EECS 151 Disc 9

Rahul Kumar (session 1)
Yukio Miyasaka (session 2)

Berkeley
UNIVERSITY OF CALIFORNIA

Contents

Memory composition

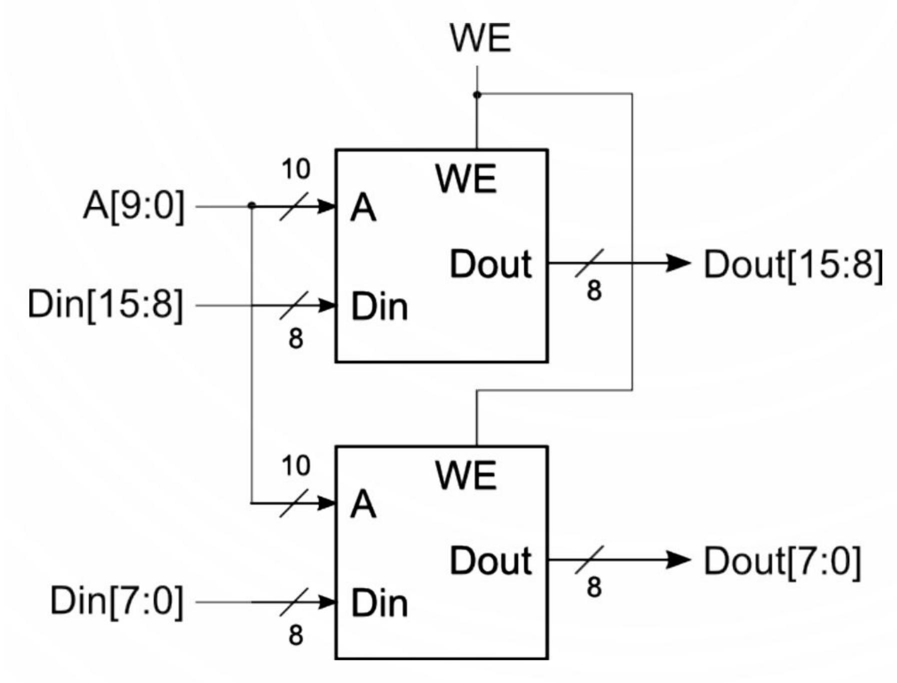
Common memory configurations:

- 1 read/write port
- 1 read/write port, 1 read-only port
- 1 write-only port, 1 read-only port

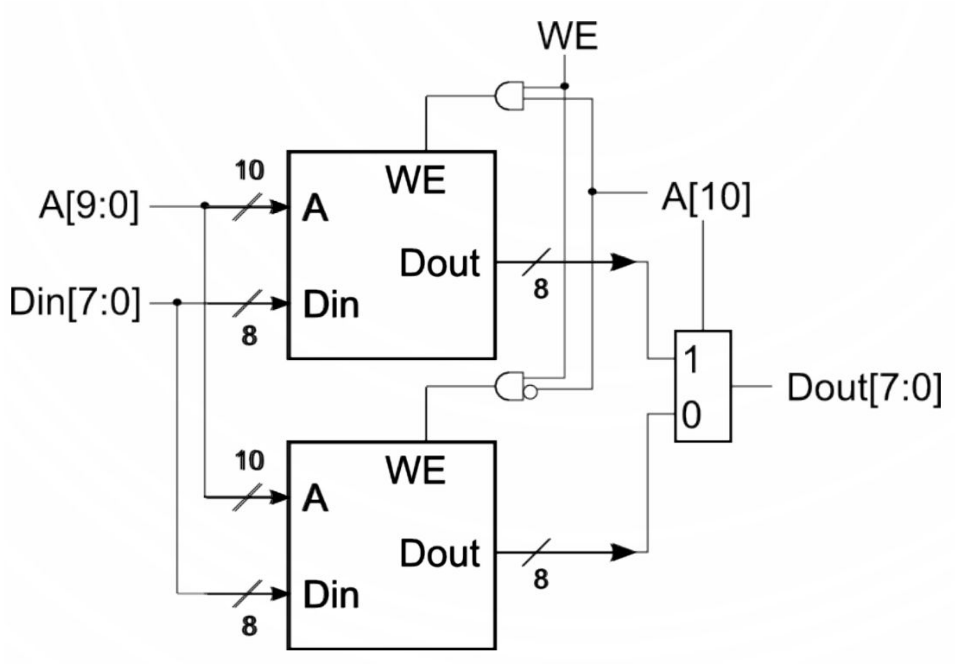
Sometimes want different configurations.

Regfile: 2 read ports, 1 write port.

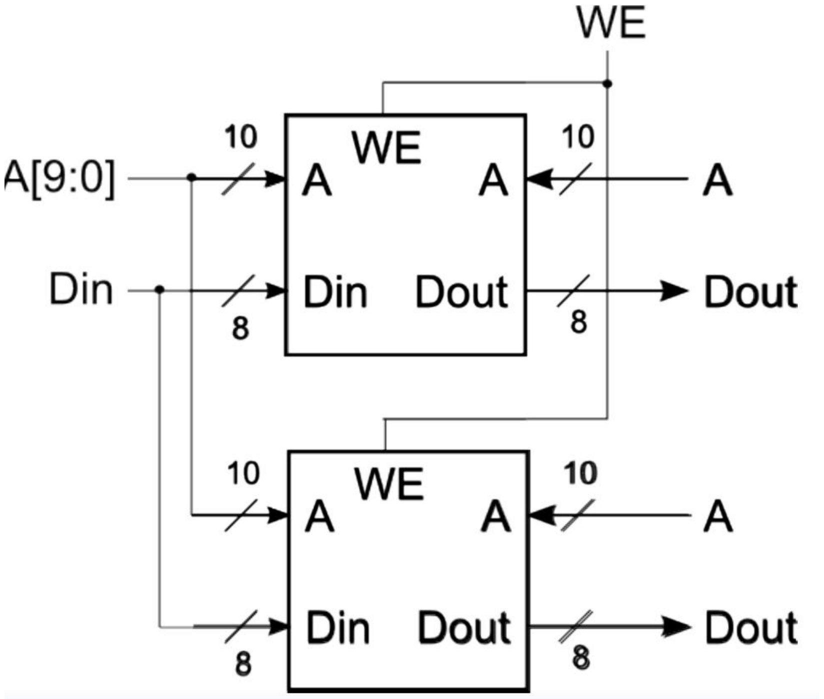
Memory composition: increasing width



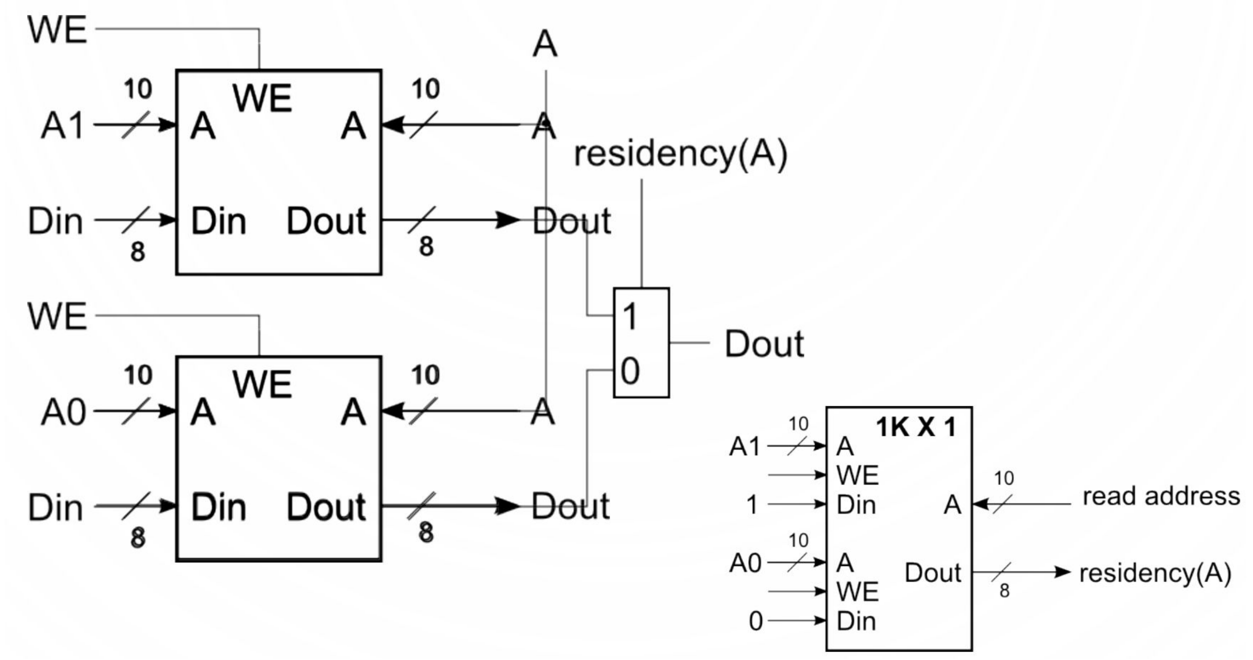
Memory composition: increasing depth



Memory composition: adding read port



Memory composition: adding write port



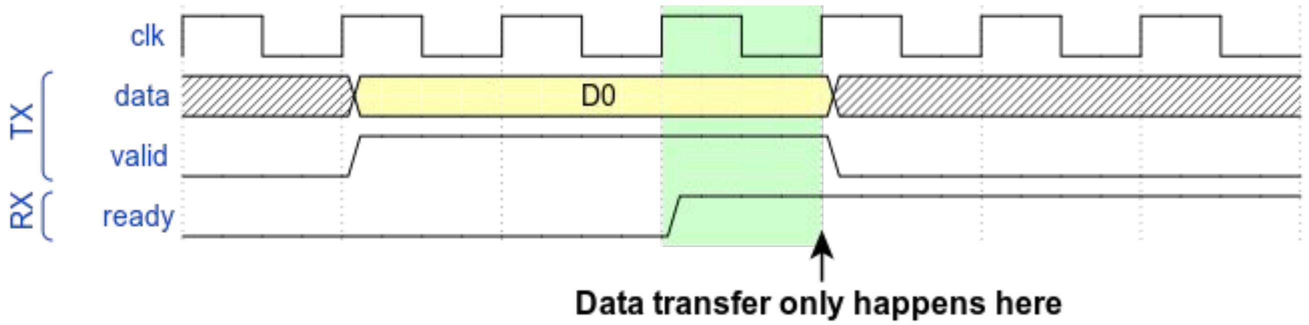
Berkeley
UNIVERSITY OF CALIFORNIA

Ready-valid interfaces

Source produces valid signal; sink produces ready signal.

Transaction occurs when ready and valid are high at a clock edge.

(If valid or ready is always 1, it can be omitted, where transaction may happen asynchronously.)



FIFOs

Read end:

- Read enable
- Empty
- Data out

Sometimes have an “almost empty” signal.

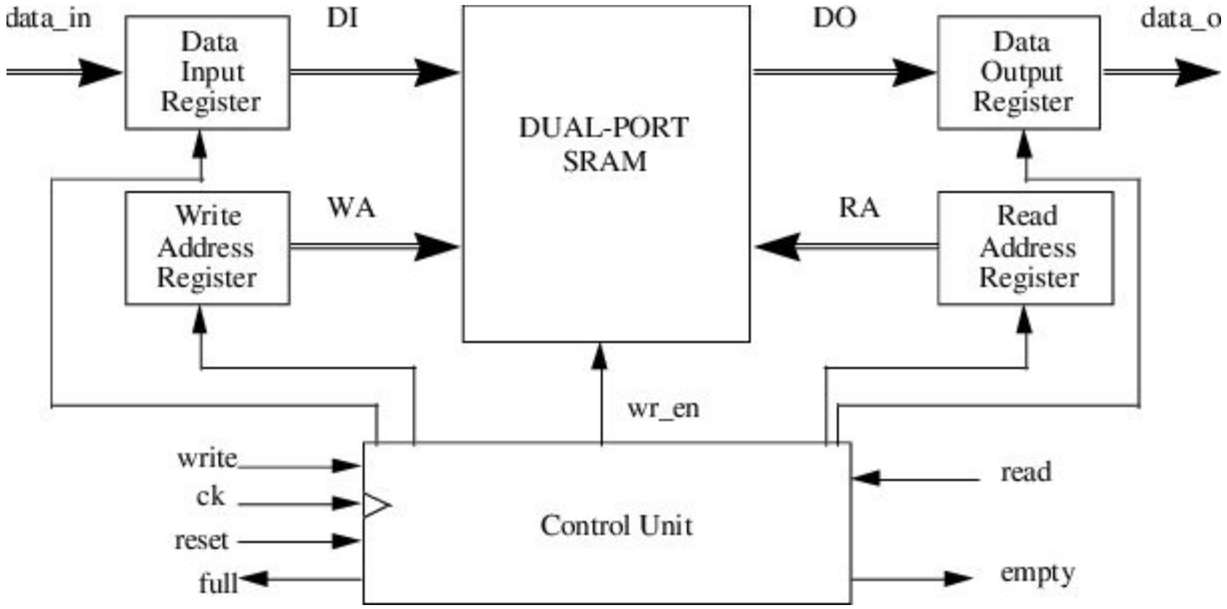
FIFOs

Write end:

- Write enable
- Full
- Data in

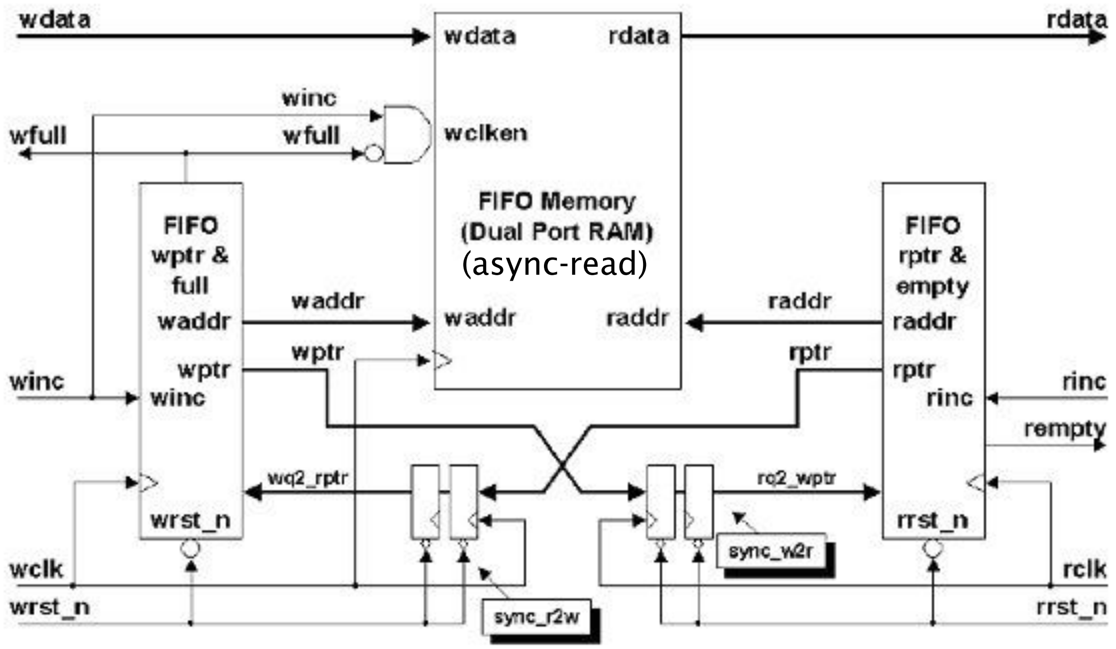
Sometimes have an “almost full” signal.

FIFOs



Berkeley
UNIVERSITY OF CALIFORNIA

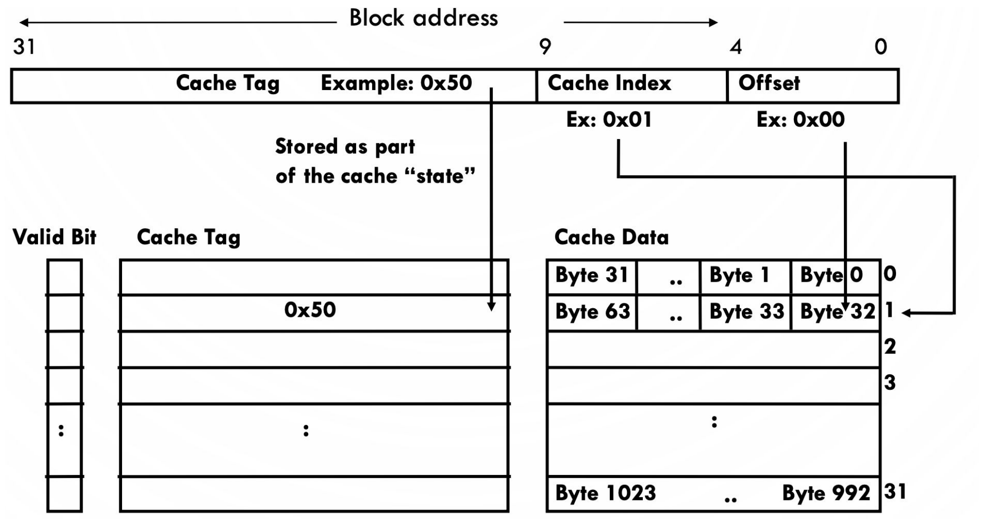
Asynchronous FIFO



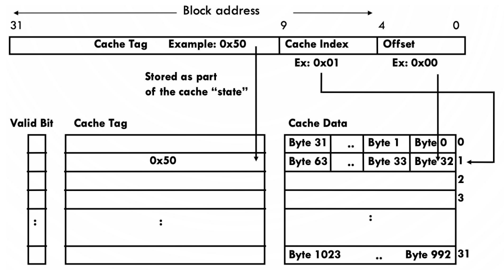
Berkeley UNIVERSITY OF CALIFORNIA

Direct-mapped caches

Main idea: a given line can only be stored in one position.

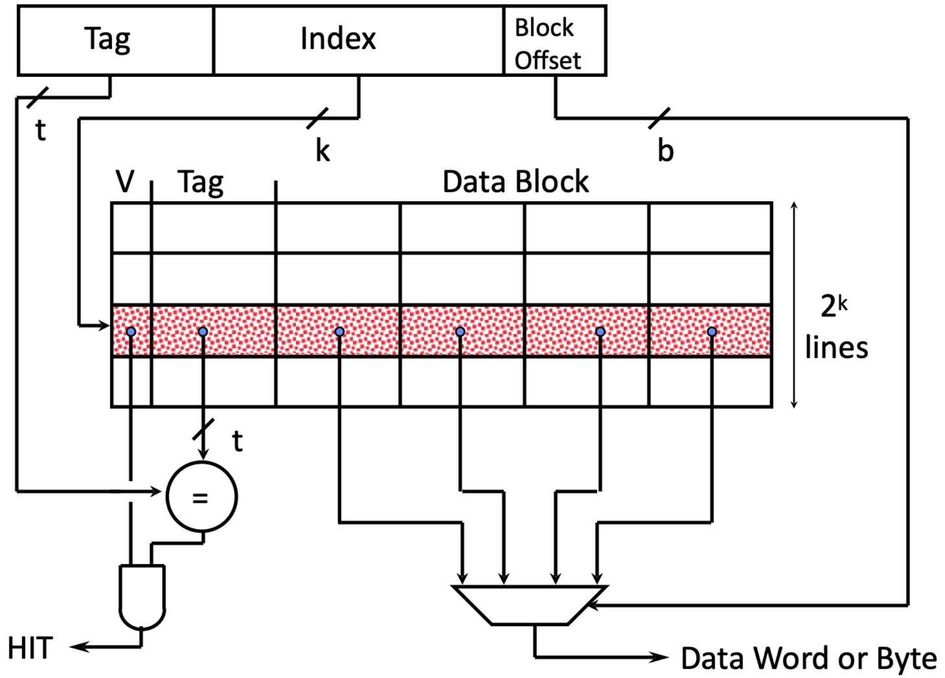


Direct-mapped caches



- Suppose there are 2^M cache lines, each holding 2^B bytes. Suppose address is A bits.
- Offset is B bits.
- Index is M bits.
- Cache tag is A – M – B bits.
- Cache stores 2^(M+B) bytes of data.

Direct-mapped caches



Direct-mapped caches

Advantages:

- Only need to check one location, so hardware implementation is relatively simple.
- Fast, low power, low area.

Main disadvantage: lower hit rate compared to higher associativity caches.

Writeback policies

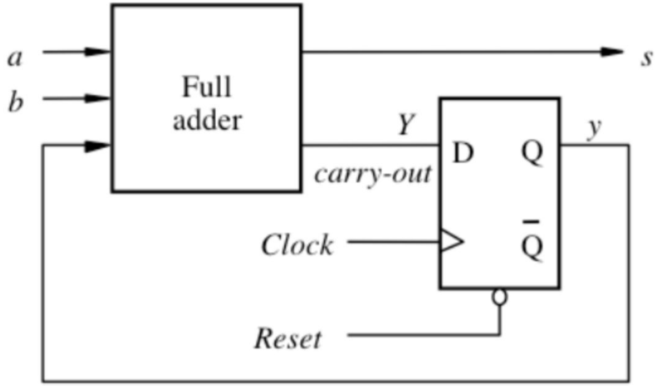
Writethrough: on every write, write to cache AND memory.

Writeback: write to memory only upon eviction of a cache line. Store a dirty bit to indicate if the line has been modified.

Loop unrolling

Example: serial adder. Performs one bit addition each cycle.

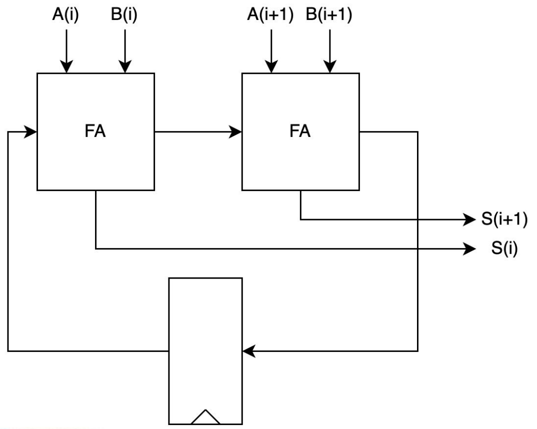
To add N-bit integers, run the adder for N cycles.



Loop unrolling

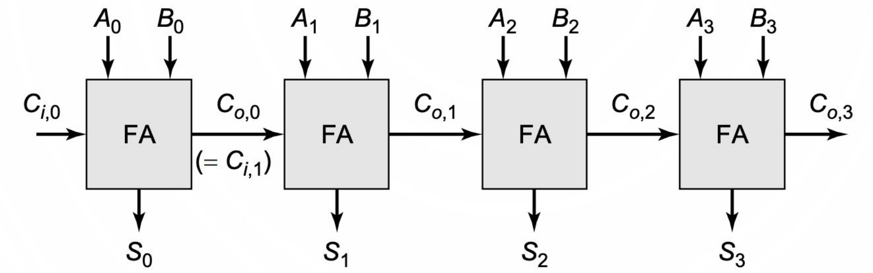
We can unroll with an interval 2. We generate 2 sum bits each cycle.

To add N-bit integers, run the adder for N/2 cycles.



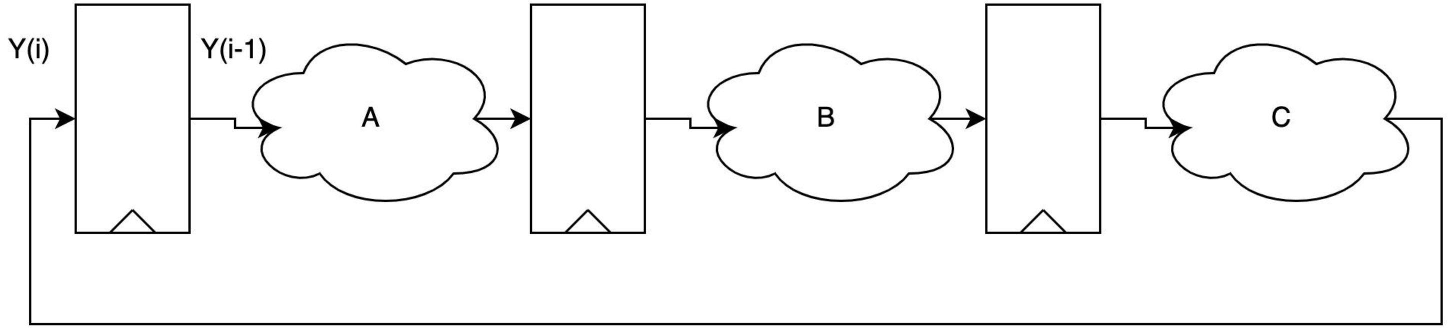
Loop unrolling

- Fully unrolling the loop results in a ripple carry adder.
- The full N-bit addition occurs in 1 cycle.
- However, the logic depth is N. To make it shallow, we could parallelize or precompute carry calculation.

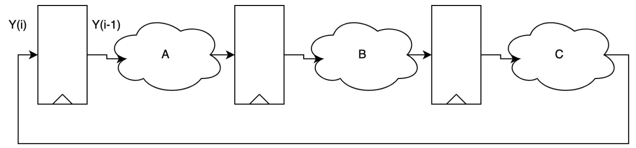


C-slowing

Example: want to compute Y[i] = C(B(A(Y[i-1]))), with pipelining.



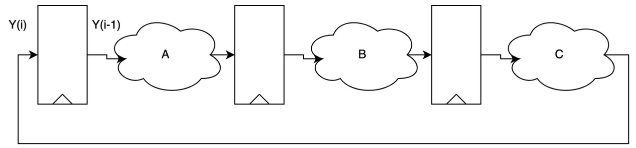
C-slowing



Many wasted cycles:

A0			A1			A2		
	B0			B1			B2	
		C0=Y1			C1=Y2			C2=Y3

C-slowing



Solution: if there are 3 independent data streams (we'll call them X, Y, and Z), we can fill the pipeline.

AX0	AY0	AZ0	AX1	AY1	AZ1	AX2	AY2	AZ2
BZ(-1)	BX0	BY0	BZ0	BX1	BY1	BZ1	BX2	BY2
CY(-1)= Y0	CZ(-1)= Z0	CX0=X1	CY0=Y1	CZ0=Z1	CX1=X2	CY1=Y2	CZ1=Z2	CX2=X3