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.

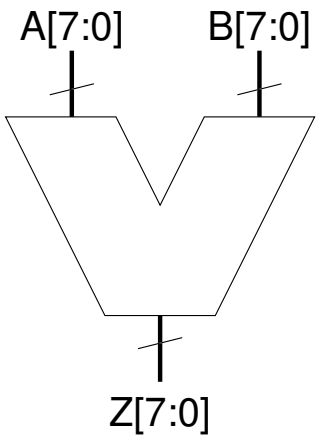
- Optimize interconnect.

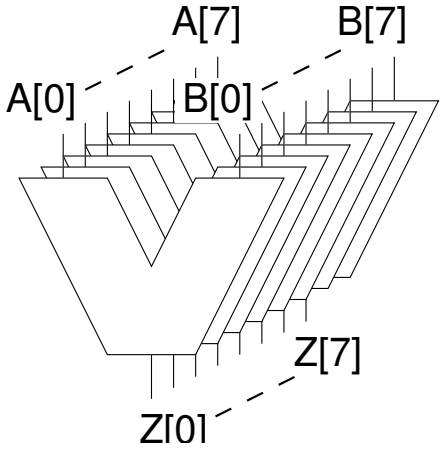
  With careful design, the requirement for routing channels between the blocks can be eliminated.

  Interconnection is by butting in 2 dimensions.

Bit Slicing

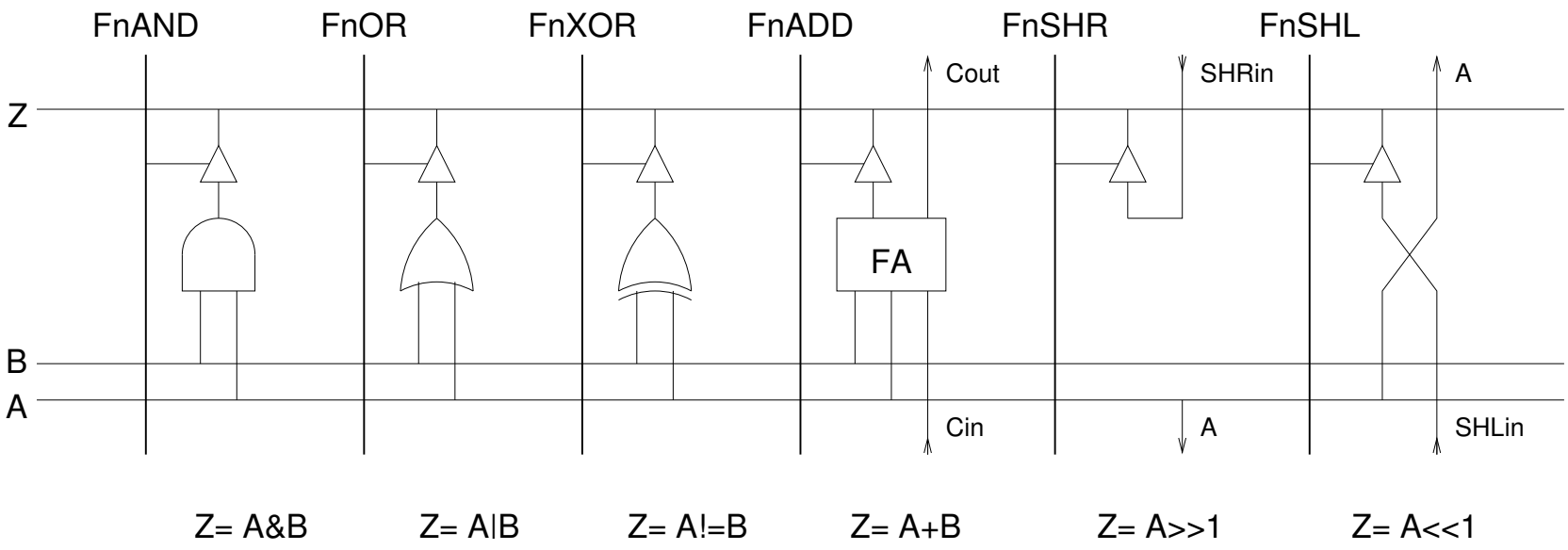
Instead of creating an ALU function by function, we create it slice by slice.





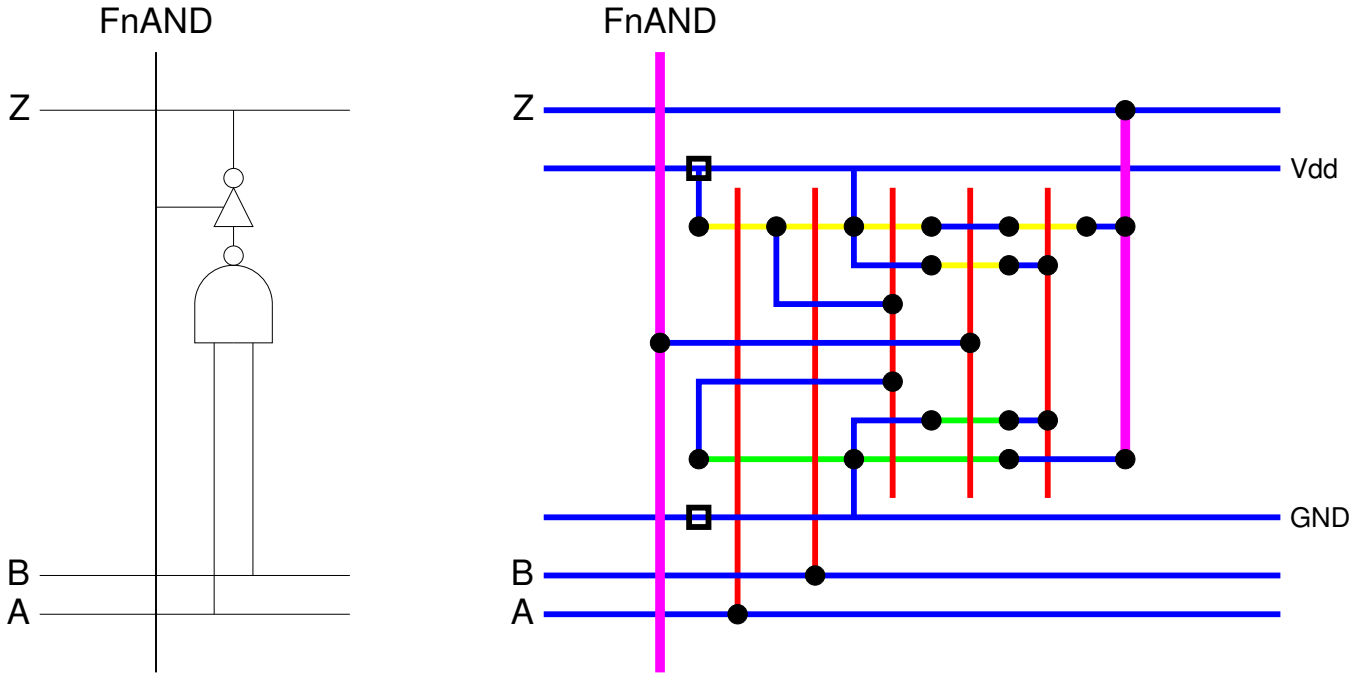
- Each *bit slice* is a full 1-bit ALU.
- *N* are used to create an *N*–bit ALU.

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.

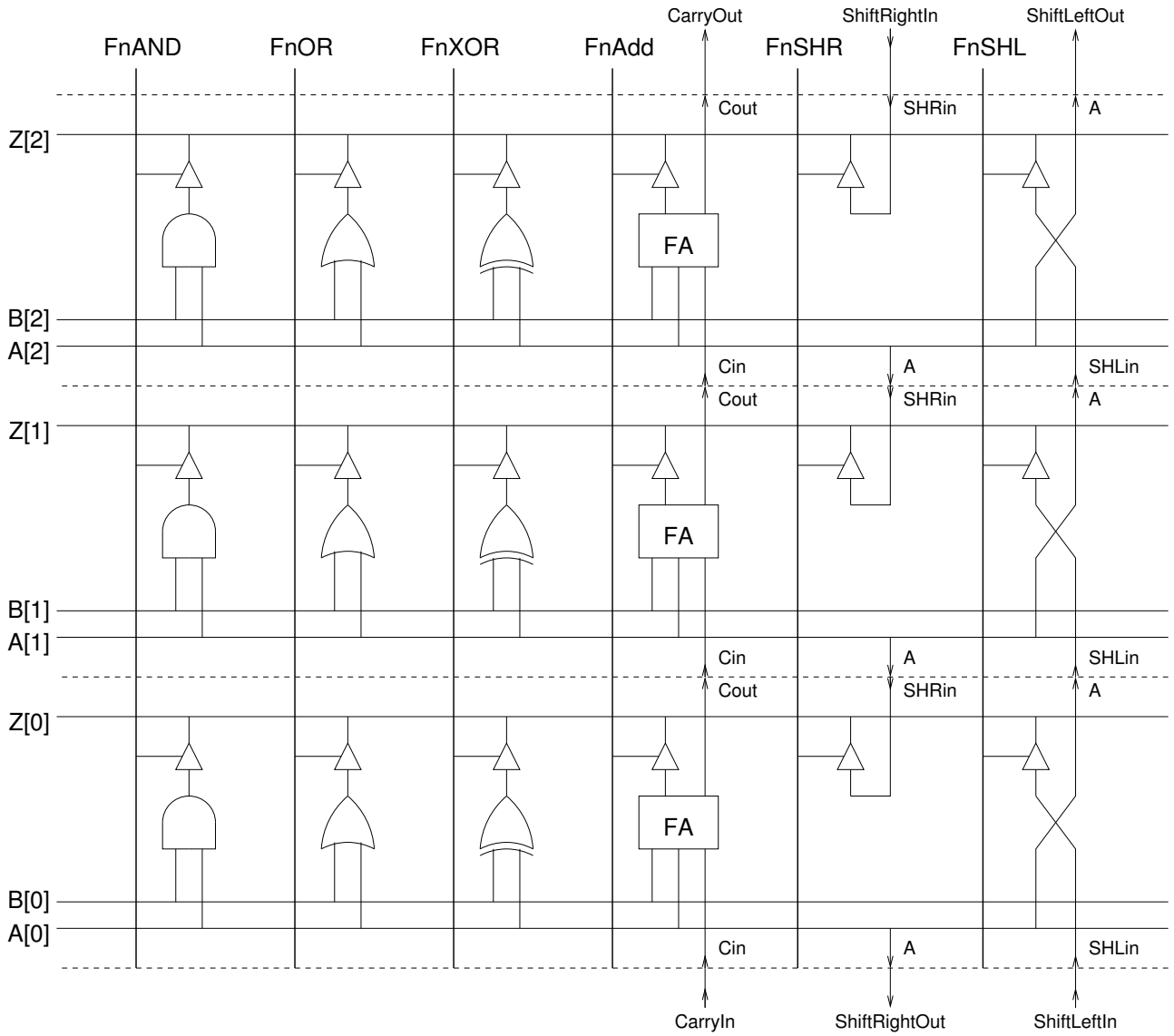
Bit Slicing



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

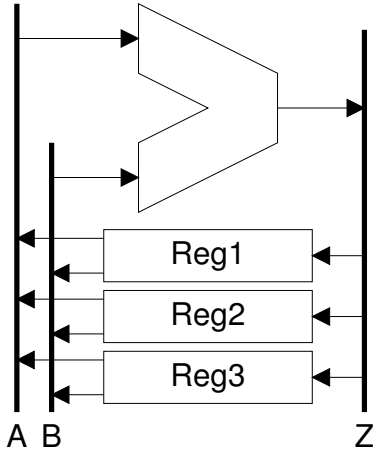
[1]bit slice designs can also be built around standard cells although the full custom approach used here gives better results

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

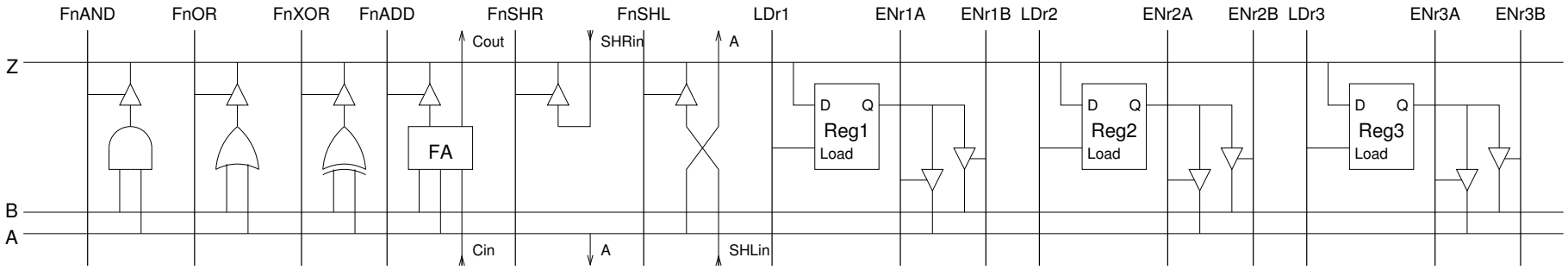


Bit Slicing

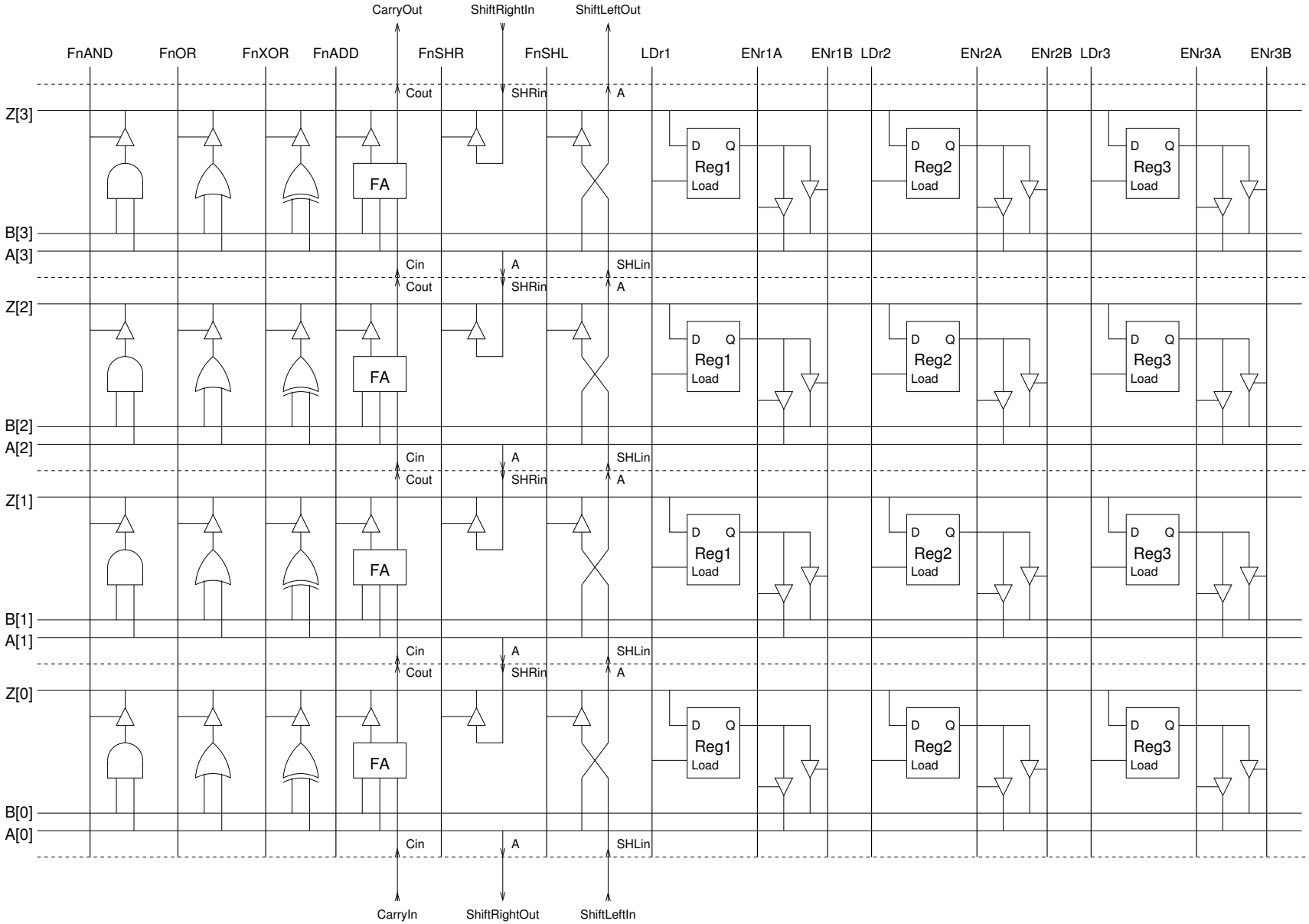
We can extend this principle to the whole *datapath*.



- 1–bit datapath.

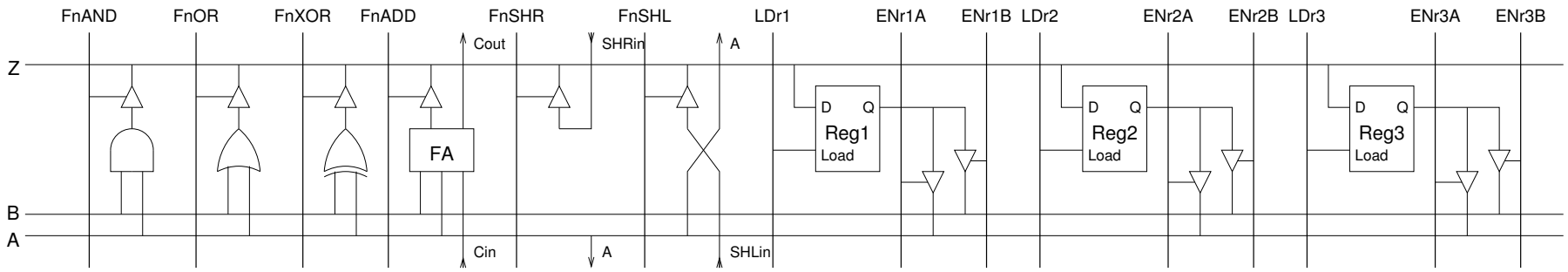


- **Bit Sliced Datapath**

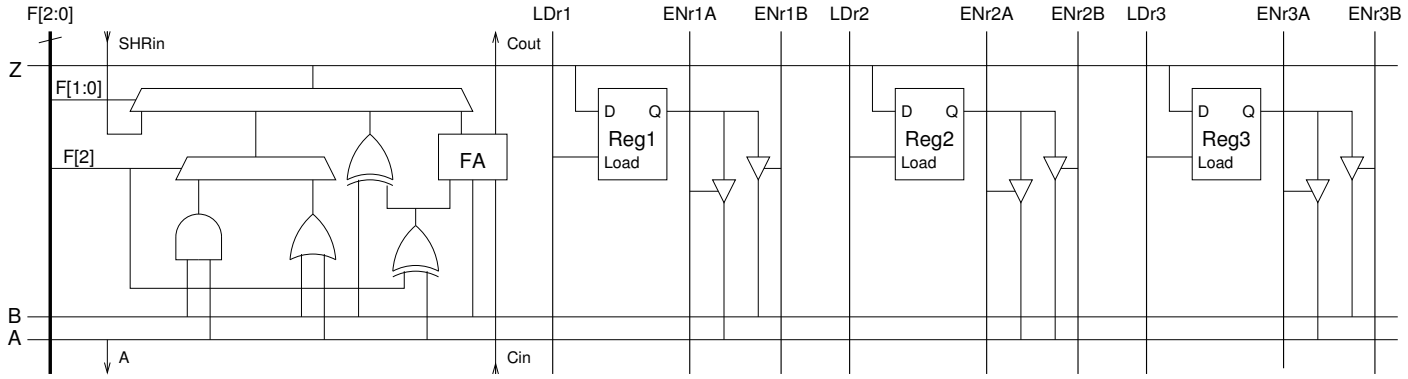


Bit Slicing

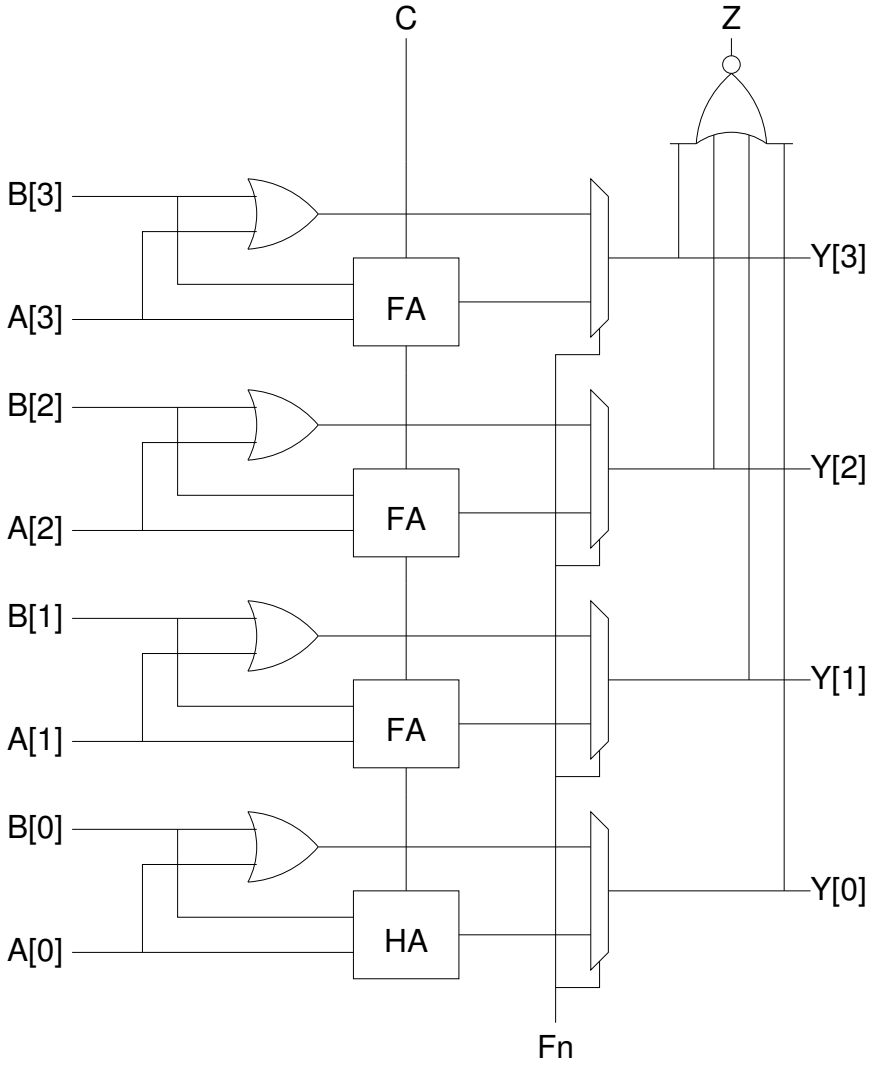
- Distributed Multiplexing



- Local Multiplexing

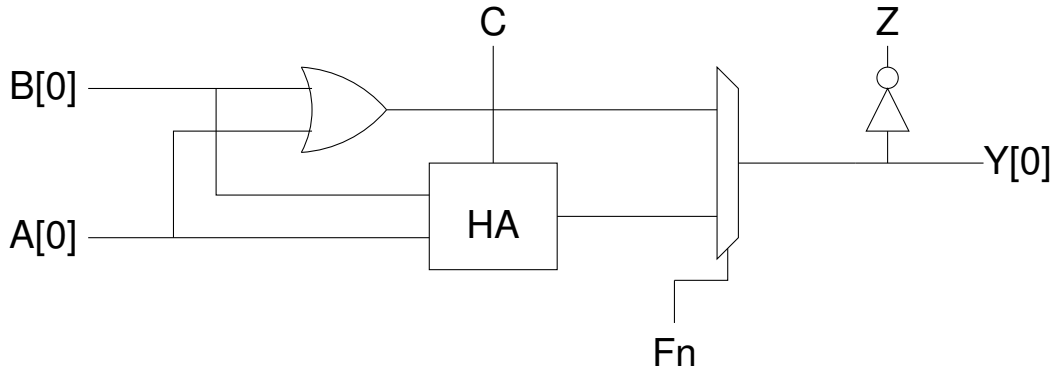


Bit Slicing - Example

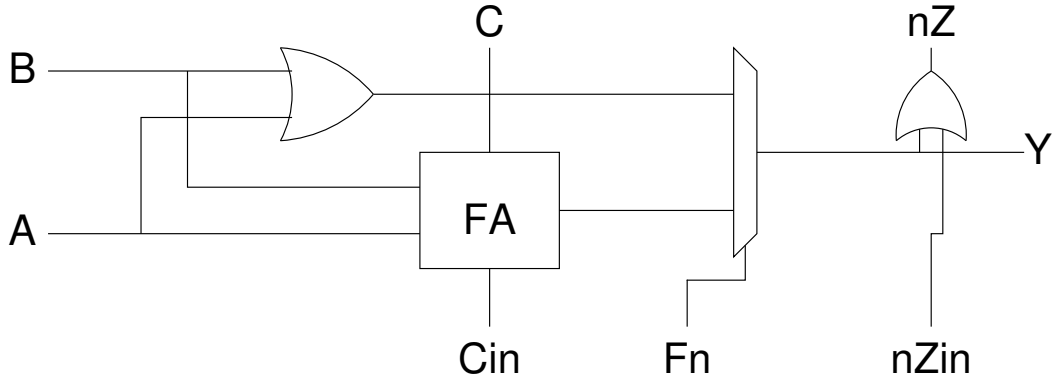


Bit Slicing - Example

1 Bit ALU (no consideration of bitslicing)



Bitslice Design for ALU



Bit Slicing - Example