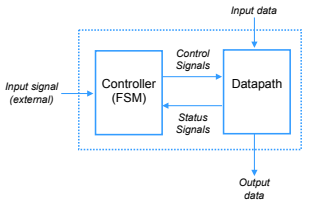
ECE 474A/57A
Computer-Aided Logic Design

Lecture 6
Algorithmic State Machines (ASMs)

Control and Datapath Interaction

- Binary information in digital system can be classified into two categories
- Data
  - Discrete elements of information manipulated by arithmetic, logic, shift, and other data processing
  - Operations implemented via digital components such as adders, decoders, muxes, etc.
- Control
  - Provides command signals that coordinate the execution of various operations in data section to accomplish desired task



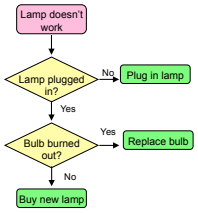
What Control Path Implements?

- Sequencing of control signals to execute algorithm implemented by circuit
- Algorithm
  - Finite set of instructions/steps to solve a problem
  - Terminates in finite time at a known end state
- Many representations

Ingredients
- 1/3 cup unsweetened cocoa
- 1/4 cup cornstarch
- 2 tablespoons butter
- 2 2/3 cups skim milk

Steps
1. Combine all ingredients in a small saucepan.
2. Heat over low heat, stirring constantly, until mixture boils. Boil gently, stirring constantly, for one minute.
3. Pour into serving dishes and chill until thickened.

Recipe



Flowchart

```
int fib(int n)
{
  if (n < 2)
    return n;
  else
    return fib(n-1) + fib(n-2);
}
```

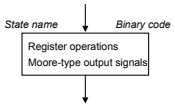
Computer Program

Flowcharts and Algorithmic State Machines (ASM)

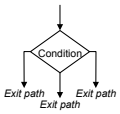
- Flowchart
  - Convenient way to graphically specify sequence of procedural steps and decision paths for algorithm
  - Enumerates sequence of operations and conditions necessary for execution
- Algorithmic State Machine (ASM)
  - Flowchart defined specifically for digital hardware algorithms
- Flowchart vs. ASM
  - Conventional flowchart
    - Sequential way of representing procedural steps and decision paths for algorithm
    - No time relations incorporated
  - ASM chart
    - Representation of sequence of events together with timing relations between states of sequential controller and events occurring while moving between steps

ASM Chart

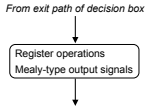
- Three basic elements
  - State box
  - Decision box
  - Conditional box
- State and decision boxes used in conventional flowcharts
  - Conditional box characteristic to ASM



State Box



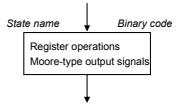
Decision Box

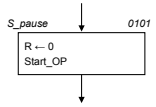


Conditional Box

State box

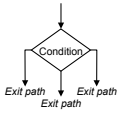
- Used to indicate states in control sequence
  - State name and binary code placed on top of box
  - Register operations and names of output signals generated in state placed inside box
- Example
  - State name: S_pause
  - Binary encoding: 0101
  - Register operation: R ← 0
    - Register R is to be cleared to 0
  - Output signal asserted: Start_OP = 1
    - Launches some operation in datapath

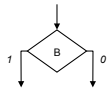




Decision Box

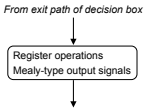
- Reflects the effect of an input
  - external or internal, input or status
- Diamond shaped box
  - Condition to be tested inside
  - Two or more outputs represent exit paths dependant on value tested
    - In binary case one path represents true the other false, represented by 1 and 0 respectively
- Example
  - Check B
    - If B is true (=1), take path marked 1
    - If B is false (=0), take path marked 0

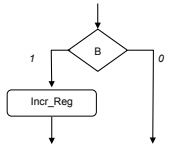




Conditional Box

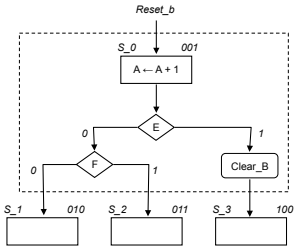
- Unique to ASM
- Inputs come from one of exit paths of decision boxes
- Register operation or outputs listed inside box generated during given state
  - Generated as Mealy-type signals
  - Associated with the state transition
- Example
  - Status of input B checked
  - Conditional operation executed depending on result coming from decision box
    - If B = 1, assert Incr_Reg signal
    - Otherwise Incr_Reg remains unchanged





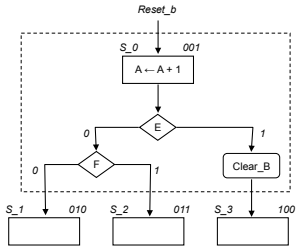
ASM Block

- Structure consisting of
  - One state box
  - All decision and conditional boxes associated with its exit paths
- Block has one entrance and any number of exits paths
- Each block in ASM dedicated to state of system during one clock cycle
- Simplifications
  - ASM Block not usually drawn because blocks are well defined
  - Can label just the "1" and omit the "0"
- ASM chart consists of one or more interconnect ASM Blocks



Interpretation of Timing Operations

- Conventional flowchart, evaluation of each follows one another
  - Reg A incremented
  - Condition E evaluated
    - If E= 1
      - clear B
      - Go to state S_3
- In ASM the entire block considered as one unit
  - All operations within block occurring during single edge transition
  - The next state evaluated during the same clock
  - System enters next state S_1, S_2, or S_3 during transition of next clock

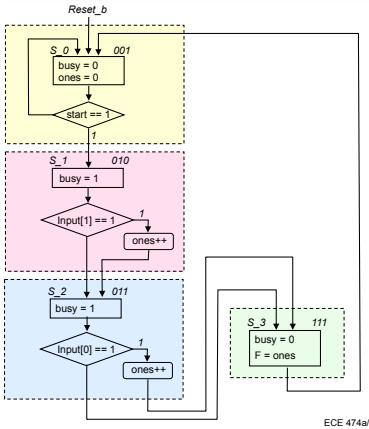


ASM Example

- Convert pseudo code to ASM chart
- Example
  - Want to detect the number of 1's in a 2-bit register called *Input*
  - *start* input indicates when to begin comparison
  - *busy* output indicates when comparison in progress
  - *ones* hold count value
  - *F* outputs result

```
S0:
   busy = 0;
   ones = 0;

   if(start == 1)
     goto S1
   else
     goto S0
S1:
   busy = 1;
   if(Input[1] == 1)
      ones ++;

   goto S2
S2:
   busy = 1;
   if(Input[0] == 1)
      ones ++;

   goto S3
S3:
   busy = 0;
   F = ones;

   goto S0
```

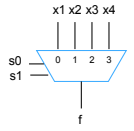

ASM Example Continued




```
S0:
   busy = 0;
   ones = 0;

   if(start == 1)
     goto S1
   else
     goto S0
S1:
   busy = 1;
   if(Input[1] == 1)
      ones ++;

   goto S2
S2:
   busy = 1;
   if(Input[0] == 1)
      ones ++;

   goto S3
S3:
   busy = 0;
   F = ones;

   goto S0
```

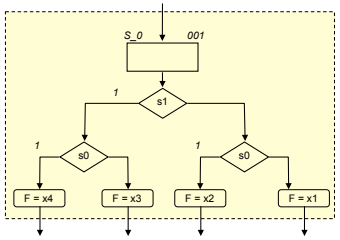
ASM – Mux

- Describe a 4x1 MUX using a ASM



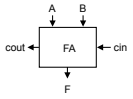
s1	s0	f
0	0	x1
0	1	x2
1	0	x3
1	1	x4

4x1 mux

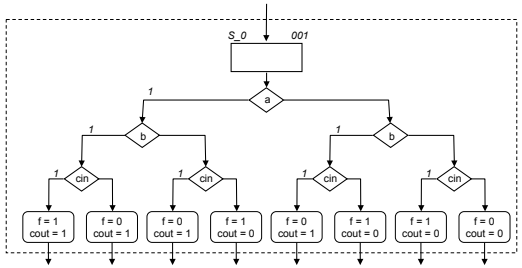


ASM – Full Adder

- Describe a 1-bit full adder using an ASM chart

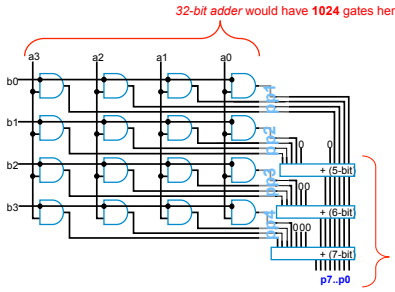


a	b	cin	f	cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Smaller Multiplier

- Multiplier in array style
  - Fast, reasonable size for 4-bit: 4*4 = 16 partial product AND terms, 3 adders
  - Rather big for 32-bit: 32*32 = 1024 AND terms, and 31 adders




```
                        a3   a2   a1   a0
                      x b3   b2   b1   b0
-------------------------------------------
                     b0a3 b0a2 b0a1 b0a0   (pp1)
                b1a3 b1a2 b1a1 b1a0   0    (pp2)
           b2a3 b2a2 b2a1 b2a0   0    0    (pp3)
+ b3a3 b3a2 b3a1 b3a0    0    0    0    (pp4)
-------------------------------------------
p7 p6   p5   p4   p3   p2   p1   p0
```

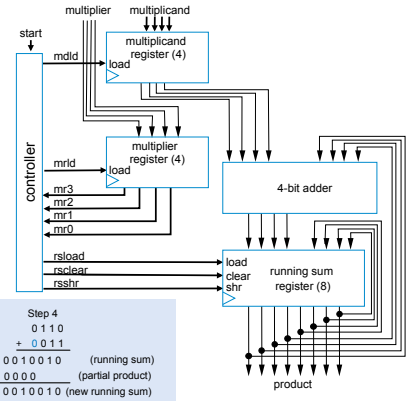

Smaller Multiplier -- Sequential (Add-and-Shift) Style

- Smaller multiplier: Basic idea
  - Don't compute all partial products simultaneously
  - Rather, compute one at a time (similar to by hand), maintain running sum

	Step 1	Step 2	Step 3	Step 4
	0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0
	+ 0 0 1 1	+ 0 0 1 1	+ 0 0 1 1	+ 0 0 1 1
(running sum)	0 0 0 0	0 0 1 1 0	0 1 0 0 1 0	0 0 1 0 0 1 0
(partial product)	+ 0 1 1 0	+ 0 1 1 0	+ 0 0 0 0	+ 0 0 0 0
(new running sum)	0 0 1 1 0	0 1 0 0 1 0	0 0 1 0 0 1 0	0 0 0 1 0 0 1 0

Smaller Multiplier -- Sequential (Add-and-Shift) Style

- Design circuit that computes one partial product at a time, adds to running sum
  - Note that **shifting** running sum right (relative to partial product) after each step ensures partial product added to correct running sum bits



Step 1	Step 2	Step 3	Step 4	
0 1 1 0	0 1 1 0	0 1 1 0	0 1 1 0	
+ 0 0 1 1	+ 0 0 1 1	+ 0 0 1 1	+ 0 0 1 1	
0 0 0 0	0 0 1 1 0	0 1 0 0 1 0	0 0 1 0 0 1 0	(running sum)
+ 0 1 1 0	+ 0 1 1 0	+ 0 0 0 0	+ 0 0 0 0	(partial product)
0 0 1 1 0	0 1 0 0 1 0	0 0 1 0 0 1 0	0 0 0 1 0 0 1 0	(new running sum)

Smaller Multiplier -- Sequential (Add-and-Shift) Style

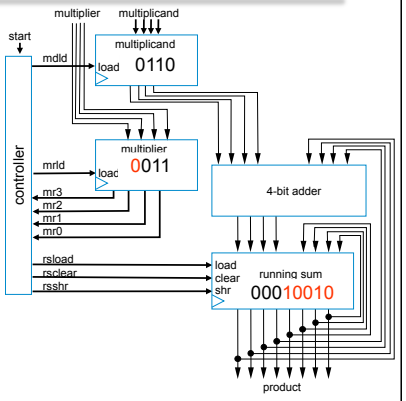
Step 0
- Set running sum to 0
- Load values

Step 1
- Check multiplier bit 0 (mr0)
  - mr0=1, add multiplicand to running sum
- Shift running sum right 1 position

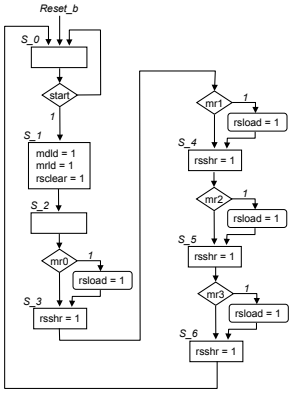
Step 2
- Check multiplier bit 1 (mr1)
  - mr0=1, add multiplicand to running sum
- Shift running sum right 1 position

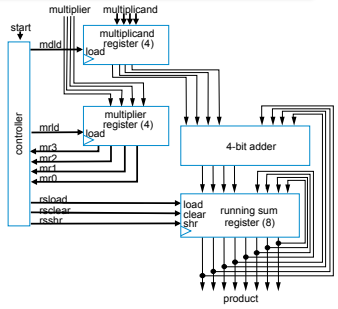
Step 3
- Check multiplier bit 2 (mr2)
- Shift running sum right 1 position

Step 4
- Check multiplier bit 3 (mr3)
- Shift running sum right 1 position



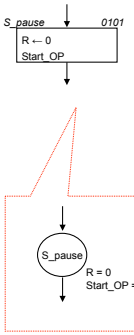
ASM – Sequential Multiplier

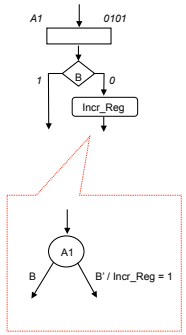


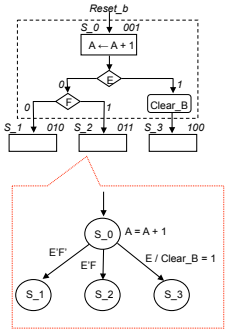


ASMs to FSMDs

- Able to convert between formats
- Once we have a FSMD, we've already seen how to implement in hardware







Not Used Much, But …

- There are commerical ASM Editors
  - Mentor Graphics
  - Summit Design, Inc.
  - Others…