
Computer Aided Digital Systems Design - EE 4743/6743

Sherif Abdelwahed

Scheduling Examples

**Department of Electrical and Computer Engineering
Mississippi State University**

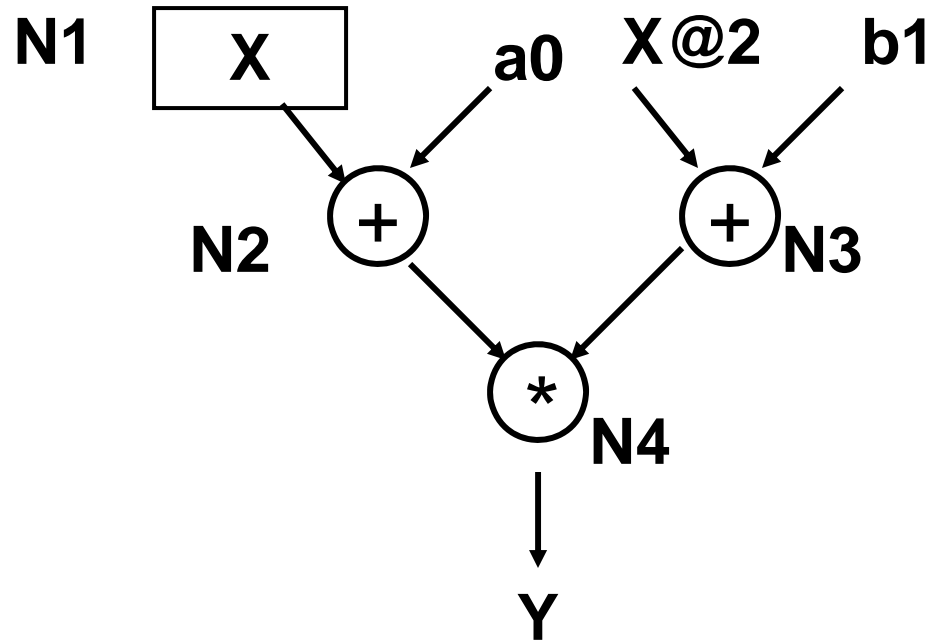
Example Problem

- Equation:

$$Y = (X + a_0) * (x @ 2 + b_1)$$

- Constraints: 1 Multiplier, 1 Adder
 - Build machine
 - Dataflow Graph
 - Scheduling
 - Resource Estimation
 - Datapath
 - ASM chart
 - Verilog code
 - Start command to begin
-

Dataflow Graph



Scheduling

Cycle Start	Adder	Multiplier	IO
#1	Idle	N3 ($x@2+b1$)	Input x
#2	Idle	N2 ($x+a0$)	
#3	N4 ($N2*N3$)	Idle	
Utilization	33%	66%	33%

Resource Estimation - Registers

- 2 coefficients (a0, b1)
 - One old input (X@2)
 - One more for intermediate input (X@1)

 - 4 at minimum
 - Define RB for X@2, RC for X@1
-

Register Scheduling

Registers: $RB=x@2$, $RC=x@1$

Clock 1:

Input X ??? Where to put this?

Add another register RA

Input X : $RA \leftarrow X$

$X@2+b1$ (N3): $RB \leftarrow RB + b1$ (don't need $x@2$ after this, destroy RB)

Register Scheduling

Registers: $RA = X$, $RB = x@2$, $RC = x@1$

Clock 2:

$X + a0$ (N2): ??? $\leftarrow RA + a0$ (nowhere to put it, so need new register)

Add another register RD

$X + a0$ (N2): $RD \leftarrow RA + a0$

Clock 3:

$N2 * N3$ (N4): $Y_out \leftarrow RB * RD$ (no new register needed)

Also:

$RC \leftarrow RA$, $RB \leftarrow RC$ (to set up next cycle)

Datapath planning: Unit sources, destinations

Mult : Left sources: RD

Right sources: RB

Adder: Left sources: a0, b1

Right sources: RA, RB

RA src: X

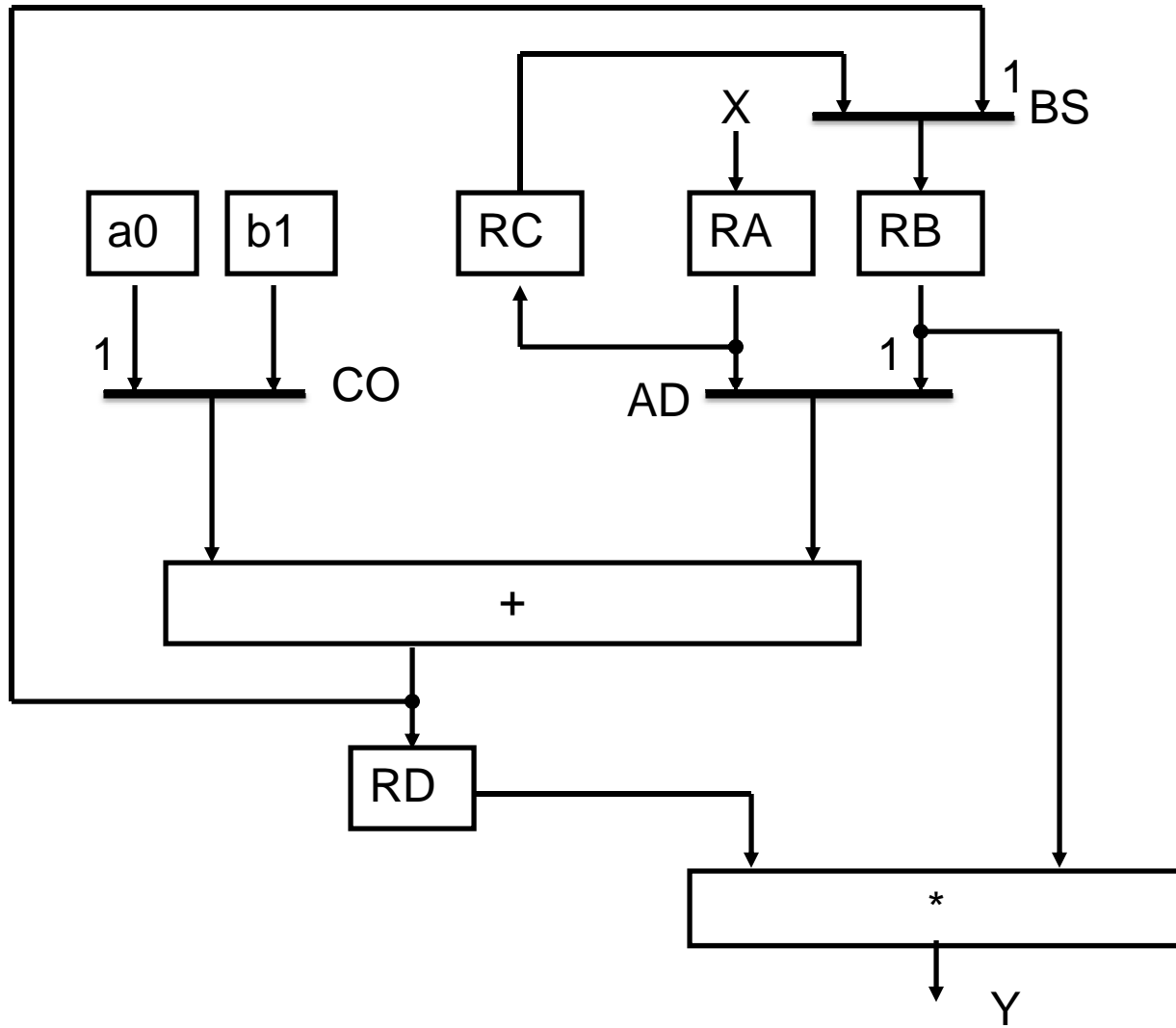
RB src: Adder, RC

RC src: RA

RD src: Adder

a0, b1 registers loaded externally from databus X

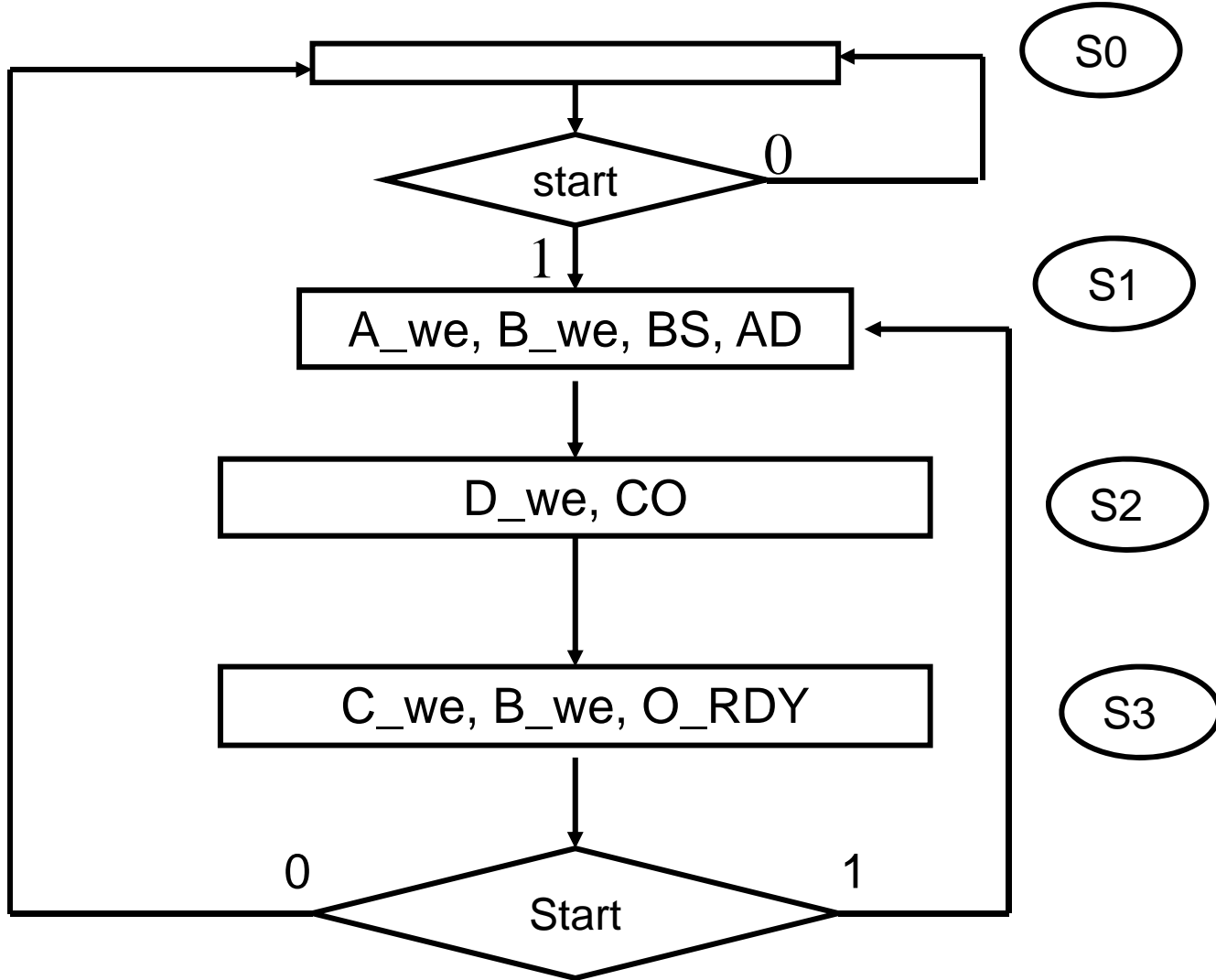
Datapath



Needed I/O

- Output from FSM:
 - Write Enable for each Register
 - Mux select lines (BS, AD, CO)
 - Output Ready (O_RDY)
 - Input to FSM:
 - Start
-

ASM Chart



FSM Verilog Design

```
Module fsm(clk, reset, start, A_we, B_we, C_we, D_we, BS, CO, AD, O_RDY,
state);

    input clk, reset;
    input start; // control inputs
    output A_we, B_we, C_we, D_we; // control outputs
    output BS, CO, AD, O_RDY; // control outputs
    output [1:0] state; // state out for debugging
    reg [1:0] pstate, nstate;

    reg A_we, B_we, C_we, D_we;
    reg BS, CO, AD, O_RDY;

    parameter S0 = 2'b00; // state encoding
    parameter S1 = 2'b01;
    parameter S2 = 2'b10;
    parameter S3 = 2'b11;

    // look at present state for debugging purposes
    assign state = pstate;
```

FSM Verilog Design

```
// updated present state with next state with DFFs
always @ (posedge clk)
    if (reset) pstate = S0;
    else pstate = nstate;

// define next state and output logic
always @ (start or pstate)
    begin
        // default assignments
        nstate = pstate;
        A_we = 0;
        B_we = 0;
        C_we = 0;
        D_we = 0;
        AD = 0;
        BS = 0;
        CO = 0;
        O_RDY = 0;
    end
```

FSM Verilog Design

```
    case (pstate)
      S0 :    if (start==1) nstate = S1;
      S1 :    begin
                A_we = 1; B_we = 1; BS    = 1;
                nstate = S2;
            end
      S2 :    begin
                D_we = 1; C0    = 1;
                nstate = S3;
            end
      S3 :    begin
                C_we = 1; B_we = 1; O_RDY = 1;
                if (start = 1) nstate = S1;
                else nstate <= S0;
            end
        default: nstate = S0;
    endcase
end
endmodule
```