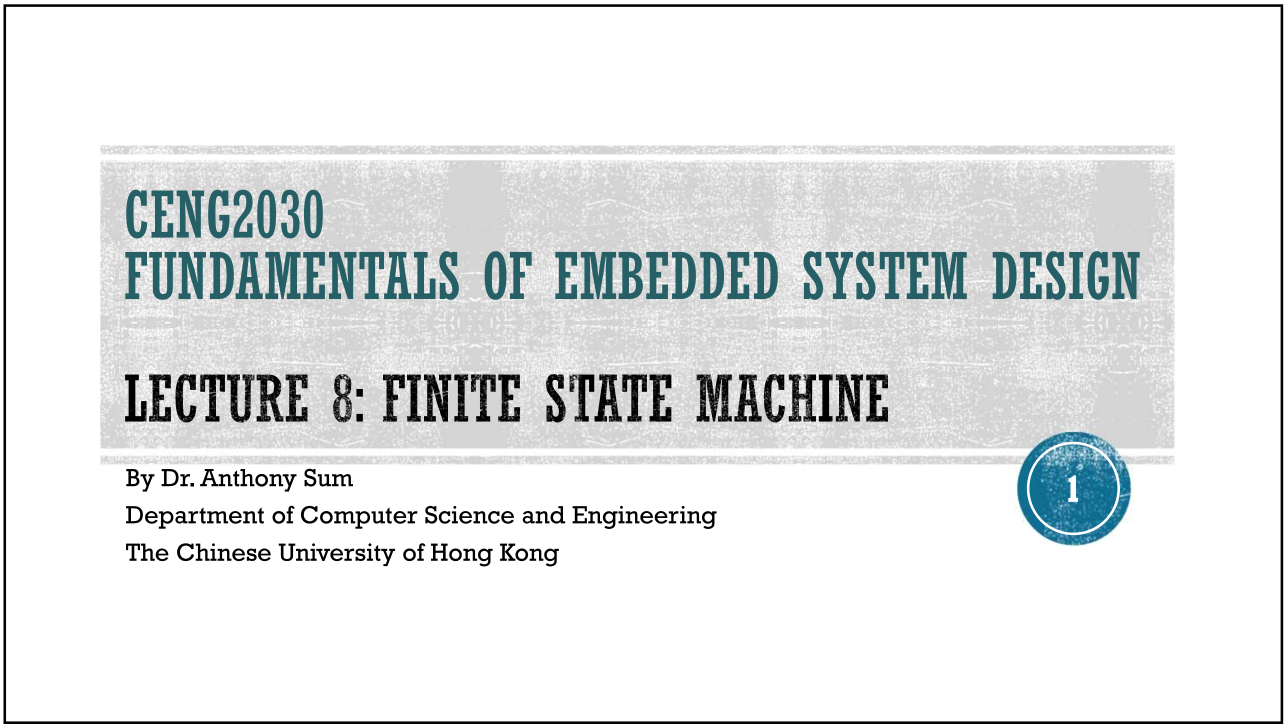
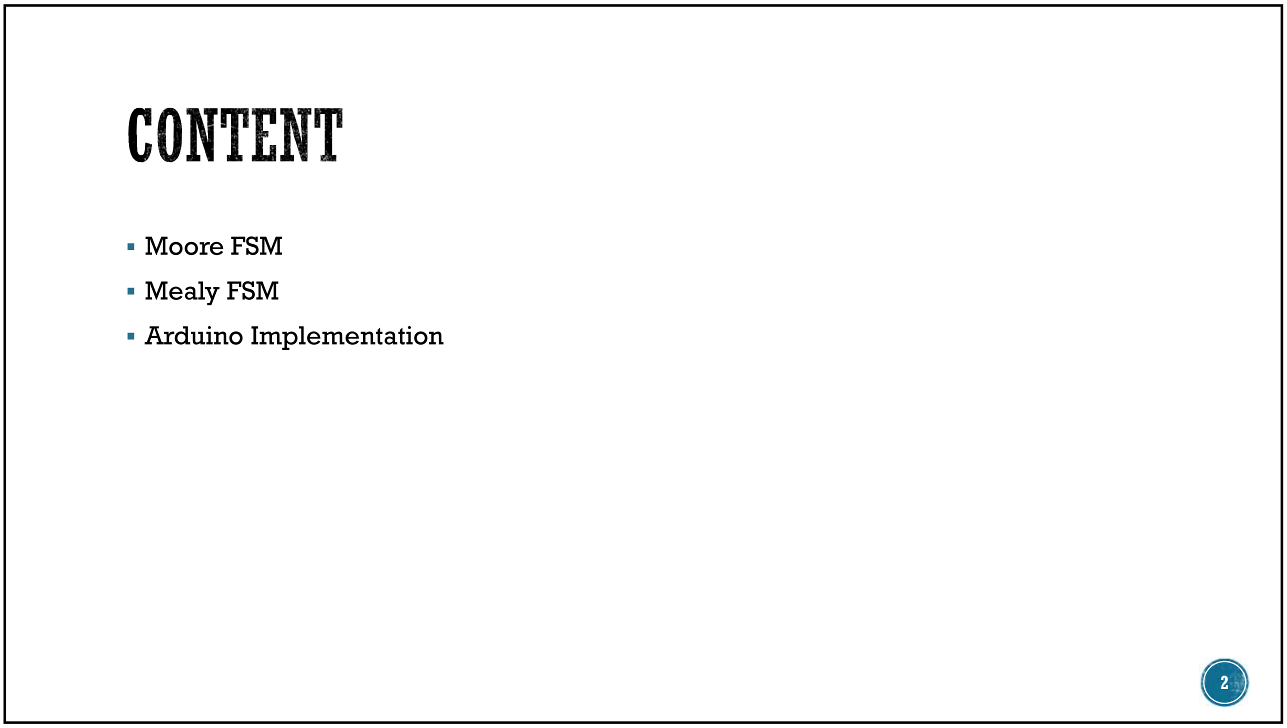


# CENG2030 FUNDAMENTALS OF EMBEDDED SYSTEM DESIGN

## LECTURE 8: FINITE STATE MACHINE

By Dr. Anthony Sum
Department of Computer Science and Engineering
The Chinese University of Hong Kong



## CONTENT

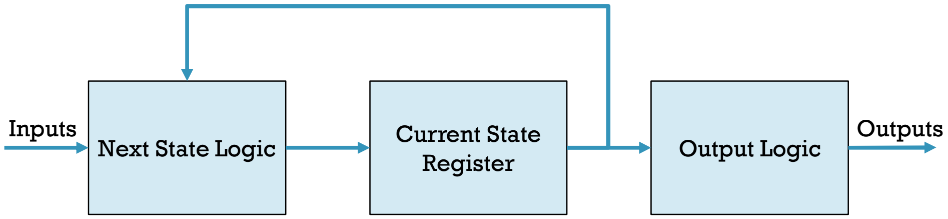
- Moore FSM
- Mealy FSM
- Arduino Implementation

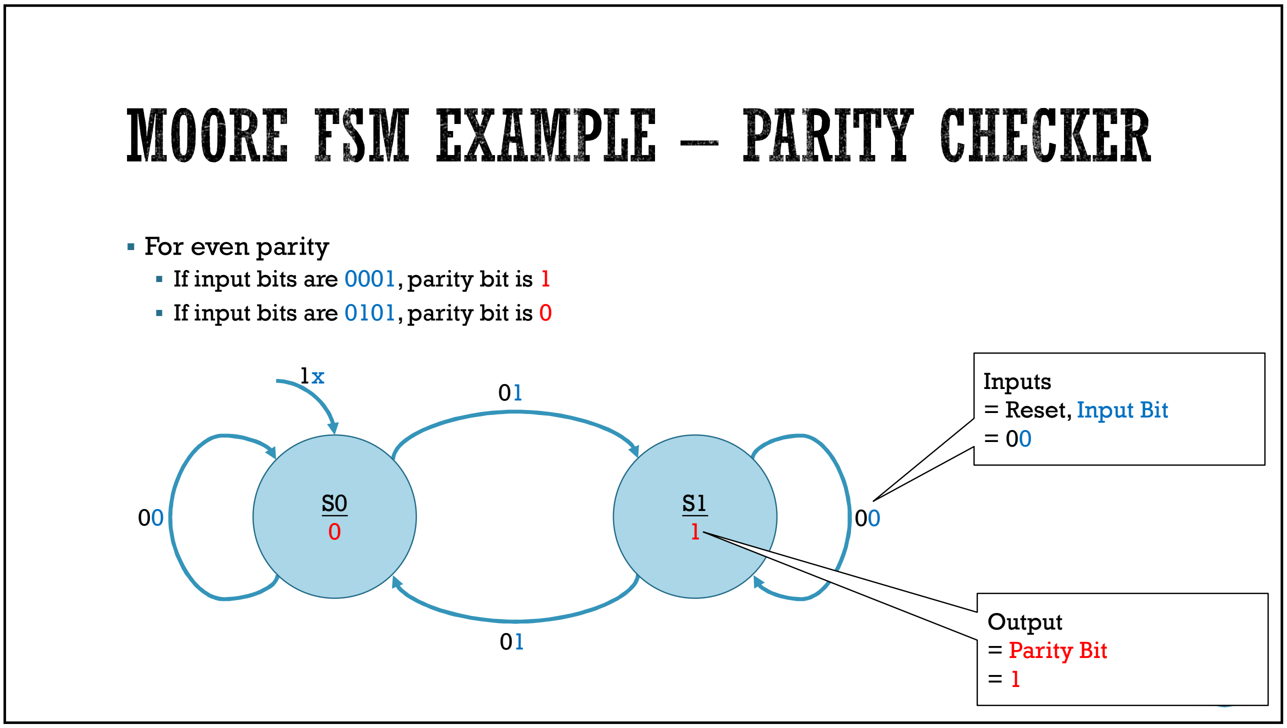
# FINITE STATE MACHINE (FSM)

- FSMs are sequential logic used in many digital systems to control the behavior of systems and dataflow paths.
- The machine is in only one state at a time.
- It can change from one state to another when initiated by a triggering event or condition; this is called a transition
- There are two types of FSMs
- Moore FSM
- Mealy FSM

# MOORE FSM

- The outputs depend on the current state only

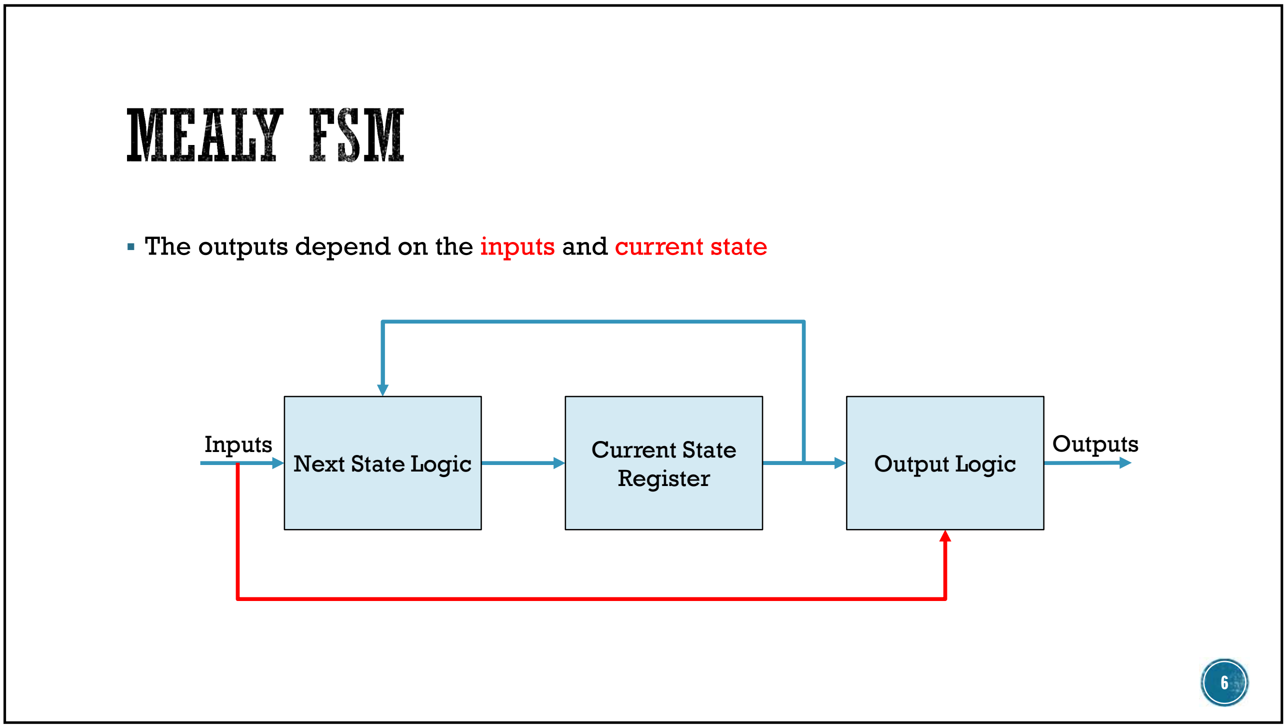




# MOORE FSM EXAMPLE – PARITY CHECKER

- For even parity
  - If input bits are 0001, parity bit is 1
  - If input bits are 0101, parity bit is 0

5



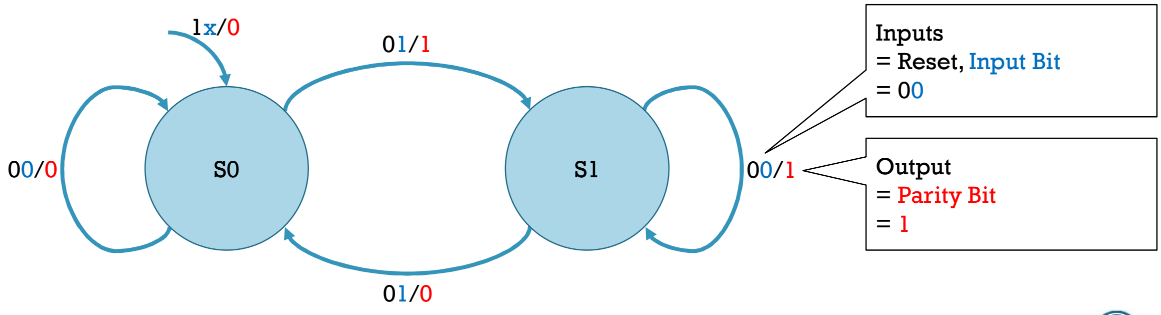
# MEALY FSM

- The outputs depend on the inputs and current state

6

# MEALY FSM EXAMPLE – PARITY CHECKER

- For even parity
  - If input bits are 0001, parity bit is 1
  - If input bits are 0101, parity bit is 0



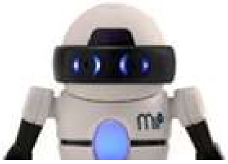
Inputs
= Reset, Input Bit
= 00

Output
= Parity Bit
= 1

# FSM EXAMPLE – ROBOT

- Suppose we have a robot that can:
  - Move Forward
  - Rotate Left
  - Rotate Right
  - Stop

# FSM EXAMPLE – STATE TABLE



- Based on two IR sensors in the front, the robot move accordingly
- The state table below shows the "Next State/Output" in different situations

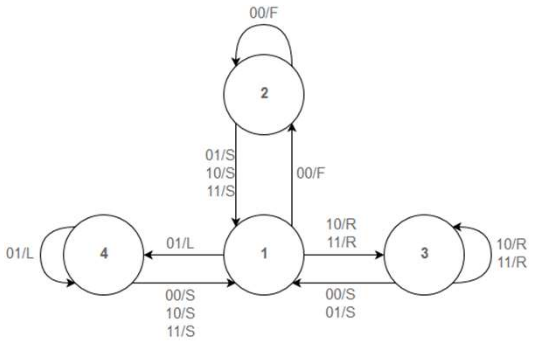
| | IR Detections | | | |
|---|---|---|---|---|
| State | 00 | 01 | 10 | 11 |
| 1: Stop (S) | 2/F | 4/L | 3/R | 3/R |
| 2: Forward (F) | (2/F) | 1/S | 1/S | 1/S |
| 3: Rotate Right (R) | 1/S | 1/S | (3/R) | (3/R) |
| 4: Rotate Left (L) | 1/S | (4/L) | 1/S | 1/S |

1 for obstacle,
0 for free space

( ) for staying in the same state

Next State / Output

# FSM EXAMPLE – MEALY MACHINE



# FSM EXAMPLE – ARDUINO

```
enum State_enum {STOP, FORWARD, ROTATE_RIGHT, ROTATE_LEFT};
enum Sensors_enum {NONE, SENSOR_RIGHT, SENSOR_LEFT, BOTH};

void state_machine_run(uint8_t sensors);
void motors_stop();
void motors_forward();
void motors_right();
void motors_left();

uint8_t read_IR();
uint8_t state = STOP;

void setup(){
}

void loop(){
  state_machine_run(read_IR());
  delay(10);
}
```

```
void motors_stop()
{
  //code for stopping motors
}

void motors_forward()
{
  //code for driving forward
}

void motors_right()
{
  //code for turning right
}

void motors_left()
{
  //code for turning left
}

uint8_t read_IR()
{
  //code for reading both sensors
}
```

```
void state_machine_run(uint8_t sensors)
{
 switch(state)
 {
  case STOP:
   motors_stop();
   if(sensors == NONE){
    motors_forward();
    state = FORWARD;
   }
   else if(sensors == SENSOR_RIGHT){
    motors_left();
    state = ROTATE_LEFT;
   }
   else{
    motors_right();
    state = ROTATE_RIGHT;
   }
   break;

  case FORWARD:
   motors_forward();
   if(sensors != NONE){
    motors_stop();
    state = STOP;
   }
   break;

  case ROTATE_RIGHT:
   motors_right();
   if(sensors == NONE || sensors == SENSOR_RIGHT){
    motors_stop();
    state = STOP;
   }
   break;

  case ROTATE_LEFT:
   motors_left();
   if(sensors != SENSOR_RIGHT)
   {
    motors_stop();
    state = STOP;
   }
   break;
 }
}
```

# ANY QUESTIONS ?