UNIVERSITY OF LIVERPOOL

Digital Systems Design ELEC373/473

ASMD charts

(Algorithmic State Machine and Datapath)

Prof J.S. Smith
Room A515;
E-mail: j.s.smith@liv.ac.uk

ASMD charts (Algorithmic State Machine and Datapath)

- ASM charts can be used to control register operations on a datapath in a sequential machine that has been partitioned into a controller and a datapath.
- A modified ASM chart is linked to the datapath by annotating each of its paths to indicate the concurrent register operations that occur in the associated datapath unit.
- ASM charts that have been linked to the datapath in this manner are called Algorithmic State Machine and Datapath (ASMD) charts.
- ASMD charts separates the design of a datapath from the design of its controller.
- An ASMD is different from an ASM in that each of the transition paths of an ASM is annotated with the associated concurrent ***register*** operations of the datapath.

ASMD Chart

- Differences between Algorithmic State Machine and Datapath (ASMD) charts & ASM charts.
  - An ASMD chart does not list register operations within a state box.
  - The edges of an ASMD charts are annotated with register operations that are concurrent with the state transition indicated by the edge.
  - An ASMD chart includes conditional boxes identifying the signals which control the register operations that annotate the edges of the chart.
  - An ASMD chart associates register operations with state transitions rather than with the state.

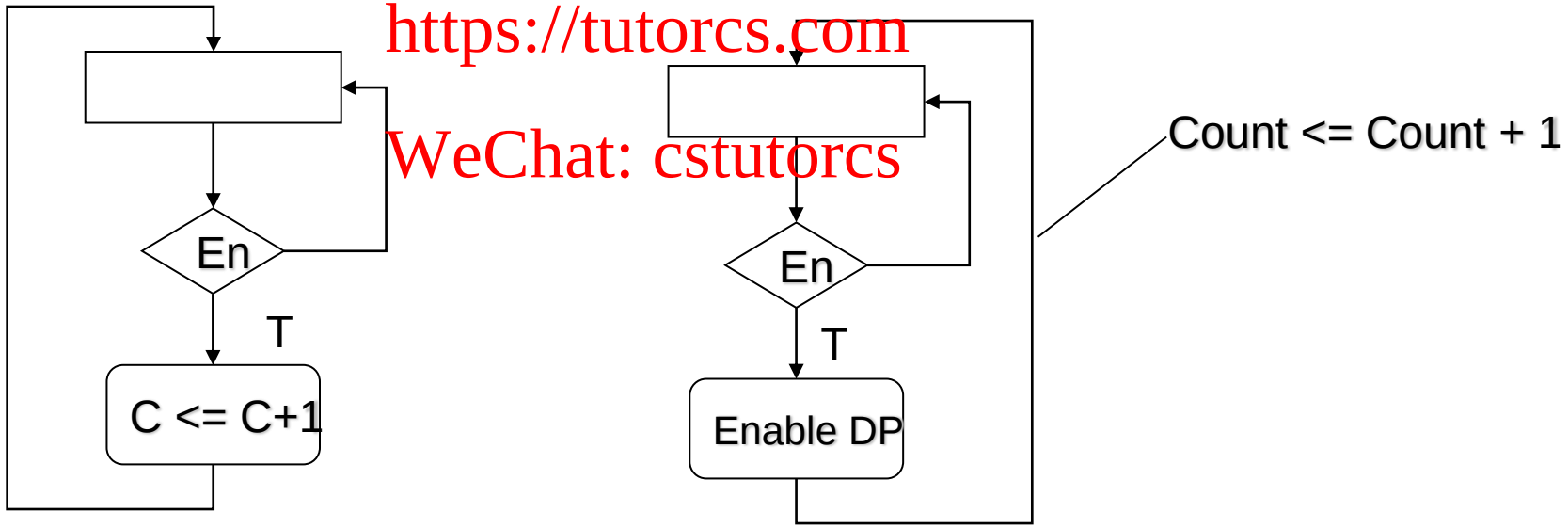
Designing ASMD Charts

- Designing an ASMD chart has three steps:
  - Form an ASM chart displaying only how the inputs to the controller determine its state transitions.
  - Convert the ASM chart to an ASMD chart by annotating the edges of ASM chart to indicate the concurrent register operations of the datapath unit.
  - Modify the ASMD chart to identify the control signals that are generated by the controller and cause the indicated register operations in the datapath unit.

ASM vs. ASMD charts for a counter with enable

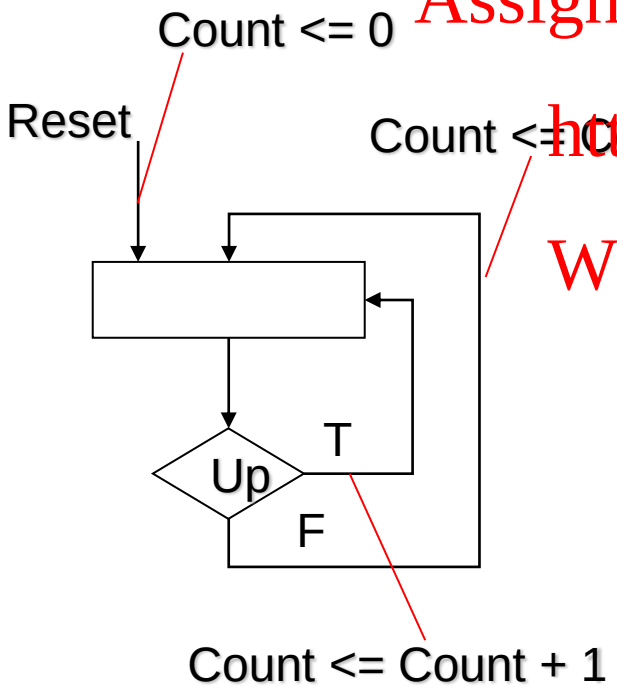
ASM chart representation

ASMD chart representation

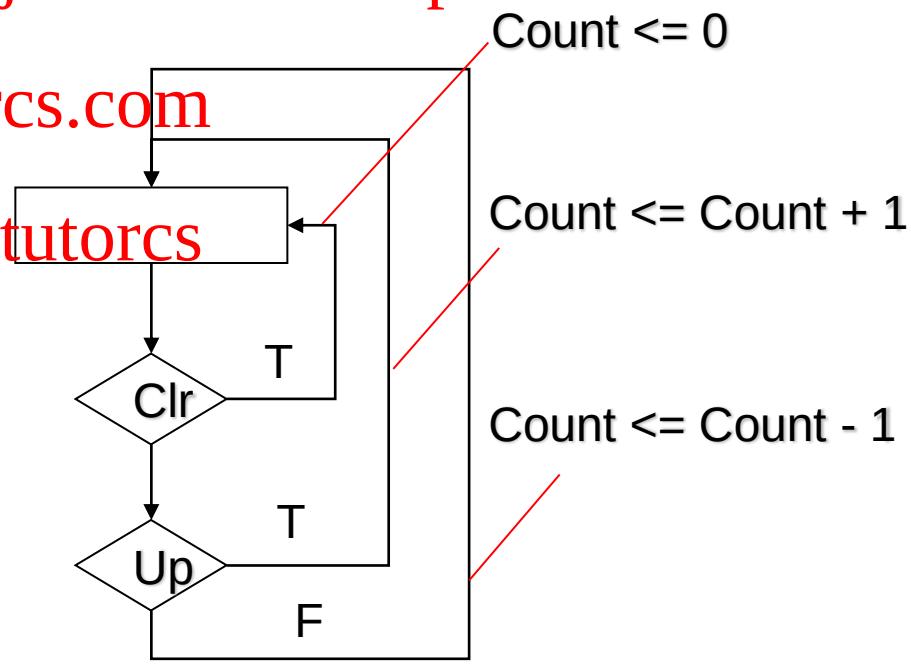


An ASMD chart for an up-down counter

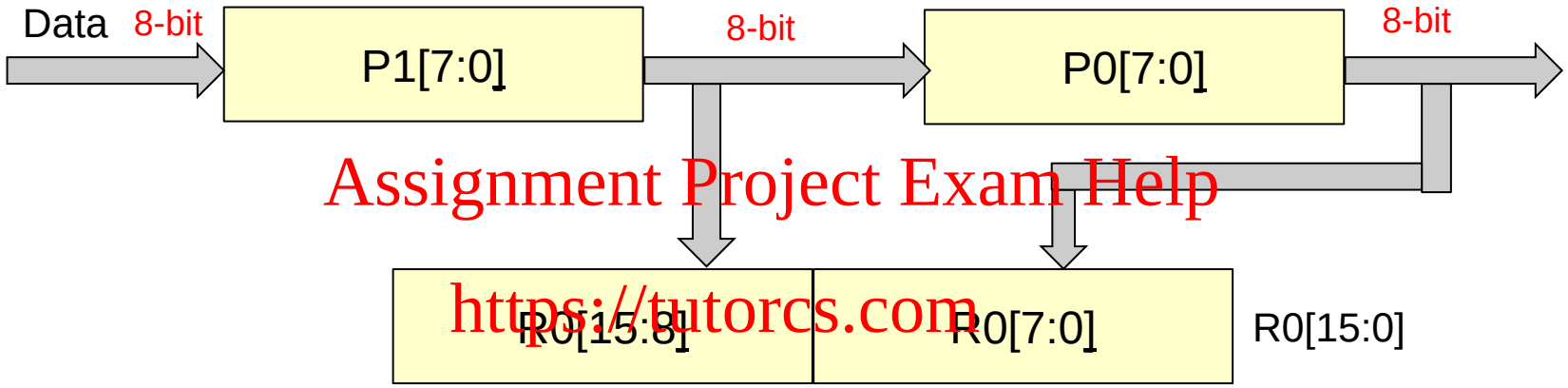
Up-down counter
with asynchronous reset



Up-down counter
with synchronous reset

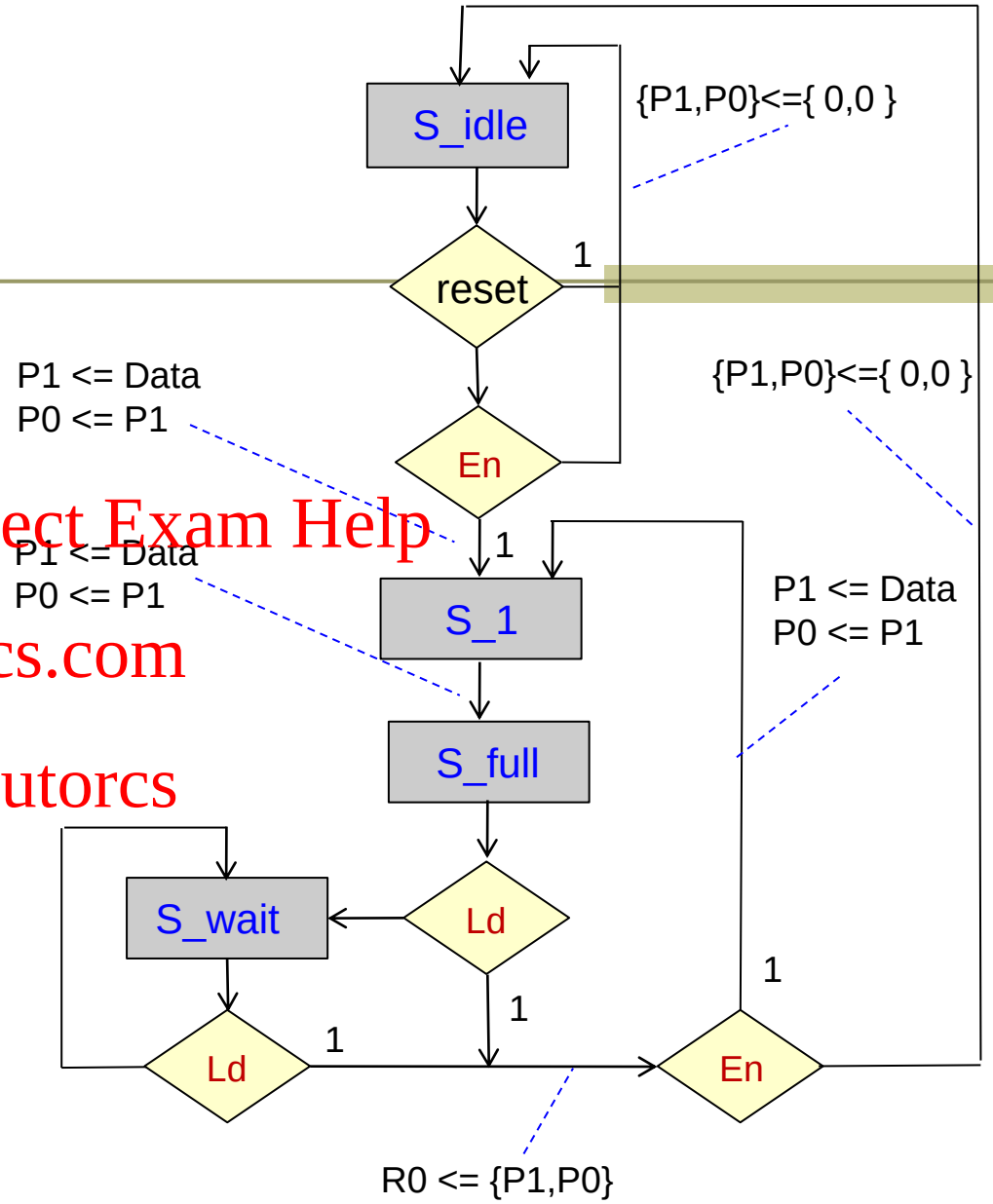


ASMD Example Two-stage Pipeline Register



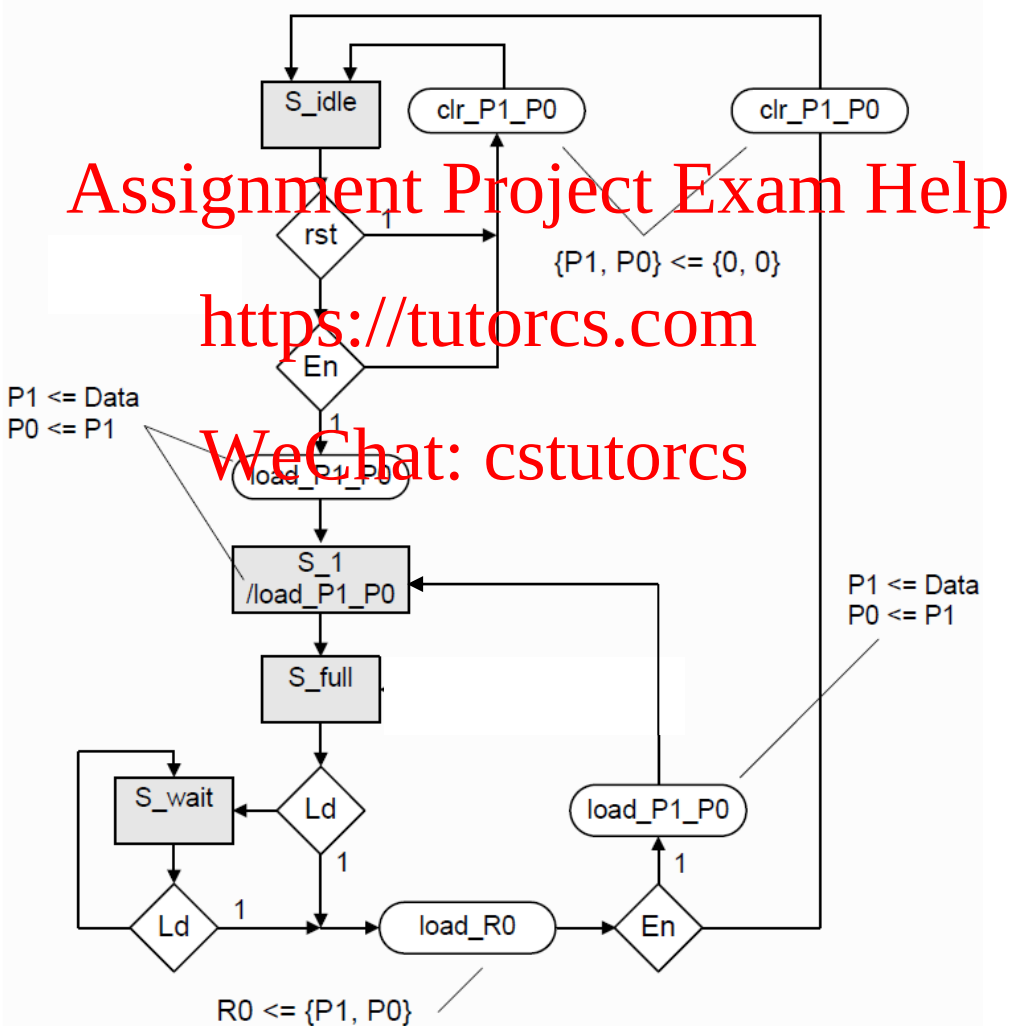
- The above architecture shows a two-stage pipeline that acts as a 2:1 decimator with parallel input and output.
- Decimators are used in digital signal processors to move data from a high-clock-rate datapath to a lower-clock-rate datapath.

- The machine has a synchronous reset to S_idle. Reset is checked in all the states but it is only shown for the S_idle state.
- With En asserted the machine transits from state S_idle to state S_1 and concurrently Data is loaded to P1 and the content of P1 is moved to P0.
- At the next clock cycle the state goes to S_full and concurrently Data is loaded to P1 and the content of P1 is moved to P0. Now the pipe is full.
- If Ld is asserted at the next clock the machine moves to S_1 while dumping the pipe into a holding register R0.
- If Ld is not asserted the machine enters S_wait.



- The conditional and unconditional output signals that interface the controller with the datapath and cause the indicated operations must be added to complete the chart.

Complete ASMD Chart



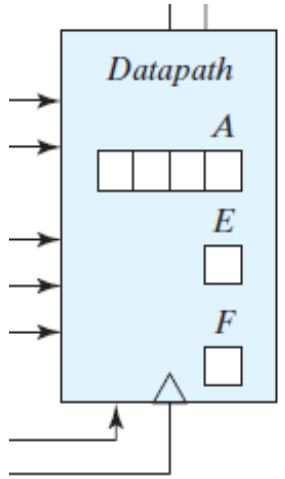
Datapath Controller Design Steps

1) Understanding of the sequential register operations that must execute on a given datapath architecture.
2) Designing an ASM chart that is controlled by the primary input signals and/or status signals from the datapath.
3) Form an ASMD chart by annotating the arcs of the ASM chart with the datapath operations associated with the state transitions of the controller.
4) Generate unconditional and conditional outputs to control the datapath.

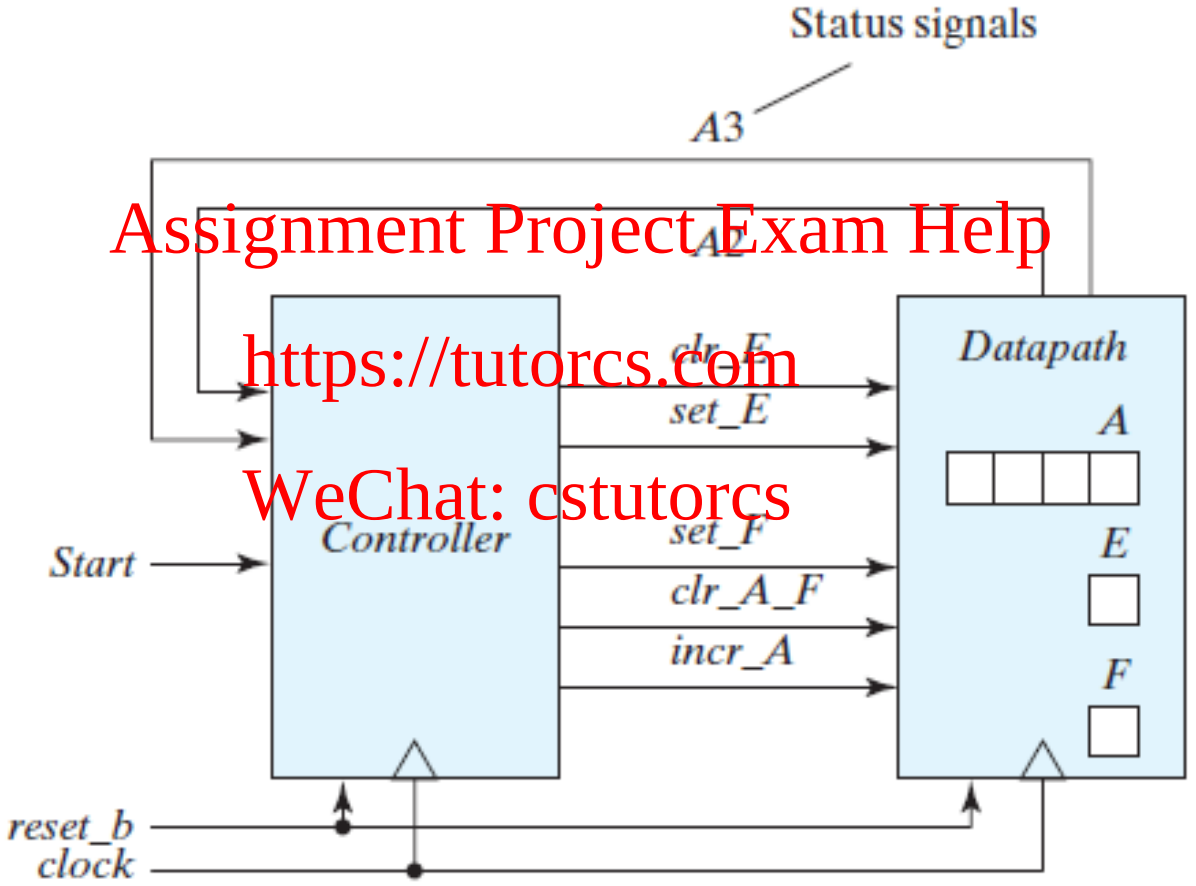
from: “Advanced Digital Design with Verilog HDL” by M. D. Ciletti

Design Example: Specifications

- The datapath unit is to consist of two JK flip-flops ‘E’ and ‘F’, and one 4-bit binary counter A[3:0]. A[3] holds the most significant bit.
- A signal, *Start*, initiates the system’s operation by clearing the counter ‘A’ and flip-flop ‘F’.
- At each subsequent clock pulse, the counter is incremented by 1 until the operations stop.
- Counter bits A2 and A3 determine the sequence of operations:
  - If A2 = 0, ‘E’ is cleared to 0 and the count continues.
  - If A2 = 1, ‘E’ is set to 1; then
    if A3 = 0, the count continues, but if A3 =1, F is set to 1
    on the next clock pulse and the system stops counting.
  - Then, if *Start* = 0, the system remains in the initial state, but if *Start* = 1, the operation cycle repeats.

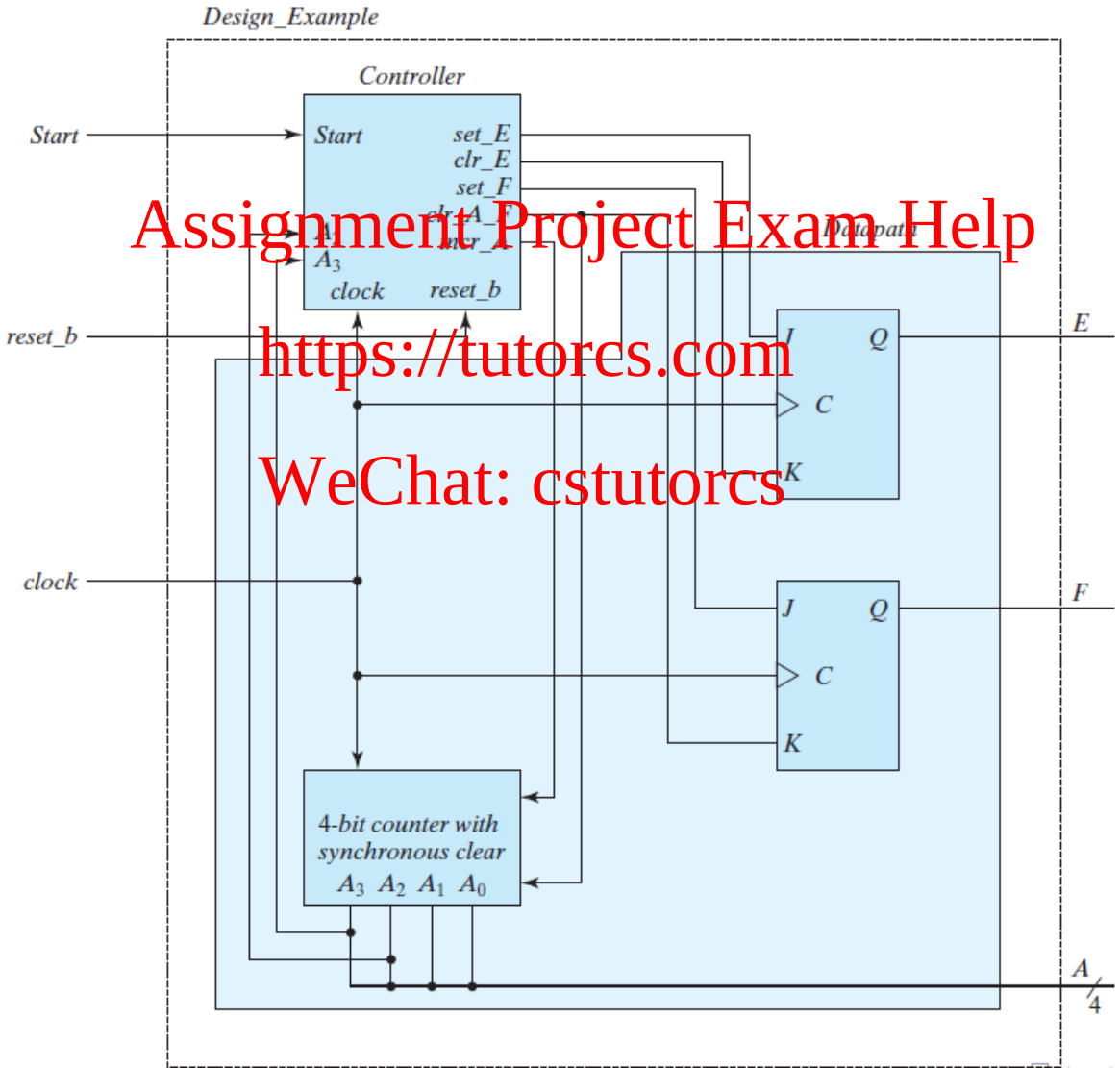


Block Diagram of the System's Architecture

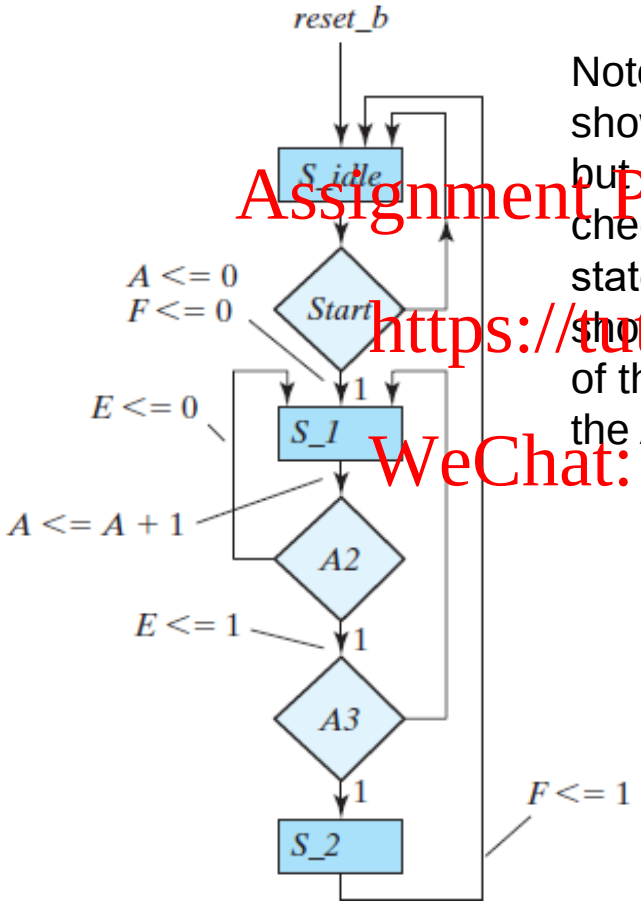


Notes: A3 denotes A[3] and A2 denotes A[2]
reset_b denotes active-low reset condition

Datapath - Controller

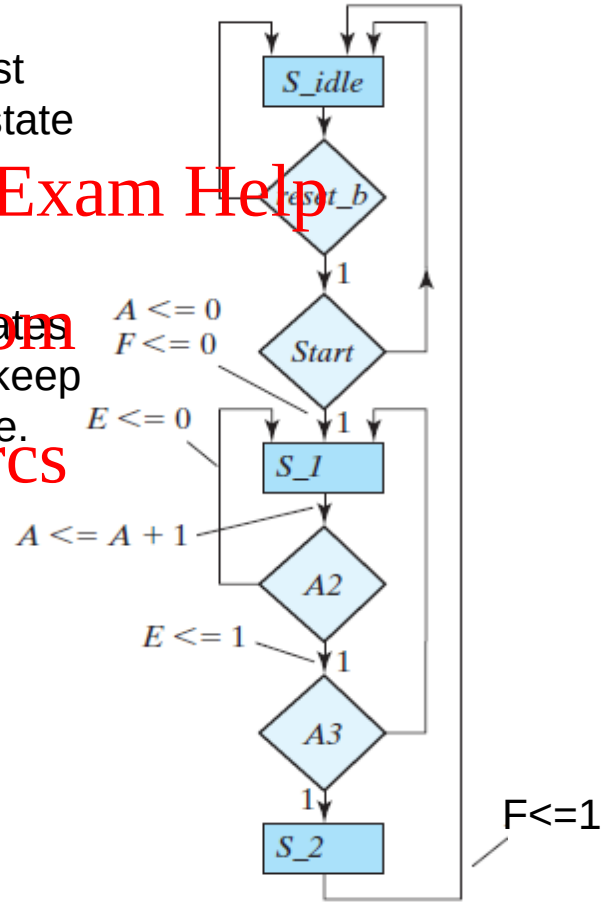


ASMD chart for controller state transition



Asynchronous reset

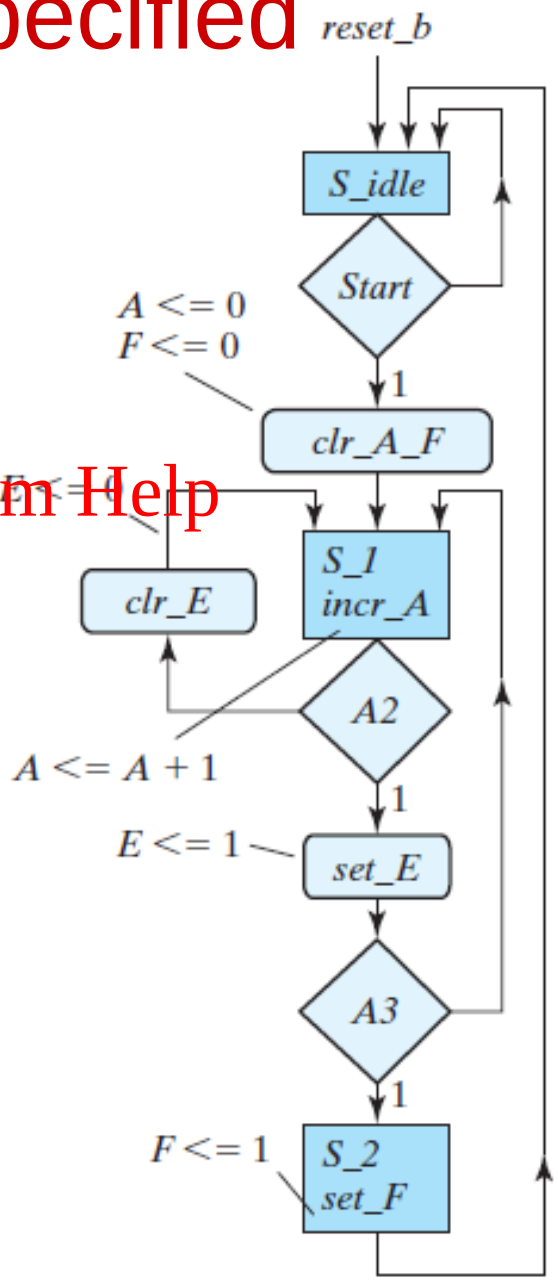
Note: Reset just shown in first state but is actually checked in all states. It's not shown in all states of the ASM to keep the ASM simple.



Synchronous reset

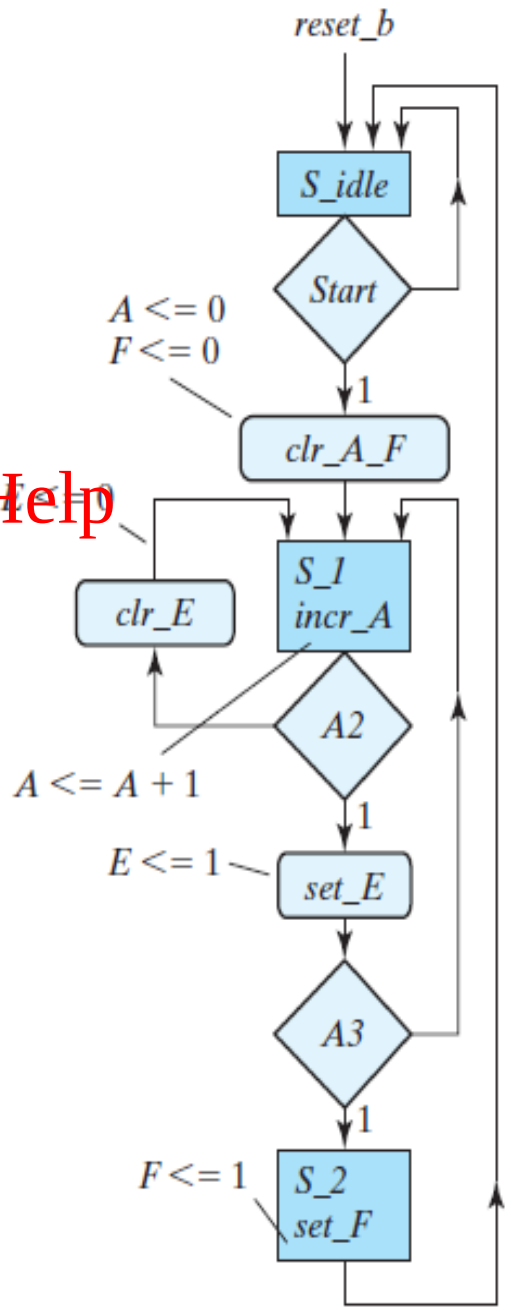
ASMD chart for a completely specified controller

- In practise, designers use the ASMD chart to write Verilog models of the controller and the datapath and then synthesise a circuit directly from the Verilog description.



```
1  module DatapathControl ( A, E, F, Start, clock, reset_b);
2      output reg [3:0]  A;
3      output reg        E, F;
4      input             Start, clock, reset_b;
5
6      reg   [1:0]       state, next_state;
7      reg   clr_E, set_E, clr_F, set_F, clr_A_F, incr_A;
8      parameter   S_idle = 2'b00, S_1 = 2'b01, S_2 = 2'b11;
9      wire  A2 = A[2], A3 = A[3];
10
11     // control unit - registers
12     always @ (posedge clock, negedge reset_b)
13        if (reset_b ==0) state <= S_idle;
14        else state <= next_state;
15
16     // control unit - combinational
17     always @ (state, Start, A2, A3)
18        begin
19           next_state = state;
20           clr_E = 0;
21           set_E = 0;
22           clr_A_F = 0;
23           set_F = 0;
24           incr_A = 0;
25           case (state)
26              S_idle: if (Start) begin next_state = S_1; clr_A_F = 1; end
27              S_1:  begin incr_A = 1; if (A2 == 0) set_E = 1;
28                                 else  begin
29                                          set_E = 1;
30                                          if (A3) next_state = S_2;
31                                       end
32                    end
33              S_2: begin set_F = 1; next_state = S_idle; end
34           endcase
35        end
36     // datapath unit
37     always @ (posedge clock)
38        begin
39           if (clr_E) E <= 0;
40           if (set_E) E <= 1;
41           if (clr_A_F) begin A <= 0; F <= 0; end
42           if (set_F) F <= 1;
43           if (incr_A) A <= A +1 ;
44        end
45 endmodule
```

Note: System stays in same state unless told to change state



Simulation