Quiz #2: Train Crossing Controller

I/O Description

Inputs:

A0 Detect that a train has entered the crossing zone

A1 Detect that a train has exited the crossing zone

A2 Sensor (Detects that the crossbar is fully lowered)

A3 Sensor (Detects that the crossbar is fully raised)

A5-A4 Speed of the train

- **00:** Slow
- **01:** Medium
- **10:** Fast
- **11:** Shinkansen (Japanese Bullet Train)

Outputs:

B0 PWM signal to raise/lower the crossbar

B1 Crossbar Direction

- **0:** Lower the crossbar
- **1:** Raise the crossbar

System Functionality (1/2)

- When the train enters the crossing zone (A0 = 1) the system detects the velocity of the train (A5-A4) and lowers the crossbar (B1 = 0). A PWM signal on B0 controls the rotational velocity of the servo that lowers the crossbar:

– Velocity = Low:	20% duty cycle
– Velocity = Medium:	40% duty cycle
– Velocity = High:	60% duty cycle
– Velocity = Shinkansen:	80% duty cycle

System Functionality (2/2)

The PWM signal stops (B0 = 0) when sensor A2 detects that the crossbar is fully lowered (A2 = 1).

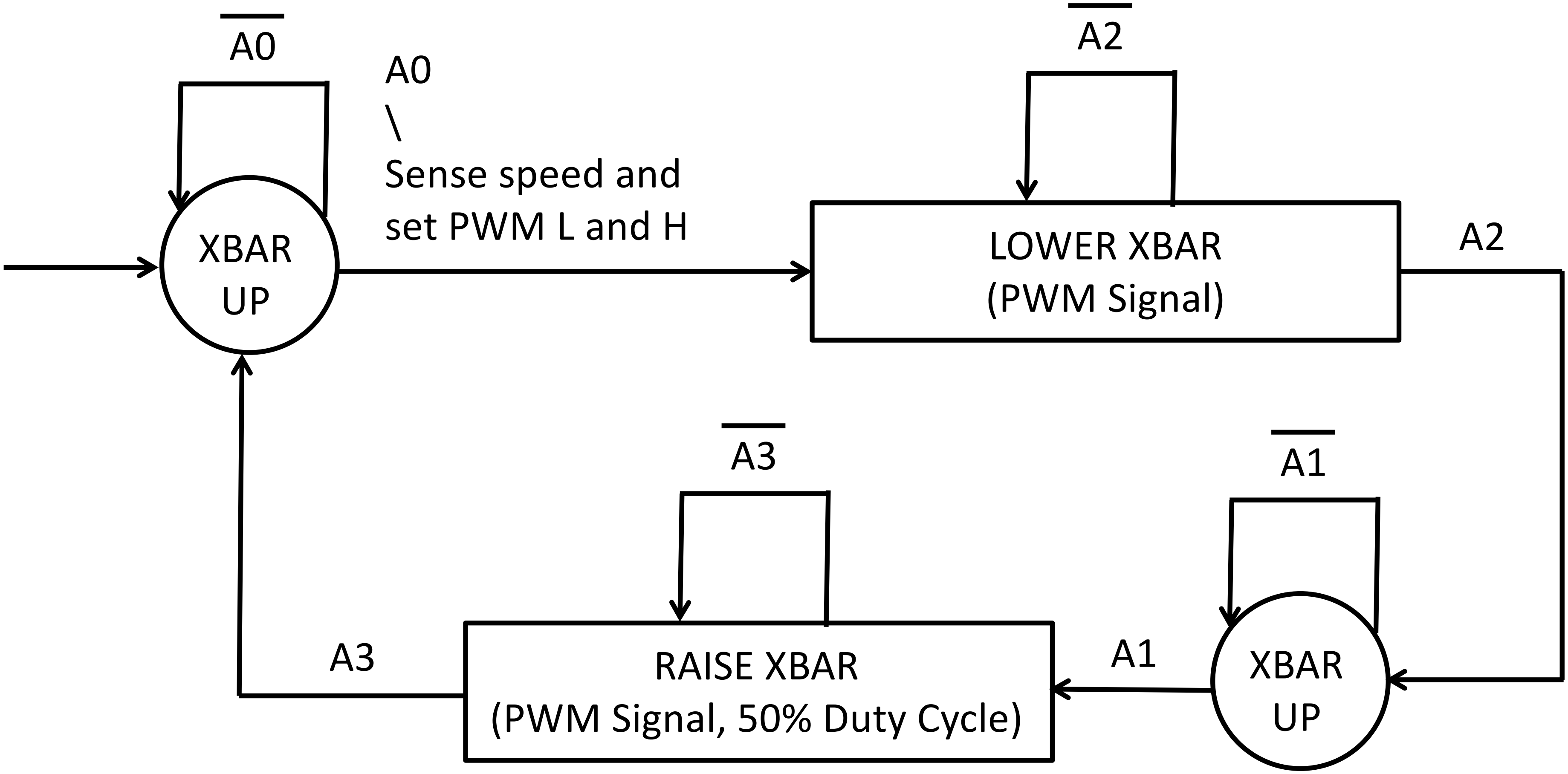
When the train exits the crossing zone (A1 = 1) the system raises the crossbar (B1 = 1) using a PWM signal on B0 with a 50% duty cycle. The PWM signal stops (B0 = 0) when sensor A3 detects that the crossbar is fully raised (A3 = 1).

PWM period:	1000 ms
System period:	100 ms

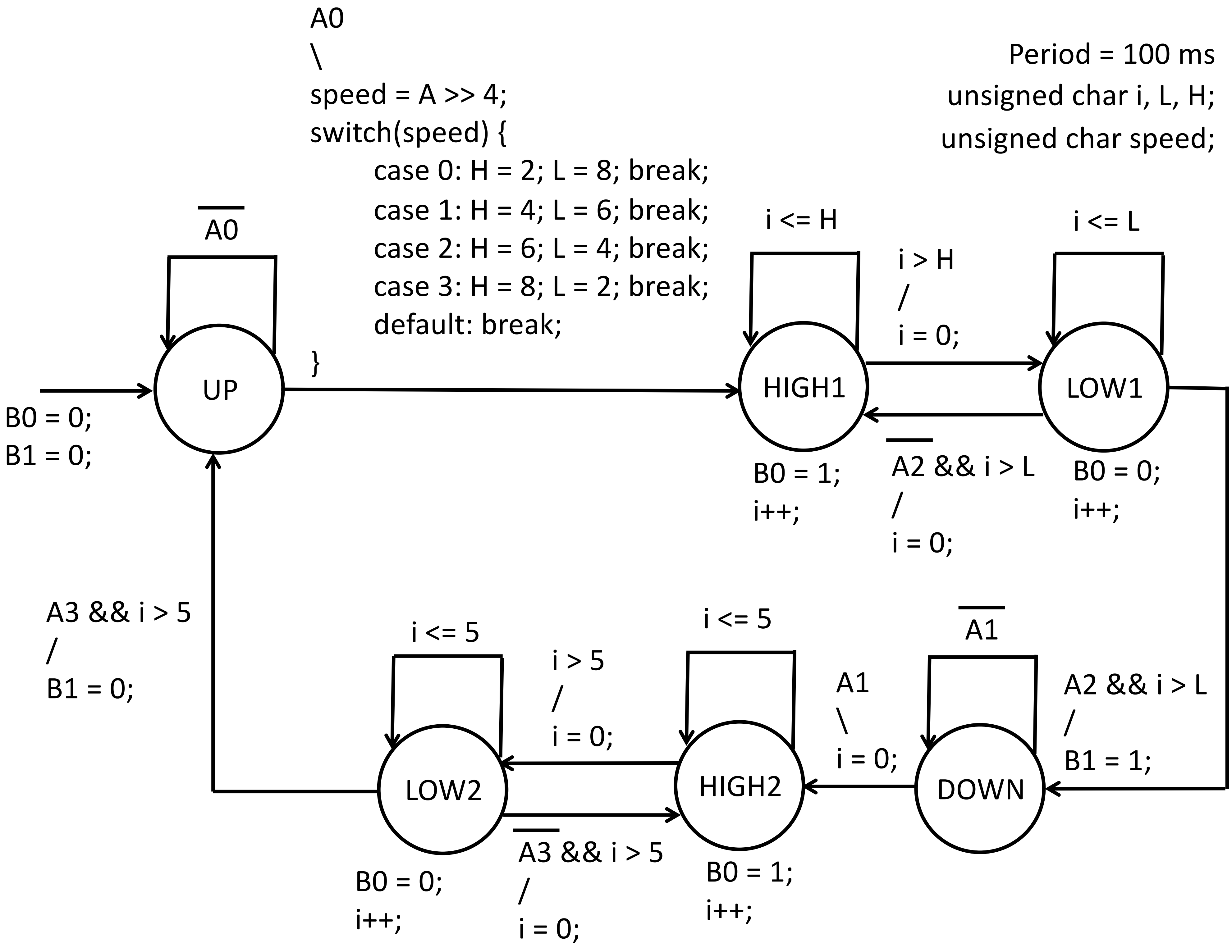
Simplifying Assumptions

- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)

SM Design Strategy (Overview)



(Complete SynchSM on Next Slide)



```
Period = 100 ms
unsigned char i, L, H;
unsigned char speed;
```

```
A0
\
speed = A >> 4;
switch(speed) {
        case 0: H = 2; L = 8; break;
        case 1: H = 4; L = 6; break;
        case 2: H = 6; L = 4; break;
        case 3: H = 8; L = 2; break;
        default: break;
}
```

Simplifying Assumptions in Detail

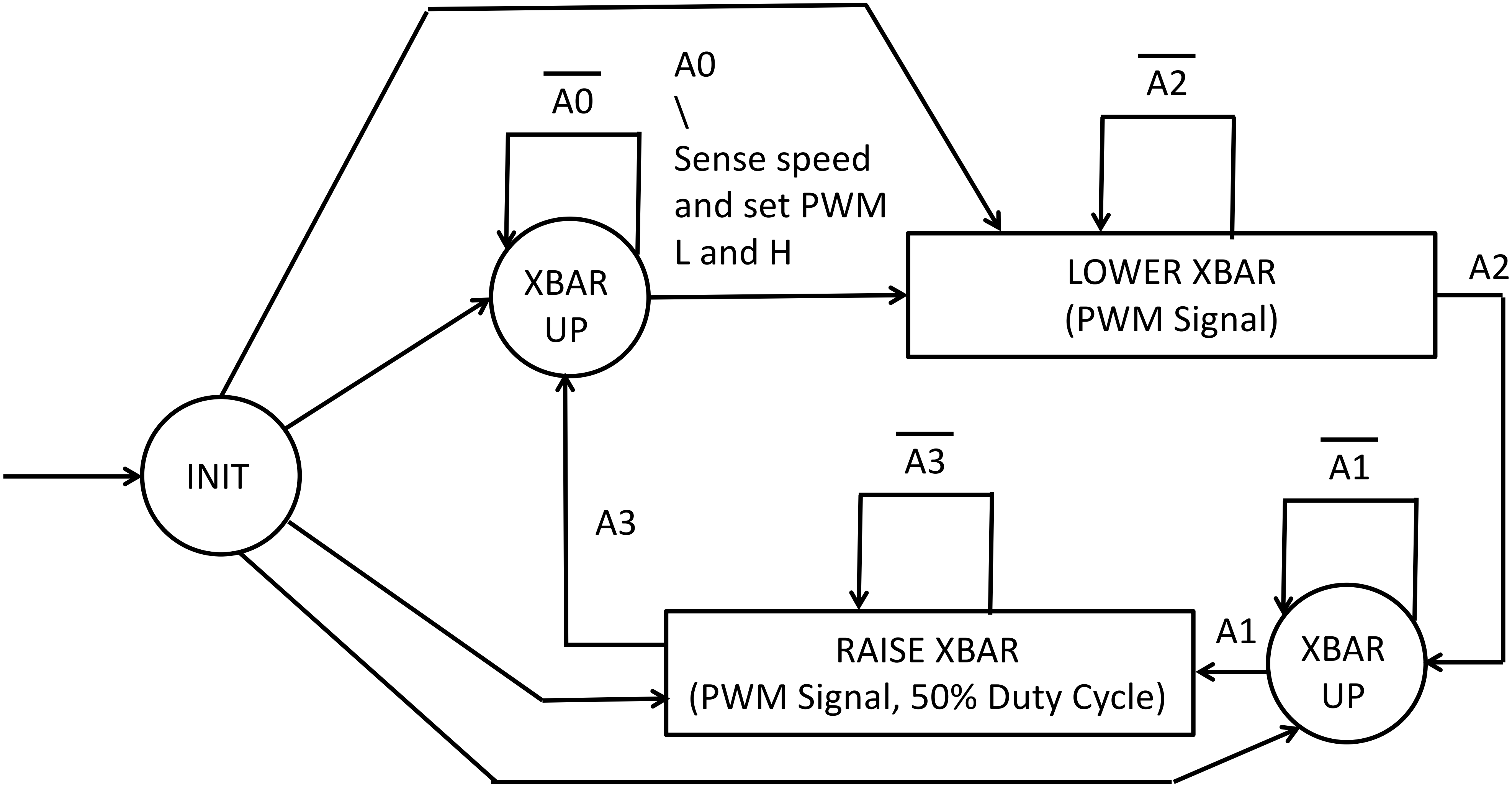
- The following slides will discuss each simplifying assumption and show how it leads to a simpler SynchSM Design

- These are needed for a realistic system, but would make the quiz too hard to complete in 20-25 minutes

Let’s Drop the Assumption in Black

- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)

SM Design Strategy



Transition out of INIT is based on an assessment of the state when the system starts
This is a non-trivial

Let’s Drop the Assumption in Black

- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)

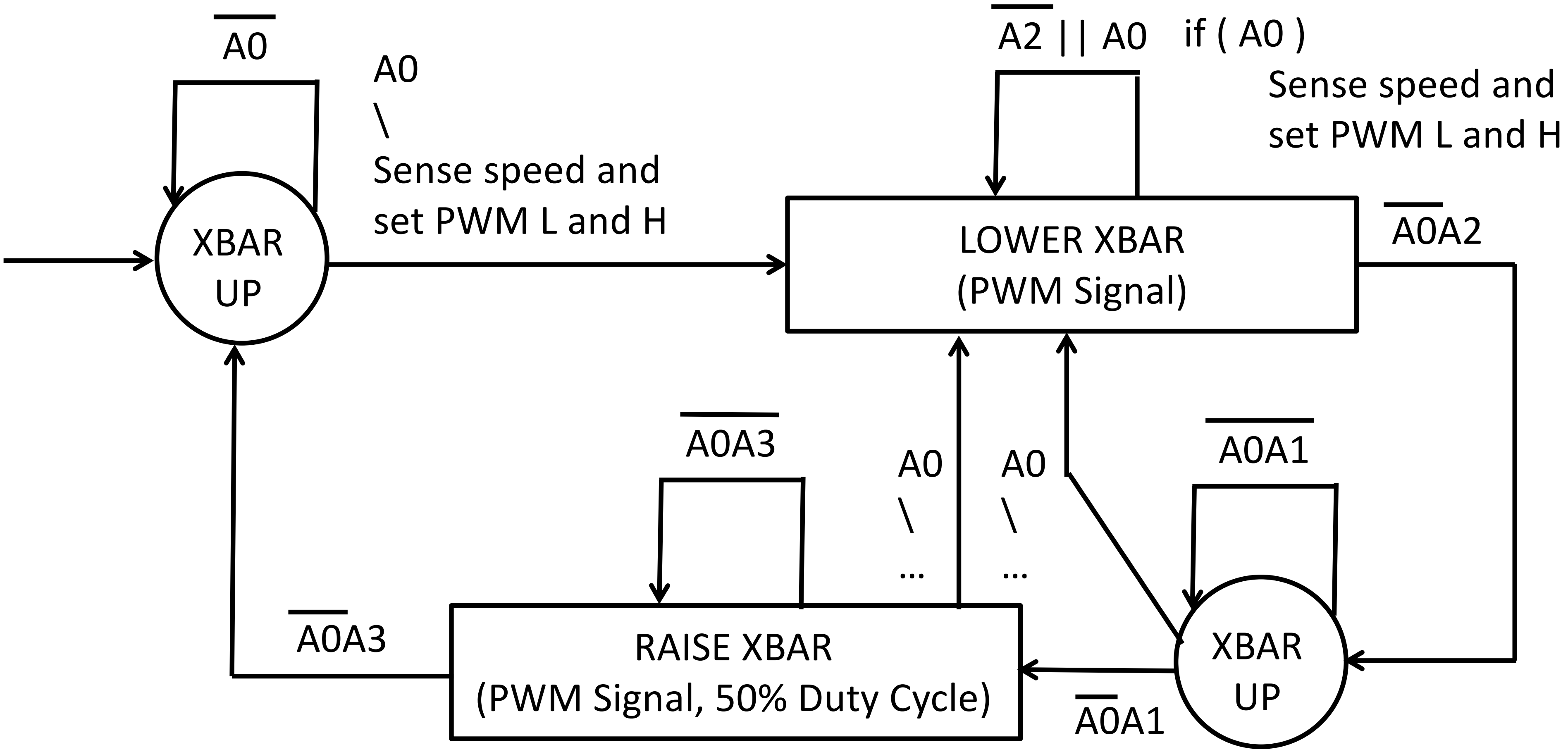
SM Design Strategy

- Without going into too much detail, we'll need to replicate the inputs/outputs per track
- We'll need multiple microcontrollers, etc.
- Assuming we have more inputs, this is a perfect example of where concurrent SynchSM would be much easier to design/debug than a single SynchSM
  - Try it out for yourself
- Plus, all complications that arise from supporting multiple trains in the crossing
  - See the next example

Let’s Drop the Assumption in Black

- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)

SM Design Strategy (Overview)

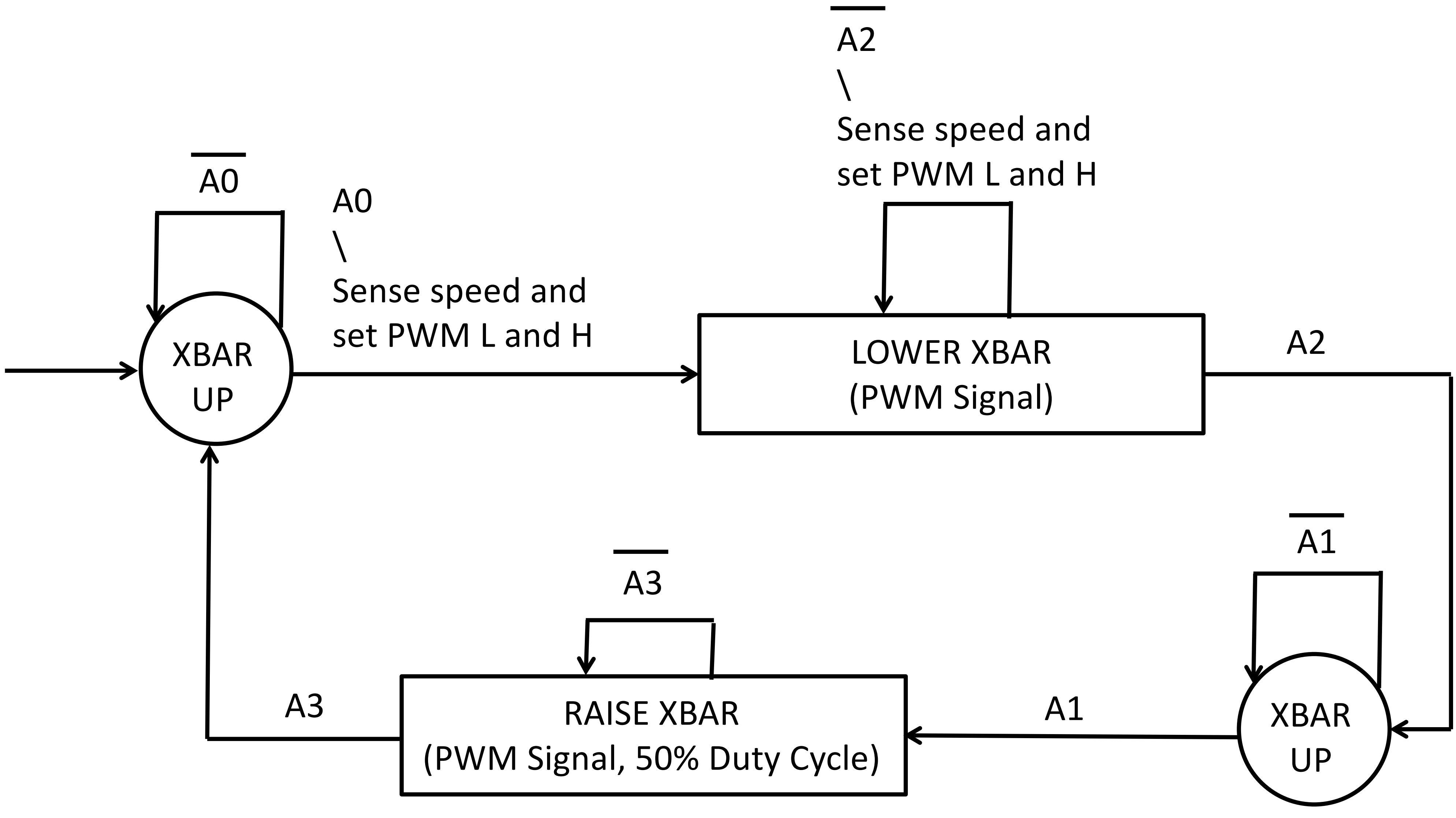


(Now A0 factors into most conditions
to progress through the SynchSM)

Let’s Drop the Assumption in Black

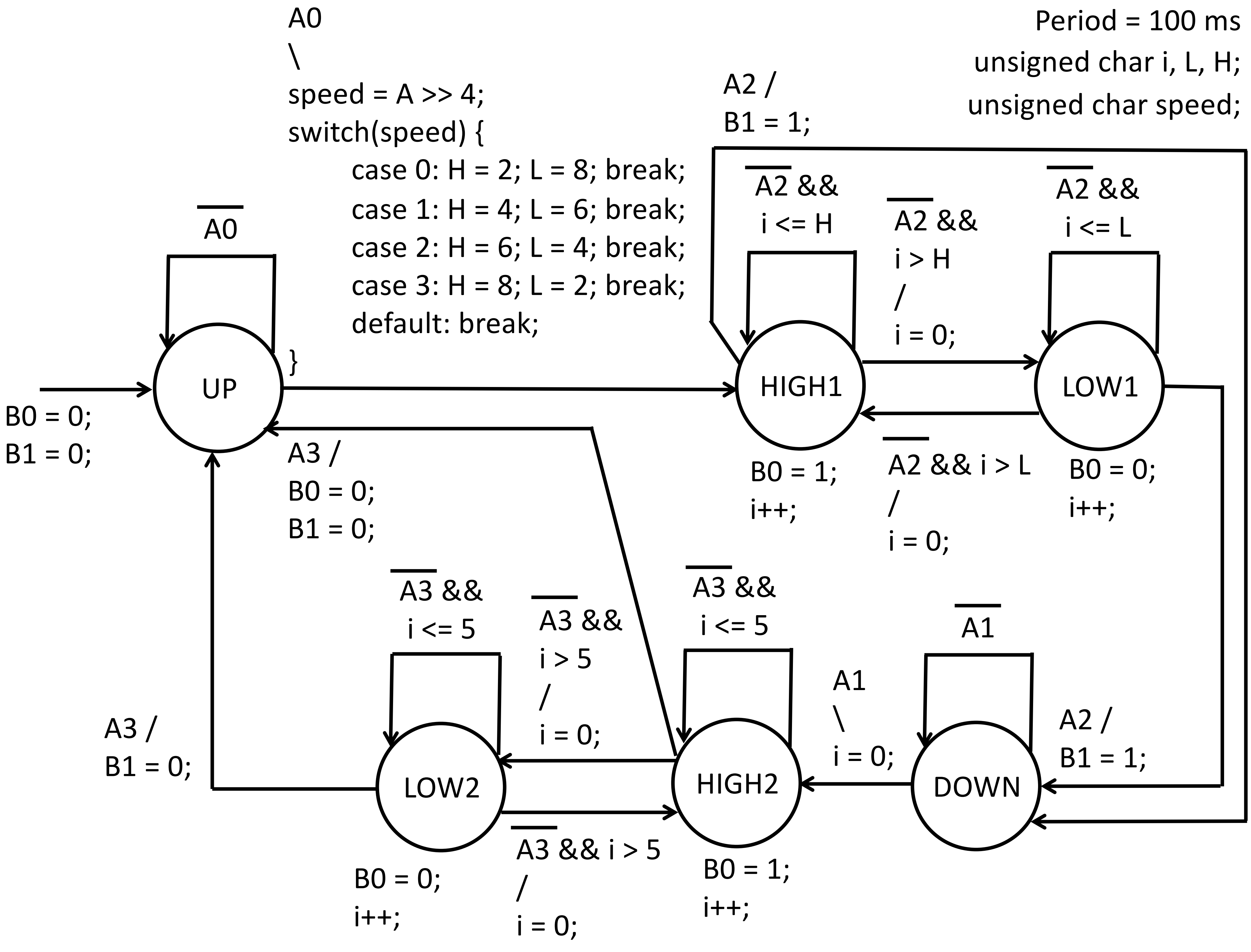
- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)

SM Design Strategy (Overview)



Let’s Drop the Assumption in Black

- No train is crossing when the system is first turned on
- There is only one train track at the crossing
- The system works correctly regardless of the direction that the train is traveling
- At most one train will travel through the detection zone at any time
- The train travels at a constant velocity throughout the crossing
- All sensors behave correctly and as expected
  - Ignore A3 while lowering the crossbar
  - Ignore A2 while raising the crossbar
- It is OK to wait until the end of a PWM cycle to detect that the crossbar is fully lowered (A2 = 1) or raised (A3 = 1)



```
Period = 100 ms
unsigned char i, L, H;
unsigned char speed;
```

```
A0
\
speed = A >> 4;
switch(speed) {
        case 0: H = 2; L = 8; break;
        case 1: H = 4; L = 6; break;
        case 2: H = 6; L = 4; break;
        case 3: H = 8; L = 2; break;
        default: break;
}
```

The Previous Slide (Summary)

- An unconscionable mess
- Internal PWM transitions now depend on A2 (lower) and A3 (raise)
- Transitions out of both the LOW and HIGH states of both PWM generators, instead of just LOW

Assumptions (Summary)

- See how each assumption adds complexity to the SynchSM design
  - Imagine building this SynchSM without **any** of these assumptions
    - Lots of subtle bugs
    - Hard to fit everything on an 8.5x11” page
  - Imagine doing the above in 20-25 minutes