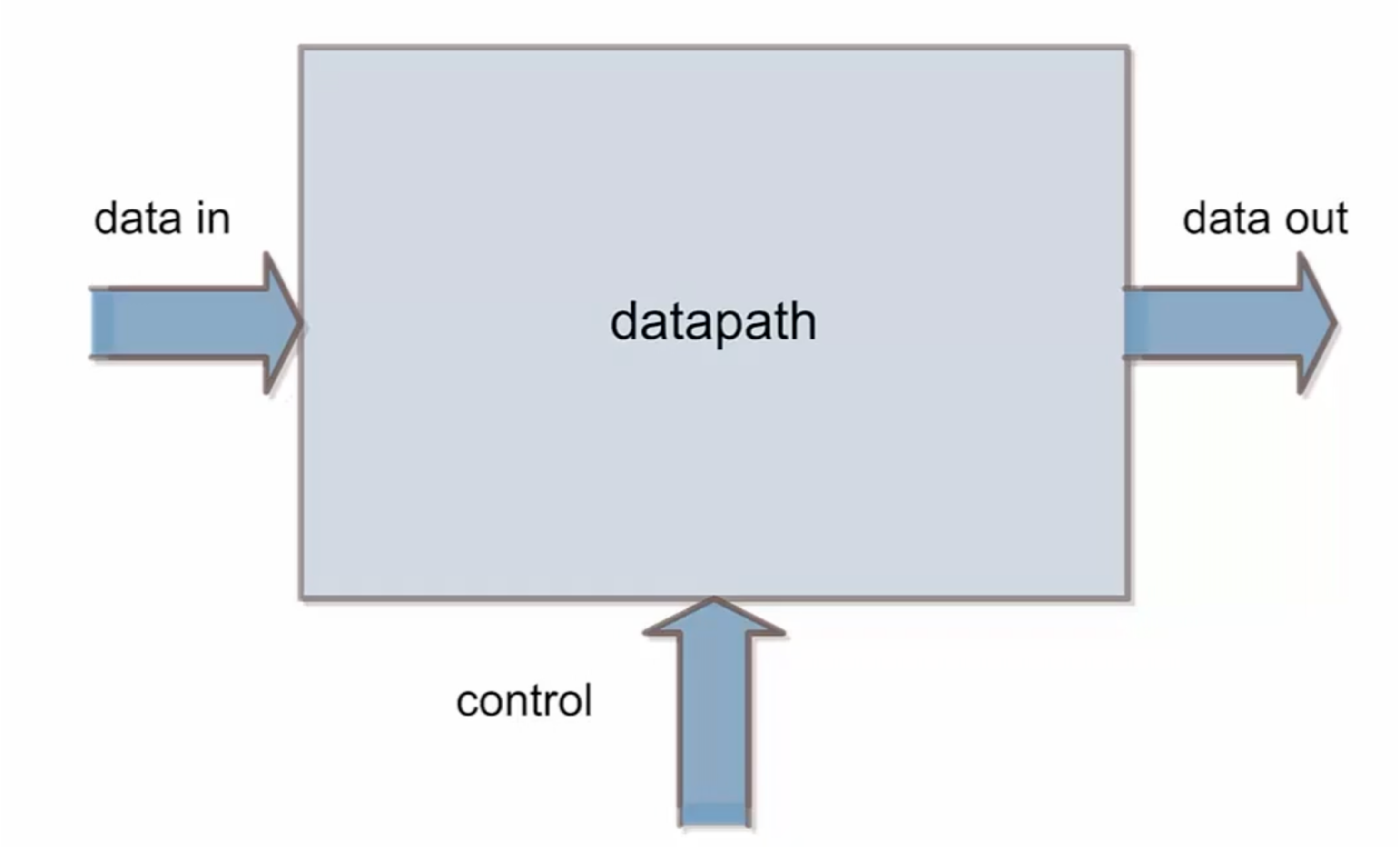
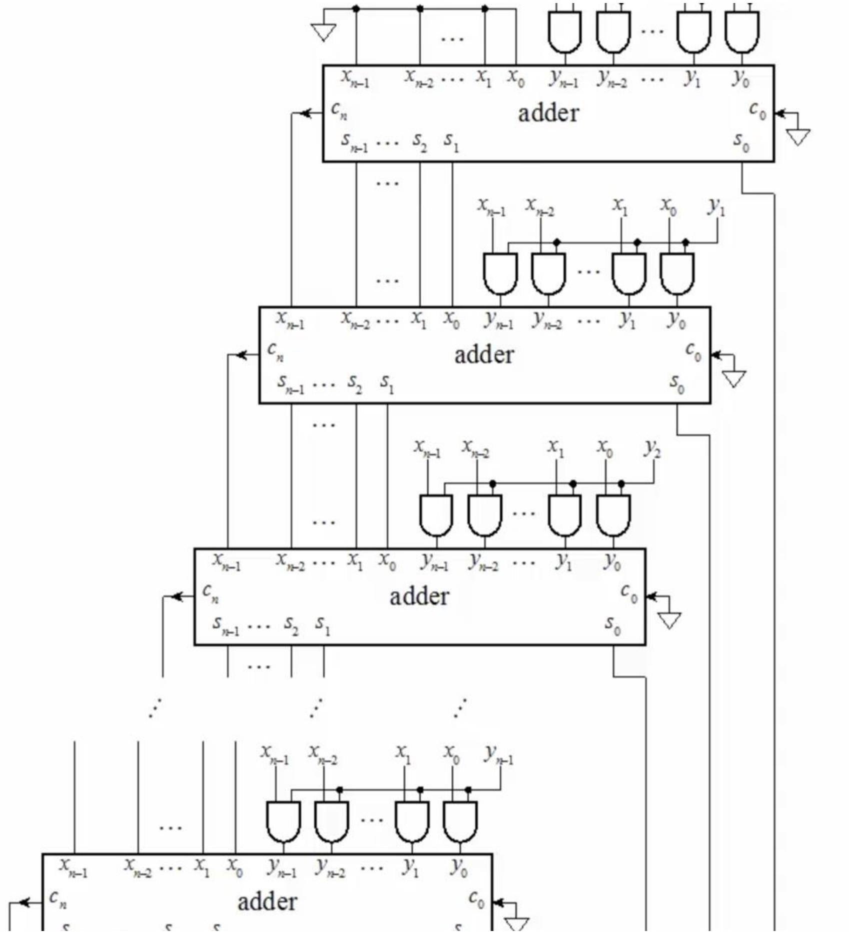
Digital Circuit Design

Li Bai

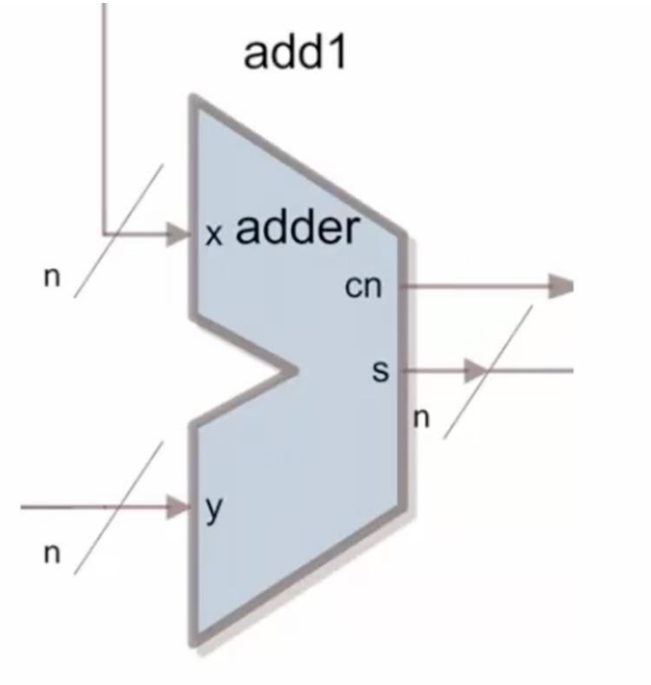
Data Path



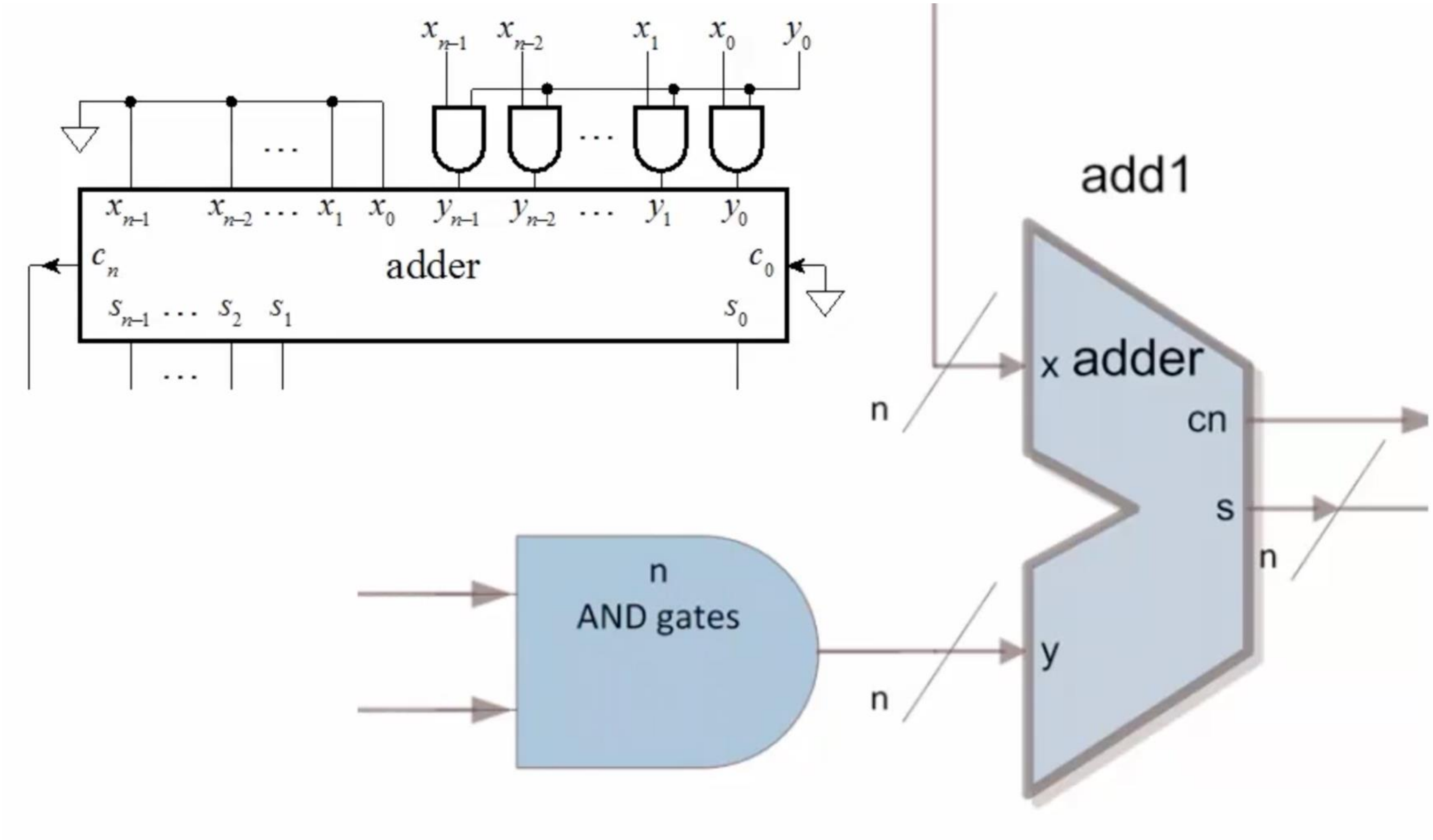
Recall the multiplication



x_{n-1} x_{n-2} … x_1 x_0 y_{n-1} y_{n-2} … y_1 y_0
c_n
adder
c_0
s_{n-1} … s_2 s_1
s_0
x_{n-1} x_{n-2} x_1 x_0 y_1
x_{n-1} x_{n-2} x_1 x_0 y_2
x_{n-1} x_{n-2} x_1 x_0 y_{n-1}

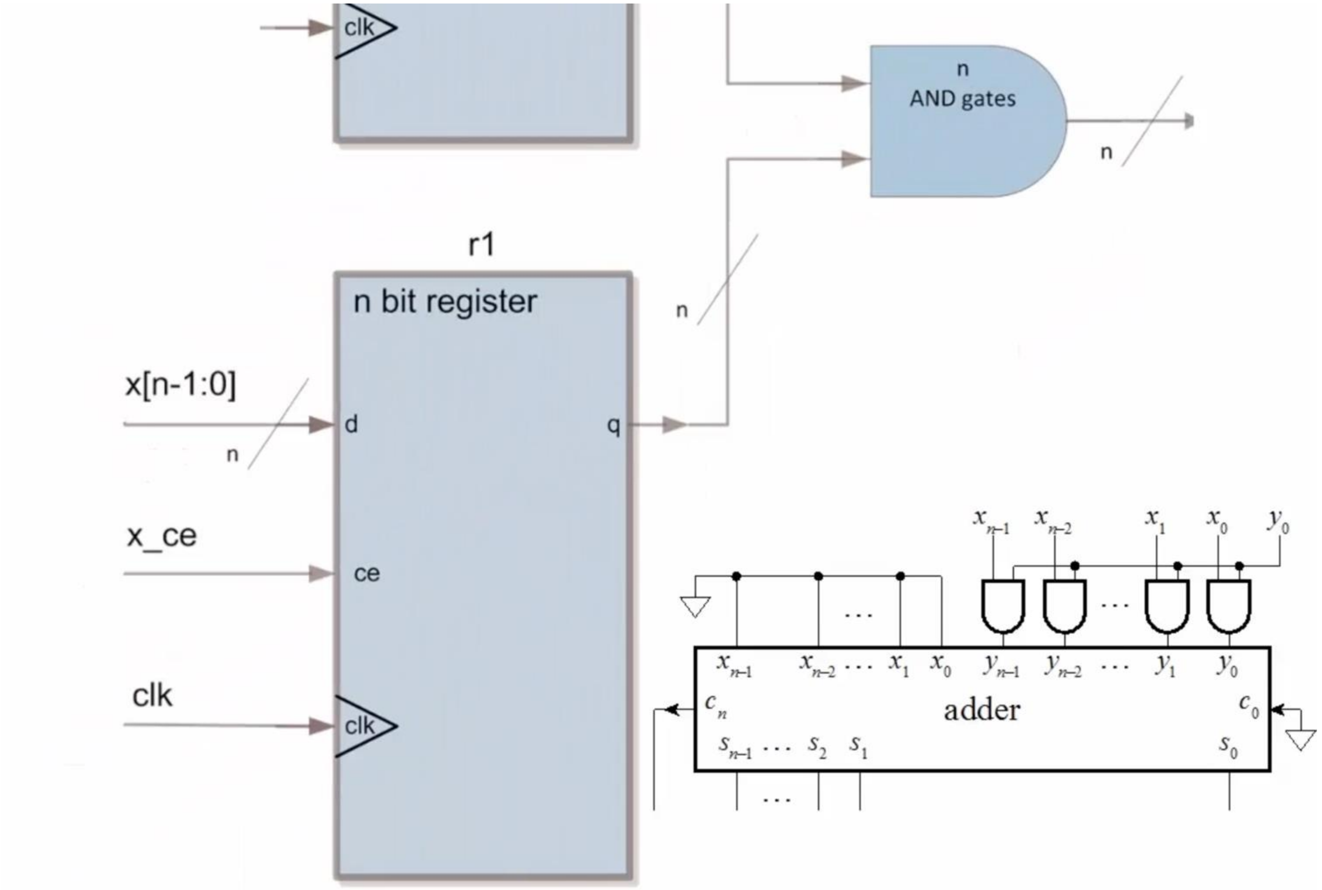


Shift only one bit and with x bus

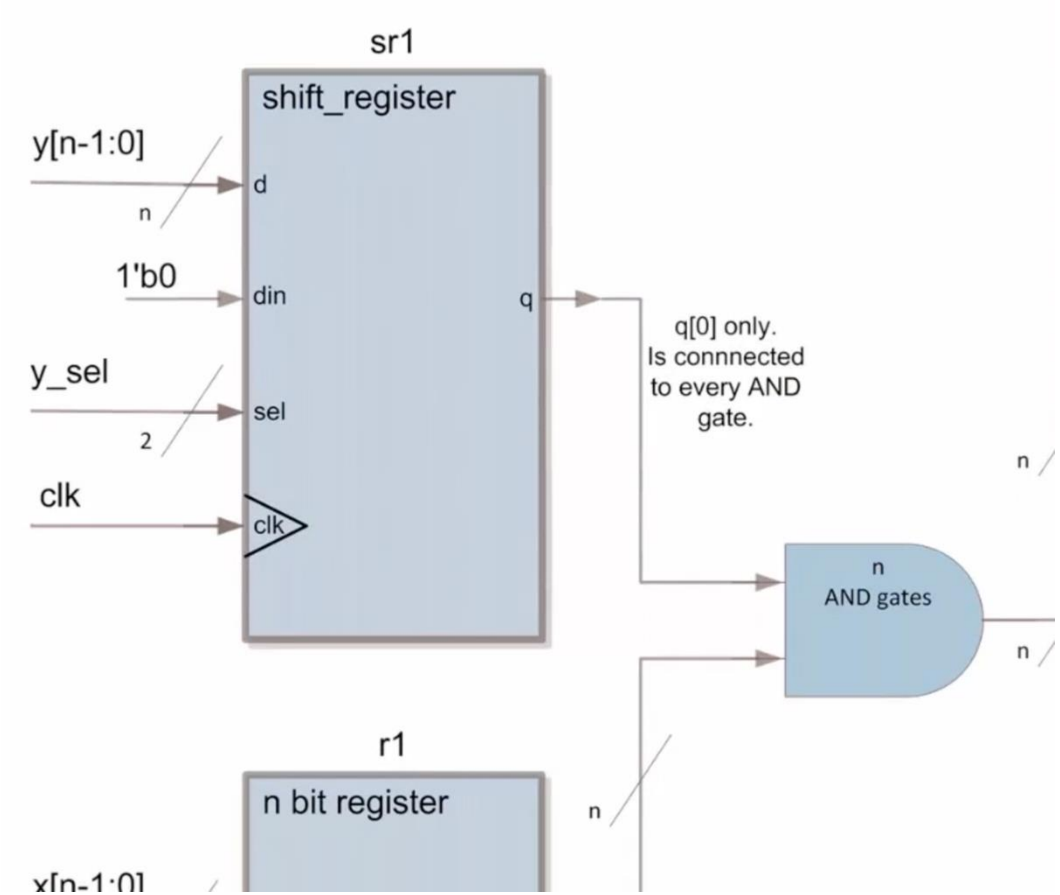


x_{n-1} x_{n-2} x_1 x_0 y_0
...
...
x_{n-1} x_{n-2} ... x_1 x_0 y_{n-1} y_{n-2} ... y_1 y_0
c_n
adder
c_0
s_{n-1} ... s_2 s_1
s_0
...
add1
x
adder
cn
s
n
n
n
y
n
AND gates
n

Use a n-bit shift register

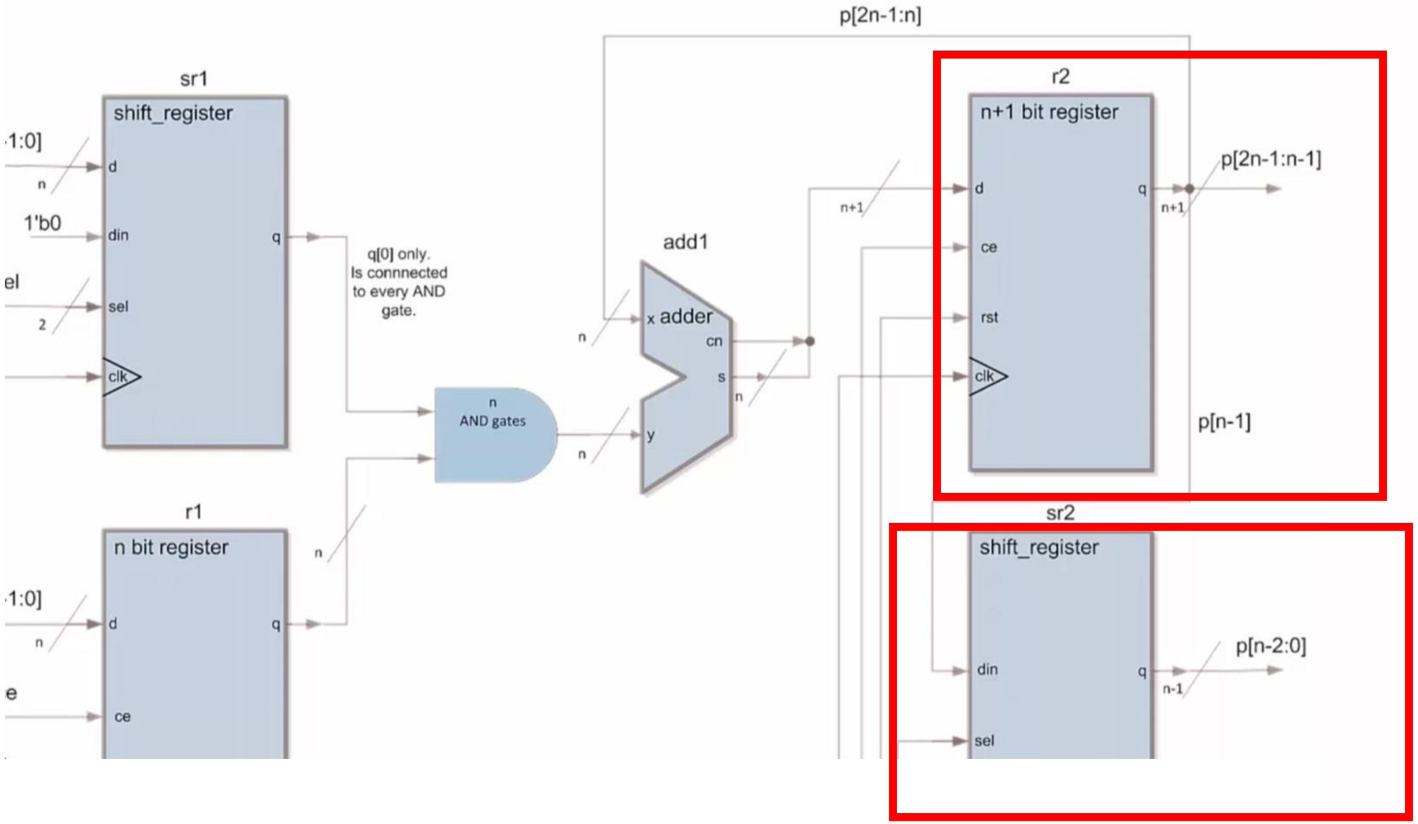


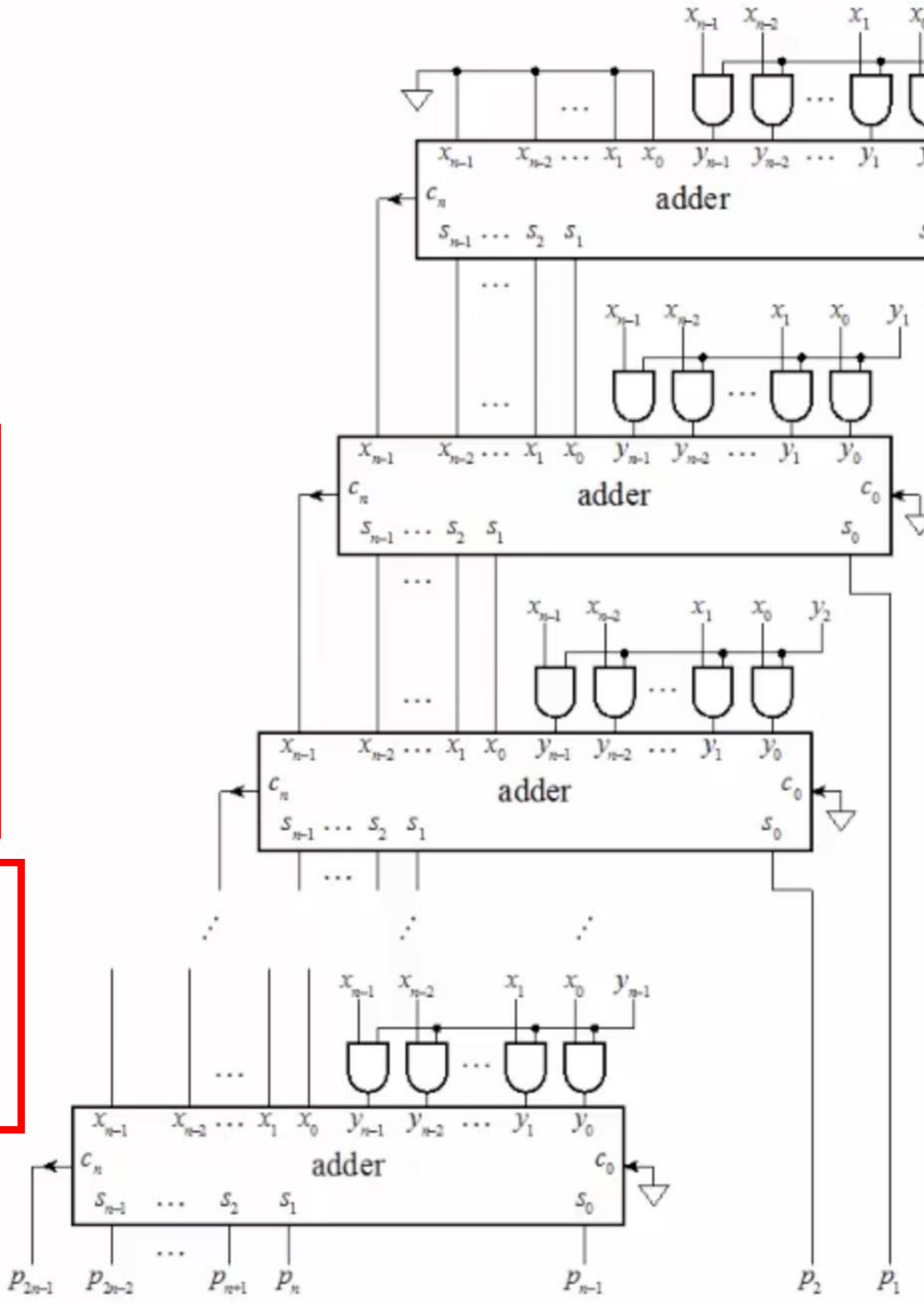
Shift y bus one bit at a time



Why y_sel has 2-bit

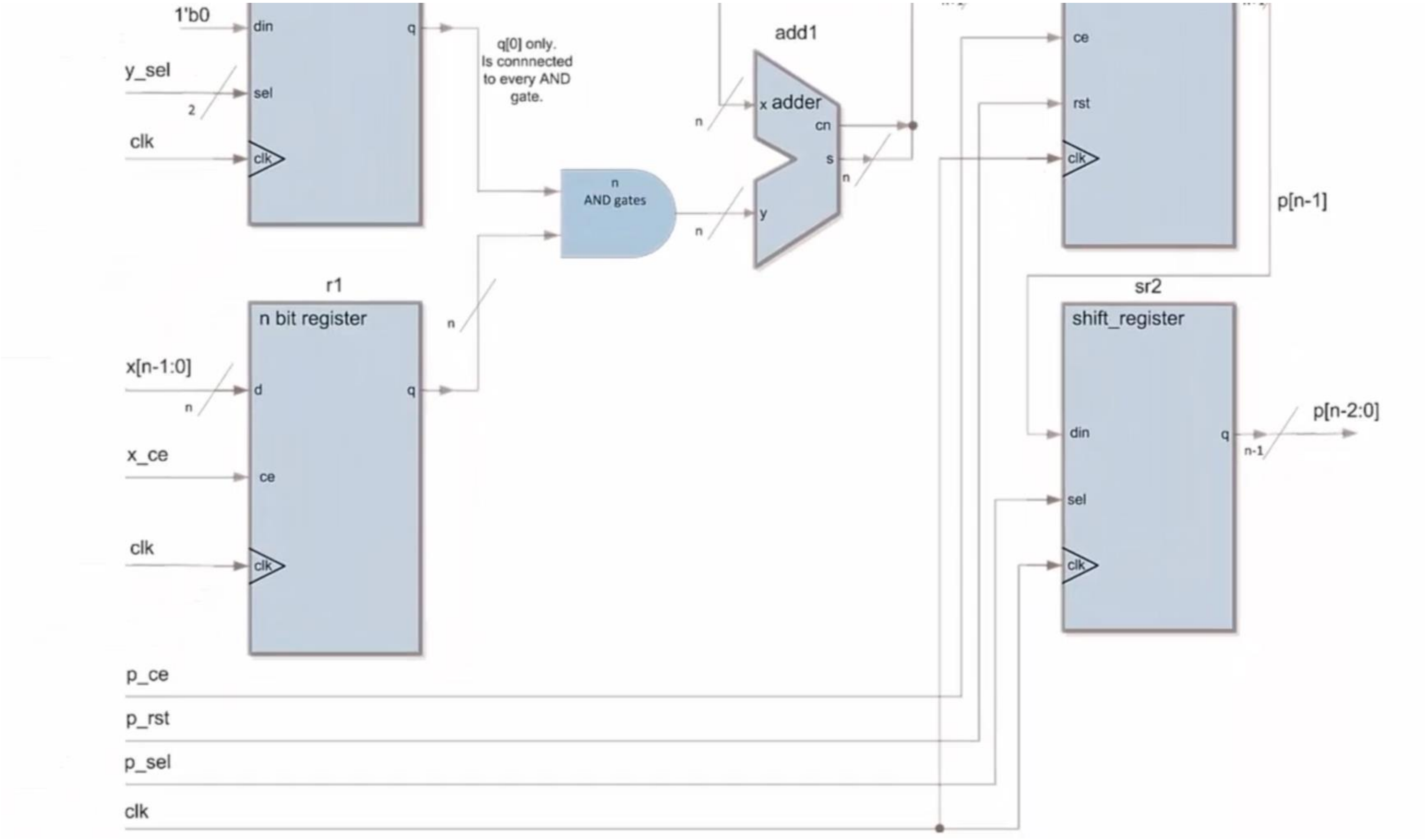
Put everything together





x_{n-1} x_{n-2} x_1 x_0
adder
x_{n-1} x_{n-2} … x_1 x_0 y_{n-1} y_{n-2} … y_1 y_0
c_n
c_0
s_{n-1} … s_2 s_1
s_0
p_{2n-1} p_{2n-2} … p_{n+1} p_n
p_{n-1}
p_2 p_1

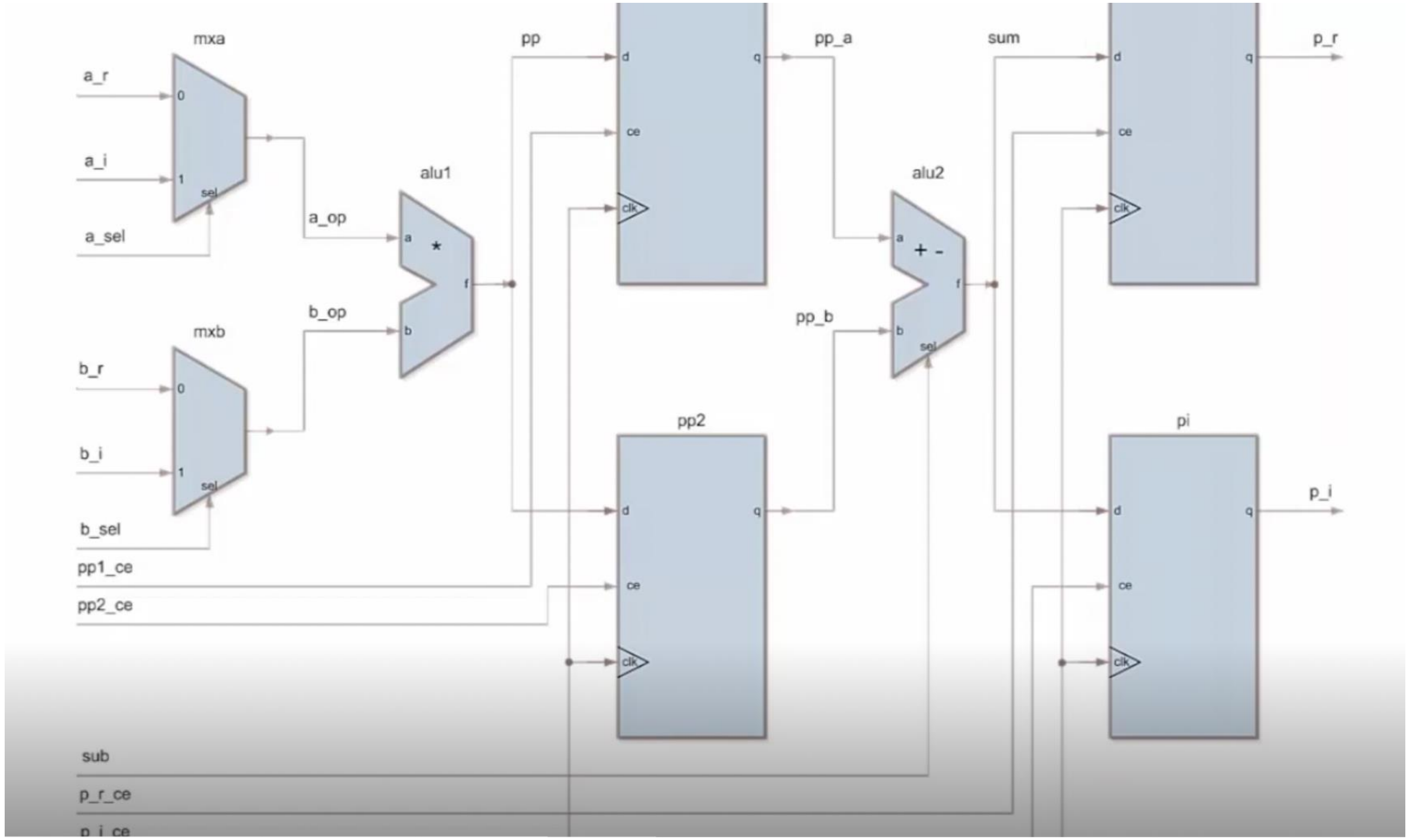
Control registers



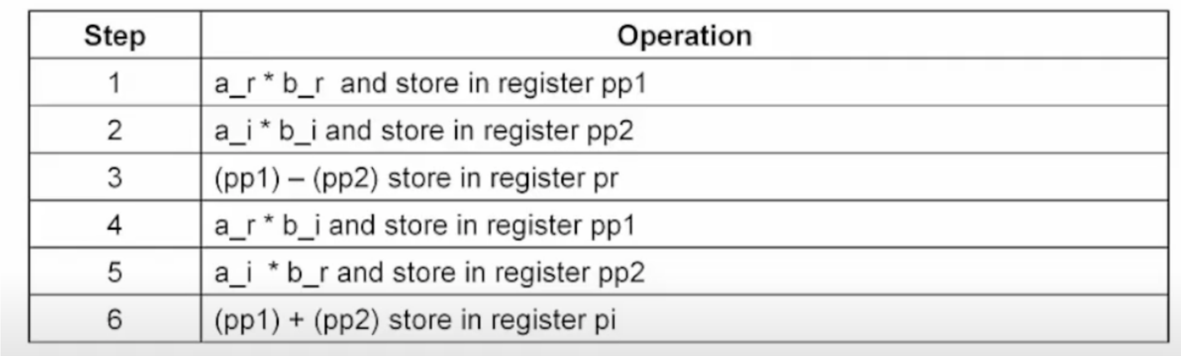
Advantage of a data path

- Use time to control data load and shift operation to achieve calculation
- You can use less number hardware
- Clock speed is important and instruction is important

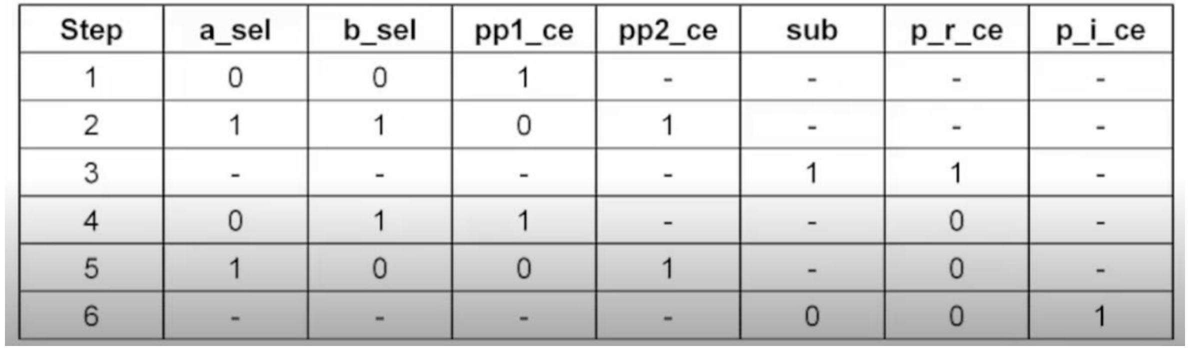
Data Path for complex multiplication



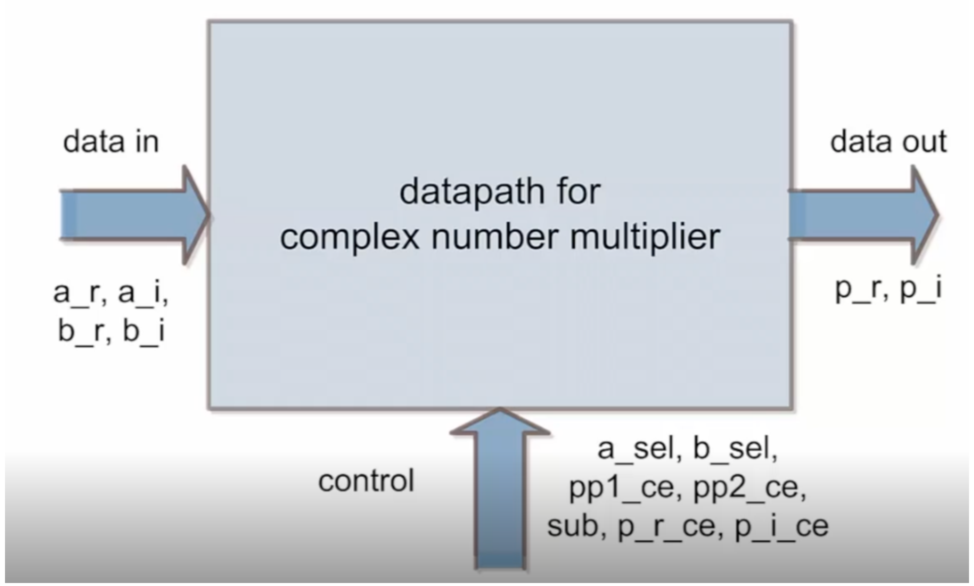
Data Path

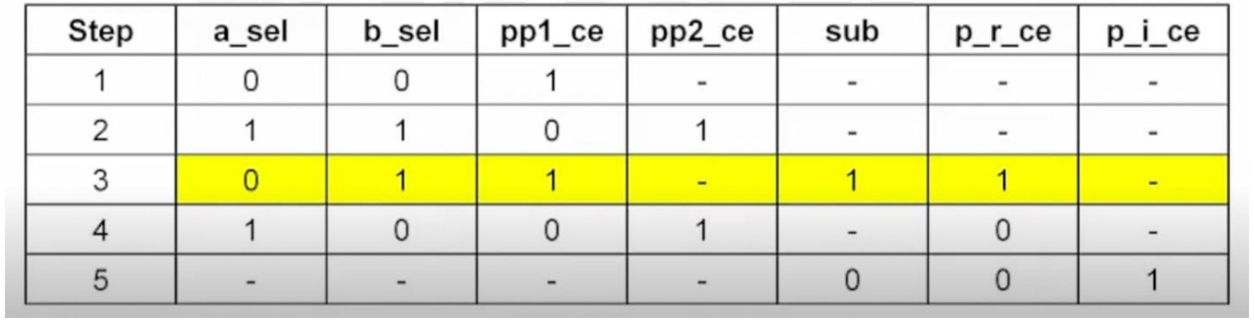


Step	Operation
1	a_r * b_r and store in register pp1
2	a_i * b_i and store in register pp2
3	(pp1) – (pp2) store in register pr
4	a_r * b_i and store in register pp1
5	a_i * b_r and store in register pp2
6	(pp1) + (pp2) store in register pi



Step	a_sel	b_sel	pp1_ce	pp2_ce	sub	p_r_ce	p_i_ce
1	0	0	1	-	-	-	-
2	1	1	0	1	-	-	-
3	-	-	-	-	1	1	-
4	0	1	1	-	-	0	-
5	1	0	0	1	-	0	-
6	-	-	-	-	0	0	1





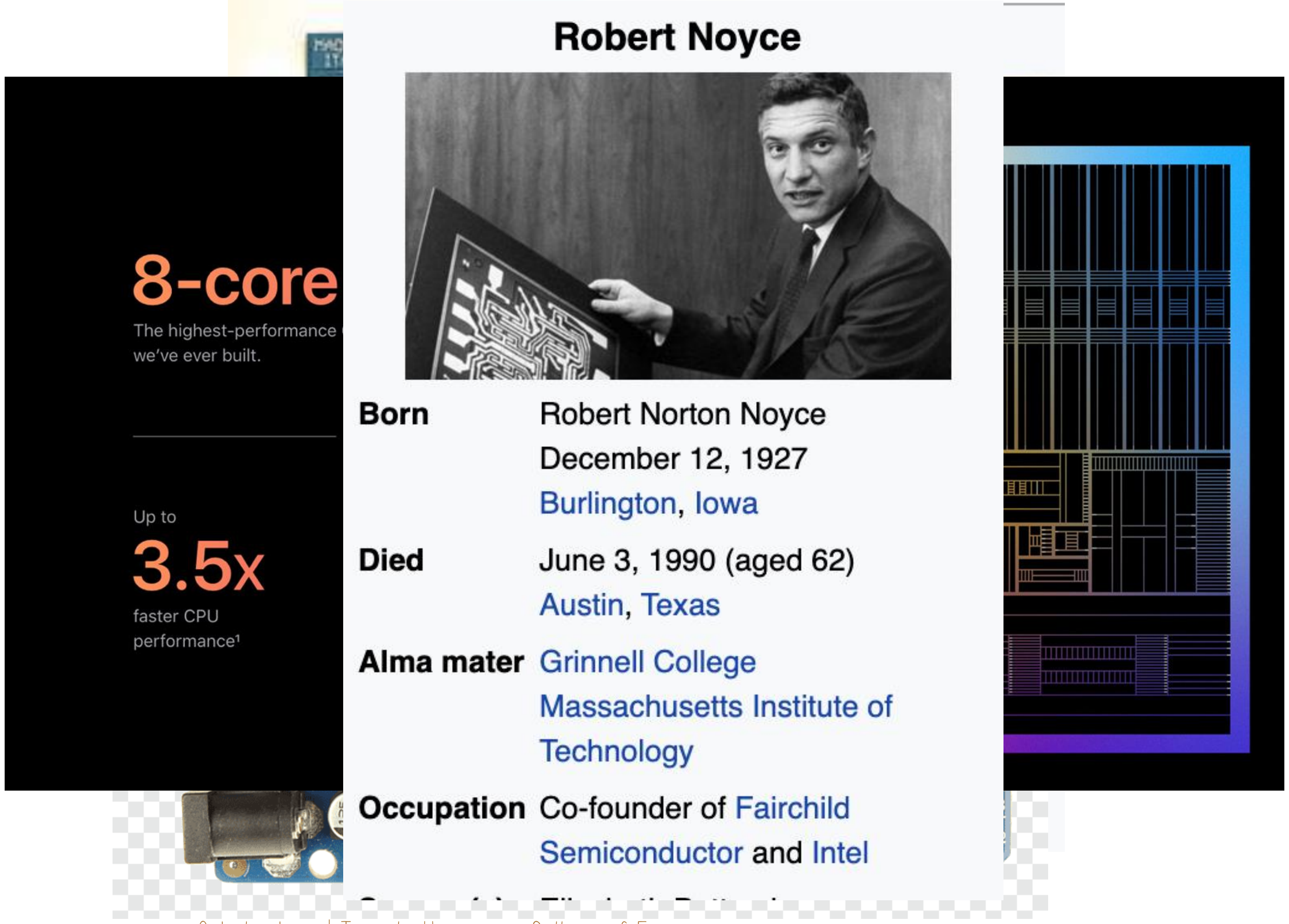
Step	a_sel	b_sel	pp1_ce	pp2_ce	sub	p_r_ce	p_i_ce
1	0	0	1	-	-	-	-
2	1	1	0	1	-	-	-
3	0	1	1	-	1	1	-
4	1	0	0	1	-	0	-
5	-	-	-	-	0	0	1

→ Instruction sets

Instruction set and processors

- CISC (Complex Instruction Set Computer)
  - Intel processor
  - Power PC processor
  - Western Digital
- RISC (Reduced Complex Instruction Set Computer)
  - ARM processor
    - Raspberry pi
    - Smart Phone
  - AVR processor (Alf and Vegard's RISC
    - (Ateml- ECE3612)
- Apple
  - Power pc -> Intel processor -> A14 Bionic -> M1 Processor

Q&A With Co-Creator of the 6502 Processor – IEEE Spectrum

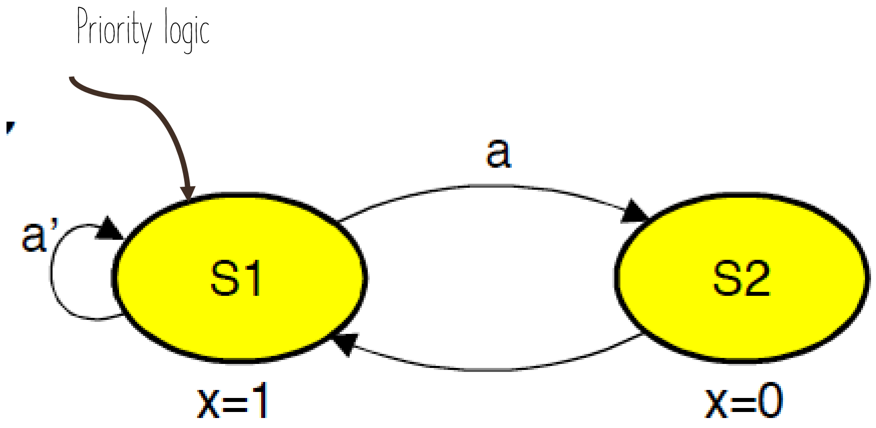


Robert Noyce

Born	Robert Norton Noyce December 12, 1927 Burlington, Iowa
Died	June 3, 1990 (aged 62) Austin, Texas
Alma mater	Grinnell College Massachusetts Institute of Technology
Occupation	Co-founder of Fairchild Semiconductor and Intel

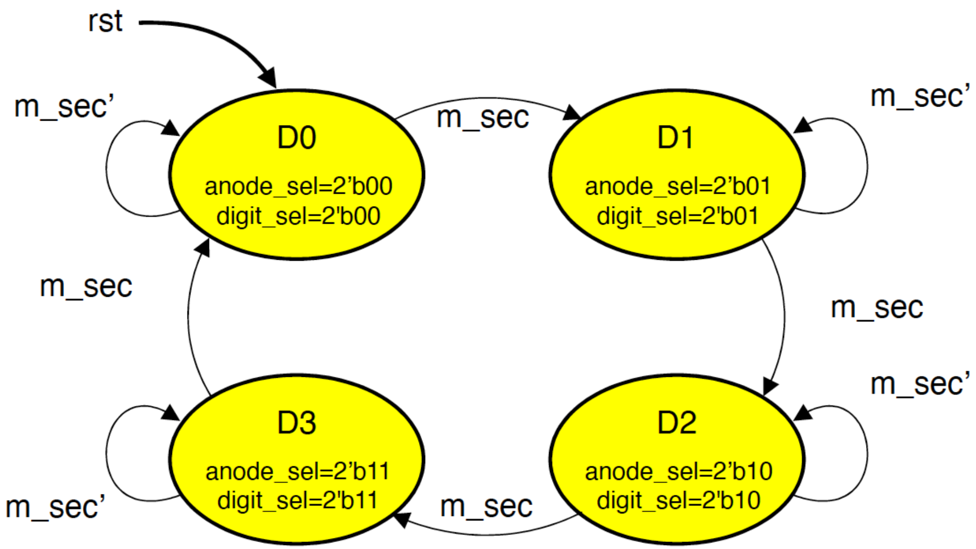
Scholarships | Temple University College of Engineering

FSM - basic example

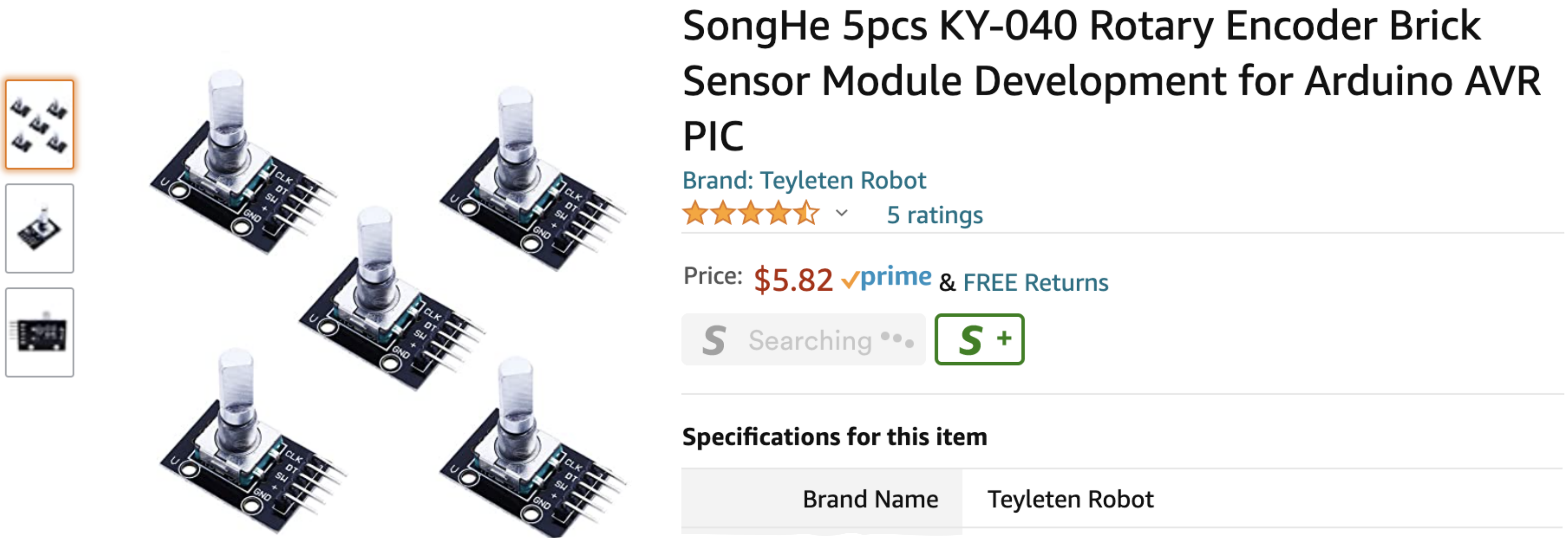


```
module simple_fsm (input a, output reg x);
  reg state, next_state;
  parameter S1=0, S2=1;
  // synchronous block
  always @(posedge clk) begin
    state <= next_state;
  end
  // combinational block
  always @* begin
    // defaults
    next_state = state;
    x = 0;
    // regular logic
    case(state)
      S1: begin
        x = 1;
        if(a==1) begin
          next_state = S2;
        end // end of if
      end // end of S1
      S2: next_state = S1;
    endcase
    // priority logic
  end // end of always
endmodule
```

Example



```
module display_fsm (input m_sec, input
rst, output reg [1:0] anode_sel, output
reg [1:0] digit_sel);
  reg  state, next_state;
  parameter S1=0, S2=1;
  // synchronous block
  always @(posedge clk) begin
    state <= next_state;
  end
  // combinational block
  always @* begin
    // defaults
    next_state = state;
    x = 0;
    // regular logic
    case(state)
      S1: begin
```



SongHe 5pcs KY-040 Rotary Encoder Brick Sensor Module Development for Arduino AVR PIC

Brand: Teyleten Robot

★★★★☆ 5 ratings

Price: $5.82 ✓prime & FREE Returns

S Searching S +

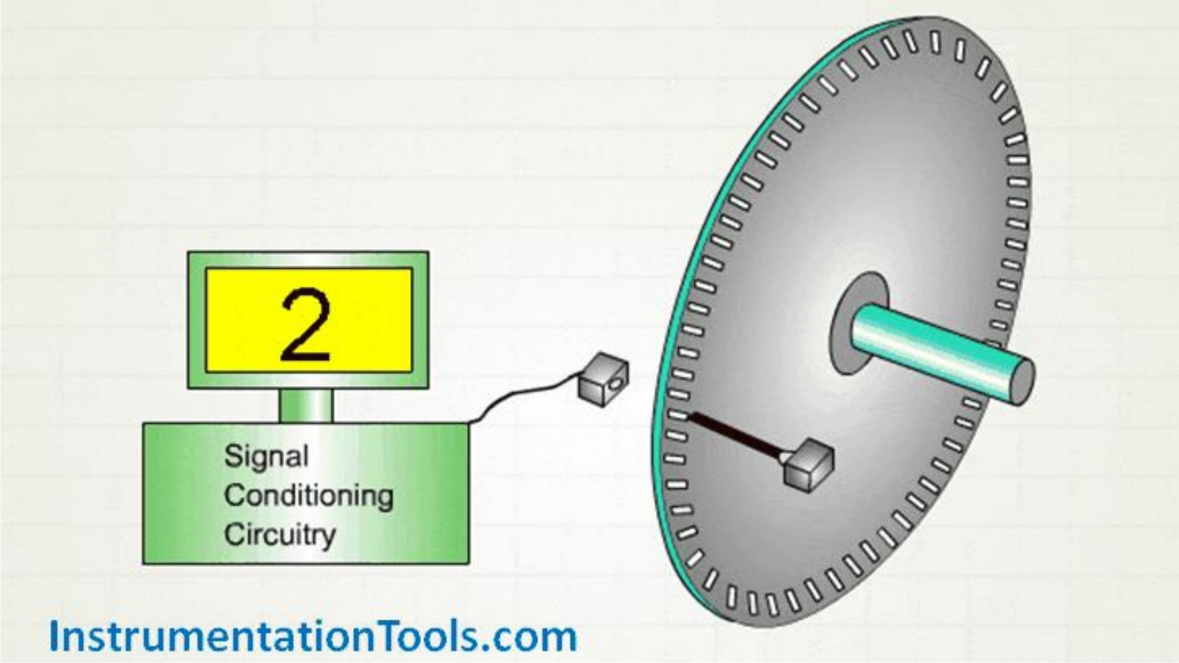
Specifications for this item

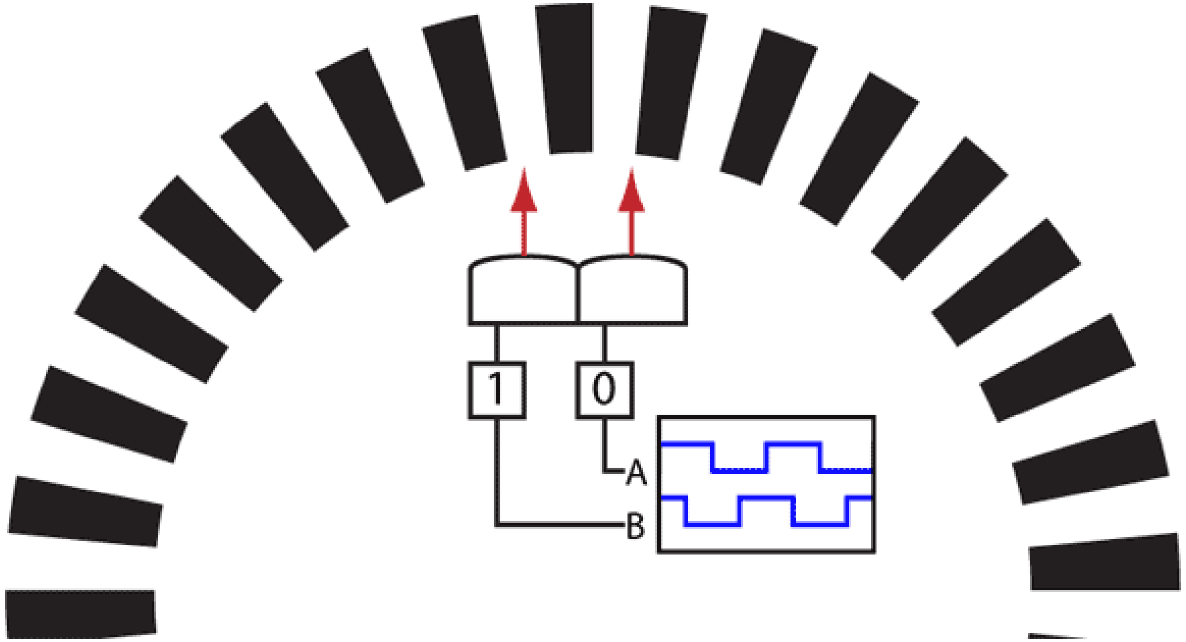
Brand Name	Teyleten Robot

Qudrature Rotary Encoder

https://www.amazon.com/SongHe-KY-040-Encoder-Development-Arduino/dp/B087ZQLLWQ

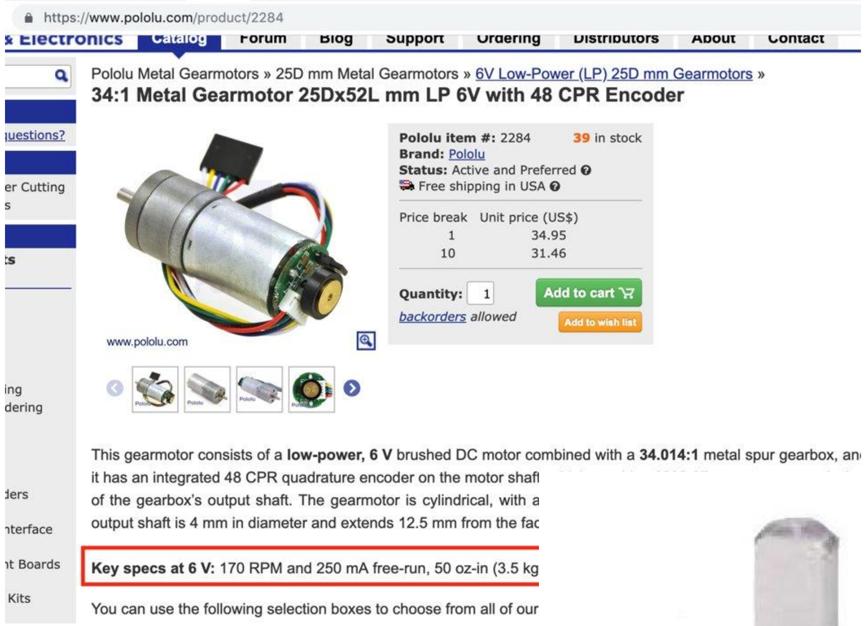
Rotary Encoder





http://www.creative-robotics.com/quadrature-intro

Rotatory Encoder Type



https://www.pololu.com/product/2284

s Electronics | Catalog | Forum | Blog | Support | Ordering | Distributors | About | Contact

questions?

er Cutting
s

ts

ing
dering

ders

nterface

nt Boards

Kits

Pololu Metal Gearmotors » 25D mm Metal Gearmotors » 6V Low-Power (LP) 25D mm Gearmotors »

34:1 Metal Gearmotor 25Dx52L mm LP 6V with 48 CPR Encoder

www.pololu.com

Pololu item #: 2284 **39** in stock
Brand: Pololu
Status: Active and Preferred
Free shipping in USA

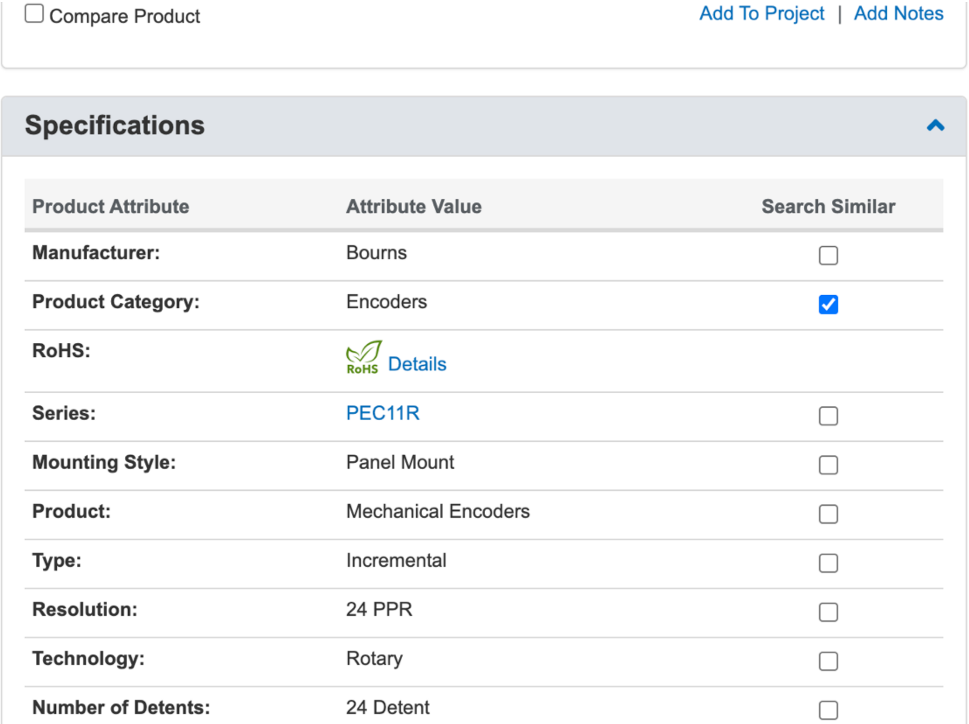
Price break	Unit price (US$)
1	34.95
10	31.46

Quantity: 1 Add to cart
backorders allowed Add to wish list

This gearmotor consists of a **low-power, 6 V** brushed DC motor combined with a **34.014:1** metal spur gearbox, an
it has an integrated 48 CPR quadrature encoder on the motor shaft
of the gearbox's output shaft. The gearmotor is cylindrical, with a
output shaft is 4 mm in diameter and extends 12.5 mm from the fac

Key specs at 6 V: 170 RPM and 250 mA free-run, 50 oz-in (3.5 kg

You can use the following selection boxes to choose from all of our



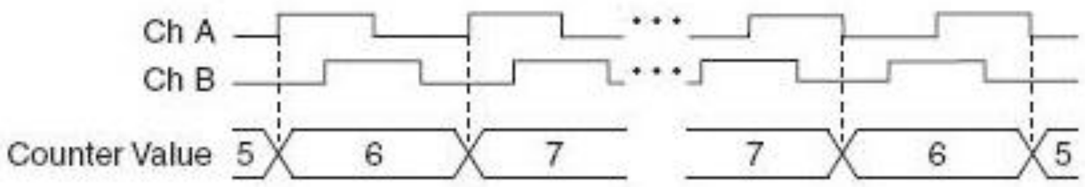
□ Compare Product   Add To Project | Add Notes

Specifications

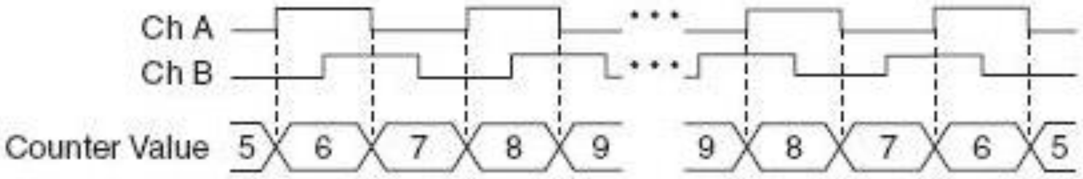
Product Attribute	Attribute Value	Search Similar
Manufacturer:	Bourns	□
Product Category:	Encoders	☑
RoHS:	RoHS Details	
Series:	PEC11R	□
Mounting Style:	Panel Mount	□
Product:	Mechanical Encoders	□
Type:	Incremental	□
Resolution:	24 PPR	□
Technology:	Rotary	□
Number of Detents:	24 Detent	□

Types of quadrature encoding

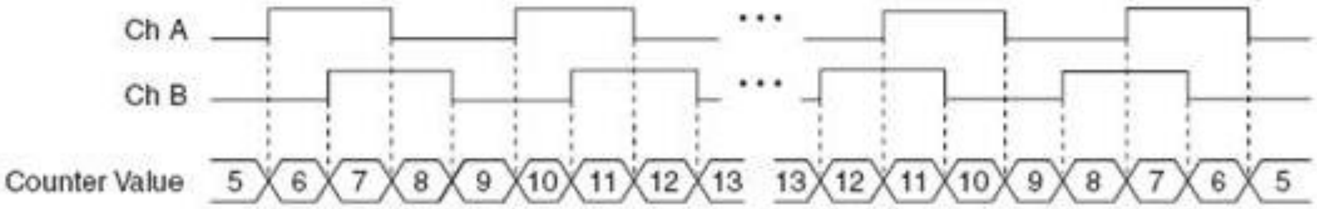
X1:



X2:

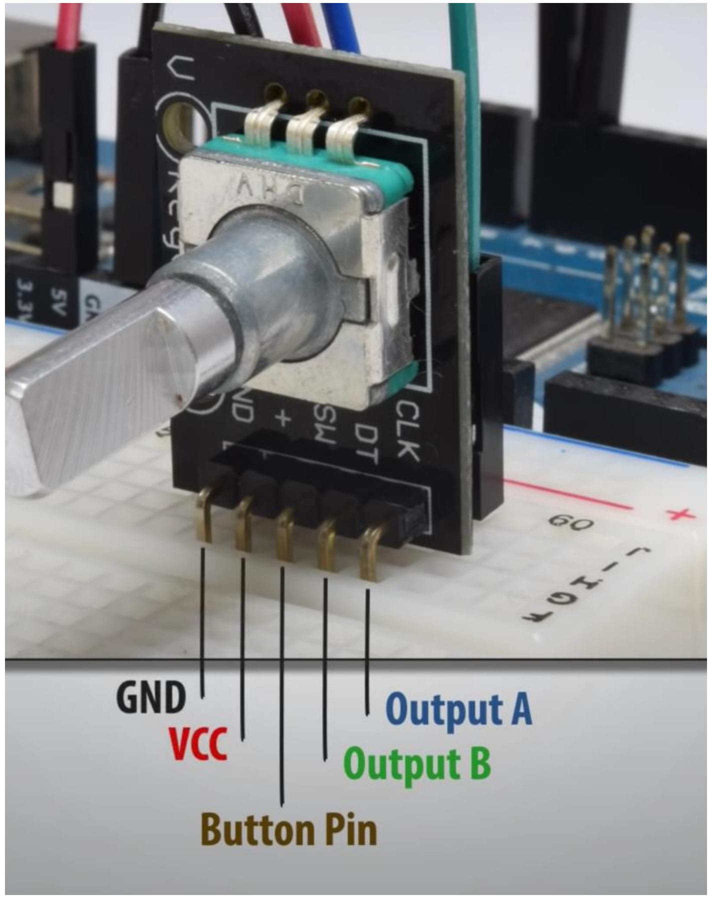


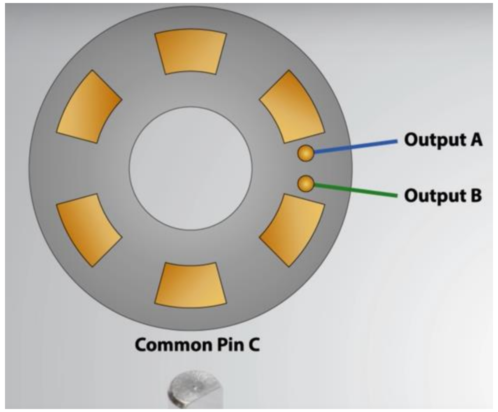
X4:

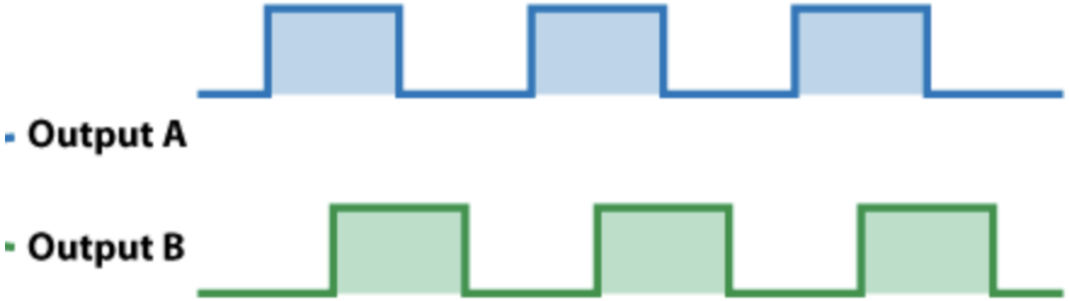


https://www.linearmotiontips.com/what-is-quadrature-encoding/

Connection of QD Rotary Encoder

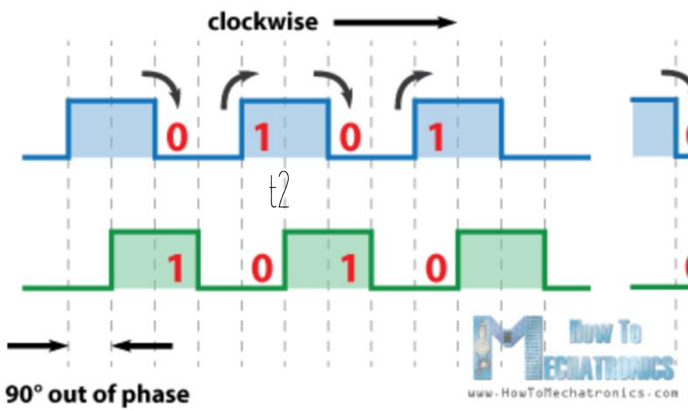


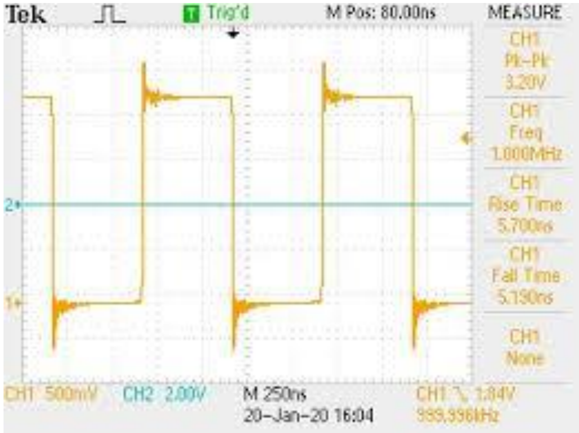




https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

How to count (x4)...





State [AB]	Next State [AB]	action
10	11	increment
11	01	increasement
01	00	increasement
00	10	increasement

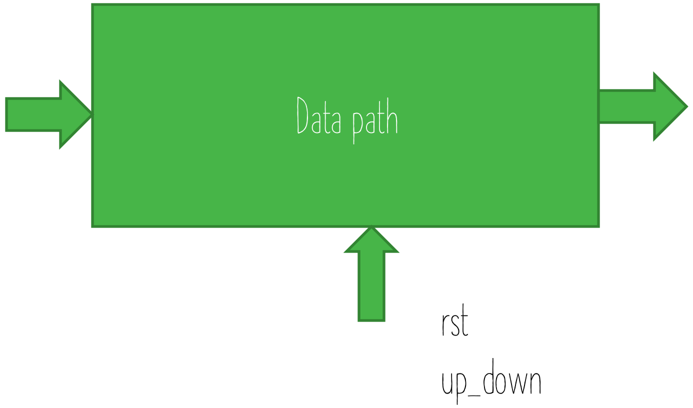
State [AB]	Next State [AB]	action
		decrement
		decrement
		decrement
		decrement

State [AB]	Next State [AB]	action
		hold
		hold
		error
		error

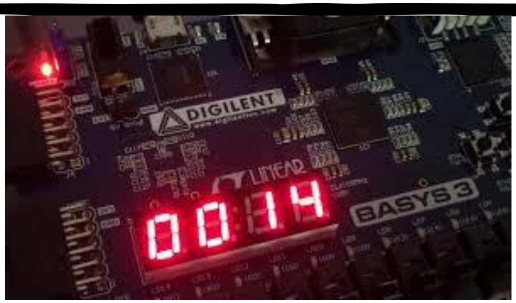
List all conditions

q_prev_a	q_prev_b	q_now_a	q_now_b	quad_ctl	function
0	0	0	0	2'b00	Hold
0	1	0	1		
1	0	1	0		
1	1	1	1		
0	0	0	1	2'b01	Count up
0	1	1	1		
1	1	1	0		
1	0	0	0		
0	0	1	0	2'b10	Count down
1	0	1	1		
1	1	0	1		
0	1	0	0		
0	0	1	1	2'b11	Error
0	1	1	0		
1	0	0	1		
1	1	0	0		

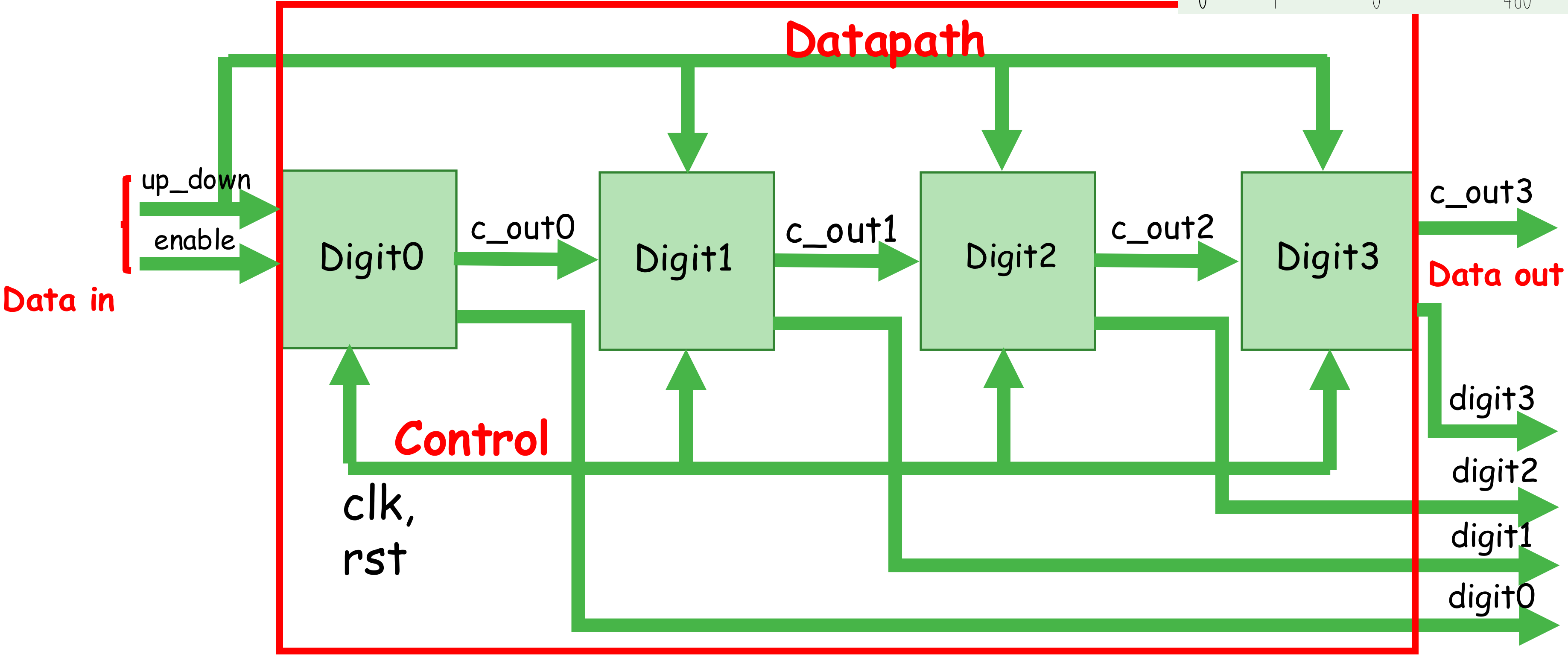
rst	carry_in	up_down	bcd	next_bcd	carry_out	Function
1	x	x	x	4'd0	x	Reset
0	0	x	4'd0 to 9	hold bcd	0	Hold
0	1	1	4'd0 to 8	bcd + 1	0	Count up
0	1	1	4'd9	4'd0	1	
0	1	0	4'd1 to 9	bcd – 1	0	Count down
0	1	0	4'd0	4'd9	1	



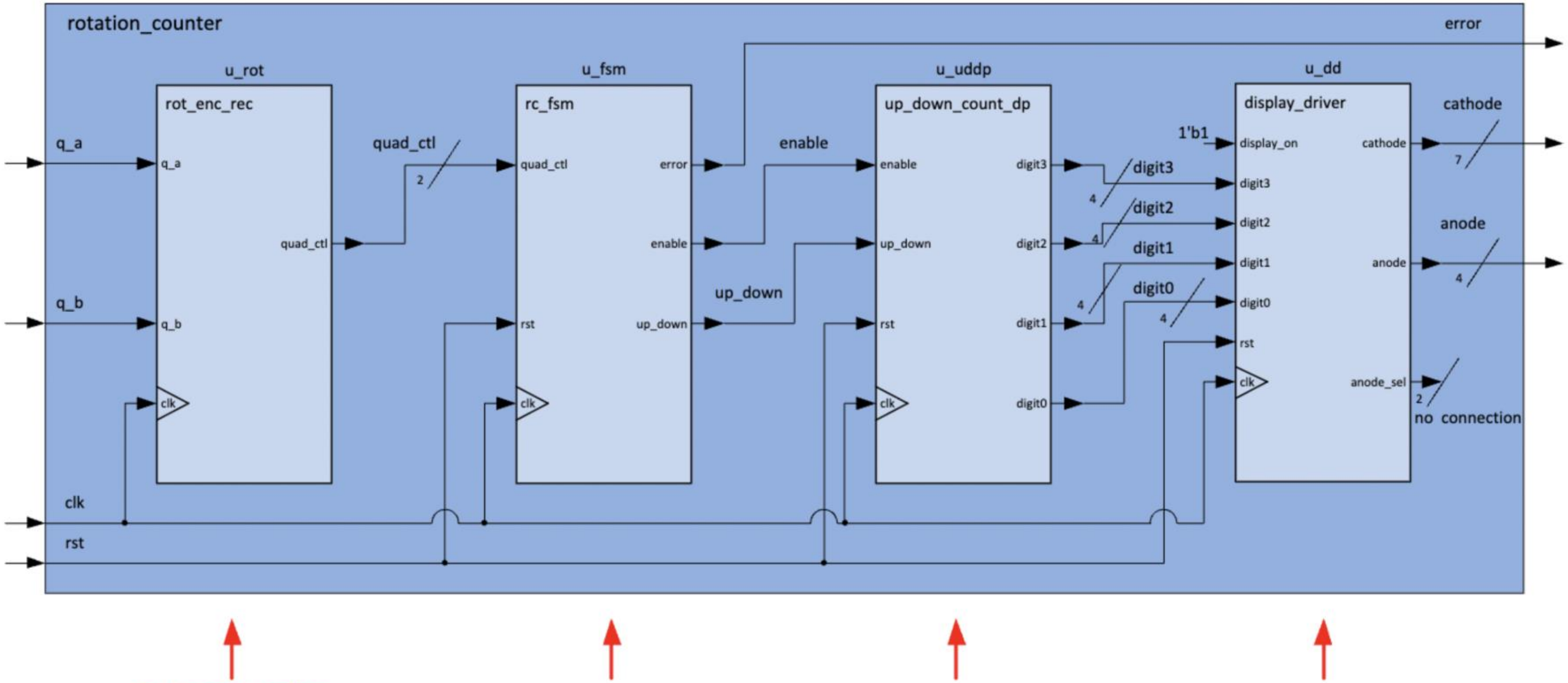
Section 2: Design for up_down_count_dp



rst	carry_in	up_down	bcd	next_bcd	carry_out	Function
1	x	x	x	4'd0	x	Reset
0	0	x	4'd0 to 9	hold bcd	0	Hold
0	1	1	4'd0 to 8	bcd + 1	0	Count up
0	1	1	4'd9	4'd10	1	
0	1	0	4'd1 to 9	bcd - 1	0	Count down
0	1	0	4'd0	4'd9	1	



Recall the lab7



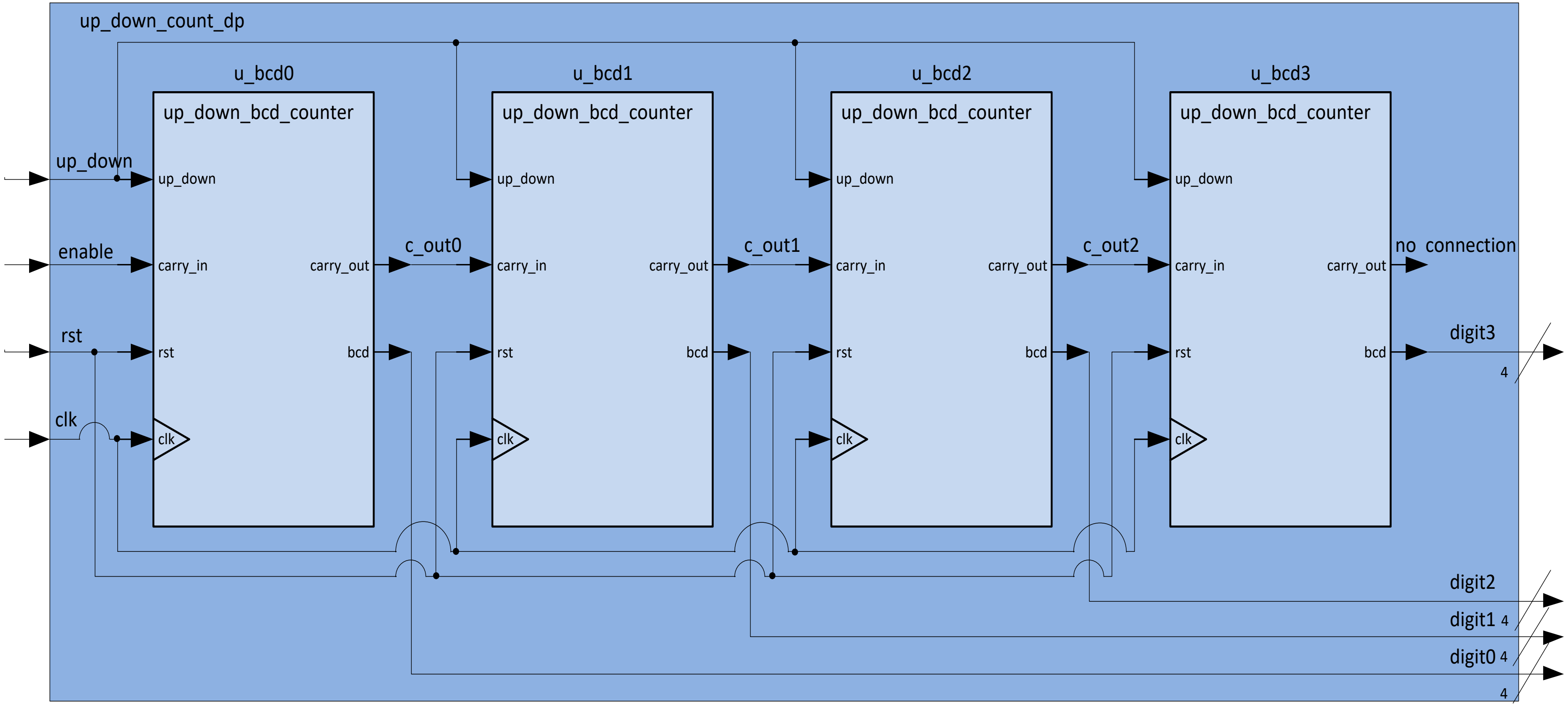
Rotary Encoder Re-coder

Controller

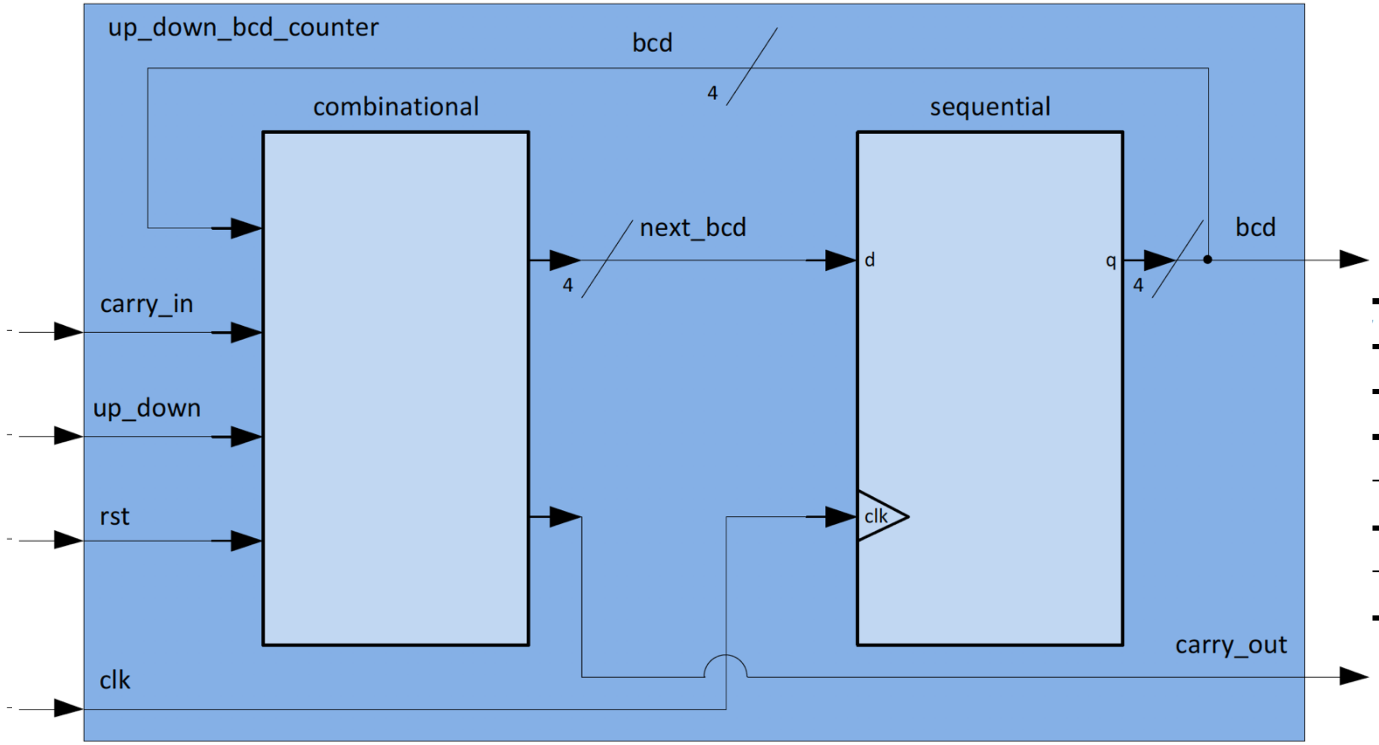
Datapath

(from prev. lab)

Create a datapath



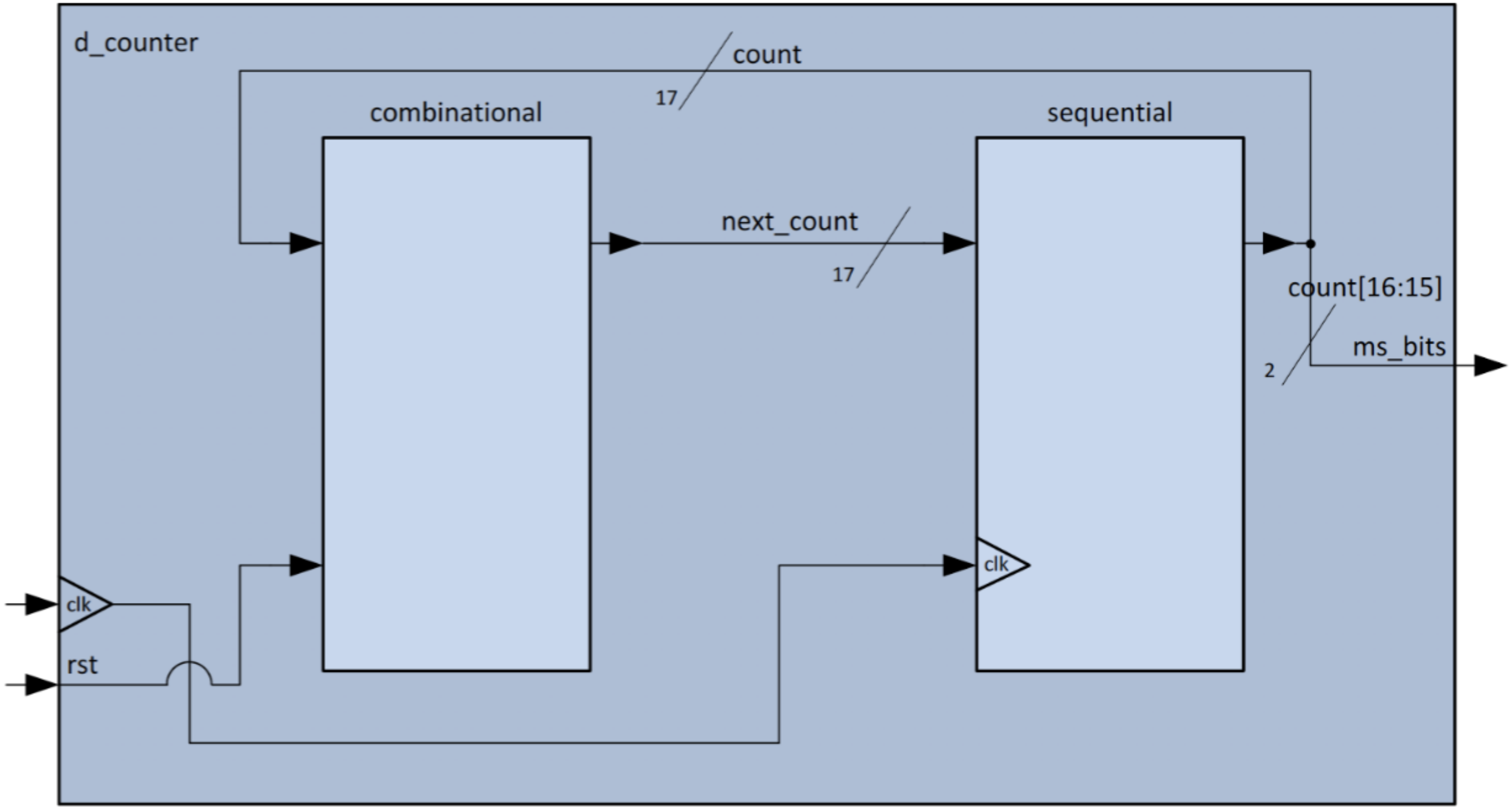
Design for up_down_bcd_counter



Bcd_counter combinational logic

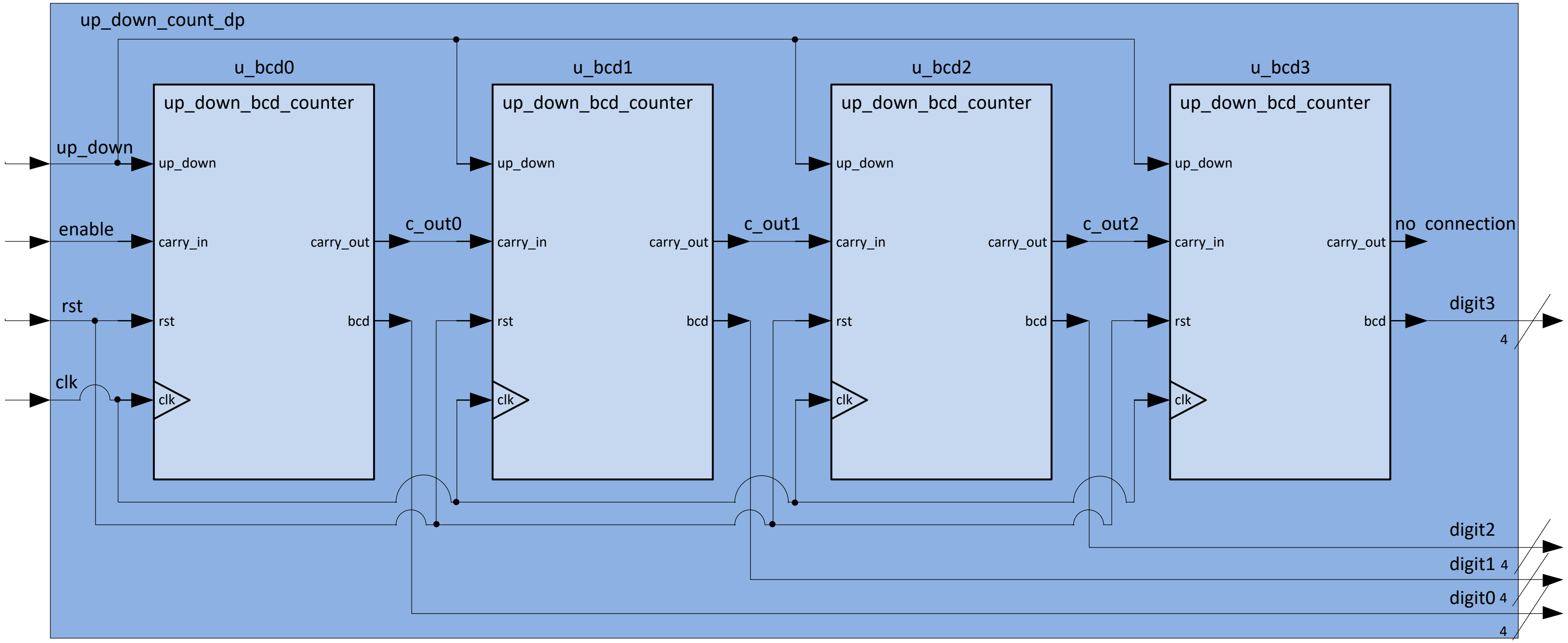
carry_in	*up_down*	*bcd*	*next_bcd*	*carry_out*	**Function**
x	x	x	4'd0	x	*Reset*
0	x	4'd0 to 9	hold bcd	0	*Hold*
1	1	4'd0 to 8	bcd + 1	0	*Count up*
1	1	4'd9	4'd10	1	
1	0	4'd1 to 9	bcd - 1	0	*Count down*
1	0	4'd0	4'd9	1	

Counter Block Diagram

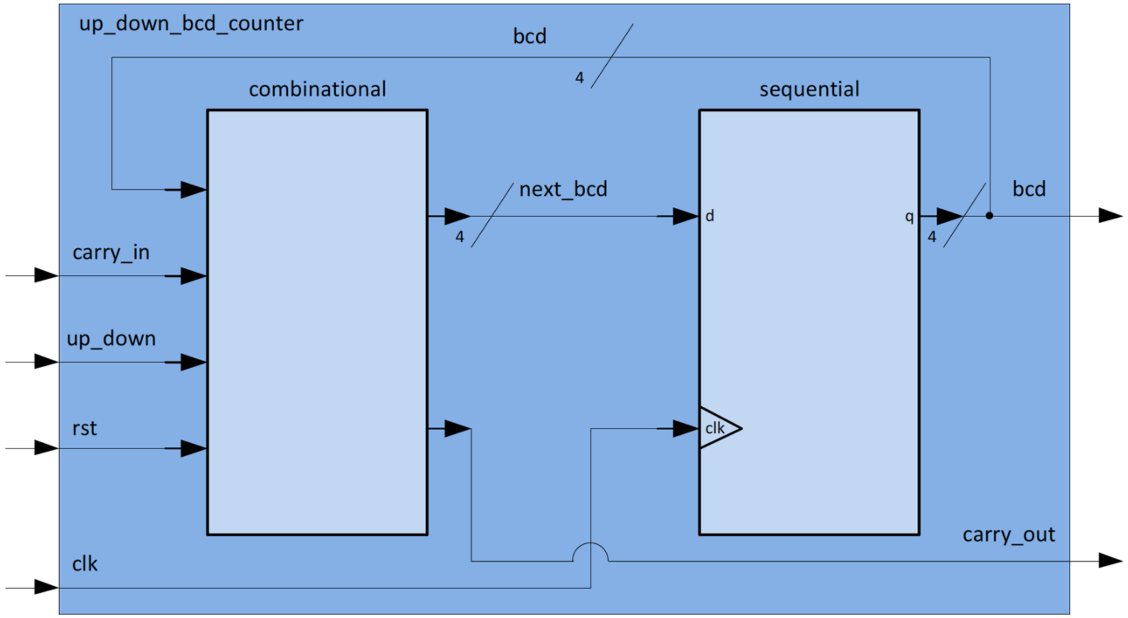


```
// Code your design here
`timescale 1ns/1ns
module cd_counter(output reg ms_bits, input wire clk, input wire rst);

  reg [16:0] count, next_count;

  // synchronous logic
  always @(posedge clk) begin
    count <= next_count;
  end

  // combinational logic
  always @* begin
    // defaults
   // count down
    next_count = count - 1;
   // priority logic
    if (rst == 1) next_count = 0;
  end

  assign ms_bits=count[16:15];
endmodule
```

Stopwatch_dp



Create stop watch



Think about 1 second watch with 50 MHz clock

Think about 1 second watch with 50 MHz clock

Onenote