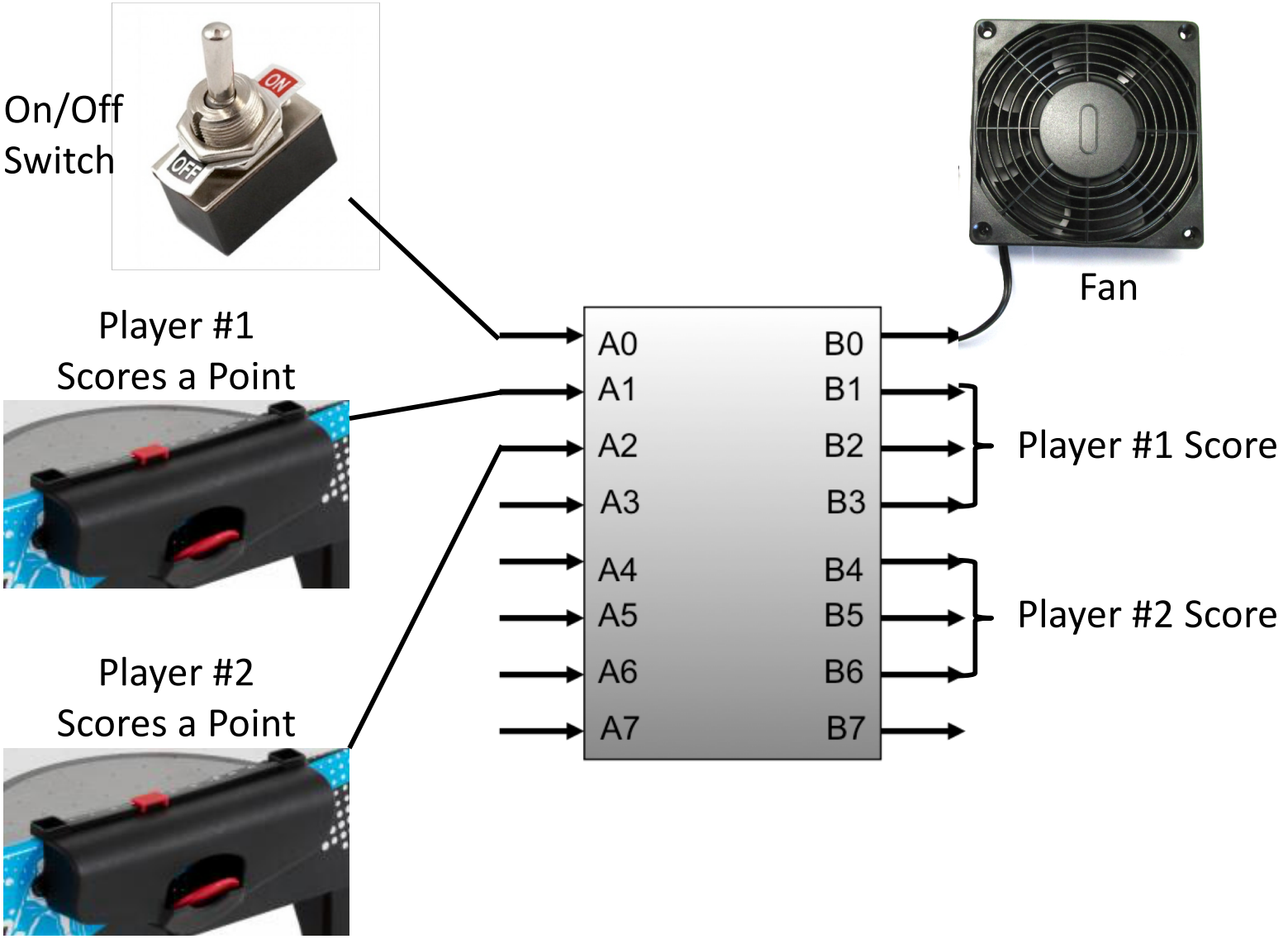
Exam #1: Air Hockey Table Controller

System Diagram



I/O Description

Inputs:

A0 – On/Off switch

A1 – Set to 1 when Player #1 scores a point; 0 otherwise

A2 – Set to 1 when Player #2 scores a point; 0 otherwise

(You can assume that A1 and A2 are never both 1 at the same time)

Outputs:

B0 – Fan control (**B0 = 1** to run the fan; **B0 = 0** when the fan is off)

B3B2B1 – Player #1 score (**B3** is the most significant bit

System Functionality (1/2)

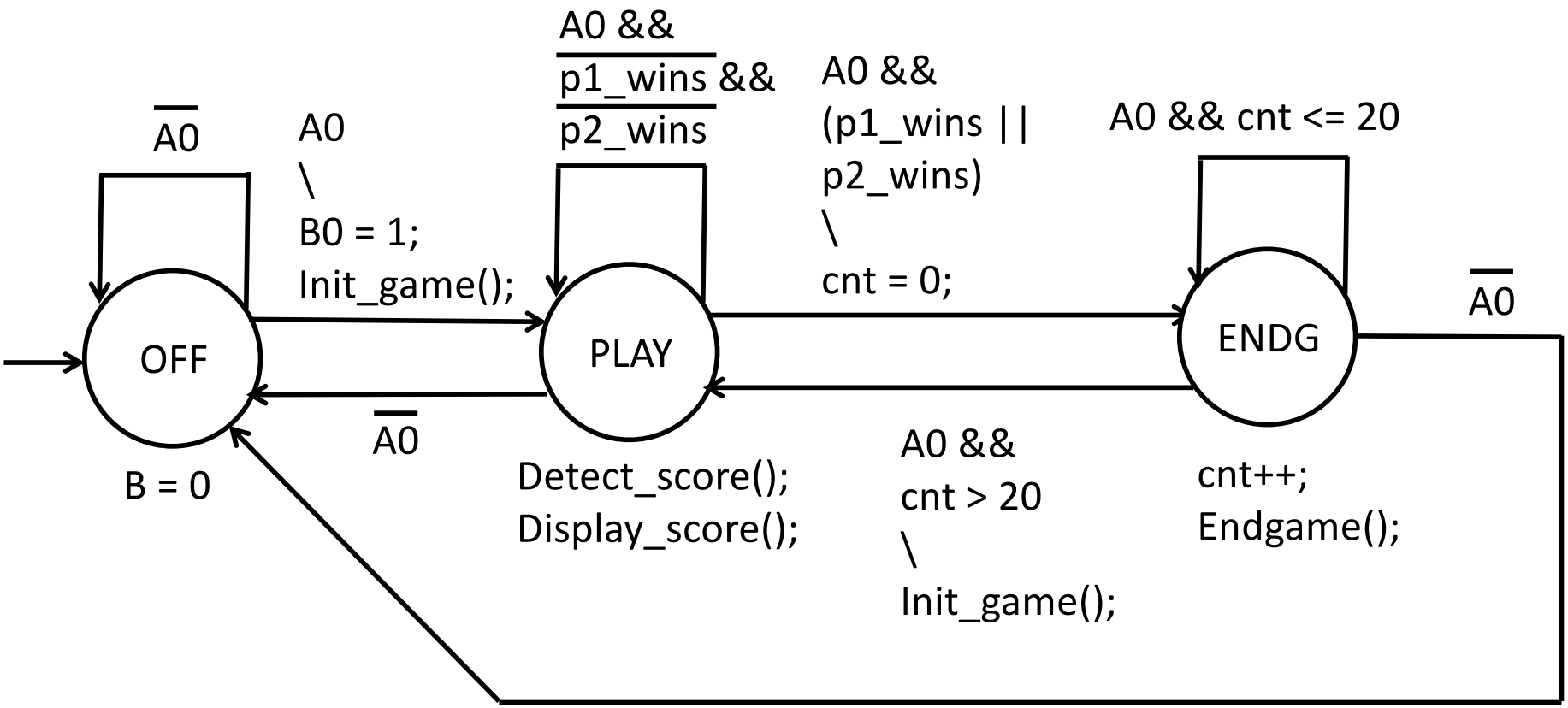
- Setting **A0=1** turns the game on.
- Setting **A0=0** during the game or the **endgame celebration** (see below) immediately turns off the system.
- The fan always runs (**B0=1**) while the game is on
- The score is initialized to 0-0 when the game begins
- During the game, the system runs the fan (**B0=1**) and displays the score of the two players on **B3B2B1** and **B6B5B4**.
- The system adds 1 point to Player #1's score when Player #1 scores a point (**A1=1**)
- The system adds 1 point to Player #2's score when Player #2 scores a point (**A2=1**)

System Functionality (2/2)

- There are two ways to win the game
  - The first player to score 7 points is the winner
  - Any player to score 3 points in a row wins, regardless of the overall score
- After a player wins, the system performs the **endgame celebration** for 20 ticks
  - The fan remains on (**B0 = 1**)
  - The winner's score flashes
    - The winner's score (**B3B2B1** or **B6B5B4**) is displayed on odd ticks
    - The winner's score is set to **000** on even ticks
  - The loser's score is set to **000**
  - After 20 ticks, the score is reset to 0-0 and a new game begins

Solution

(See Next 3 slides for SM variable declaration and function bodies)



SM Variables and Initialization

```
unsigned char p1_pts;
unsigned char p1_in_a_row;
unsigned char p1_wins;
unsigned char cnt;

unsigned char p2_pts;
unsigned char p2_in_a_row;
unsigned char p2_wins;

void Init_game() {
    p1_pts = p2_pts = 0;
    p1_in_row = p2_in_a_row = 0;
    p1_wins = p2_wins = 0;
}
```

Scoring Function

```
Detect_score() {
	if(A1) {
		p1_pts++;
		p1_in_a_row++;
		p2_in_a_row = 0;
		p1_wins = p1_pts == 7 || p1_in_a_row == 3;
	}
	else if (A2) {
		p2_pts++;
		p2_in_a_row++;
		p1_in_a_row = 0;
		p2_wins = p2_pts == 7 || p2_in_a_row == 3;
	}
}
```

Display Function

Display_score()

{ B = 0x01 | p1_pts << 1 | p2_pts << 4; }

Endgame Function

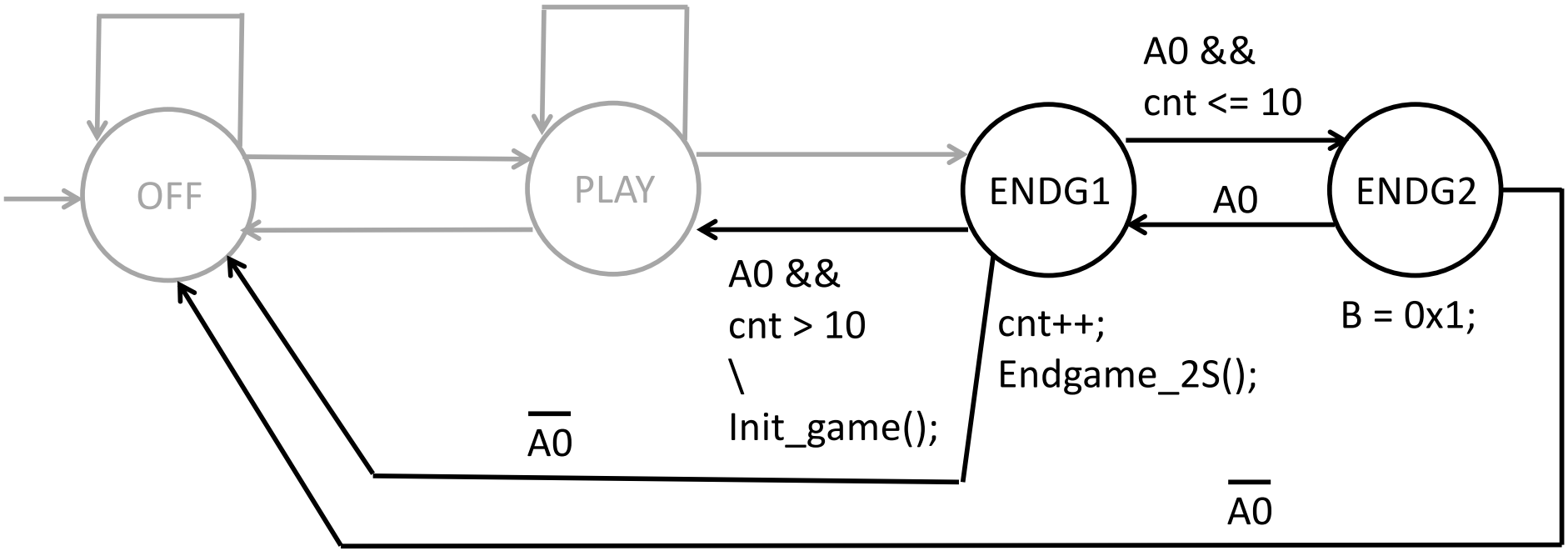
```
Endgame() {
    // Even tick
    if( cnt % 2 == 0 ) B = 0x01;

    // Odd tick; Player #1 won
    else if (p1_wins) B = 0x01 | p1_pts << 1;

    // Odd tick; Player #2 won
    else B = 0x01 | p2_pts << 4;
}
```

Alternative #1 (Sketch)

- Two Endgame States

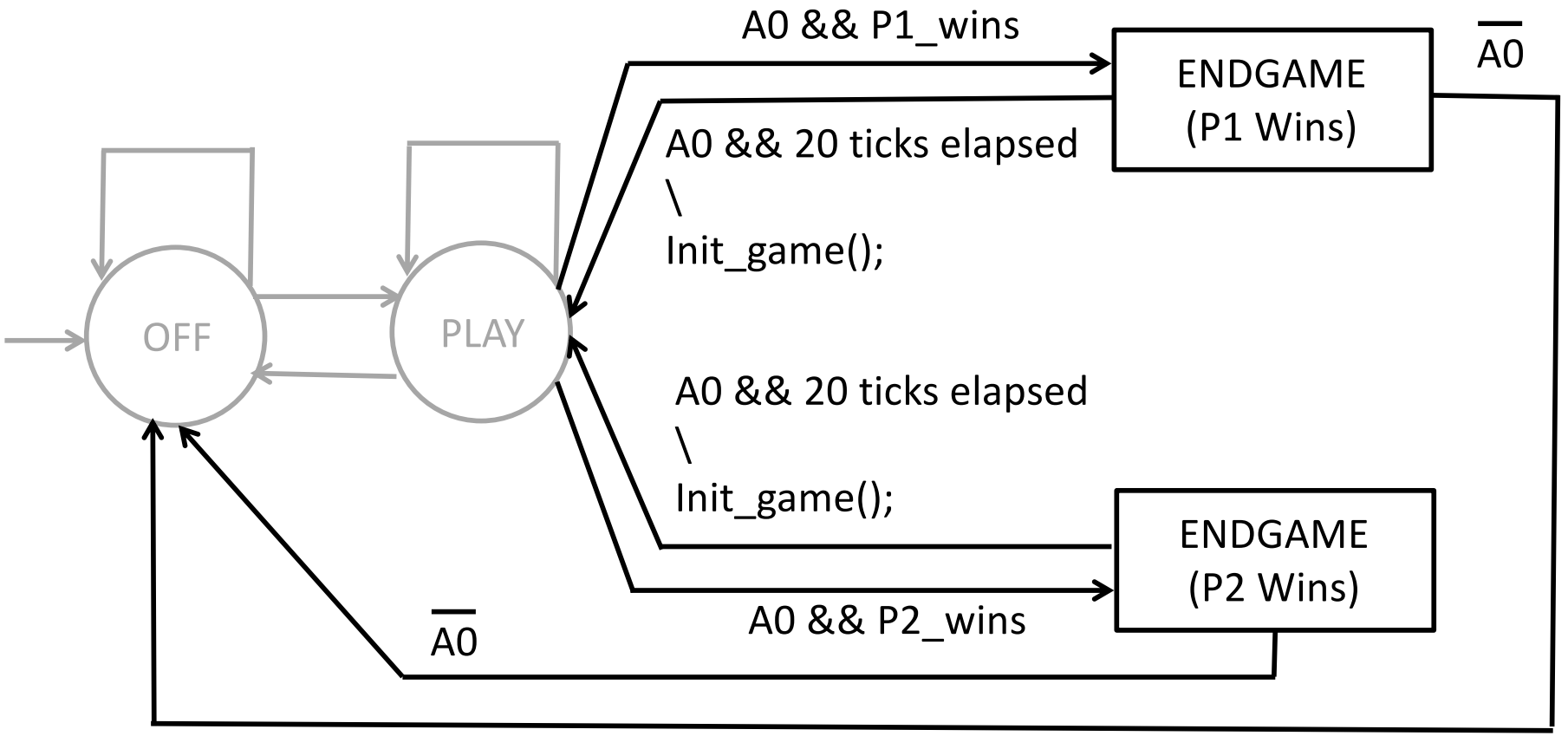


2-State Endgame Function

```
Endgame_2S() {
    if (p1_wins) B = 0x01 | p1_pts << 1;
    else B = 0x01 | p2_pts << 4;
}
```

Alternative #2 Sketch

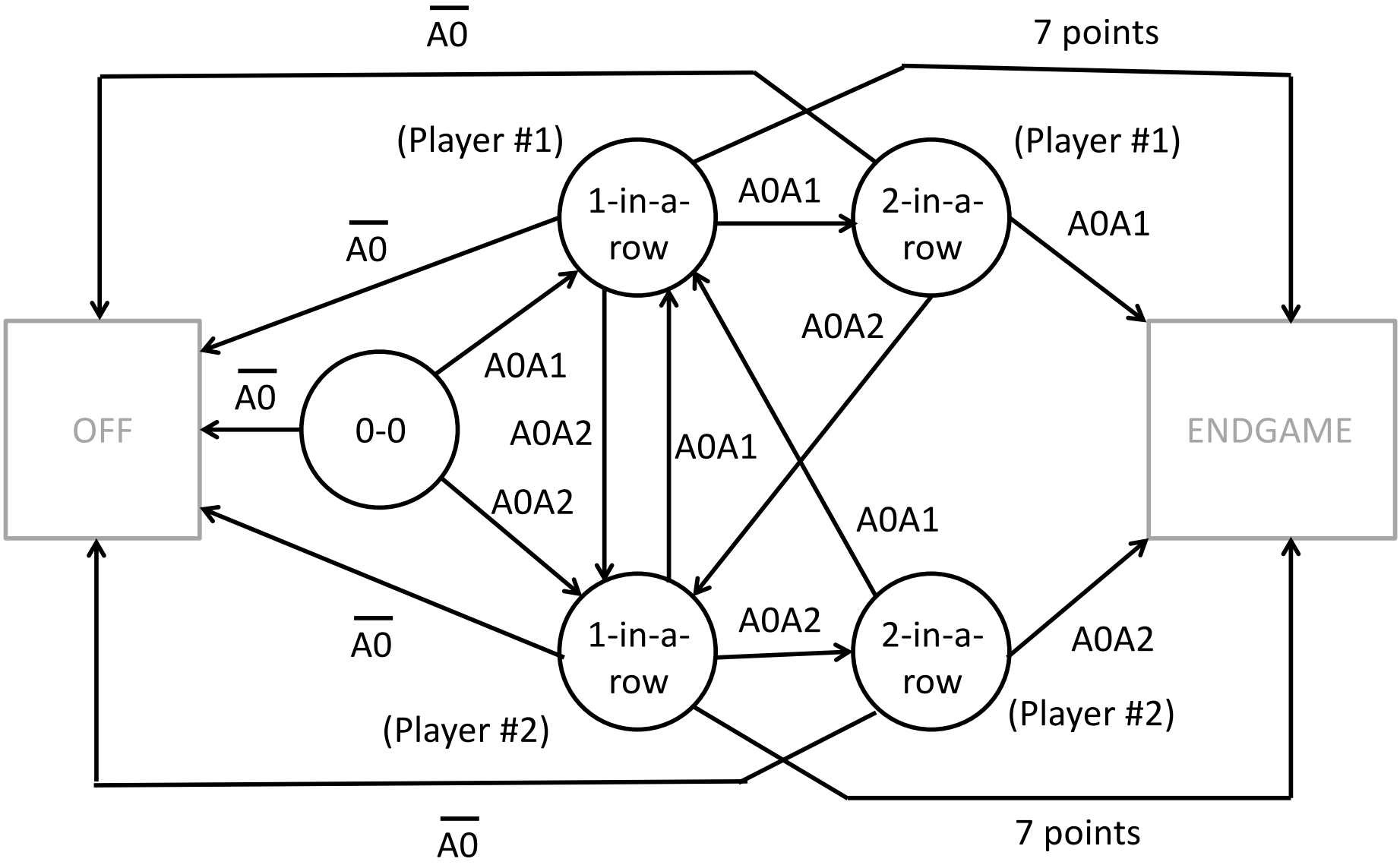
- Per-Winner EndGames
  - Could be 1- or 2-state Endgames



One Other SM Design Alternative

- Use state machine logic to detect if Player #1 or Player #2 scores 3 points in a row
  - Sketch on Next slight
  - Bottom line: it's a mess

Alternative #3 (Partial Sketch)



Why Are the Hints Important?

- **Hint #1**: It is OK to transition into the **endgame** one tick **after** detecting that a player has scored a point

- Let's assume that we cannot wait for one tick. Then we must do the following at once:
  - Detect if a point was scored
  - Update local variables (_pts, _in_a_row)
  - Detect if there was a winner
  - If there was a winner, transition to the endgame

- This would be a convoluted mess
  - Details on the next few slide

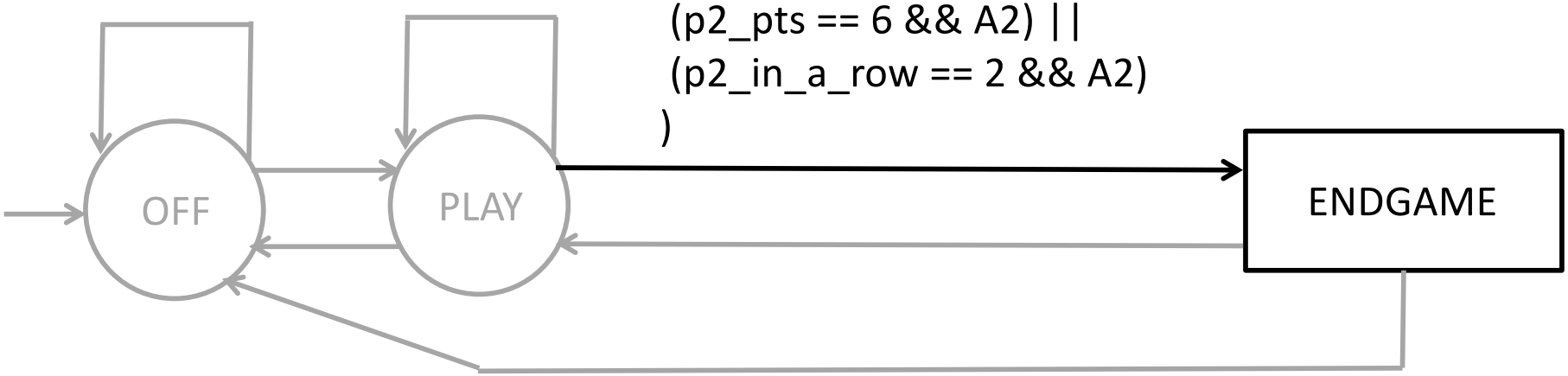
Simultaneously Detecting a Score and a Win

- Win by Scoring 7 points
  - Must detect: 6 points plus a new score
  - Condition: p1_pts == 6 && A1
- Win by Scoring 3 points
  - Must detect: 2 points-in-a-row plus a new score
  - Condition: p1_in_a_row == 2 && A1
- Similar story for Player #2

Simultaneously Detecting a Score and a Win (Sketch)

- This is a mess! Horrible for an exam!

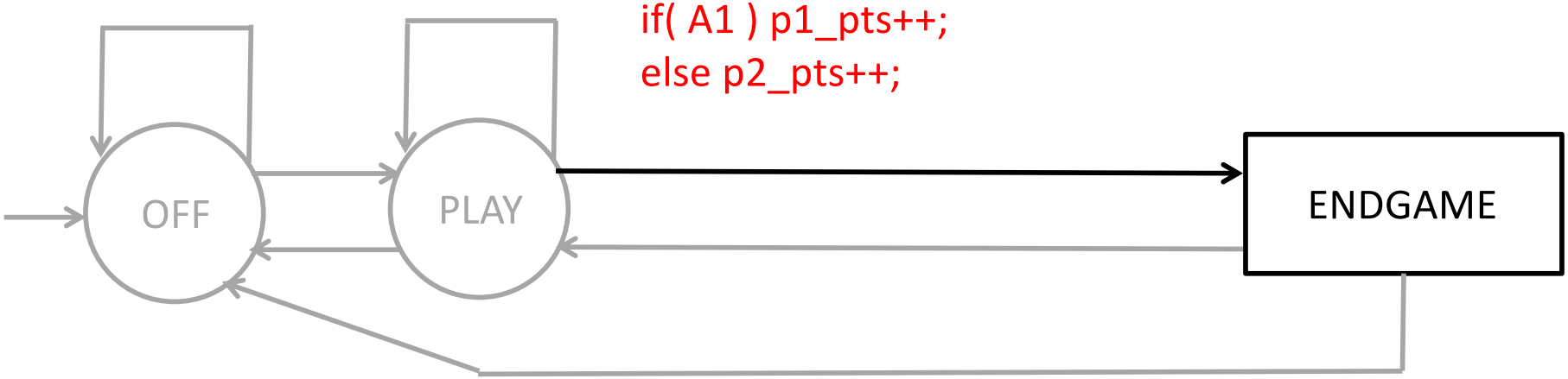
```
A0 &&
(
  // Player #1 Win Criteria
 (p1_pts == 6 && A1) ||
 (p1_in_a_row == 2 && A1)
 ||
 // Player #2 Win Criteria
 (p2_pts == 6 && A2) ||
 (p2_in_a_row == 2 && A2)
)
```



Simultaneously Detecting a Score and a Win (Sketch)

- Must update winner's score for Endgame display

```
A0 &&
(
 (p1_pts == 6 && A1) ||
 (p1_in_a_row == 2 && A1)
 ||
 (p2_pts == 6 && A2) ||
 (p2_in_a_row == 2 && A2)
)
\
if( A1 ) p1_pts++;
else p2_pts++;
```



Simultaneously Detecting a Score and a Win (Sketch)

- What if there is a score, but no winner?
  - Similar condition, but negate victory detection

```
A0 &&
!((p1_pts == 6 && A1) ||
(p1_in_a_row == 2 && A1)
||
(p2_pts == 6 && A2) ||
(p2_in_a_row == 2 && A2))
```

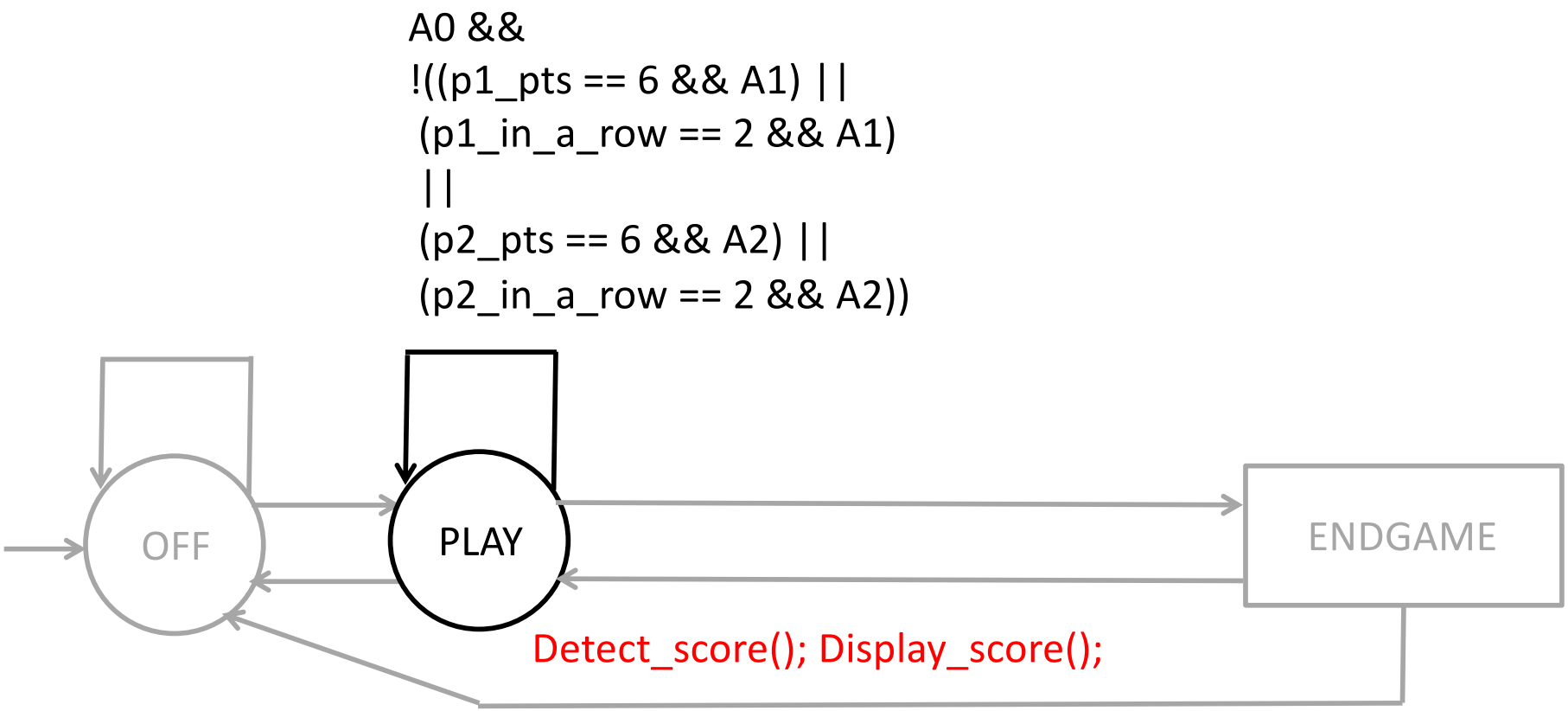
Simultaneously Detecting a Score and a Win (Sketch)

- Don't forget to detect the score and update the game state!
  - Option #1: Transition Action

```
A0 &&
!((p1_pts == 6 && A1) ||
 (p1_in_a_row == 2 && A1)
 ||
 (p2_pts == 6 && A2) ||
 (p2_in_a_row == 2 && A2))
\
Detect_score(); Display_score();
```

Simultaneously Detecting a Score and a Win (Sketch)

- Don't forget to detect the score and update the game state!
  - Option #2: State Action



Why Are the Hints Important?

- **Hint #2**: You can assume that the values on the inputs (A0, A1, A2) never change during a **tick()**. In other words, if you read A0 multiple times during a **tick()**, you will always read the same value. (The value of A0 may change prior to the **next tick()**.)

```
A0 &&
!((p1_pts == 6 && A1) ||
 (p1_in_a_row == 2 && A1)
 ||
 (p2_pts == 6 && A2) ||
 (p2_in_a_row == 2 && A2))
\
Detect_score();
```

You could optimize the logic to read A1 once

You could optimize the logic to read A2 once

You need to read A1 and A2 again

It is not possible to update the _pts and _in_a_row variables in the condition

Why Are the Hints Important?

- **Hint #1**: It is OK to transition into the **endgame** one tick **after** detecting that a player has scored a point
- **Hint #2**: You can assume that the values on the inputs (A0, A1, A2) never change during a **tick()**. In other words, if you read A0 multiple times during a **tick()**, you will always read the same value. (The value of A0 may change prior to the **next tick()**.)

- **Summary**
  - **Hint #1** leads to the initial set of solutions. In this case, you don't need Hint #2.
  - **Hint #2** is needed in case you didn't follow **Hint #1**