Digital Logic Circuits 'Common Combinational Circuits'

ELEC2200

Summer 2009

David J. Broderick
brodedj@auburn.edu
http://www.auburn.edu/~brodedj
Office: Broun 360

Design Validation

- Design is an important but small part of our jobs
- Before design is completed
    - Development of requirements
- After design is completed
    - Design Validation
    - Documentation of work

Design Validation

- 800 people designed Pentium 4
    - Documentation is important!
- Insufficient design verification
    - Design error in floating-point division in Pentium processor in mid-1990's
    - Intel recalled 5 million devices
    - Cost: $500,000,000
- **Check your answers**

Hardware Description Languages

- Netlist (ASL)
    - Connections via signal name
- Schematic
    - Connections either explicit of via signal name
    - Produces a netlist for simulation
- Higher level language (VHDL or Verilog)
    - Synthesis to gate level netlist
- All of these result in design descriptions that can be used for simulation

AUSIM

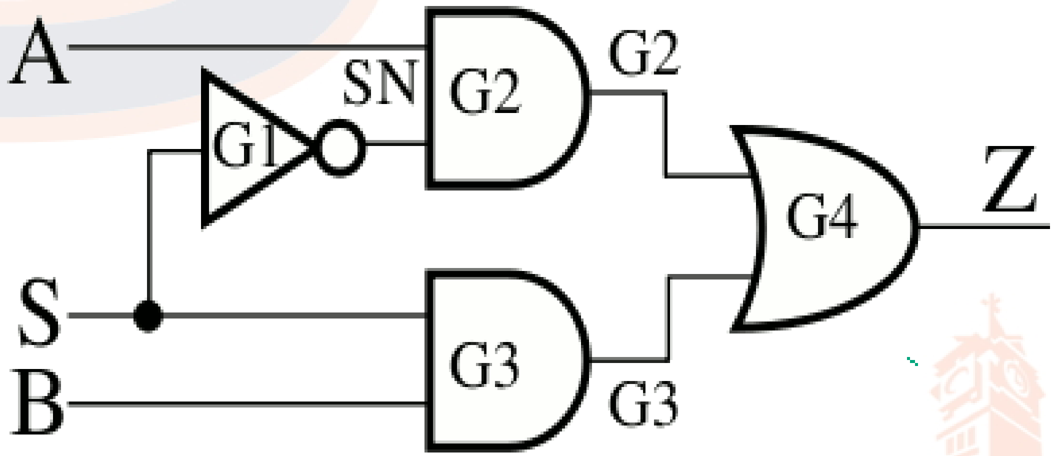
- Netlist description of design
- 4 files involved in a design
  - Input:
    - example.asl – netlist for design
    - example.vec – simulation input vectors
  - Output:
    - example.aud – analysis results
    - example.out – output of simulation

ASL

- Hardware Description Language (Netlist)
- example.asl:

```
ckt: MUX in: A B S out: Z ;
```

Notice:
circuit name
inputs
outputs
spaces

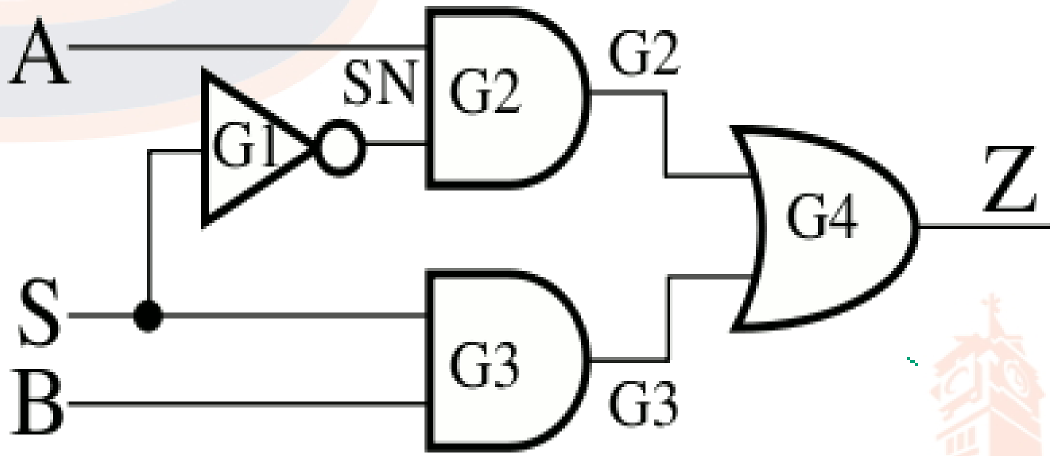


ASL

- Hardware Description Language (Netlist)
- example.asl:

```
ckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
```

Notice:
gate type
gate name
inputs
outputs

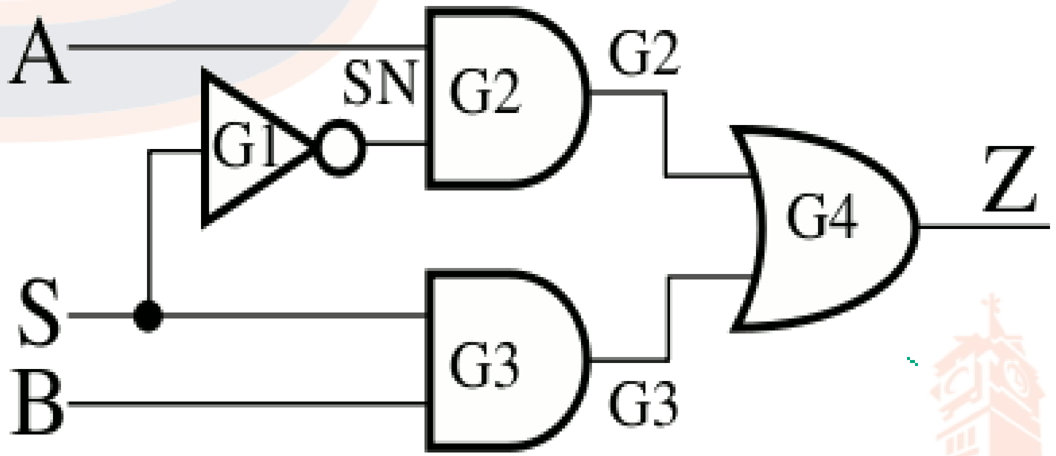


ASL

- Hardware Description Language (Netlist)
- example.asl:

```
ckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
```

Notice:
Output signal name matches



VEC

- Each line is one input combination
- Order of inputs from example.asl preserved
- Lines starting with '#' are comments and transferred verbatim to the output file

VEC

- example.vec:

```
# These are all of my test inputs ;
  000
  001
  010
  011
  100
  101
  110
  111
```

- This is a natural order for test vectors but not required

OUT

- example.out:

```
# ABS Z ;
# These are all of my test inputs ;
  000 0
  001 0
  010 0
  011 1
  100 1
  101 0
  110 1
  111 1
```

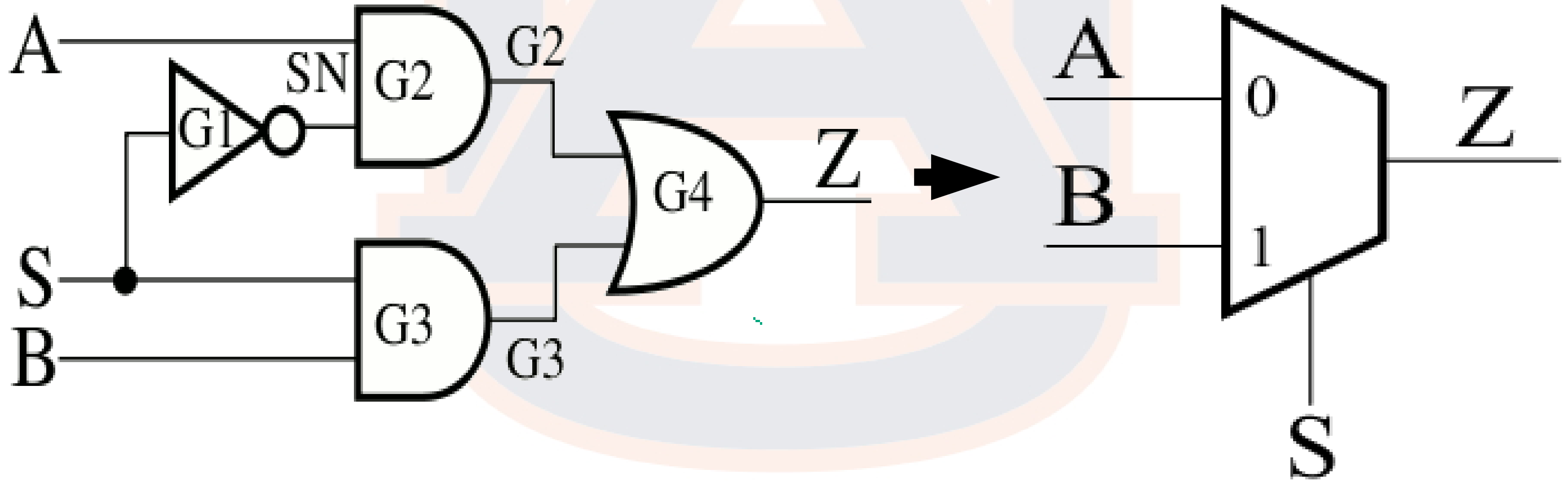
- Similar to .vec file
- Are outputs what we expected?

AUD

- Analysis Results
- Contains all 4 metrics
    - # of gates
    - # of inputs/outputs
    - G_{del} through worst case path
    - P_{del} through worst case path
- Let's take a look...

Subcircuits

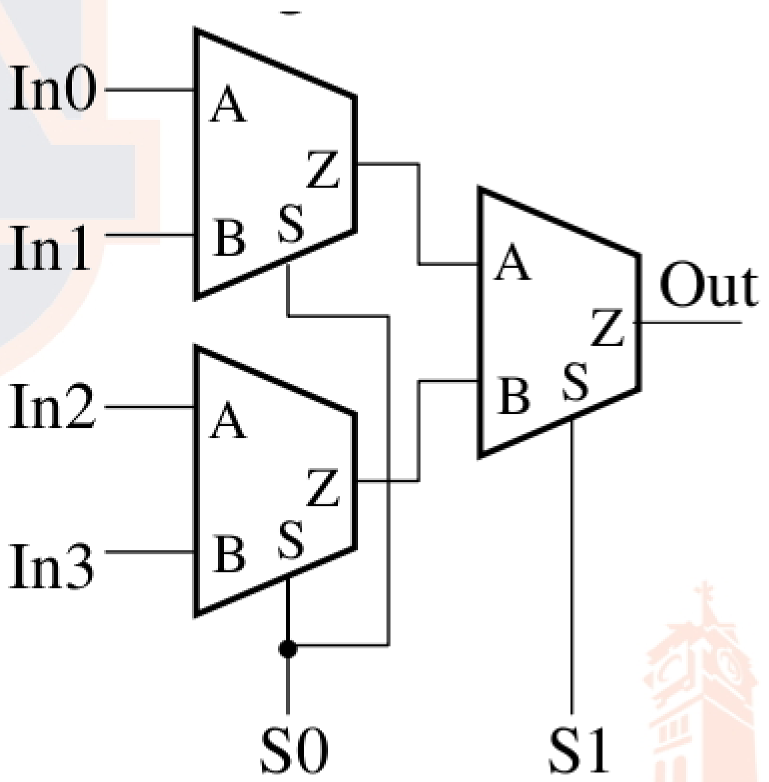
- Modular design makes things faster
- Let's take our 2-to-1 MUX and build a 4-to-1 MUX



Subcircuits

- A 4-to-1 MUX
- MUX4.asl:

```
subckt: MUX in: A B S out: Z ;
   not: G1 in: S out: SN ;
   and: G2 in: A SN out: G2 ;
   and: G3 in: S B out: G3 ;
   or: G4 in: G2 G3 out: Z ;
```



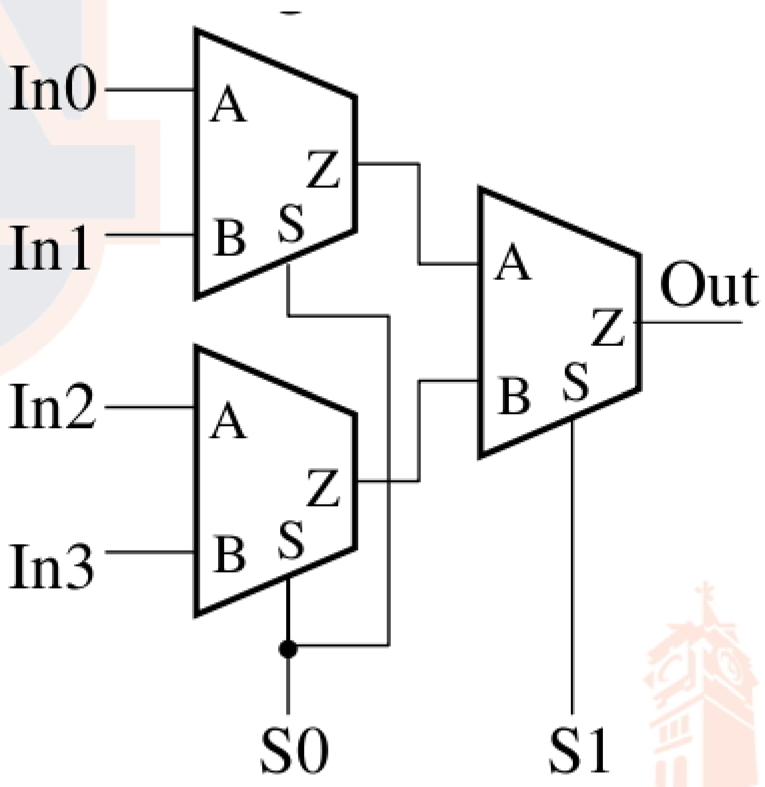
- Subcircuit taken from our previous example

Subcircuits

- A 4-to-1 MUX
- MUX4.asl:

```
subckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
```

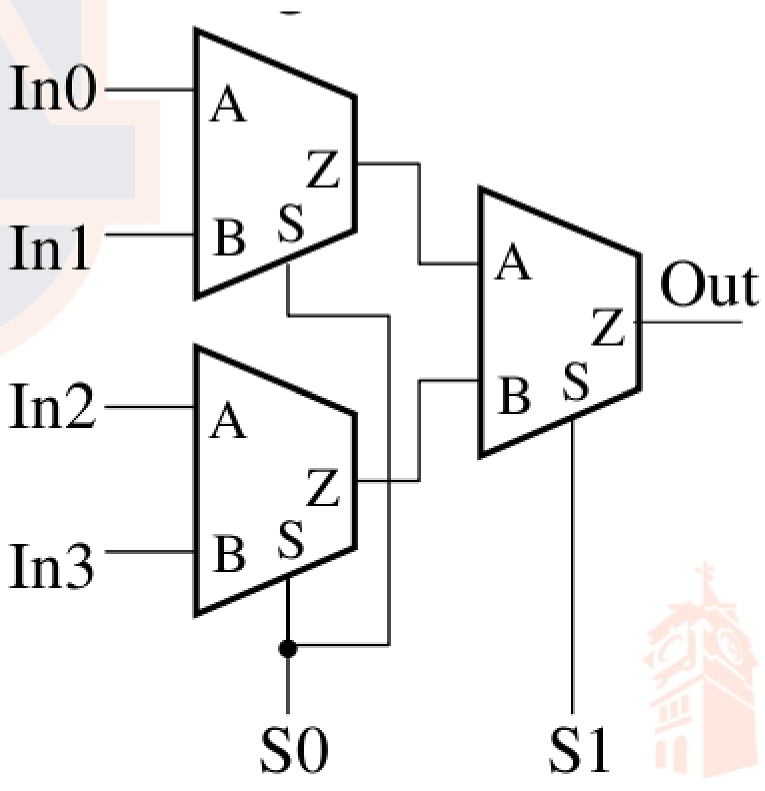
- Subcircuit taken from our previous example



Subcircuits

- A 4-to-1 MUX
- MUX4.asl:

```
subckt: MUX in: A B S out: Z ;
not: G1 in: S out: SN ;
and: G2 in: A SN out: G2 ;
and: G3 in: S B out: G3 ;
or: G4 in: G2 G3 out: Z ;
ckt: MUX4 in: In[0:3] S[0:1] out: Out ;
MUX: M1 in: In0 In1 S0 out: M1 ;
MUX: M2 in: In2 In3 S0 out: M2 ;
MUX: M3 in: M1 M2 S1 out: Out ;
```



Validation of MUX4

- MUX4.vec:
- Could use natural order of vectors 000000 to 111111
- Could also group vectors to ease validation

Timing Issues

- AUSIM validates steady-state behavior well
- Transient errors also possible
- Delays differ through various paths, can lead to timing hazards (AKA glitches)
    - Static-1 hazard: produces a glitch to logic 0
    - Static-0 hazard: produces a glitch to logic 1

Static Hazards

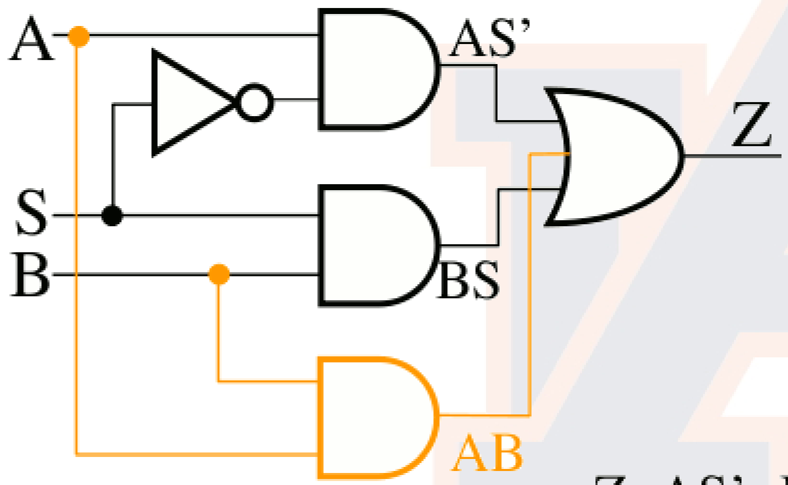
- Static-1 Hazard caused by a pair of inputs that:
    - Are logically adjacent
    - Both produce an output of 1
    - Cause a momentary logic 0 (called a 0 glitch) during a transition from one input value to the other
- Static-0 Hazard caused by a pair of inputs that:
    - Are logically adjacent
    - Both produce an output of 0
    - Cause a momentary logic 1 (called a 1 glitch) during a transition from one input value to the other

Static Hazards

- Important properties:
    - A properly designed 2-level AND-OR (SOP) circuit has no static-0 hazards, but may have static-1 hazards
    - A properly designed 2-level OR-AND (POS) circuit has no static-1 hazards, but may have static-0 hazards

Static Hazard Example

Multiplexer logic diagram



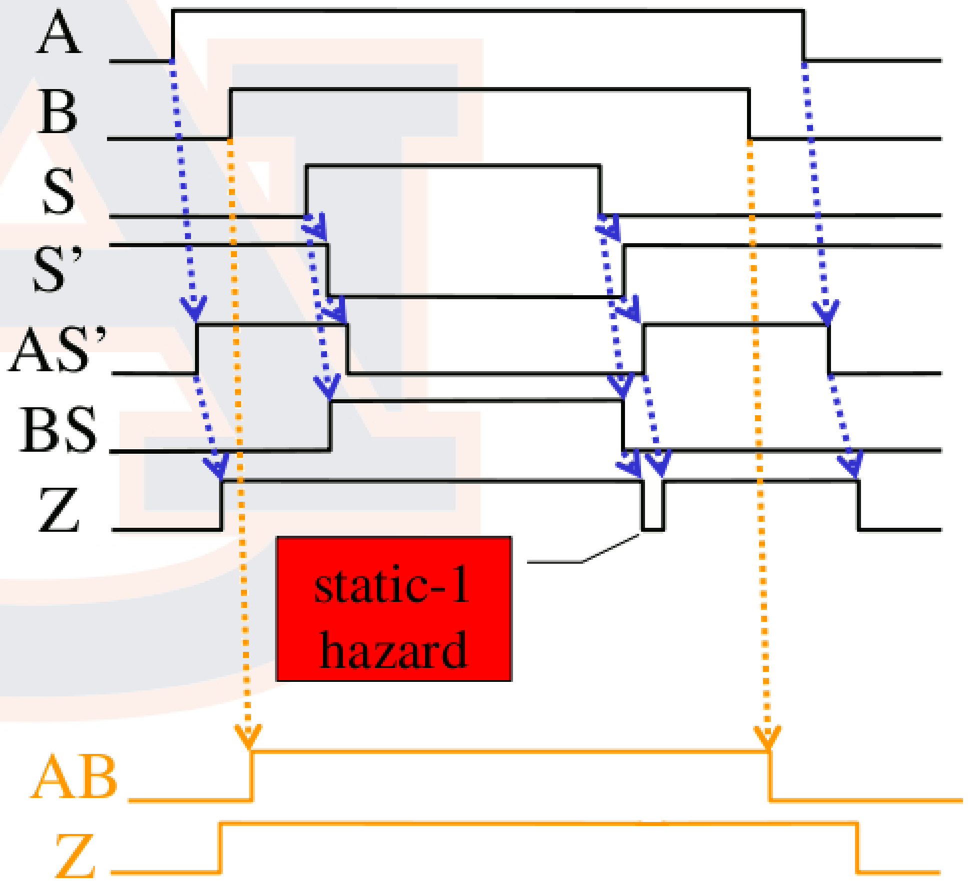
Z=AS'+BS

S \ AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Note: this group removes hazard

Z=AS'+BS+AB

Timing diagram



Hazard-free multiplexer output

David J. Broderick