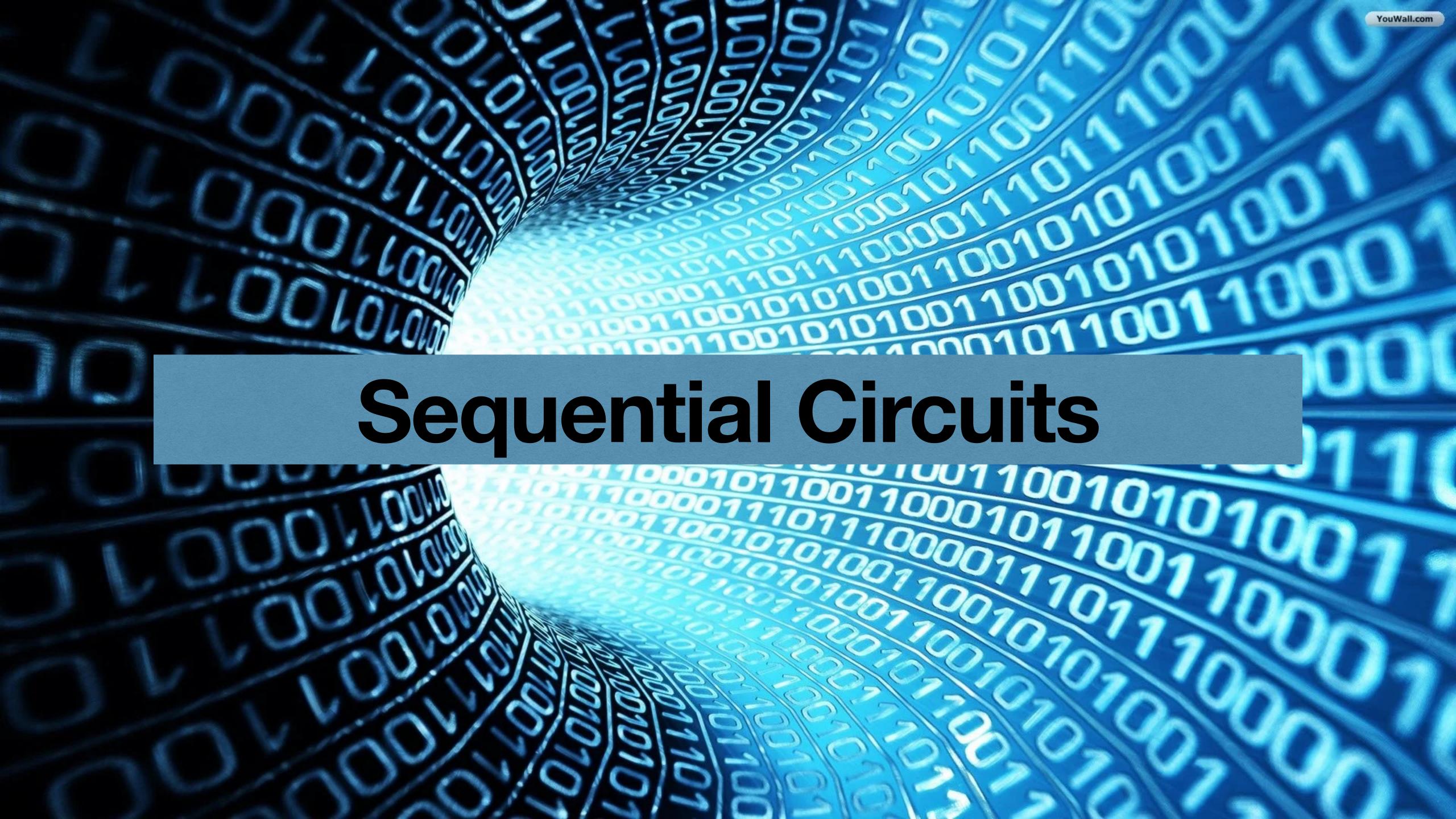
# Digital Logic Design + Computer Architecture

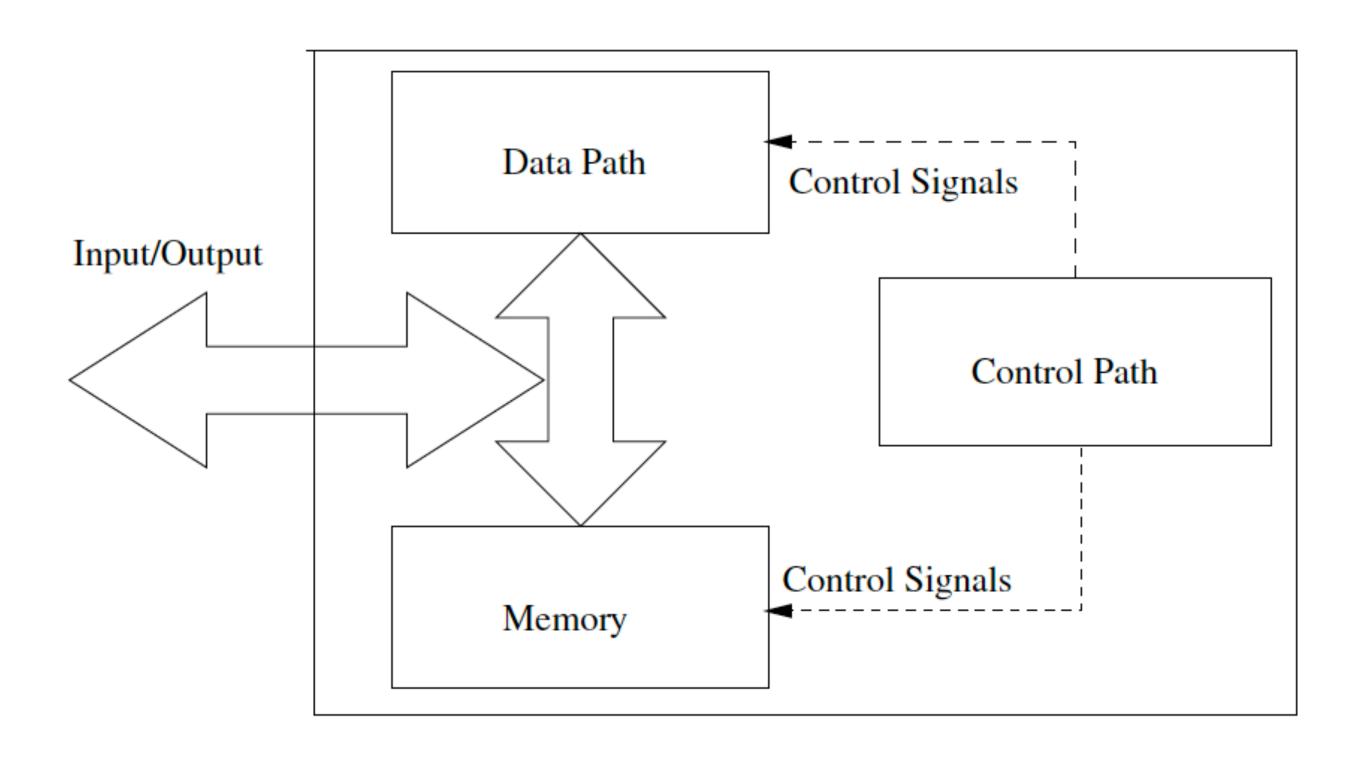
Sayandeep Saha

Assistant Professor
Department of Computer
Science and Engineering
Indian Institute of Technology
Bombay





## General View of a Hardware: Once Again



## Binary GCD Algorithm

```
Input: Integers u and v
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
1 while (u!=v) do
      if u and v are even then
         z = 2gcd(u/2, v/2)
 \bf 3
      end
      else if (u is odd and v is even) then
 5
         z = gcd(u, v/2)
 6
      end
 7
      else if (u is even and v is odd) then
 8
          z = gcd(u/2, v)
 9
      end
10
      else
11
         if (u \geq v) then
12
             z = \gcd((u - v)/2, v)
13
         end
14
          else
15
    z = \gcd(u, (v - u)/2)
      \mathbf{end}
17
      \mathbf{end}
18
19 end
```

## Binary GCD Algorithm: Closer to Hardware

26 end

```
Input: Integers u and v
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
 1 while (u!=v) do
      if u and v are even then
         z = 2gcd(u/2, v/2)
      end
      else if (u is odd and v is even) then
         z = gcd(u, v/2)
      end
      else if (u is even and v is odd) then
         z = gcd(u/2, v)
 9
      end
10
      else
11
         if (u \geq v) then
12
             z = \gcd((u - v)/2, v)
13
         end
14
         else
15
             z = \gcd(u, (v - u)/2)
16
         end
17
      end
18
19 end
```

```
Input: Integers u and v
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
 1 register X_R, Y_R;
 2 X_R = u; Y_R = v; count = 0;
 3 while (X_R! = Y_R) do
       if (X_R[0] = 0 \text{ and } Y_R[0] = 0) then
           X_R = \text{right shift}(X_R)
           Y_R = \text{right shift}(Y_R)
           count = count + 1
       end
       else if (X_R[0] = 1 \text{ and } Y_R[0] = 1) then
           Y_R = \text{right shift}(Y_R)
10
       end
11
       else if (X_R[0] = 0 \text{ and } Y_R[0] = 1) then
12
           X_R = \text{right shift}(X_R)
13
       \mathbf{end}
14
       else
15
           if (X_R \ge Y_R) then
16
               X_R = \text{right shift}(X_R - Y_R)
17
           end
18
           else
19
               Y_R = \text{right shift}(Y_R - X_R)
20
           end
\bf 21
22 end
23 while (count > 0) do
       X_R = \text{left shift}(X_R)
       count = count - 1
```

## Binary GCD Algorithm: Closer to Hardware Input: Integers u and v

26 end

```
Output: Greatest Common Divisor of u and v: z = gcd(u, v)
   Input: Integers u and v
                                                                                        1 register X_R, Y_R;
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
                                                                                        2 X_R = u; Y_R = v; count = 0;
 1 while (u!=v) do
                                                 How to interpret as hardware?
                                                                                        3 while (X_R! = Y_R) do
       if u and v are even then
                                                                                        if (X_R[0] = 0 \text{ and } Y_R[0] = 0) then
           z = 2gcd(u/2, v/2)
                                                                                               \longrightarrow X_R = \text{right shift}(X_R)
       end
                                                                                               Y_R = \text{right shift}(Y_R)
       else if (u is odd and v is even) then
                                                                                                 count = count + 1
                                                                                              end
           z = gcd(u, v/2)
                                                                                              else if (X_R[0] = 1 \text{ and } Y_R[0] = 1) then
       end
                                                                                                  Y_R = \text{right shift}(Y_R)
                                                                                       10
       else if (u is even and v is odd) then
                                                         Keep the
                                                                                              end
                                                                                       11
           z = gcd(u/2, v)
 9
                                                        count of the
                                                                                              else if (X_R[0] = 0 \text{ and } Y_R[0] = 1) then
                                                                                       12
       end
10
                                                         2's getting
                                                                                                  X_R = \text{right shift}(X_R)
                                                                                       13
       else
11
                                                         multiplied
                                                                                              \mathbf{end}
                                                                                       14
           if (u \geq v) then
12
                                                                                              else
                                                                                       15
               z = \gcd((u - v)/2, v)
13
                                                                                                 if (X_R \ge Y_R) then
                                                                                       16
           end
                                                                                                     X_R = \text{right shift}(X_R - Y_R)
14
                                                                                       17
           \mathbf{else}
                                                                                                  end
15
                                                                                       18
               z = \gcd(u, (v - u)/2)
                                                                                                  else
16
                                                                                                     Y_R = \text{right shift}(Y_R - X_R)
           end
17
                                                                                                  end
       end
18
                                                                                       22\end
19 end
                                                                                       23 while (count > 0) do
                                                                                            X_R = \text{left shift}(X_R)
                                                                                             count = count - 1
```

## Constructing the Datapath

```
Input: Integers u and v
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
 1 register X_R, Y_R;
 2 X_R = u; Y_R = v; count = 0;
 3 while (X_R! = Y_R) do
       if (X_R[0] = 0 \text{ and } Y_R[0] = 0) then
          X_R = \text{right shift}(X_R)
          Y_R = \text{right shift}(Y_R)
          count = count + 1
       end
       else if (X_R[0] = 1 \text{ and } Y_R[0] = 1) then
           Y_R = \text{right shift}(Y_R)
10
       end
11
       else if (X_R[0] = 0 \text{ and } Y_R[0] = 1) then
\bf 12
           X_R = \text{right shift}(X_R)
13
       end
14
       else
15
           if (X_R \ge Y_R) then
16
               X_R = \text{right shift}(X_R - Y_R)
17
           end
18
           else
19
               Y_R = \text{right shift}(Y_R - X_R)
20
           end
22 end
23 while (count > 0) do
       X_R = \text{left shift}(X_R)
       count = count - 1
26 end
```

#### Required hardware components

• Two Registers for holding u and v (sequential)

#### Datapath elements

- Subtractor
- Complementer
- Right shifter
- Left shifter
- Counter (sequential)
- Multiplexors to route the control signals

## Constructing the Controller

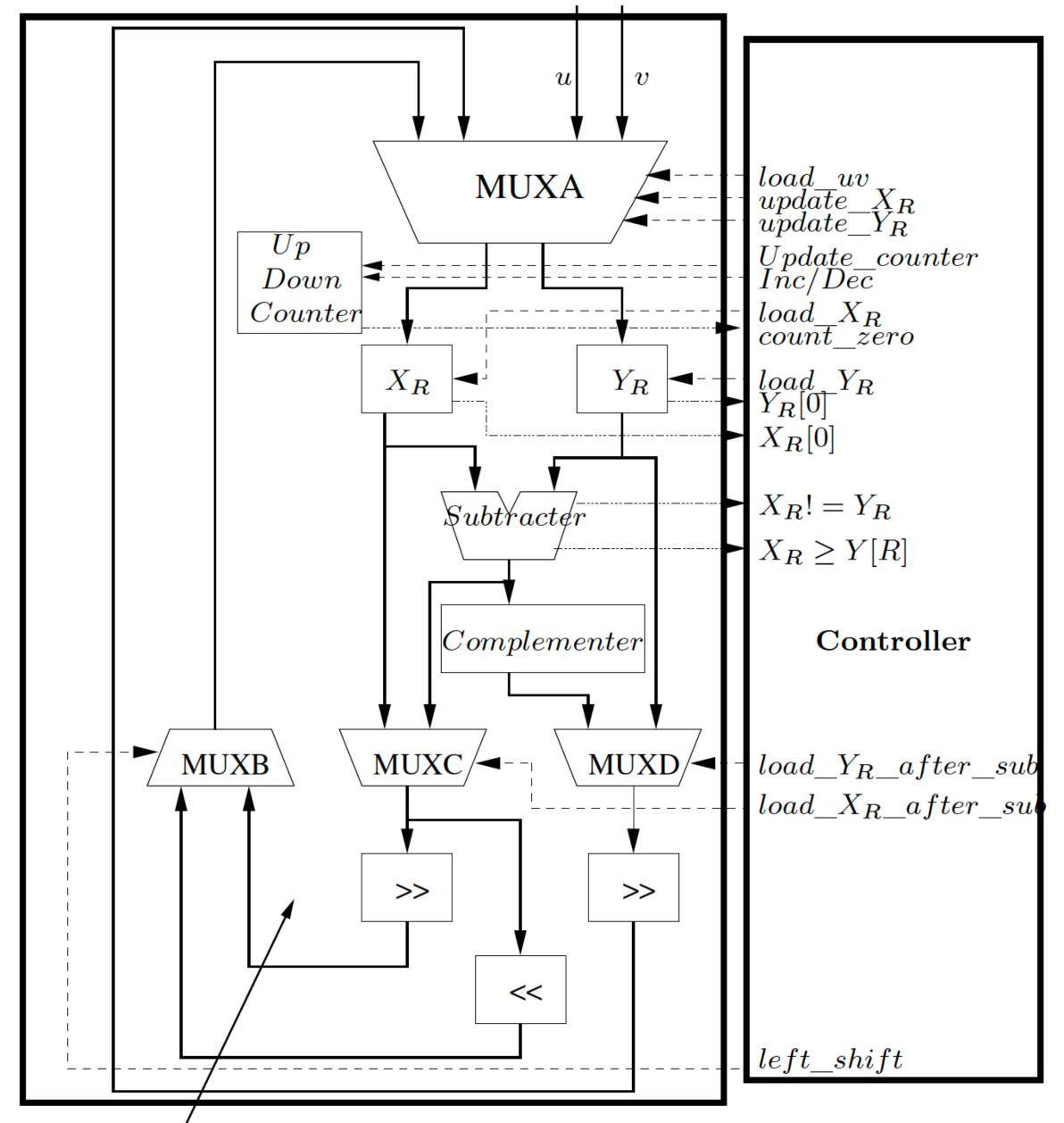
```
1 register X_R, Y_R;
 2 X_R = u; Y_R = v; count = 0;
                                                                                    /* State 0 */
 3 while (X_R! = Y_R) do
       if (X_R[0] = 0 \text{ and } Y_R[0] = 0) then
                                                                                    /* State 1 */
           X_R = \text{right shift}(X_R)
           Y_R = \text{right shift}(Y_R)
           count = count + 1
       \mathbf{end}
       else if (X_R[0] = 1 \text{ and } Y_R[0] = 1) then
                                                                                    /* State 2 */
           Y_R = \text{right shift}(Y_R)
10
       end
11
       else if (X_R[0] = 0 \text{ and } Y_R[0] = 1) then
                                                                                    /* State 3 */
\bf 12
           X_R = \text{right shift}(X_R)
13
       end
14
                                                                                    /* State 4 */
       else
15
           if (X_R \geq Y_R) then
16
               X_R = \text{right shift}(X_R - Y_R)
17
           end
18
           else
19
               Y_R = \text{right shift}(Y_R - X_R)
20
           end
\mathbf{21}
22 end
23 while (count > 0) do
                                                                                    /* State 5 */
       X_R = \text{left shift}(X_R)
\bf 24
       count = count - 1
25
```

26 end

#### Observation

- Whenever there is an if-else block, we allocate a state
- Why? Because they dictate the control flow
- The while loop is handled by the counter
- Give the counter value to the controller and it tells you when to stop
- Note: there might be cases when you have no if-else statements, but you have to sequentialize the computation
  - Remember the 4a+b example.
  - Whenever you need to sequentialise, allocate states.

## Constructing the Hardware Diagram



```
Input: Integers u and v
   Output: Greatest Common Divisor of u and v: z = gcd(u, v)
 1 register X_R, Y_R;
 2 X_R = u; Y_R = v; count = 0;
 3 while (X_R! = Y_R) do
       if (X_R[0] = 0 \text{ and } Y_R[0] = 0) then
           X_R = \text{right shift}(X_R)
           Y_R = \text{right shift}(Y_R)
           count = count + 1
       \mathbf{end}
 8
       else if (X_R[0] = 1 \text{ and } Y_R[0] = 1) then
           Y_R = \text{right shift}(Y_R)
10
       end
11
       else if (X_R[0] = 0 \text{ and } Y_R[0] = 1) then
           X_R = \text{right shift}(X_R)
13
       \mathbf{end}
14
       \mathbf{else}
15
           if (X_R \ge Y_R) then
16
               X_R = \text{right shift}(X_R - Y_R)
17
           end
18
           else
19
               Y_R = \text{right shift}(Y_R - X_R)
20
           end
\bf 21
22 end
23 while (count > 0) do
       X_R = \text{left shift}(X_R)
       count = count - 1
26 end
```

## **Constructing the State Transition Table**

Present	Next State					Output Signals										
State						load	update	update	load	load	$load\_X_R$	$load\_Y_R$	Update	Inc	left	count
	0	100_	110_	101_	111_	uv	$X_R$	$Y_R$	$X_R$	$Y_R$	$after\_sub$		counter	/Dec	shift	zero
$S_0$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	1	0	0	1	1	0	0	0	_	_	_
$S_1$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	0	1	1	1	1	0	0	1	1	_	_
$S_2$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	0	0	1	0	1	0	0	0	_	_	
$S_3$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	0	1	0	1	0	0	0	0	_	_	
$S_4$																
$(X_R \geq Y_R)$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	0	1	0	1	0	1	0	0		_	
$S_4$																
$(X_R < Y_R)$	$S_5$	$S_1$	$S_2$	$S_3$	$S_4$	0	0	1	0	1	0	1	0			_
$S_5$	$S_5$	$S_5$	$S_5$	$S_5$	$S_5$	0	0	0	0	0	0	0	1	0	1	0

#### • Controller receives 4 input signals

- XR != YR
- XR[0]
- YR[0]
- XR >= YR

## Now go and code it down