Combinational Logic Design Process

1. **Understand the Problem**
   - **what is the circuit supposed to do?**
   - **write down inputs (data, control) and outputs**
   - **draw block diagram or other picture**
2. **Formulate the Problem in terms of a truth table or other suitable design representation**
   - **truth table, Boolean Algebra, Verilog, etc.**
3. **Choose Implementation Target**
4. **Follow Implementation Procedure**
   - **K-maps, Boolean algebra, Quartus synthesis**

Process Line Control Example

Statement of the Problem

Rods of varying length (+/-10%) travel on conveyor belt
Mechanical arm pushes rods within spec (+/-5%) to one side
Second arm pushes rods too long to other side
Rods too short stay on belt

3 light barriers (light source + photocell) as sensors

Design combinational logic to activate the arms

Understanding the Problem

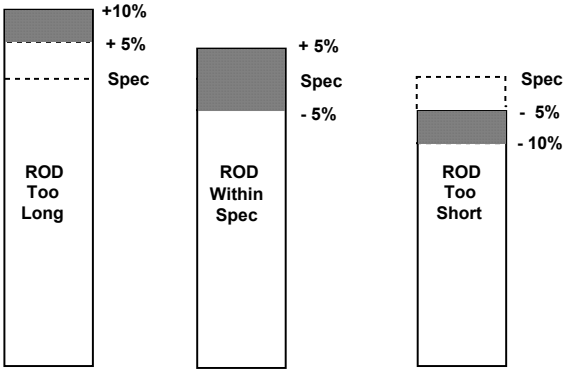
Inputs are three sensors, outputs are two arm control signals

Assume sensor reads "1" when tripped, "0" otherwise

Call sensors A, B, C

Draw a picture!

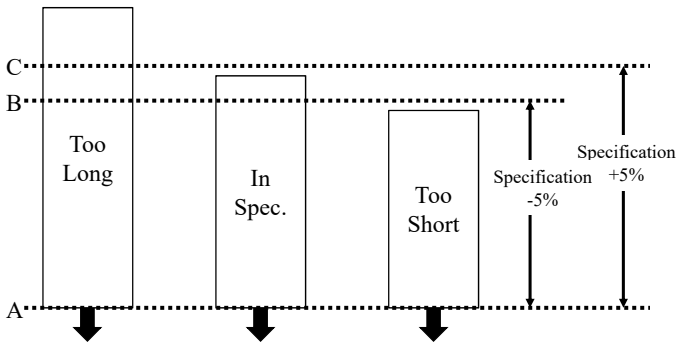
Process Line Control Example (cont.)



Where to place the light sensors A, B, and C to distinguish among the three cases?

Assume that A detects the leading edge of the rod on the conveyor

Process Line Control Example (cont.)



A to B distance place apart at specification - 5%

A to C distance placed apart at specification +5%

Process Line Control Example (cont.)

A	B	C	Meaning	Accept	Long
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Logical Function Unit

Create a unit that can compute the AND, OR, or XOR of two inputs A and B, based upon control lines C0 and C1.

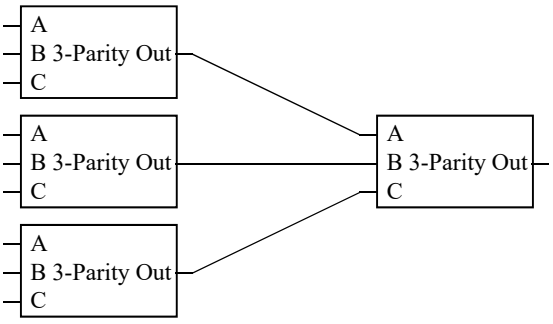
Similar to the main computation unit in a Microprocessor

Logical Function Unit (cont.)

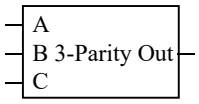
- Implementation:

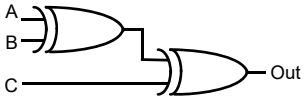
Debugging Complex Circuits

- Complex circuits require careful debugging
  - Rip up and retry?
- Ex. Debug a 9-input odd parity circuit
  - True if an odd number of inputs are true



Debugging Complex Circuits (cont.)





Debugging Approach

- Test all behaviors.
  - All combinations of inputs for small circuits, subcircuits.
- Identify any incorrect behaviors.
- Examine inputs and outputs to find earliest place where value is wrong.
  - Typically, trace backwards from bad outputs, forward from inputs.
  - Look at values at intermediate points in circuit.
- DO NOT RIP UP, DEBUG!