Design of Digital Circuits and Systems
ASM with Datapath I

Instructor: Justin Hsia

Teaching Assistants:

Colton Harris
Deepti Anoop
Gayathri Vadhyan
Jared Yoder
Lancelot Wathieu
Matthew Hung

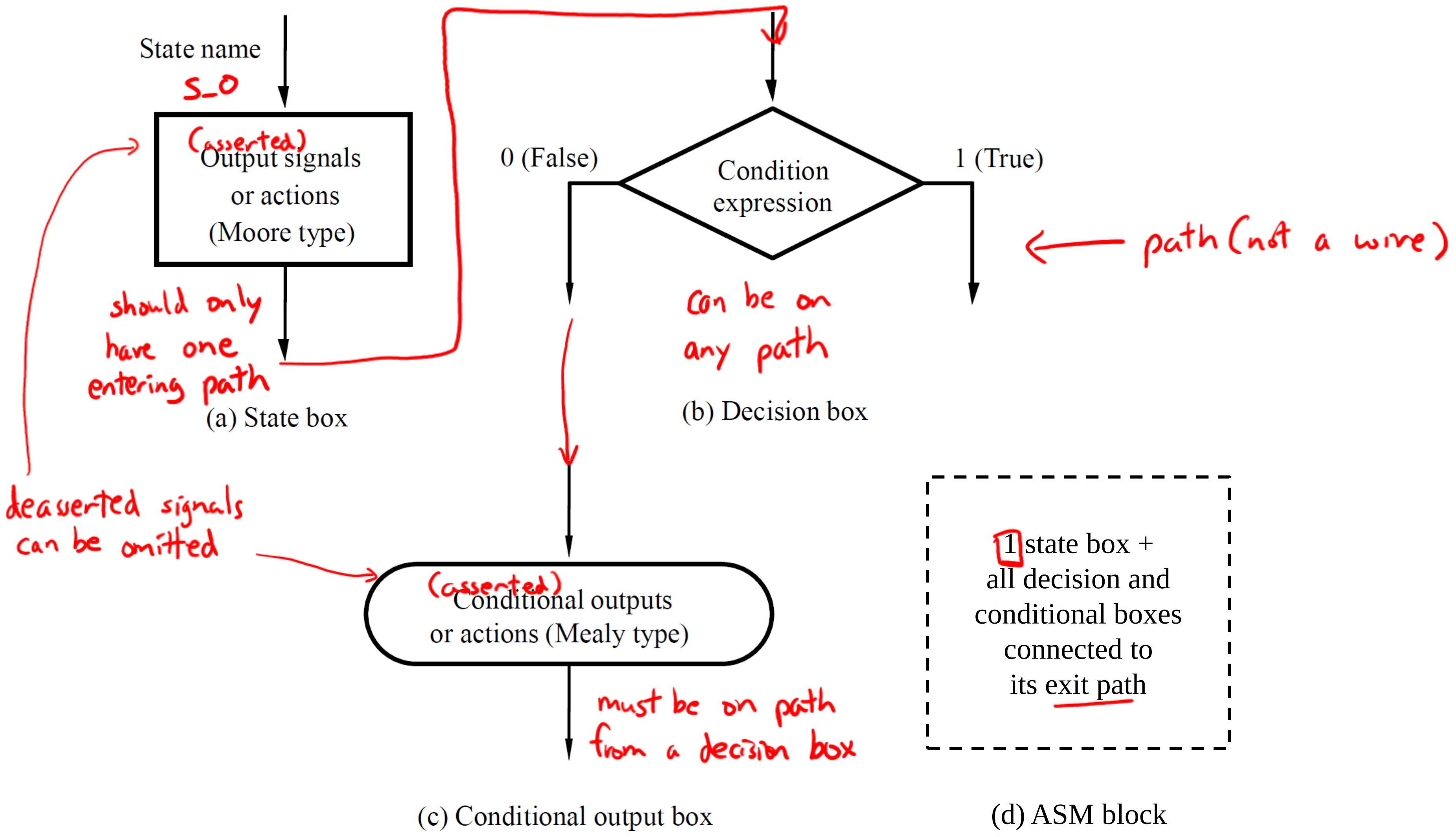
Relevant Course Information

- Homework 2 late deadline tonight (4/11)
- Homework 3 due next Friday (4/19)
  - FIFO buffers & ASM charts

- Lab 2 reports due Friday (4/12), demos 4/15-19
  - Same lab demo slots for whole quarter
- Lab 3 due 4/26
  - Lab 3 + 4 are really ~1.5 weeks long, so don’t wait!

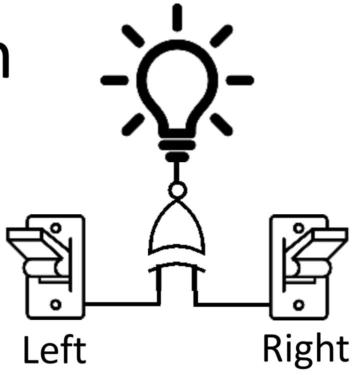
- Quiz 2 next Thursday (4/18)
  - Memory (ROM, RAM, reg files)

Review: ASM Chart


State name
S_0
(asserted) Output signals or actions (Moore type)
should only have one entering path
(a) State box
0 (False)
Condition expression
1 (True)
path (not a wire)
can be on any path
(b) Decision box
deasserted signals can be omitted
(asserted) Conditional outputs or actions (Mealy type)
must be on path from a decision box
(c) Conditional output box
1 state box + all decision and conditional boxes connected to its exit path
(d) ASM block

Review Question: 3-way Switch

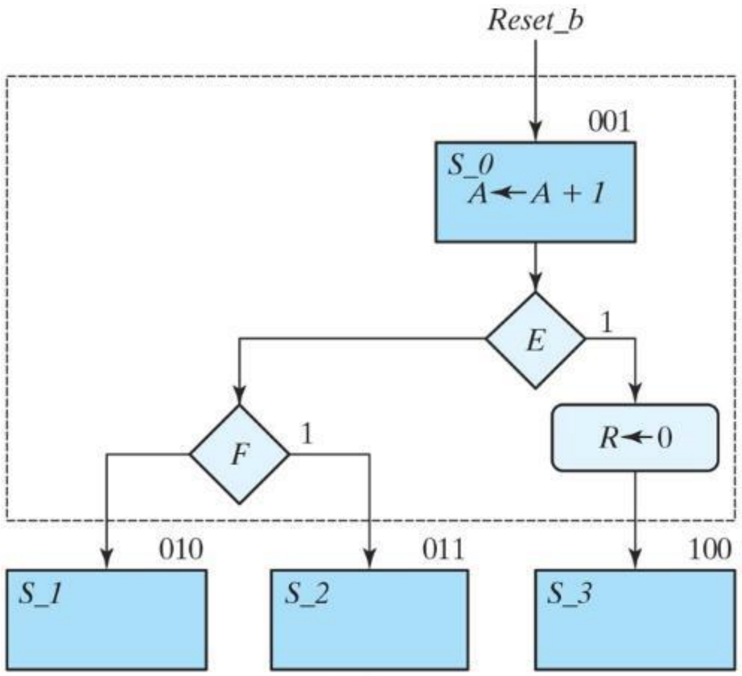
- Create an ASM chart for a 3-way switch system using *Mealy*-type output
  - `LTog` and `RTog` pulse `1` when switch is flipped/toggled, output called `light`



ASMD Charts

- An **A**lgorithmic **S**tate **M**achine with a **D**atapath chart is created by adding RTL operations to an ASM chart
  - Timing of operations can be confusing – *NOT* a flowchart

- School of Thought #1:
  - RTL operations are triggered by control signals, so they can appear anywhere an output signal can:

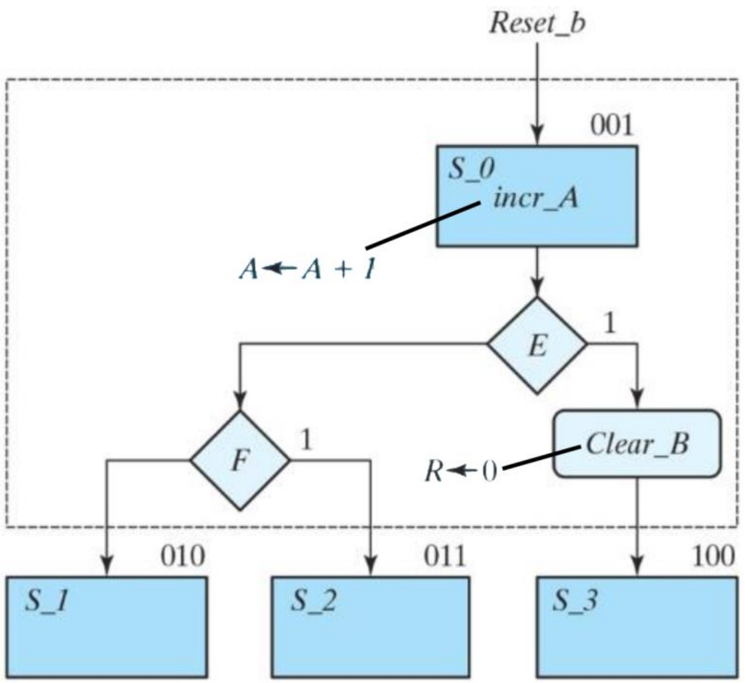


ASMD Charts

- An **A**lgorithmic **S**tate **M**achine with a **D**atapath chart is created by adding RTL operations to an ASM chart
  - Timing of operations can be confusing – *NOT* a flowchart

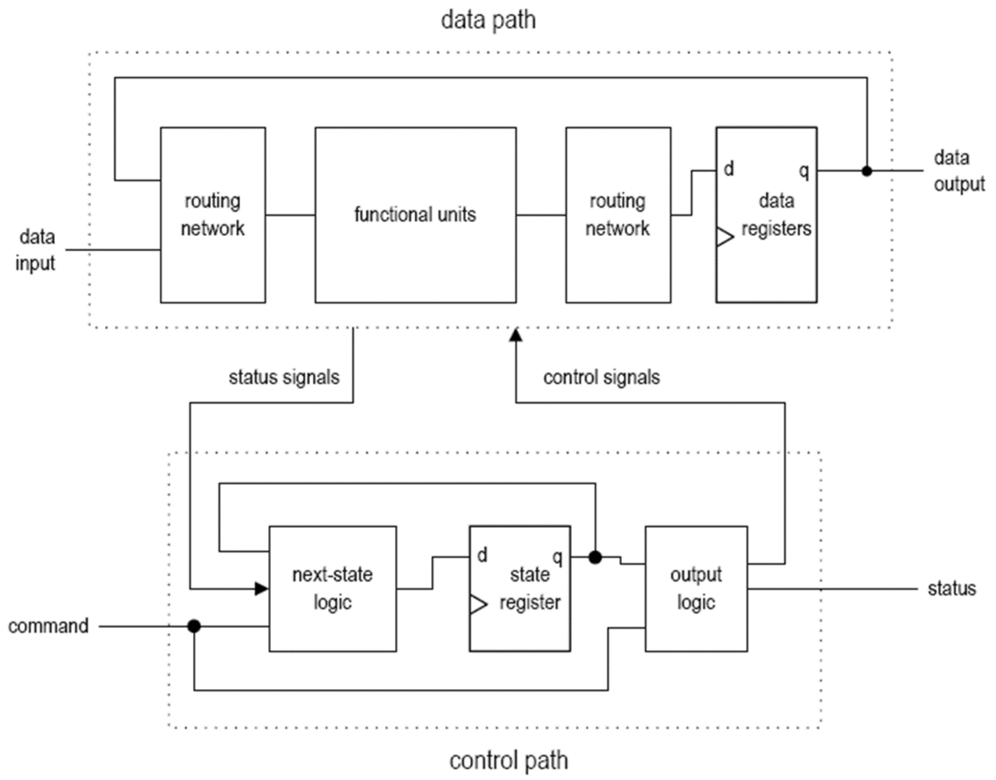
- School of Thought #2:
  - It's clearer to separate control signals (Control) from RTL operations (Datapath)

- There isn't a set standard
  - You may see both or variants
  - ***We use School of Thought #2***



ASMD Hardware

- State transitions and RTL operations are both controlled by the clock
  - It's often helpful to remember the underlying hardware – registers!

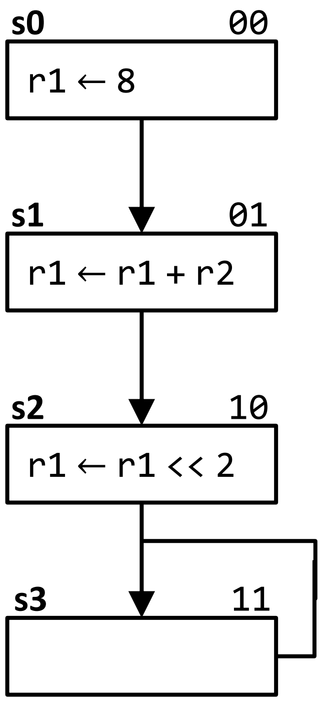


ASMD Hardware

- State transitions and RTL operations are both controlled by the clock
  - It’s often helpful to remember the underlying hardware – registers!
- The behavior of both state and data registers depend on the current control state
  - Can conceptually think of as a MUX to the registers’ inputs that uses the current state as its selector bits

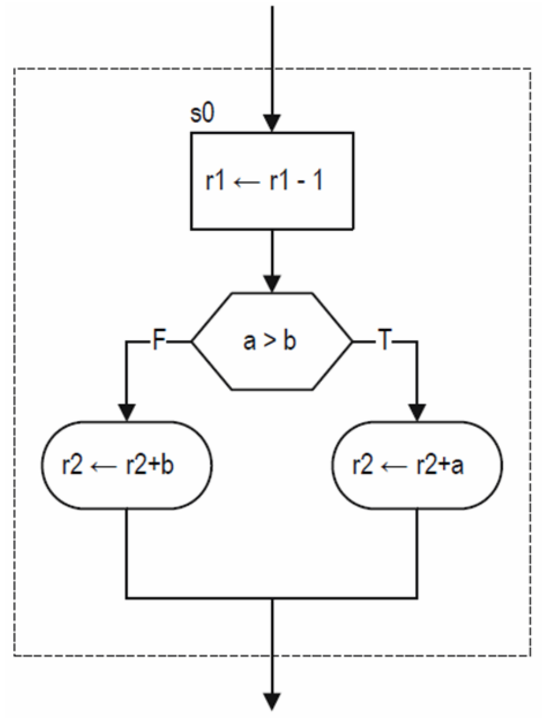
Hardware Example #1

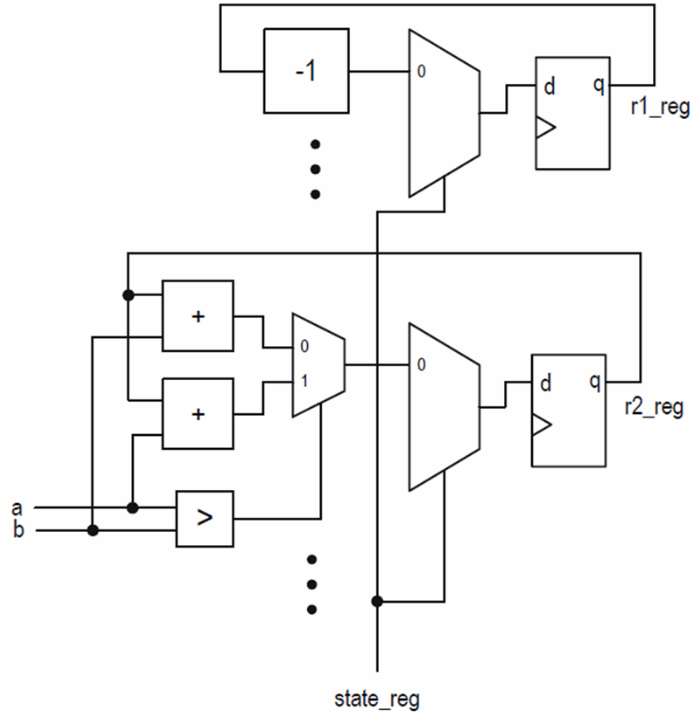
- State transitions and RTL operations are both controlled by the clock
  - It’s often helpful to remember the underlying hardware – registers!



Hardware Example #2

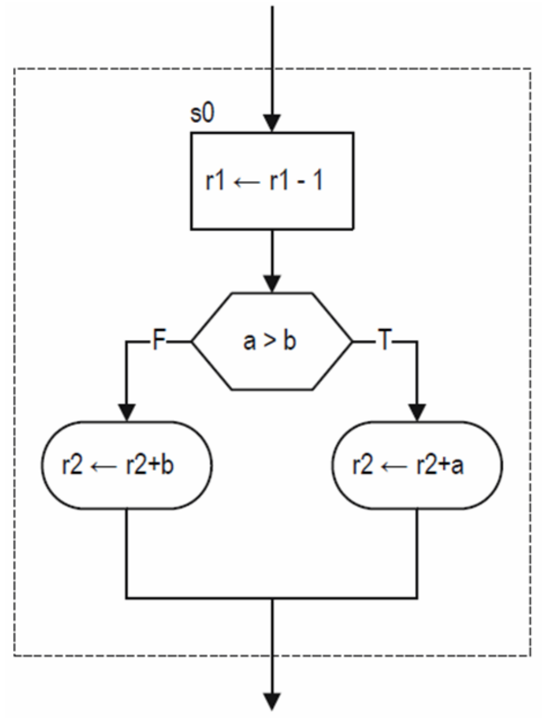
- State transitions and RTL operations are both controlled by the clock
  - It's often helpful to remember the underlying hardware – registers!

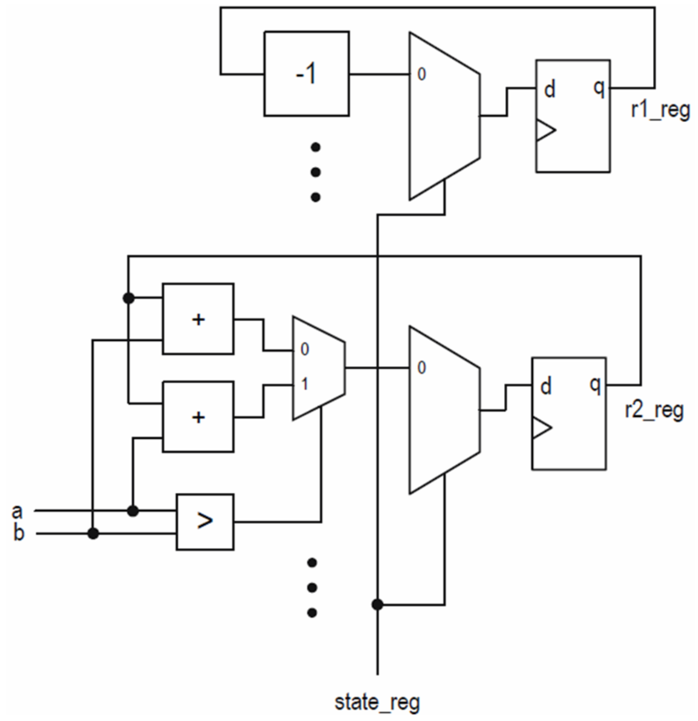




ASMD Timing

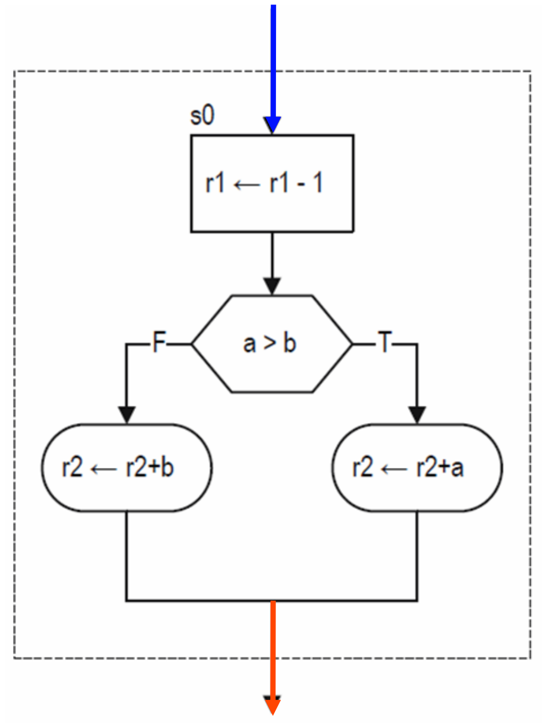
- Everything (registers!) within an ASM block occurs simultaneously at the *next* clock trigger
  - Differs from a flowchart – changes occur at state *exit* rather than *entrance*

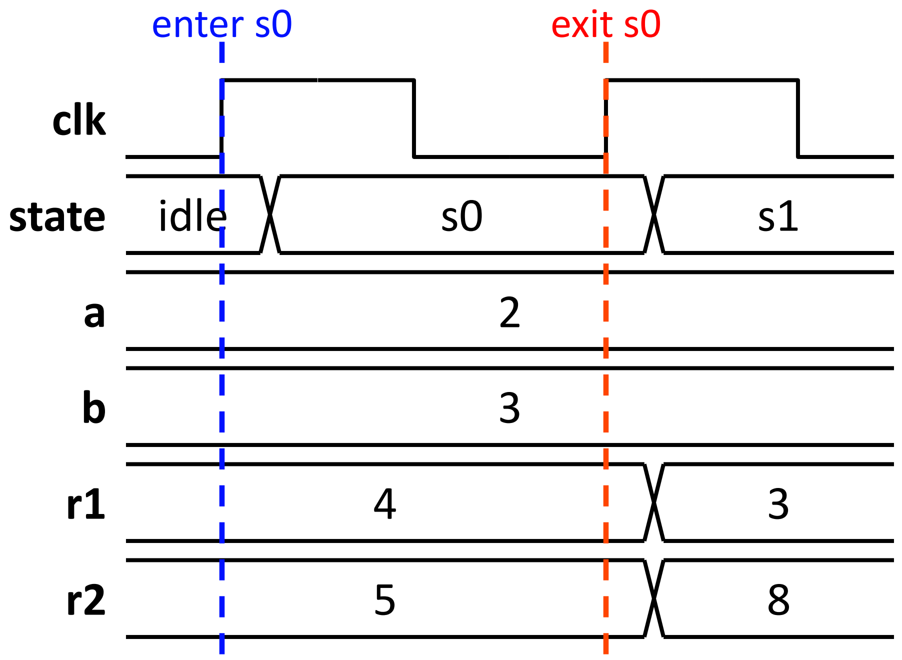




ASMD Timing

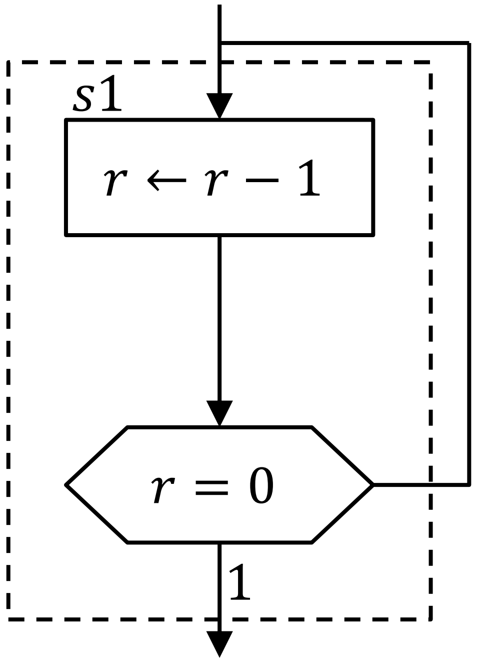
- Everything (registers!) within an ASM block occurs simultaneously at the *next* clock trigger
  - Differs from a flowchart – changes occur at state *exit* rather than *entrance*





ASMD Timing Question

- What value will be stored in r when we transition from state $s1$ to the next state? -1, 0, 1

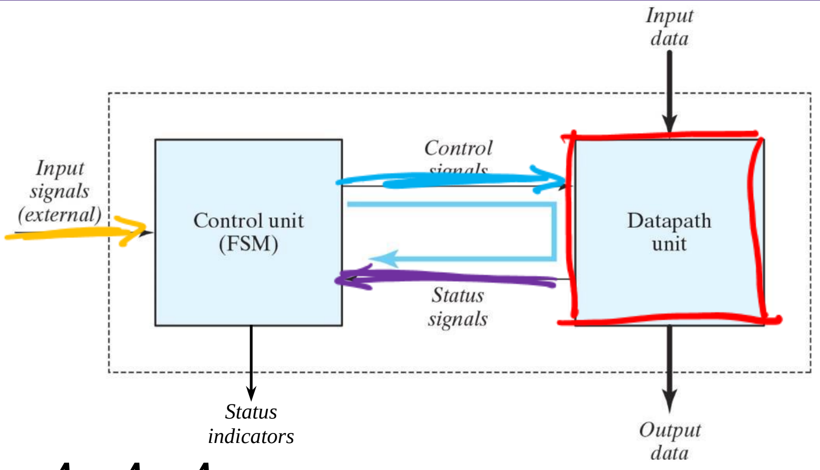


Short Tech Break

ASMD Design Procedure

- From problem description or algorithm pseudocode:
  1) **Identify necessary datapath components and operations**
  2) **Identify states and signals that cause state transitions** (external inputs and status signals), based on the necessary sequencing of operations
  3) **Name the control signals** that are generated by the controller that cause the indicated operations in the datapath unit
  4) **Form an ASM chart for your controller**, using states, decision boxes, and signals determined above
  5) **Add the datapath RTL operations** associated with each control signal

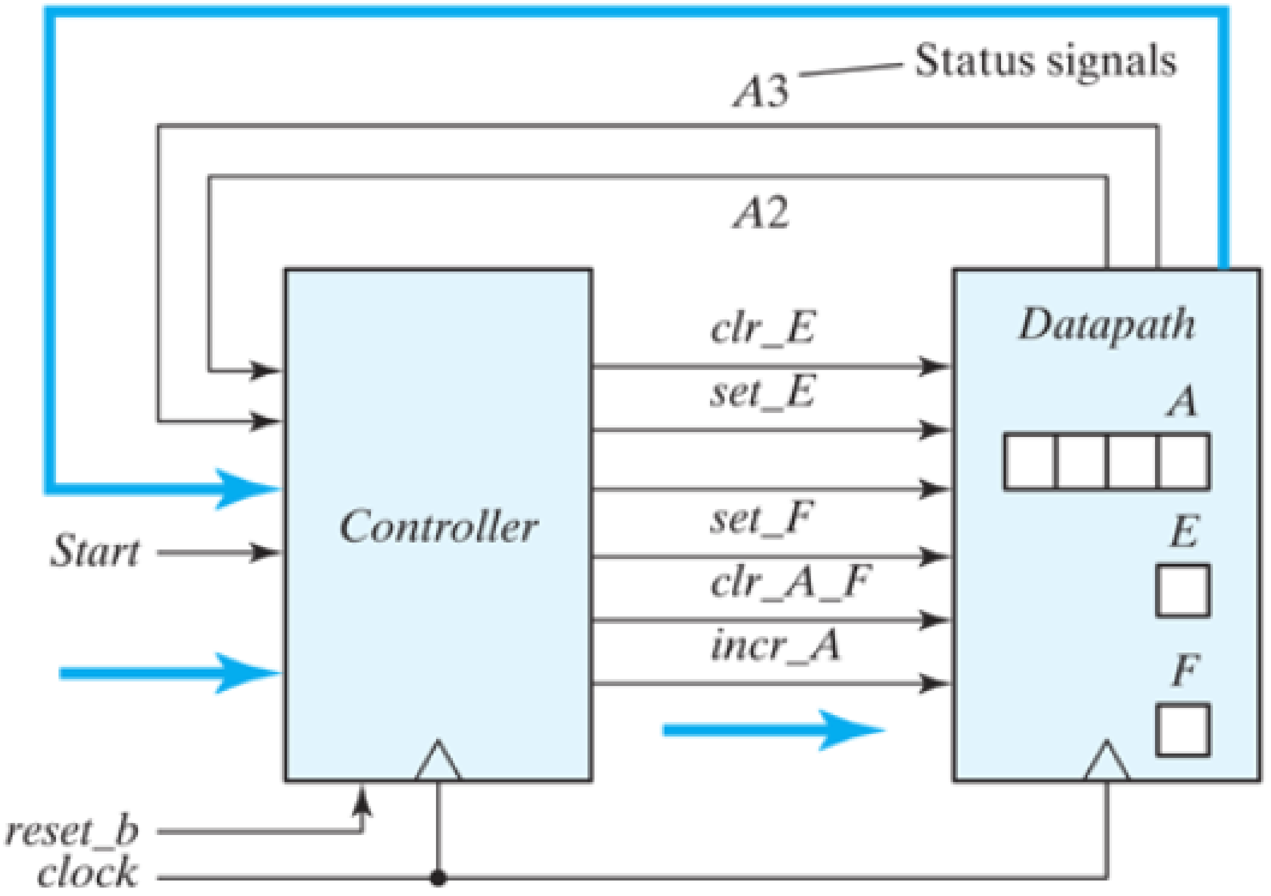
Design Example #1



- System specification:
  - Flip-flops E and F
  - 4-bit binary up-counter $A = 0\mathrm{b}A_3A_2A_1A_0$
  - Active-low reset signal $reset\_b$ puts us in state S_idle, where we remain while signal $Start = 0$
  - $Start = 1$ initiates the system's operation by clearing A and F. At each subsequent clock pulse, the counter is incremented by 1 until the operations stop.
  - Bits A_2 and A_3 determine the sequence of operations:
    - If $A_2 = 0$, set E to 0 and the count continues
    - If $A_2 = 1$, set E to 1; additionally, if $A_3 = 0$, the count continues, otherwise, wait one clock pulse to set F to 1 and stop counting (*i.e.*, back to S_idle)

Design Example #1

- The system can be represented by the following block diagram:



Design Example #1 (ASM → ASMD Chart)

- Synchronous or asynchronous reset?

Design Example #1 (Timing)

- Sequence of operations:

Counter				Flip-Flops			
A_3	A_2	A_1	A_0	E	F	**Conditions**	**State**
X	X	X	X	1	X	*Start*	*S_idle*
0	0	0	0	1	0	$A_2 = 0, A_3 = 0$	*S_count*
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
0	1	0	0	0	0	$A_2 = 1, A_3 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
1	0	0	0	1	0	$A_2 = 0, A_3 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
1	1	0	0	0	0	$A_2 = 1, A_3 = 1$	
1	1	0	1	1	0		*S_F*
1	1	0	1	1	1		*S_idle*

Design Example #1 (Logic)

- Controller:
  - State Table:

	Present State		Inputs			Next State		Outputs				
Present-State Symbol	P_1	P_0	*Start*	A_2	A_3	N_1	N_0	*set_E*	*clr_E*	*set_F*	*clr_A_F*	*incr_A*
S_idle	0	0	0	X	X	0	0	0	0	0	0	0
S_idle	0	0	1	X	X	0	1	0	0	0	1	0
S_count	0	1	X	0	X	0	1	0	1	0	0	1
S_count	0	1	X	1	0	0	1	1	0	0	0	1
S_count	0	1	X	1	1	1	1	1	0	0	0	1
S_F	1	1	X	X	X	0	0	0	0	1	0	0

  - Logic:

$N_1 =$

$N_0 =$

$set\_E =$

$clr\_E =$

$set\_F =$

$clr\_A\_F =$

$incr\_A =$

Short Tech Break

Design Example #1 (SV, Controller)

```
module controller (set_E, clr_E, set_F, clr_A_F,
                   incr_A, A2, A3, Start, clk,
                   reset_b);

   // port definitions
   input  logic Start, clk, reset_b, A2, A3;
   output logic set_E, clr_E, set_F, clr_A_F, incr_A;

   // define state names and variables


   // controller logic w/synchronous reset
```

Design Example #1 (SV, Controller)

```
    // next state logic









    // output assignments









endmodule  // controller
```

Design Example #1 (SV, Datapath)

```
module datapath (A, E, F, clk, set_E, clr_E, set_F, clr_A_F,
                 incr_A);

   // port definitions
   output logic [3:0] A;
   output logic E, F;
   input  logic clk, set_E, clr_E, set_F, clr_A_F, incr_A;

   // datapath logic










endmodule  // datapath
```

Design Example #1 (SV, Top-Level Design)

```
module top_level (A, E, F, clk, Start, reset_b);

   // port definitions
   output logic [3:0] A;
   output logic E, F;
   input  logic clk, Start, reset_b;

   // internal signals
   logic set_E, clr_E, set_F, clr_A_F, incr_A;

   // instantiate controller and datapath
   controller c_unit (.set_E, .clr_E, .set_F,
                      .clr_A_F, .incr_A, .A2(A[2]),
                      .A3(A[3]), .Start, .clk,
                      .reset_b);
   datapath d_unit (.*);

endmodule  // top_level
```