Repetitive Logic

Where logic blocks are duplicated within a system, there is much to gain from optimization.

- Optimize single block for size.

  The effect is amplified by the number of identical blocks.

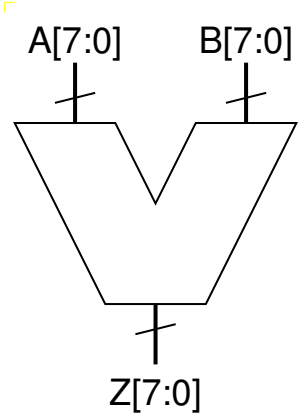
- Optimize interconnect.

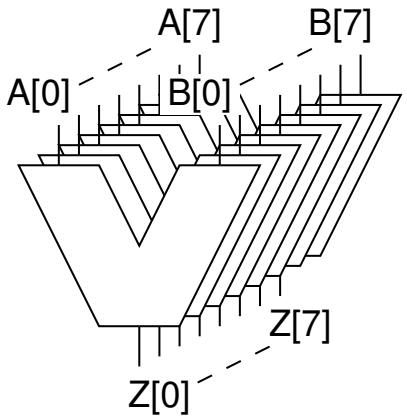
  With careful design, the requirement for routing channels between the blocks can be eliminated.

  Interconnection is by butting in 2 dimensions.

Bit Slicing

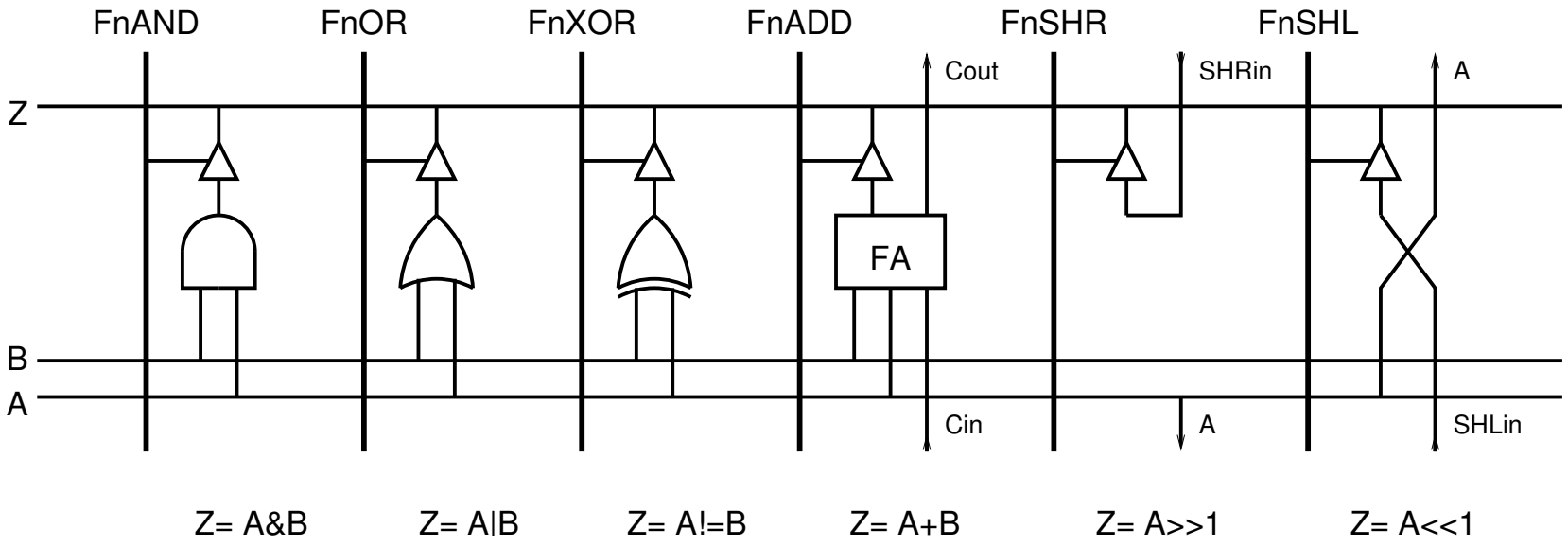
Instead of creating an ALU function by function, we create it slice by slice.





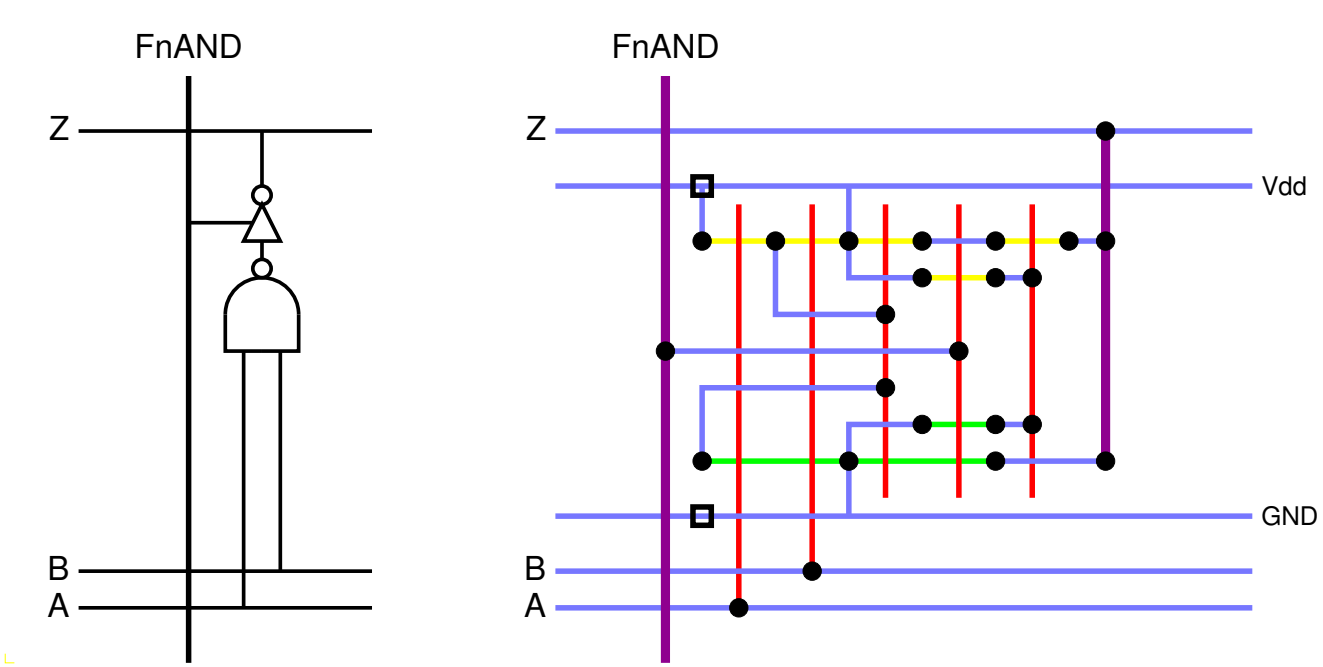
- Each *bit slice* is a full 1-bit ALU.
- N are used to create an N–bit ALU.

Bit Slicing



- Simple 1–bit ALU.
- The control lines select which function of A and B is fed to Z.
- Some functions, e.g. Add, require extra data I/O.

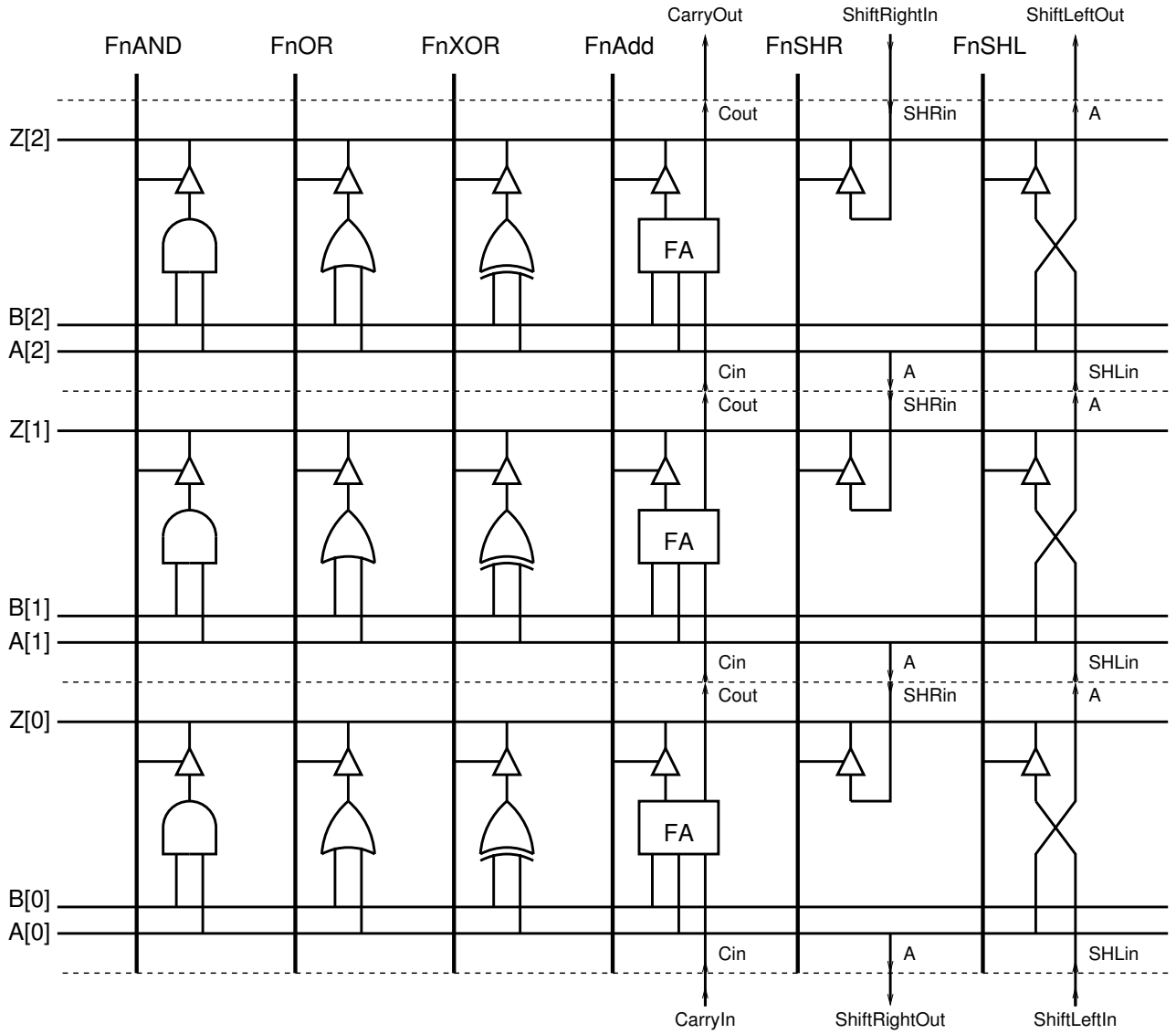
Bit Slicing



- Data busses horizontal, control lines vertical.
- Compact gate matrix implementation[1].

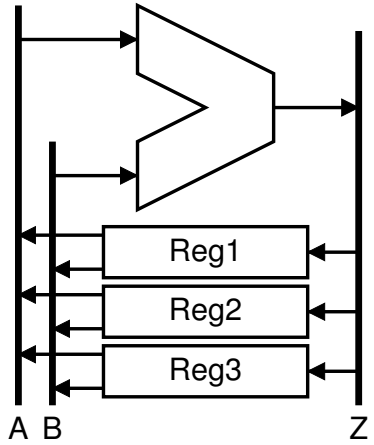
[1]bit slice designs can also be built around standard cells although the full custom approach used here gives better results

- Bit Sliced ALU (3-bits, scalable).

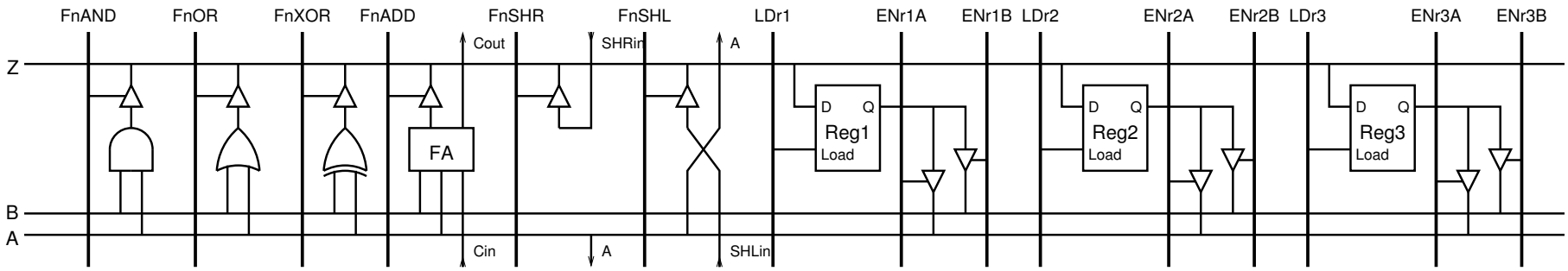


Bit Slicing

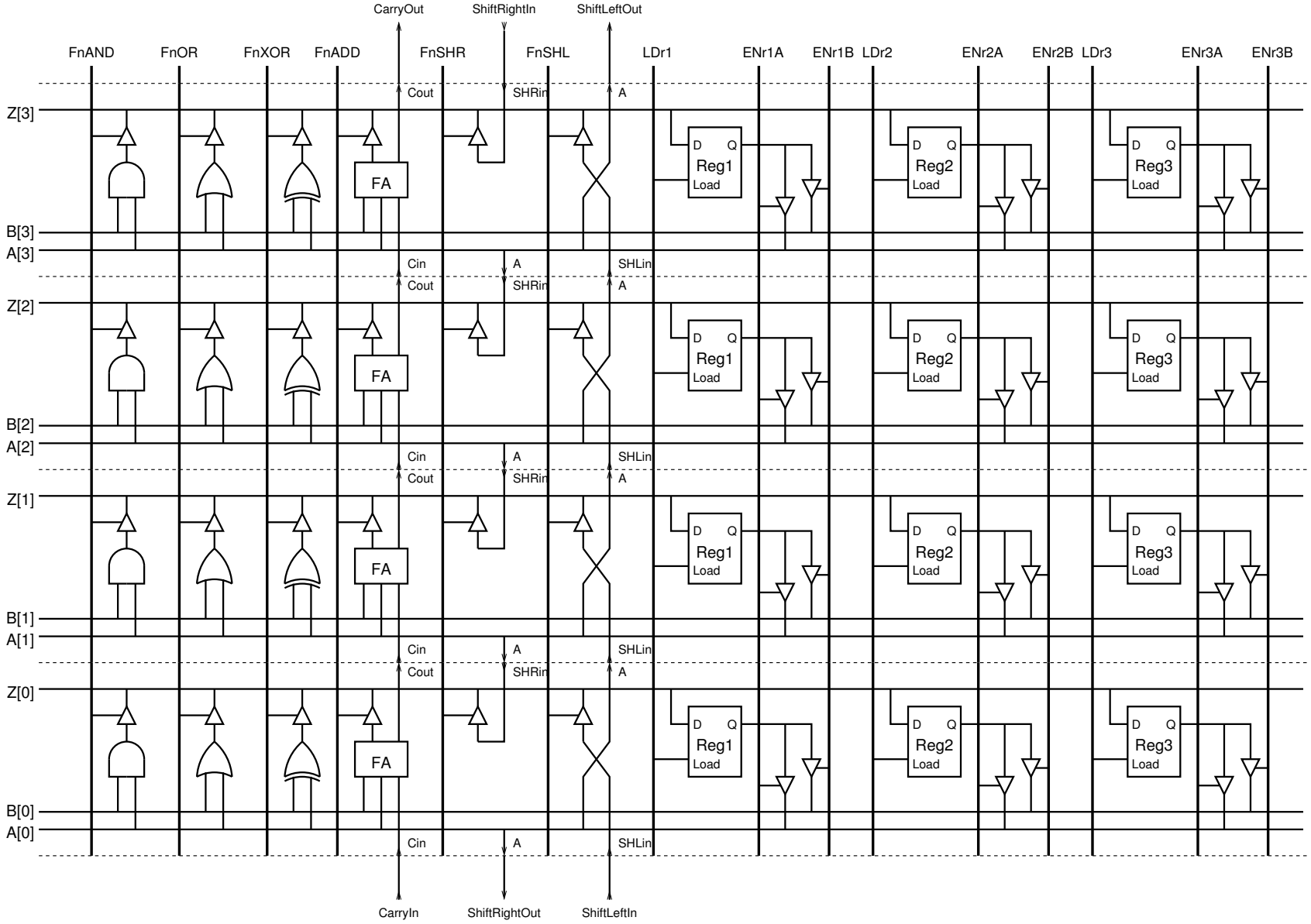
We can extend this principle to the whole *datapath*.



- 1–bit datapath.

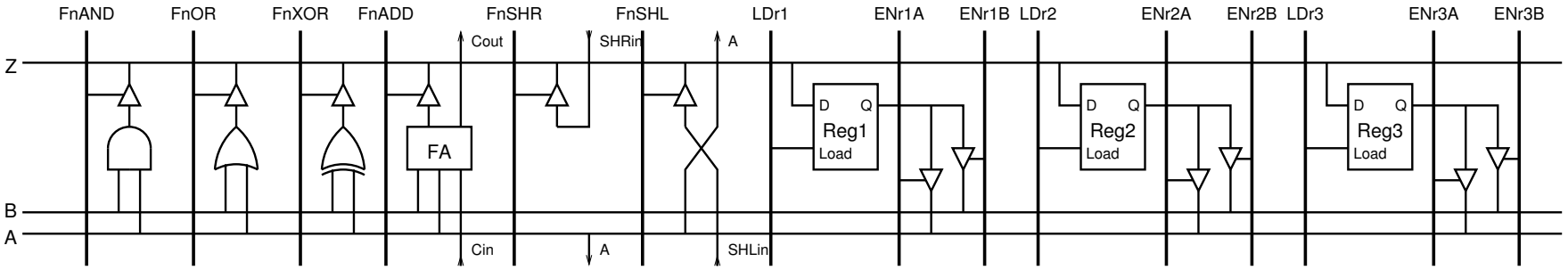


- **Bit Sliced Datapath**

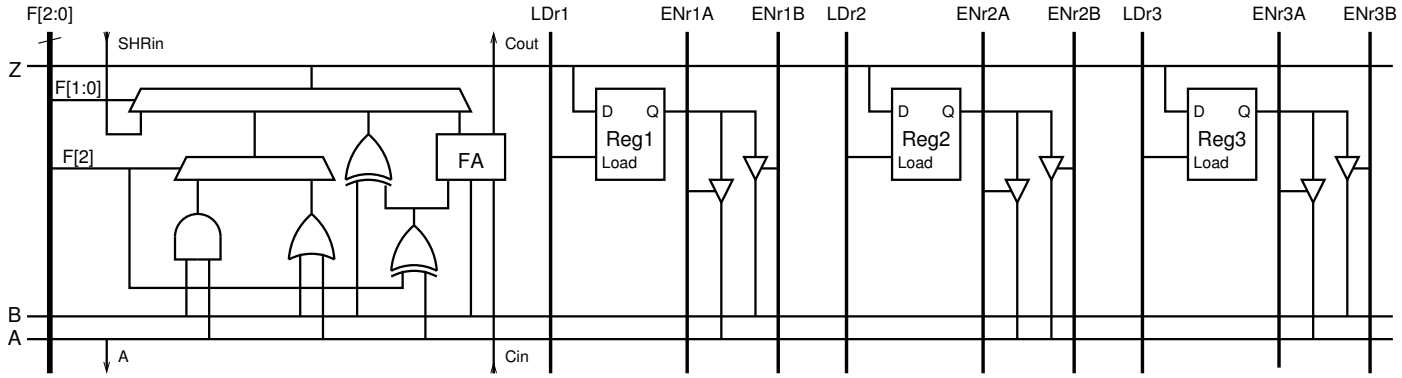


Bit Slicing

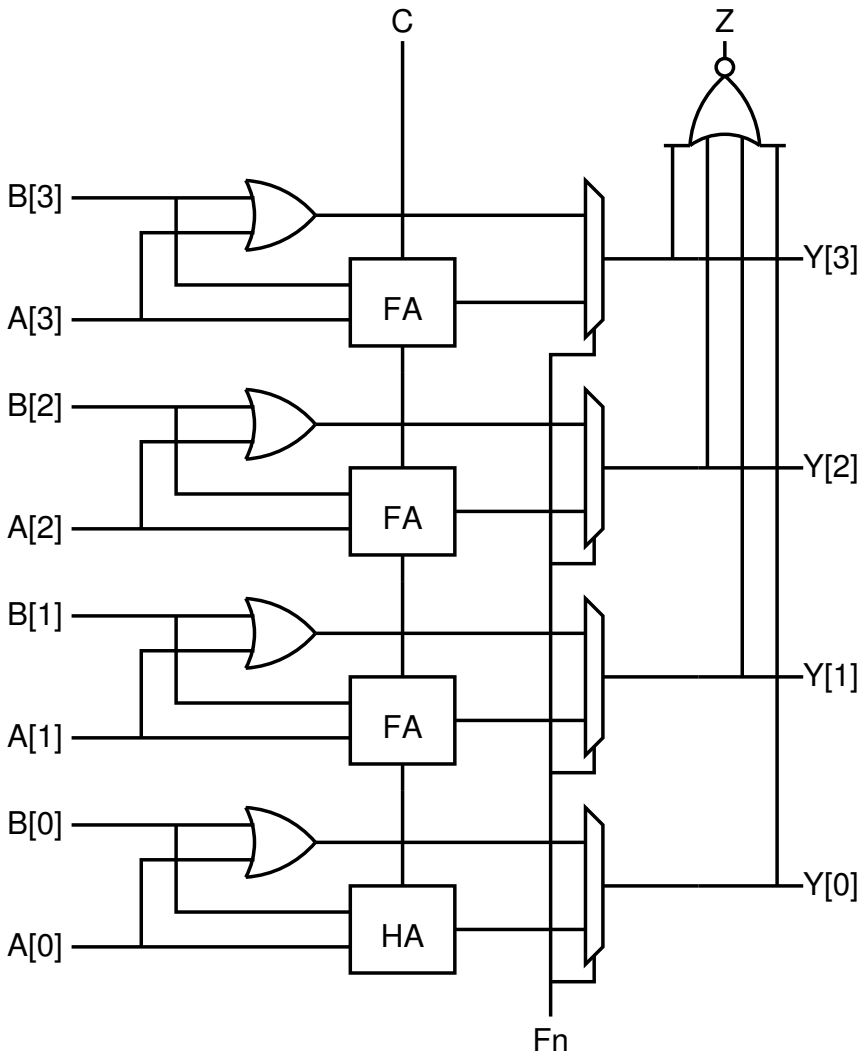
- Distributed Multiplexing



- Local Multiplexing

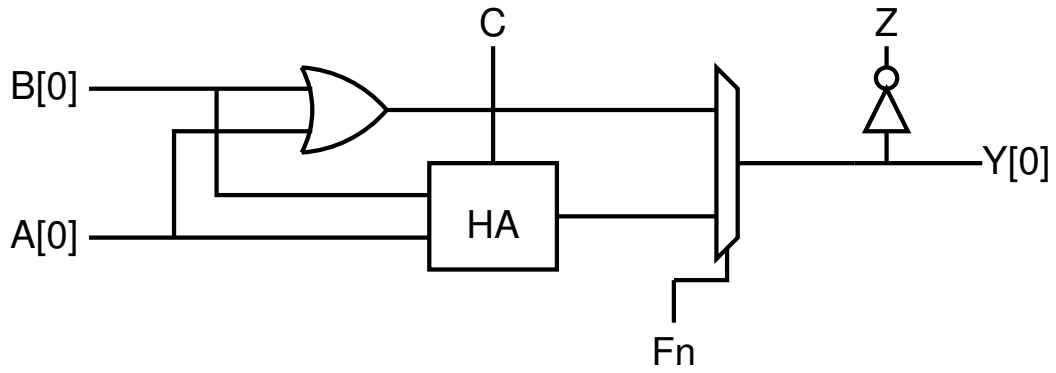


Bit Slicing - Example

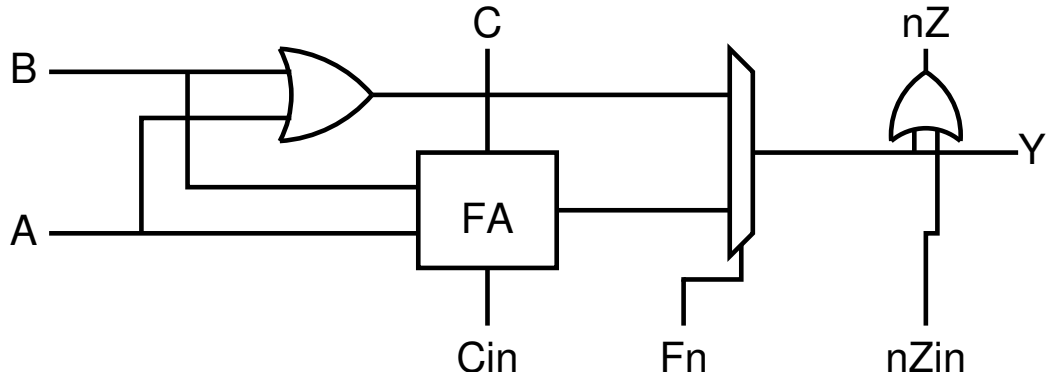


Bit Slicing - Example

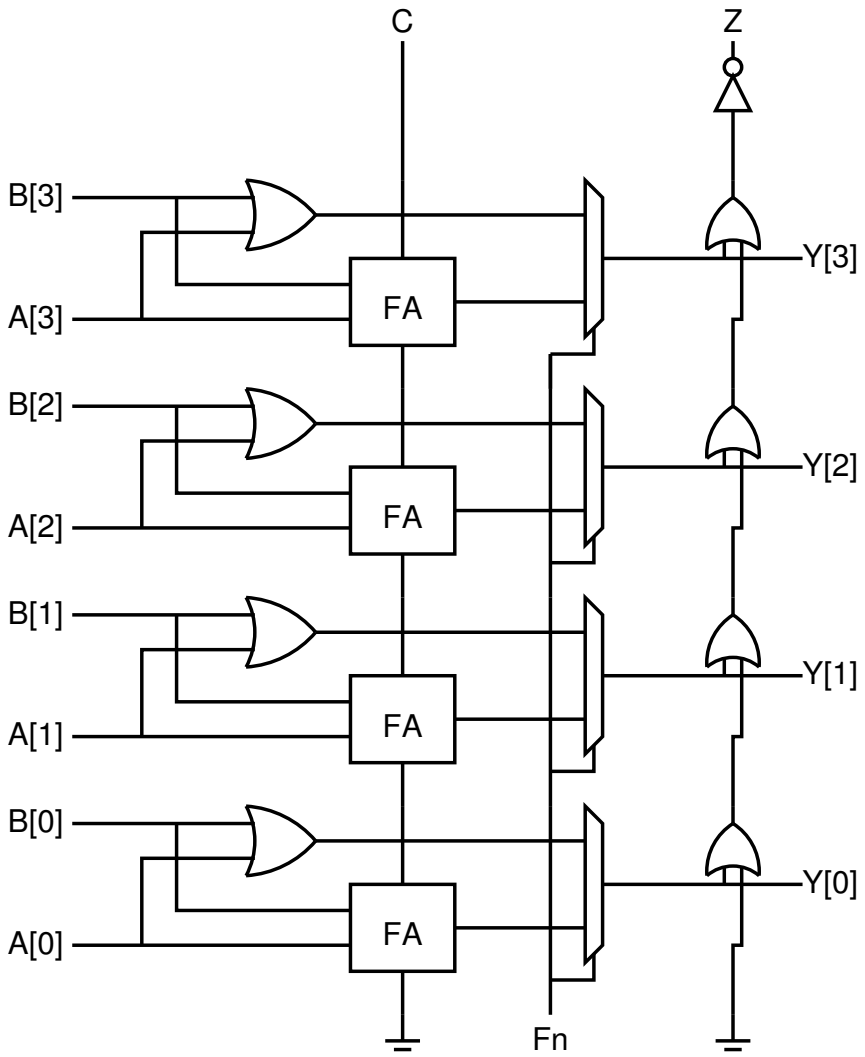
1 Bit ALU (no consideration of bitslicing)



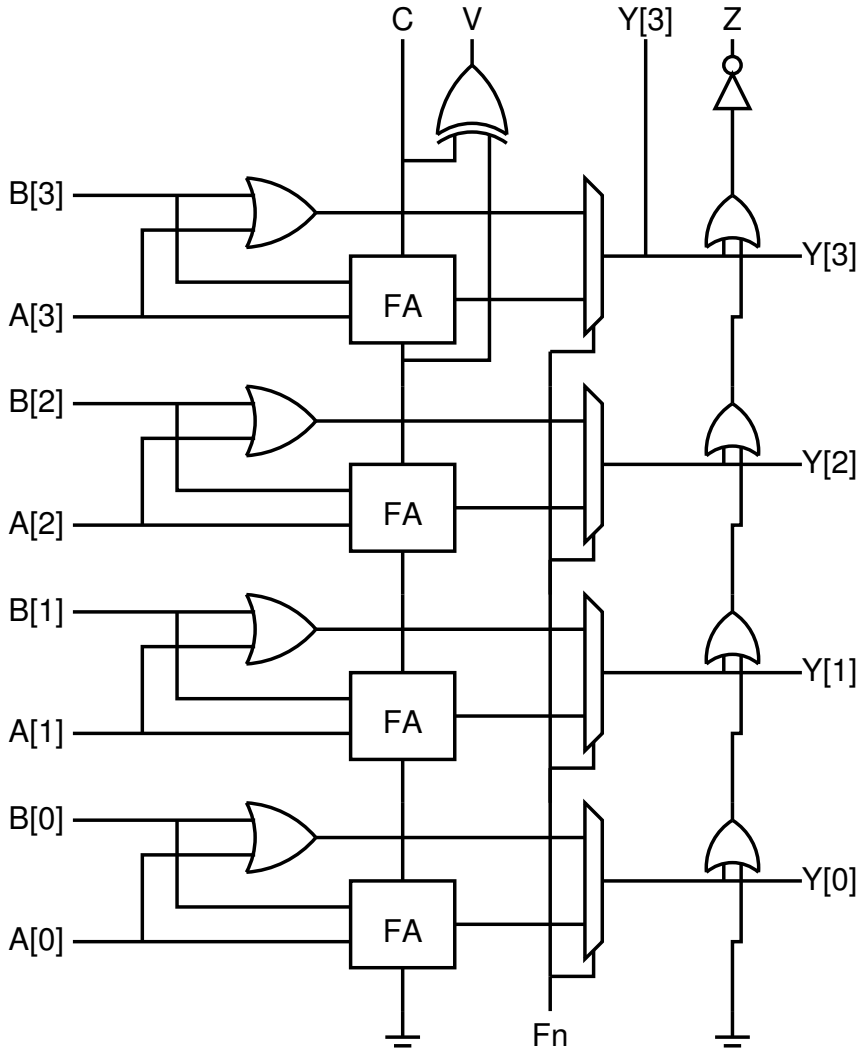
Bitslice Design for ALU



Bit Slicing - Example

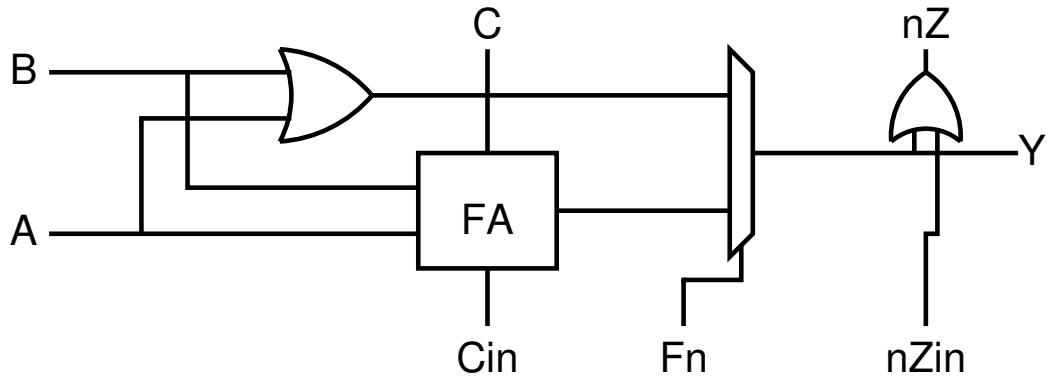


Bit Slicing



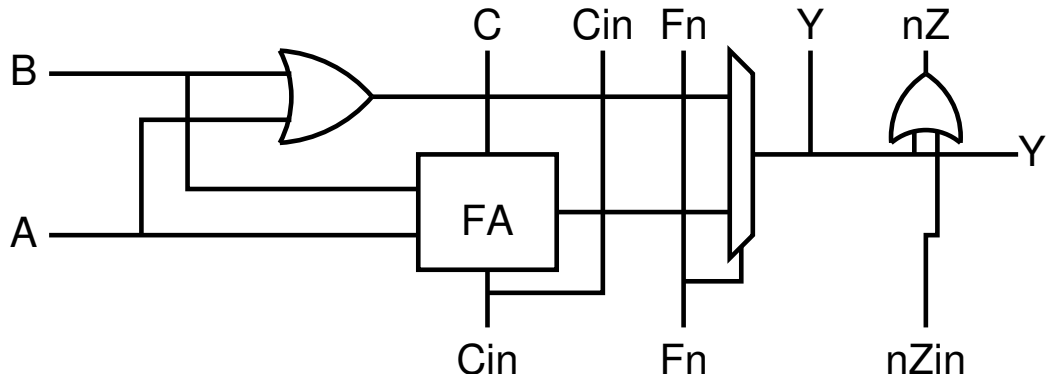
Bit Slicing

Original Bitslice Design for ALU

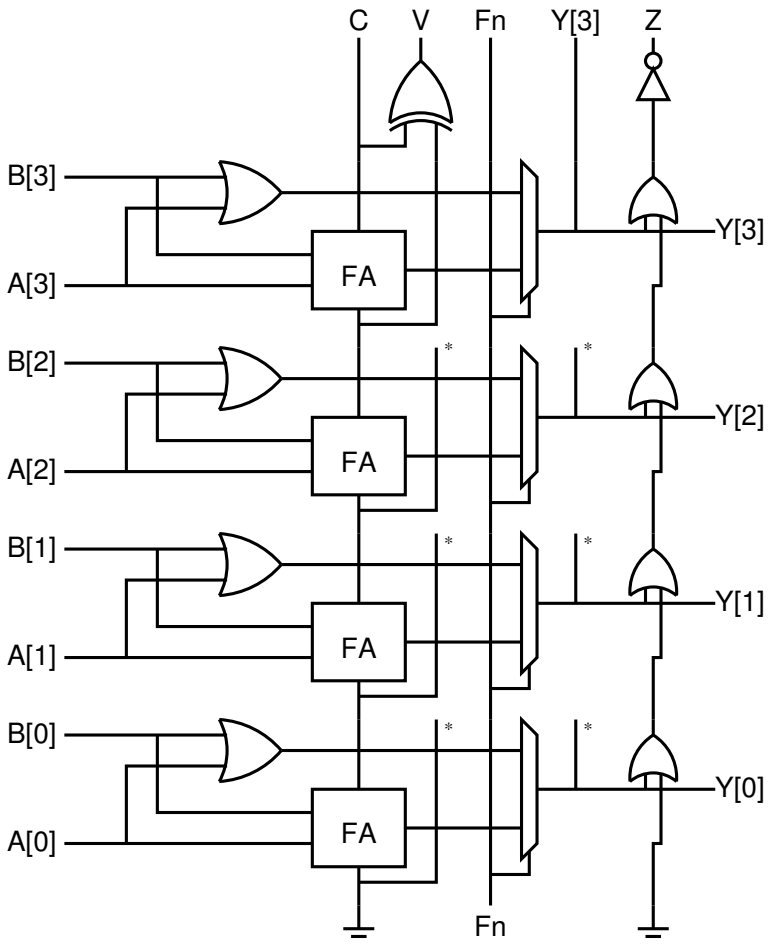


Modified Bitslice Design for ALU

- Wiring is arranged so that no over cell routing is required.



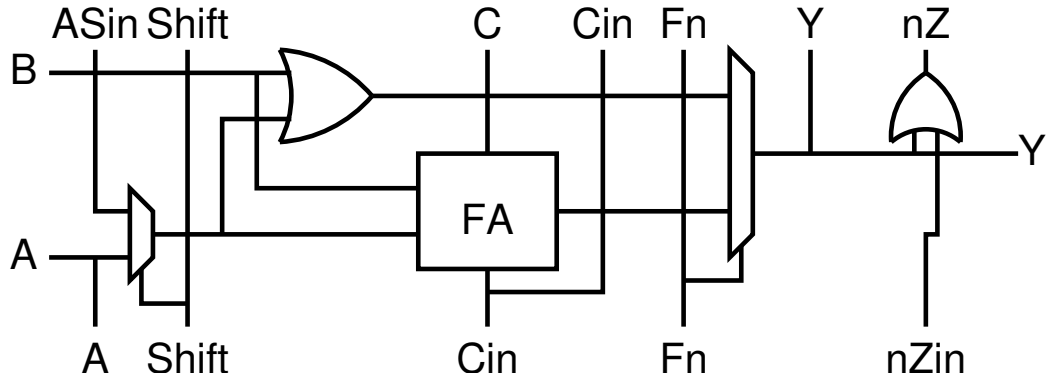
Bit Slicing



*a number of wires are left unconnected

Bit Slicing

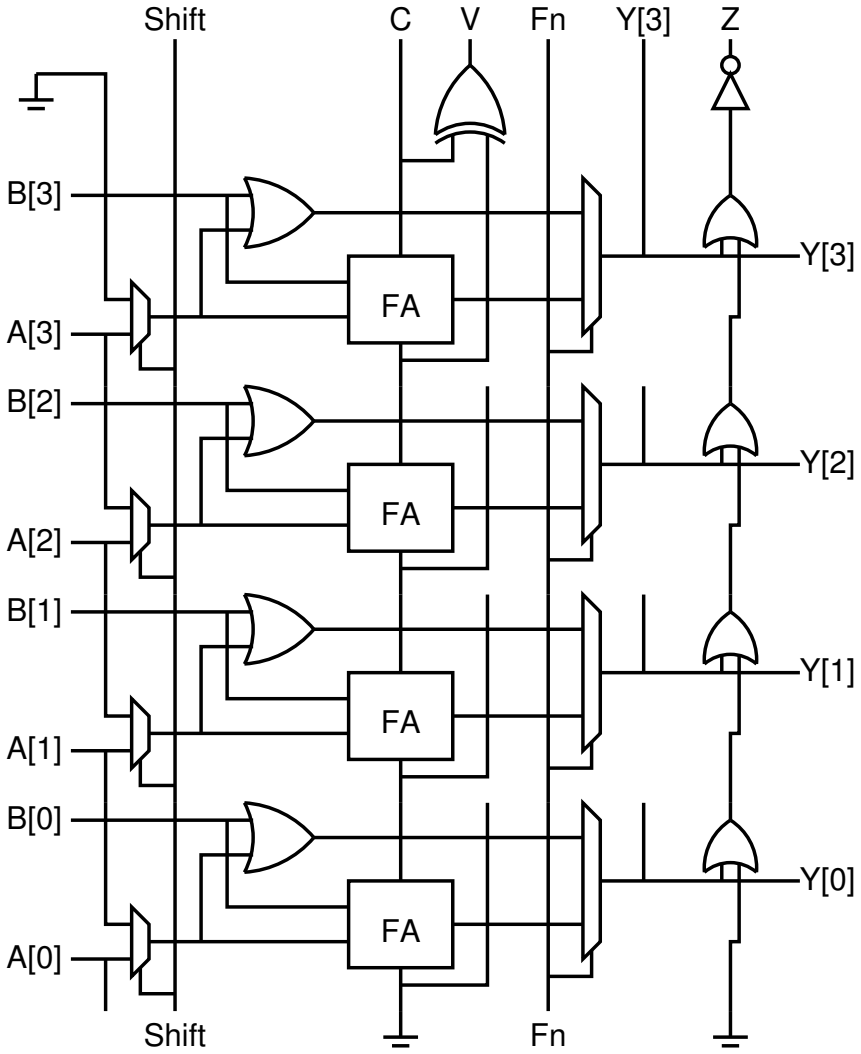
Single Bit Shift



This simple ALU bitslice now supports 4 functions

- `Y = A + B`
- `Y = A | B`
- `Y = (A >> 1) + B`
- `Y = (A >> 1) | B`

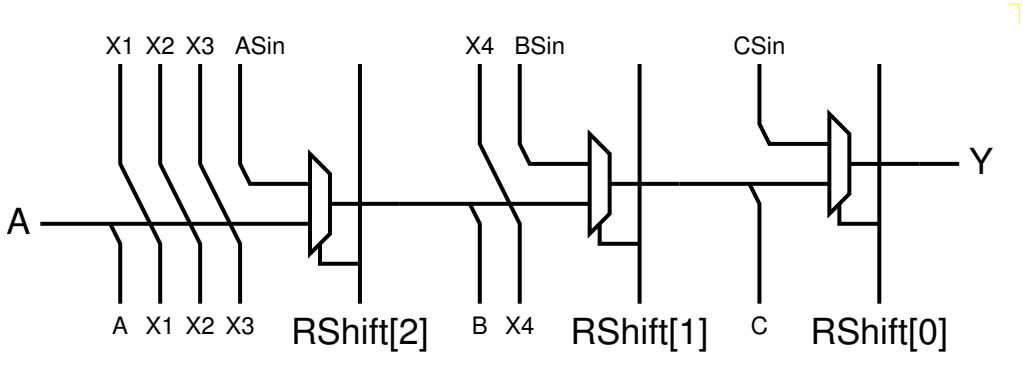
Bit Slicing


Shift
C
V
Fn
Y[3]
Z
B[3]
A[3]
B[2]
A[2]
B[1]
A[1]
B[0]
A[0]
FA
FA
FA
FA
Y[3]
Y[2]
Y[1]
Y[0]
Shift
Fn

Bit Slicing

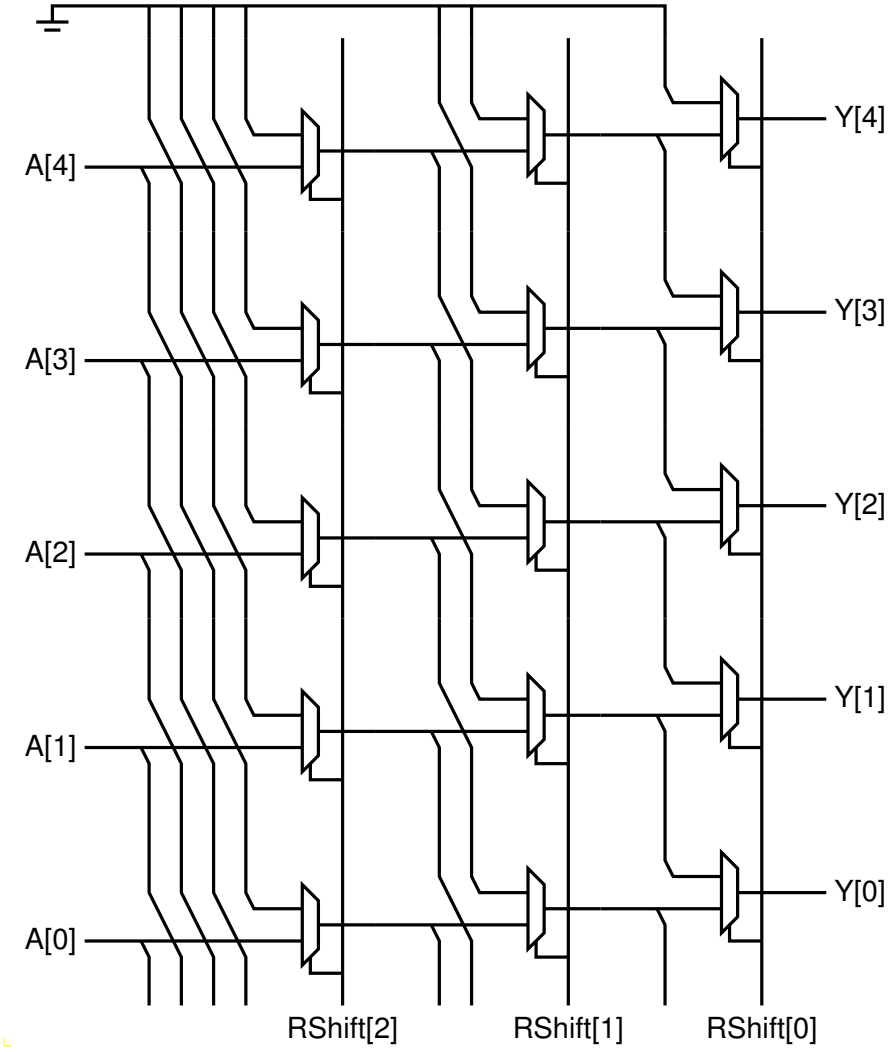
Multi-Bit Shift

This barrel shifter bitslice includes wiring for 4 bit right shift, wiring for 2 bit right shift and wiring for 1 bit right shift.



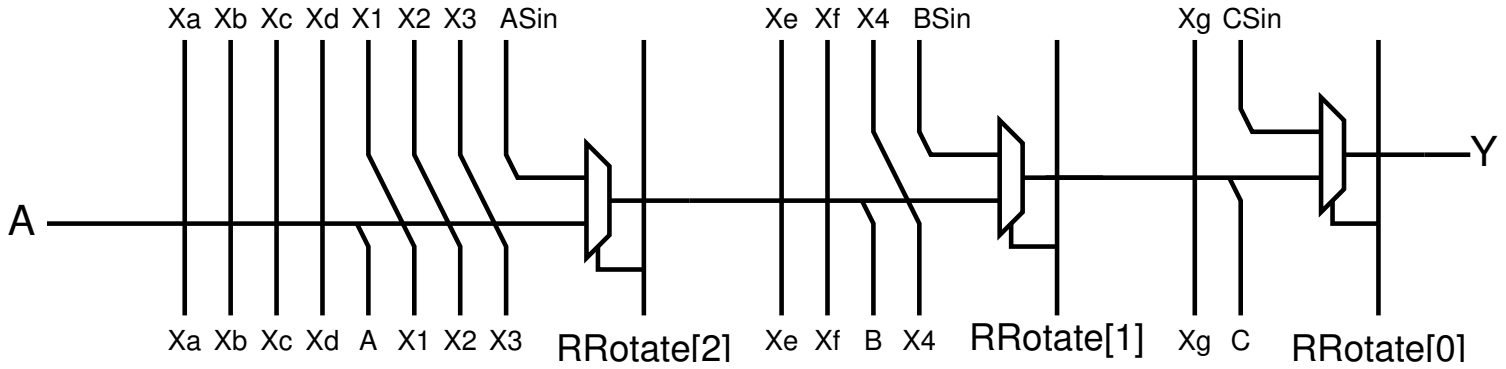
- `Y = A >> RShift[2:0]`

Bit Slicing

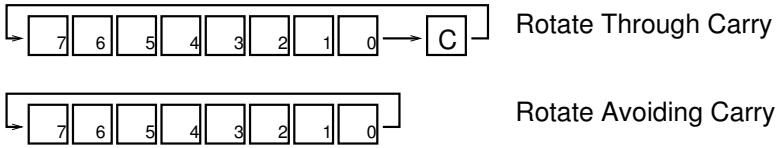


Multi-Bit Rotate

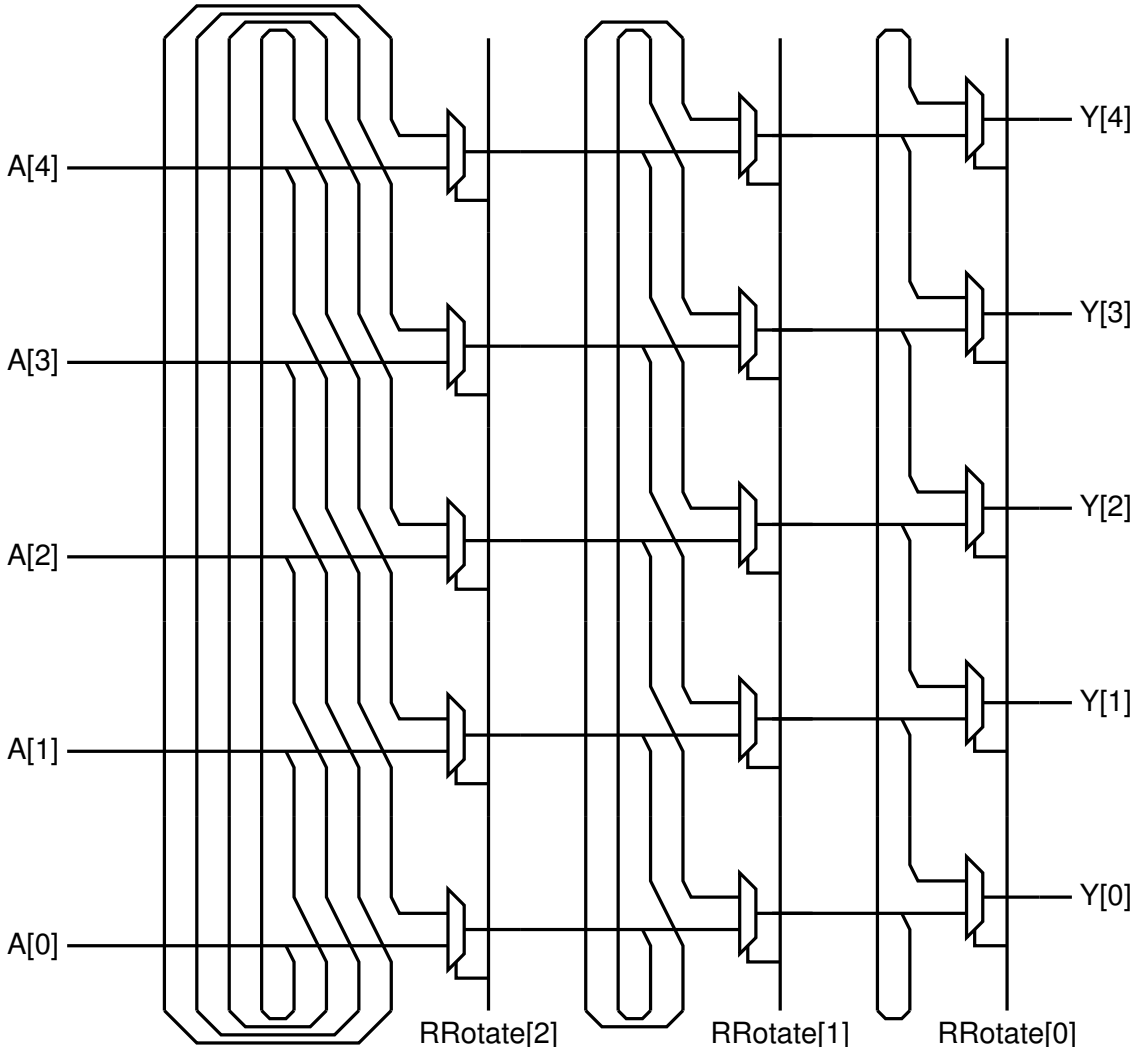
By providing extra feedthrough paths we can create a bitslice for multi-bit rotation.



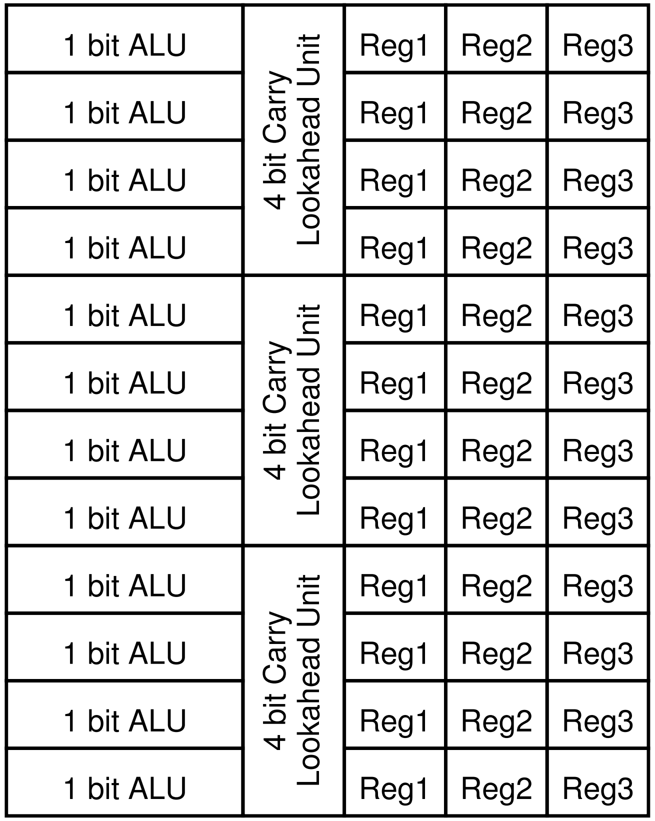
N.B. Most single bit rotate functions will *rotate through the carry*. This barrel rotation, based on the SPARC implementation, does not make use of the carry for input or output.

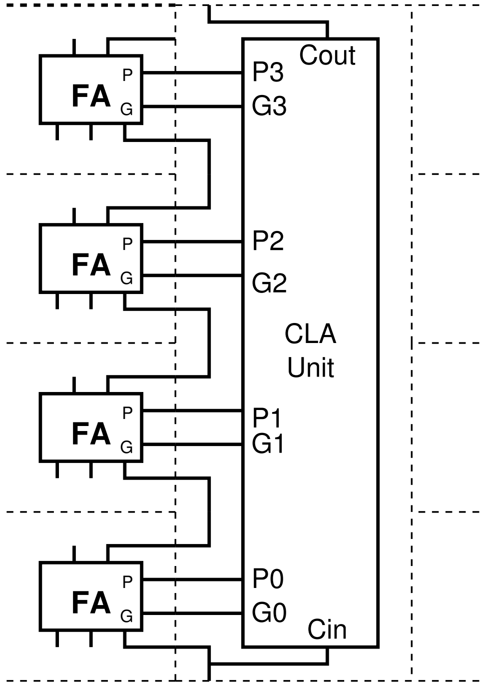


Bit Slicing



Bit Slicing





- Bitslice Exceptions

  Where full bitslicing is not suitable we attempt to disrupt the bitslice as little as possible.