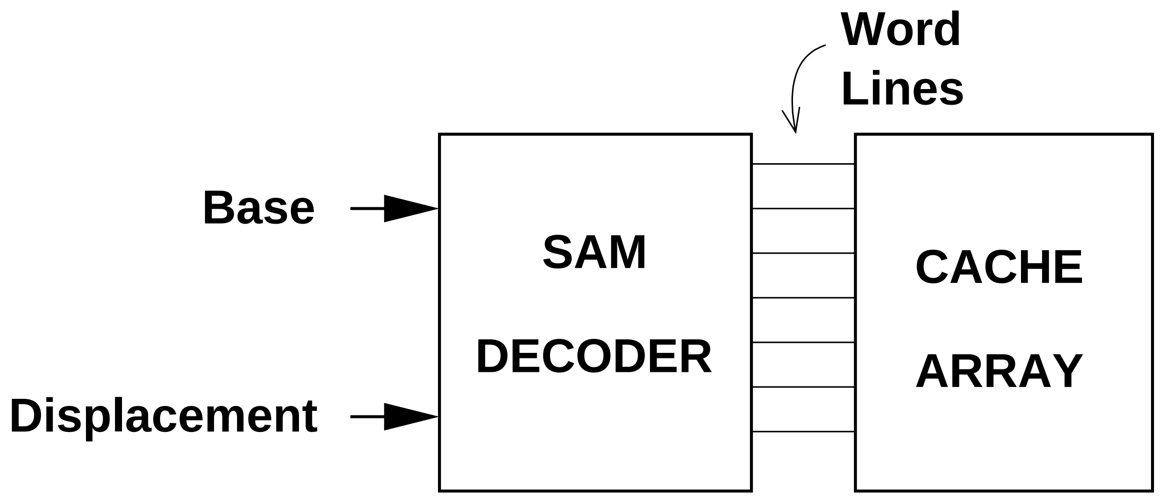
LOAD Addresses

- Cache index must be in a unique representation (2's Complement).



LOAD Addresses, contd.

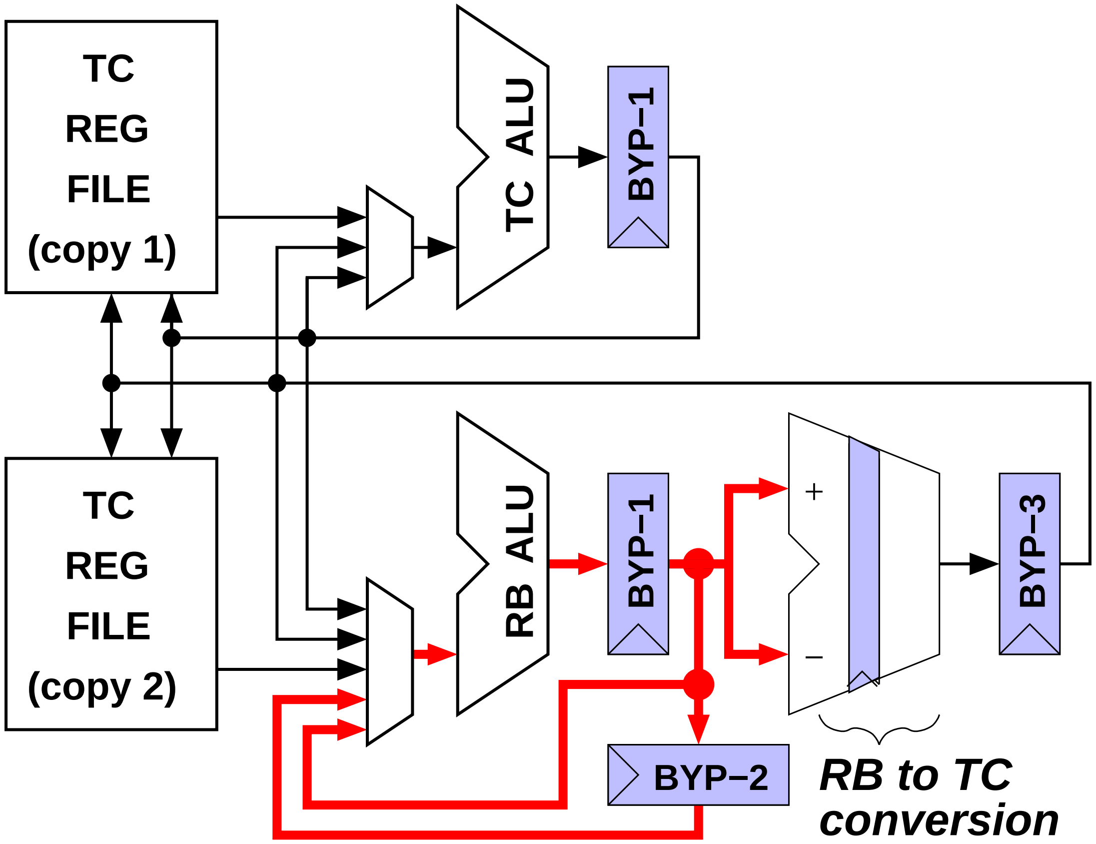
- Sum Addressed Memory *[Lynch, Lauterbach, Chamdani]* avoids Base + Displacement addition.
- Modifed SAM avoids Redundant Base (2 vectors) + 2's Complement Displacement (1 vector) addition.



Execution Core Configurations

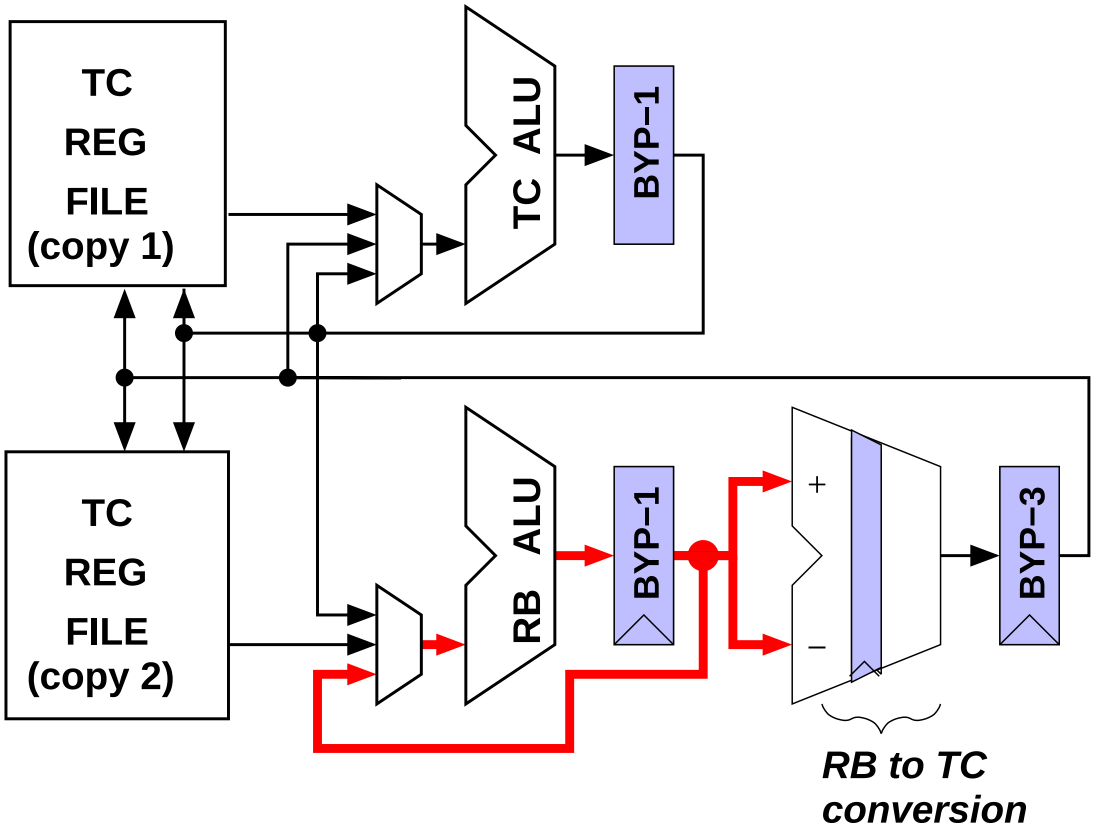
- Only 2's Complement register files.
- Both 2's Complement and Redundant Binary register files.

Using Only 2's Complement Register Files



- Output of RB ALU feeds into format converter.
- Extra levels of bypass may be needed.

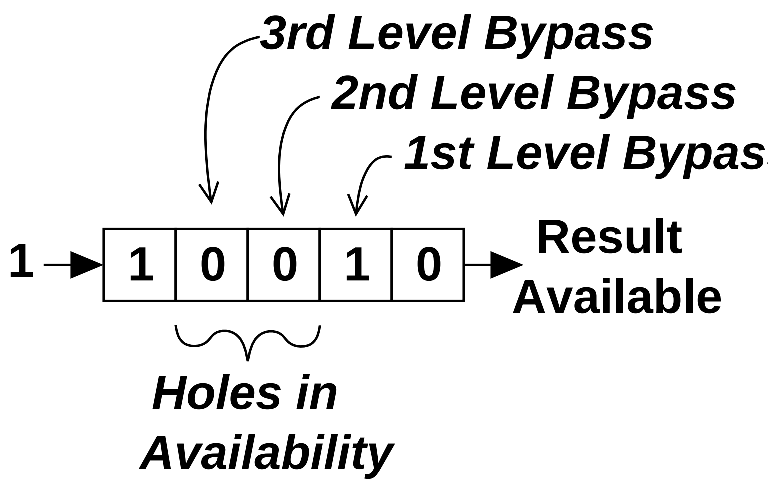
Limited Bypass Network



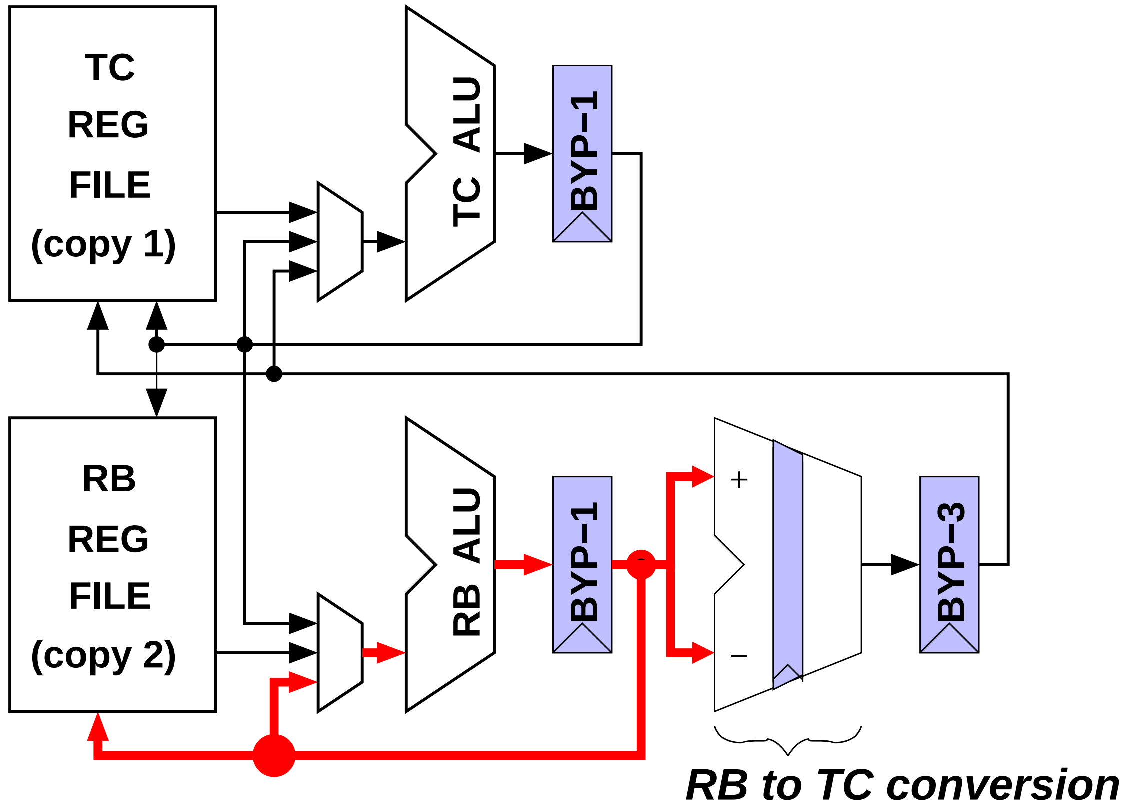
- No 2nd or 3rd level bypass from RB ALUs to RB ALUs.
- "Holes" in result availability [Cruz, González, Valero, Topham].

Scheduling for Holes in Availability

- An instruction can execute when all of its source operands are available.
- A shift register indicates when a result is available.
- Example for no 2nd or 3rd level bypass:



Using Both 2's Complement and Redundant Binary Register Files



- No extra bypass paths.
- RB register file is wider.

Impact of Format Conversions

- 20% of instructions produce redundant outputs.
- 25% of instructions require 2's Complement inputs.
- Format conversion is required when producer's output is Redundant *and* consumer requires 2's Complement inputs.
- On average, less than 3% of all dynamic instructions were delayed because of a format conversion.

Machine Assumptions

- Fetch, Decode, Issue 8 instructions per cycle
- 128 reservation station entries
- General purpose functional units:

(1) 2 clusters of eight with 1 cycle delay between clusters

(2) 1 cluster of four (execution bottleneck)

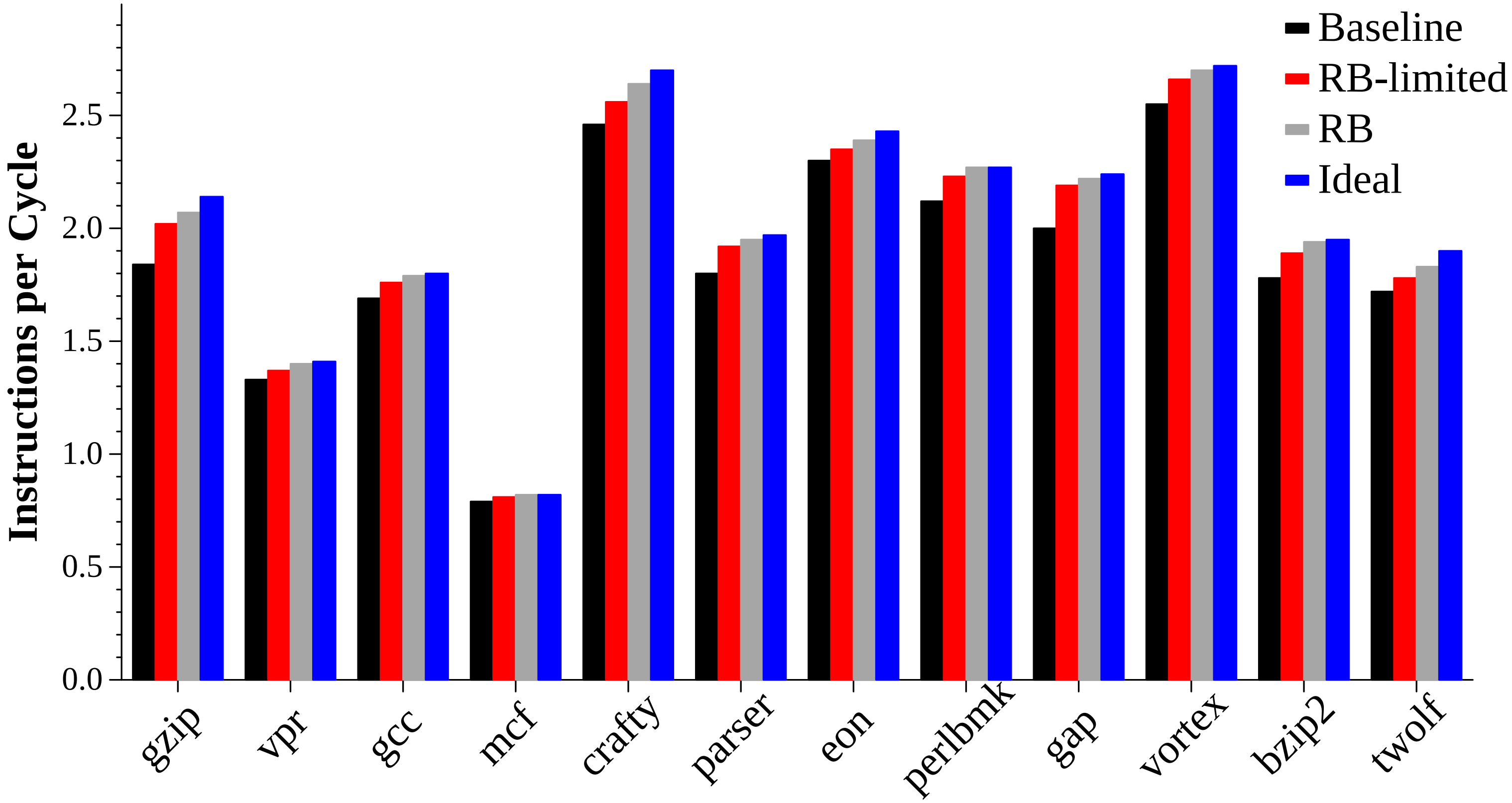
Both configurations use the same front end.

Machine Models

- **Baseline**: 2-cycle, pipelined 2's Complement arithmetic operations
- **Redundant Binary (RB)**: Both RB and TC register files.
  - *Full 2's complement add takes 2X as long as RB add [Makino et al., Nagendra et al.].*
  - 1-cycle RB operations, 2 cycles for RB $\rightarrow$ TC conversion.
- **Ideal**: 1-cycle arithmetic operations
- **RB-limited**: TC register files, limited bypass network

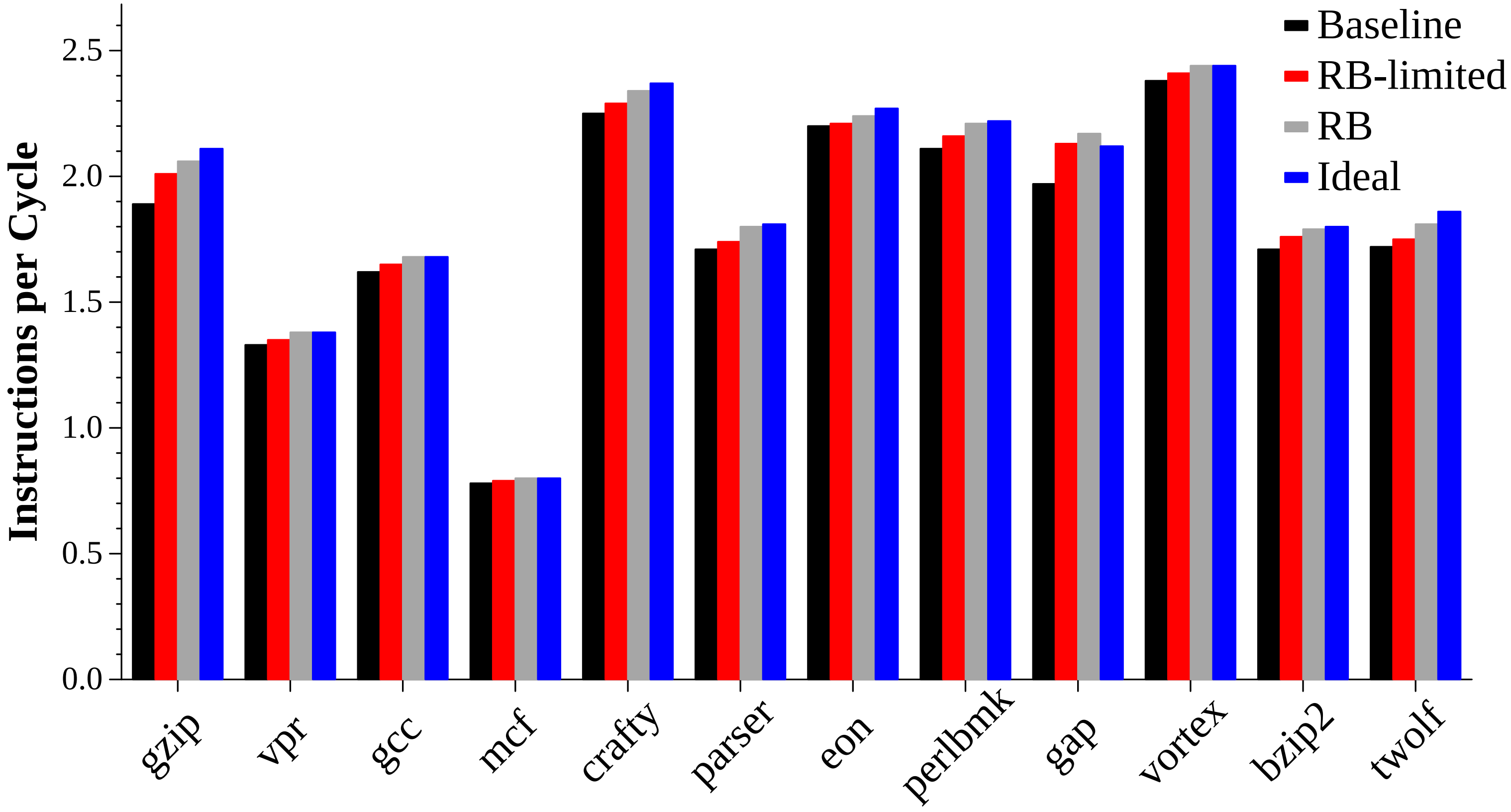
All machines have the same execution bandwidth (in terms of IPC)

IPC with 8 Clustered Functional Units



- 8% difference between Baseline and Ideal
- RB was 7% higher than Baseline, 1% from Ideal
- RB-Limited was 5% higher than Baseline, 3% from Ideal
- SPECint95: 11% difference between Baseline and Ideal

IPC with 4 Functional Units



- 5% difference between Baseline and Ideal
- RB was 4.5% higher than Baseline, 0.5% from Ideal
- RB-Limited was 2.5% higher than Baseline
- SPECint95: 7% difference between Baseline and Ideal

Conclusion

- Difference in IPC between 1-cycle adders and 2-cycle adders: 5% to 11%.
- IPC of machines with Redundant Binary adders is within 2% to 0.5% of the IPC of machines with 1-cycle 2's Complement adders.
- Execution cores built from redundant binary adders can be clocked at a higher frequency.
- Significant increase in performance for latency critical applications.

Classification of Last Bypasses