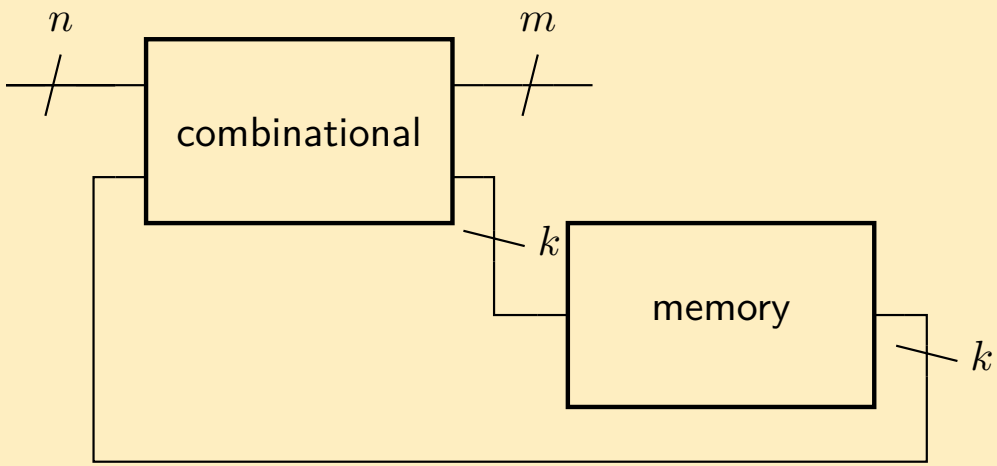
Flipflops

January 29, 2025

Sequential Systems

- A sequential system contains two parts
  - A combinational system
  - Memory
- (Hence, a **computer**!)
- Something like this:

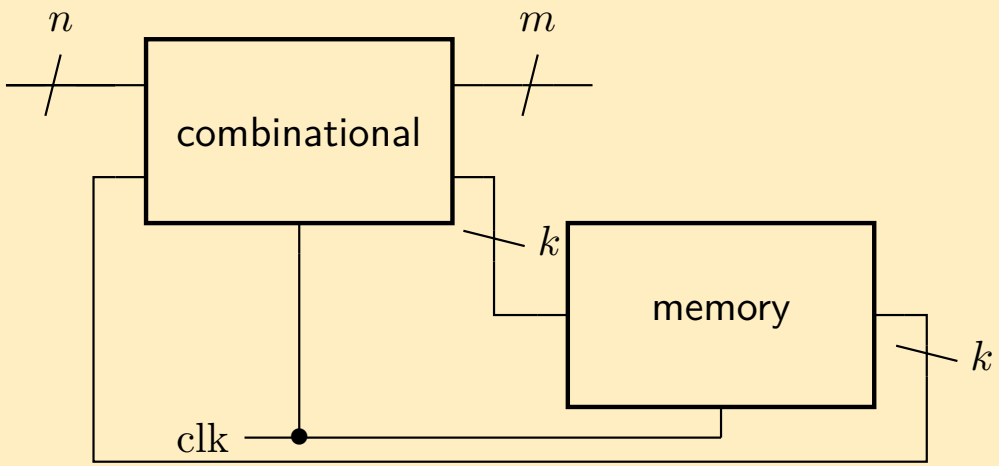
Asynchronous



Sequential Systems

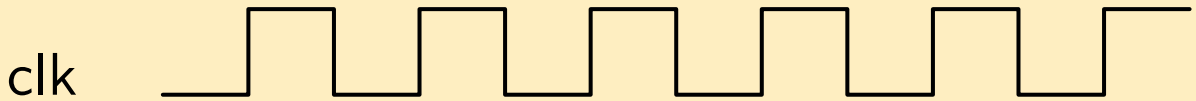
- A sequential system contains two parts
  - A combinational system
  - Memory
- (Hence, a **computer**!)
- Something like this:

Synchronous



Clock

Clock signal



- A Clock generating circuit is **analog**
- We supply the `clock` module
- We have no interest in the `veriloging` of `clock`

Clock testbench I

Listing 1: clock_tb.v

```
‘timescale 1ns/1ns

module clock_tb;
  wire clk;
  clock #(20ns) c(clk);  // 20ns cycle
  initial begin
    $dumpfile("clock.vcd");
    $dumpvars;
    $monitor($time, " ", clk);
    #5000;
    $finish;
  end
endmodule;
```

(Fast) Memory units

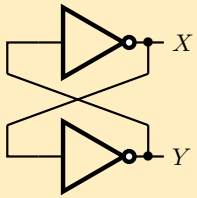
Types

- Latches
- Flipflops

Computer **main** memory uses different comonents

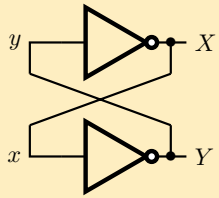
- Latches and flipflops present their value constantly
- Memory chips present their value **only** on demand
  - (and are basically analog)

Feedback loop



How to analyze?

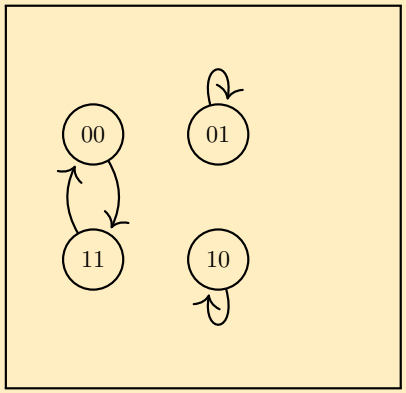
Feedback loop



State table

x	y	X	Y
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0

State Diagram

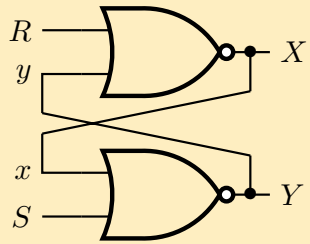


- Stable states yield memory
- No control of the stored value
- Race conditions!

That was **the** Trick

A feedback loop was translated to the combinational method!

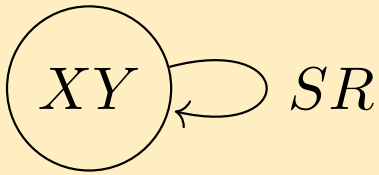
SR-latch

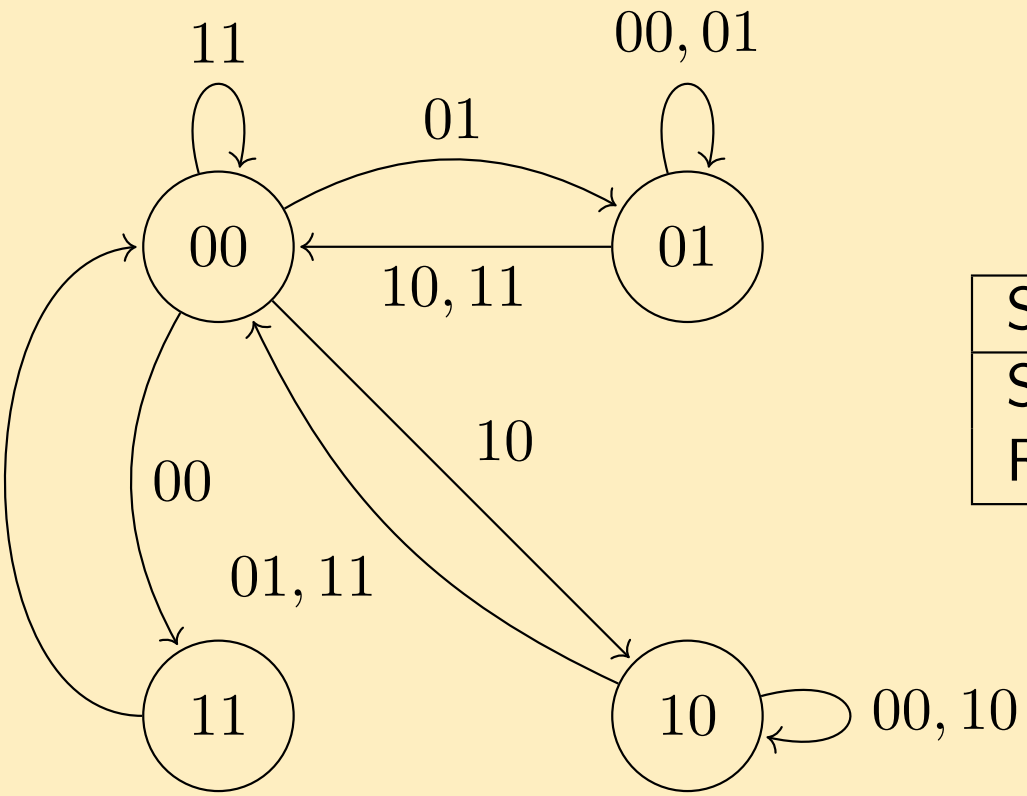


S	R	x	y	X	Y
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	0

S	R	x	y	X	Y
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

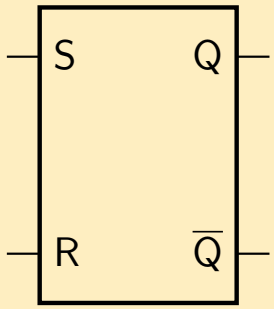
SR-Latch State Diagram





State	Code
Set	10
Reset	01

SR-Latch Component



Characteristic Table

S	R	Q^*
0	0	Q
0	1	0
1	0	1
1	1	??

Characteristic Equation

$$Q^* = S + \bar{R}Q$$

Q^* is arbitrary when returning from $SR = 11$ to $SR = 00$

SR-Latch

- Race conditions can happen
  - The latch is active **all the time**
- Synthesizing circuits is possible
- We will prefer simpler (e.g., D-latch) components

SR Latch I

Listing 2: demo/02_srLatch/srLatch.v

```
`timescale 1ns/1ns

module srLatch(
  output q,
  output qn,
  input s,
  input r
);

  wire n0, n1;
  norGate ng0(n0, r, n1);
  norGate ng1(n1, s, n0);
```

SR Latch II

```
  assign q  = n0;
  assign qn = n1;

endmodule
```

SR Latch I

Listing 3: demo/02_srLatch/srLatch_tb.v

```
`timescale 1ns/1ns

module srLatch_tb;
  logic s,r;
  wire q, qn;

  srLatch sr(q, qn, s ,r);

  initial begin
    logic x,y;

    $dumpfile("srLatch.vcd");
    $dumpvars;
```

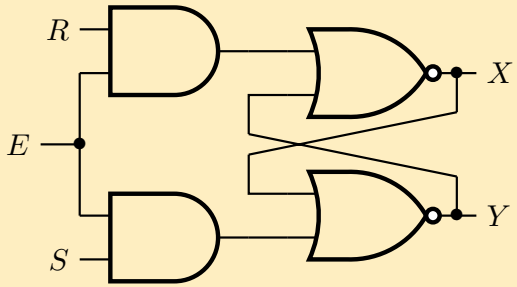
SR Latch II

```
x = 0;
y = 0;

repeat (2) begin
  repeat (2) begin
    s = x;
    r = y;
    #40ns;

    s = 0;
    r = 0;
    #40ns;
    y = y + 1;
```

SR Latch III

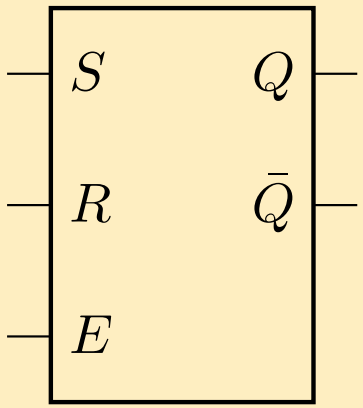
```
        end
        x = x + 1;
    end

    #1000ns;
    $finish;
  end

endmodule
```

Gated SR-Latch



- $E = 0$: Behavior as if $SR = 00$
- $E = 1$: Behavior as a simple SR-latch

Component



Gated SR Latch I

Listing 4: demo/03_sreLatch/srLatch.v

```
`timescale 1ns/1ns

module sreLatch(
  output q,
  output qn,
  input e,
  input s,
  input r
);

  wire se, re;
  andGate as(se, e, s);
  andGate ar(re, e, r);
```

Gated SR Latch II

```
  wire n0, n1;
  norGate ng0(n0, re, n1);
  norGate ng1(n1, se, n0);

  assign q  = n0;
  assign qn = n1;

endmodule
```

Gated SR Latch I

Listing 5: demo/03_sreLatch/sreLatch_tb.v

```
`timescale 1ns/1ns

module sreLatch_tb;
  logic e, s,r;
  wire q, qn;

  sreLatch sre(q, qn, e, s ,r);

  initial begin
    logic x,y;

    $dumpfile("sreLatch.vcd");
    $dumpvars;
```

Gated SR Latch II

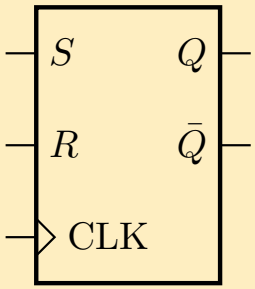
```
x = 0;
y = 0;
e = 0;

repeat (4) begin
  repeat (2) begin
    repeat (2) begin
      s = x;
      r = y;
      #40ns;

      s = 0;
      r = 0;
```

Gated SR Latch III

```
            #40ns;
            y = y + 1;
          end
          x = x + 1;
        end
        e = e + 1;
      end

      #1000ns;
      $finish;
    end

endmodule
```

SR-Flipflop

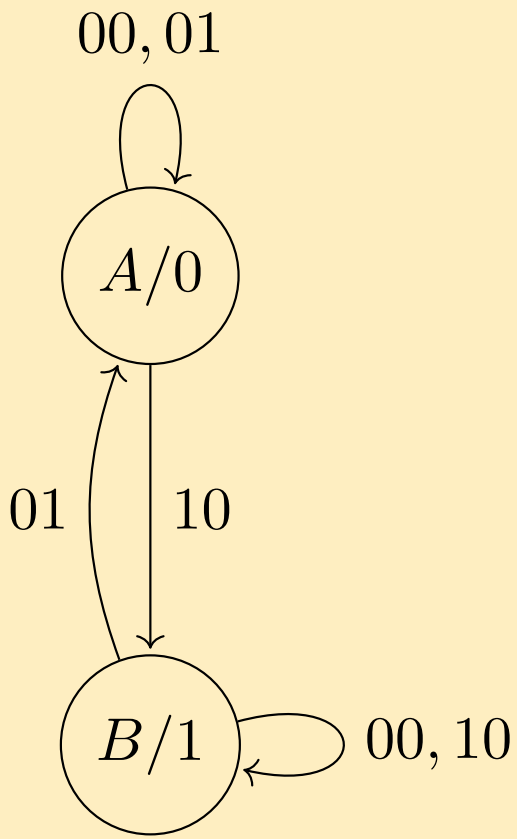
Component



Q can change only on the raising edge of CLK

$\bar{Q}$ is exposed because it **exists** internally

SR-Flipflop State Diagram



SR-Flipflop Characteristic Table and Equation

Characteristic Table

S	R	$Q(t+1)$
0	0	Q
0	1	0
1	0	1
1	1	??

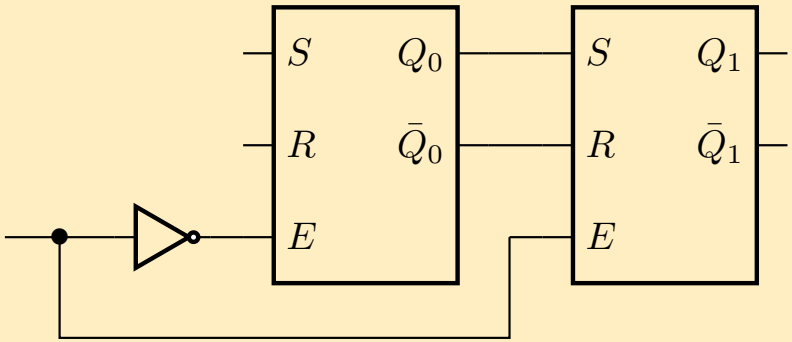
Characteristic Equation

$$Q(t+1) = S + \bar{R}Q$$

when $S = 0$ or $R = 0$

SR-Flipflop Implementation Example

Master-Slave Implementation



Q_1 changes **only** when E switches from 0 to 1!

SR Flipflop I

Listing 6: demo/04_srFF/srFF.v

```
`timescale 1ns/1ns

module srFF(
  output q,
  output qn,
  input s,
  input r,
  input clk
);


  wire clkn;
  notGate n(clkn, clk);
```

SR Flipflop II

```
  wire q1,q1n;
  sreLatch sr1(q1,q1n,clkn,s,r);
  sreLatch sr2(q,qn,clk,q1,q1n);

endmodule
```

SR Flipflop I

Listing 7: demo/04_srFF/srFF_tb.v

```
`timescale 1ns/1ns

module srFF_tb;
  logic s,r;
  wire q, qn;

  wire clk;

  clock #(100) c(clk);

  srFF sr(q, qn, s, r, clk);

  logic x,y,even;
```

SR Flipflop II

```
initial begin

  $dumpfile("srFF.vcd");
  $dumpvars;

   x = 0;
   y = 0;
   s = 0;
   r = 0;
   even = 0;

  #2000ns;
  $finish;
end
```

SR Flipflop III

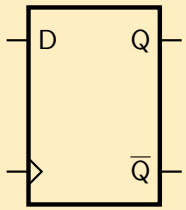
```
always @(posedge clk) begin
  if (even == 0) begin
    s = 0;
    r = 0;
  end else begin
    y = y + 1;
    if (y == 0) begin
      x = x + 1;
      if (x == 0) y = y + 1;
    end
    s = x;
    r = y;
  end
  even = even + 1;
```

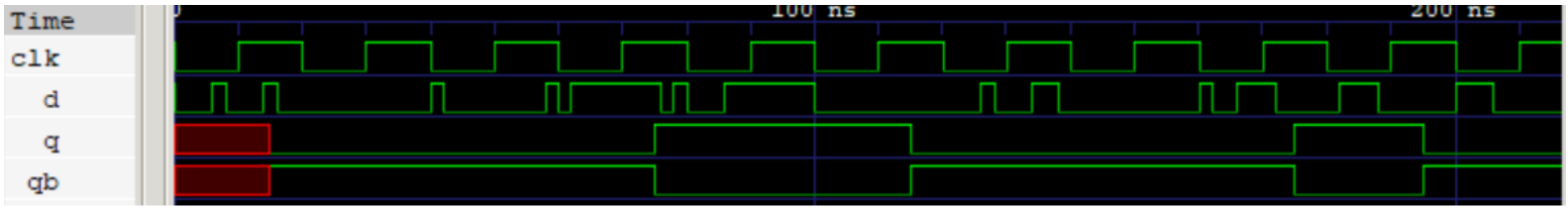
SR Flipflop IV

```
   end

endmodule
```

(Carmi) Lecture 14 reached here

D-Flipflop

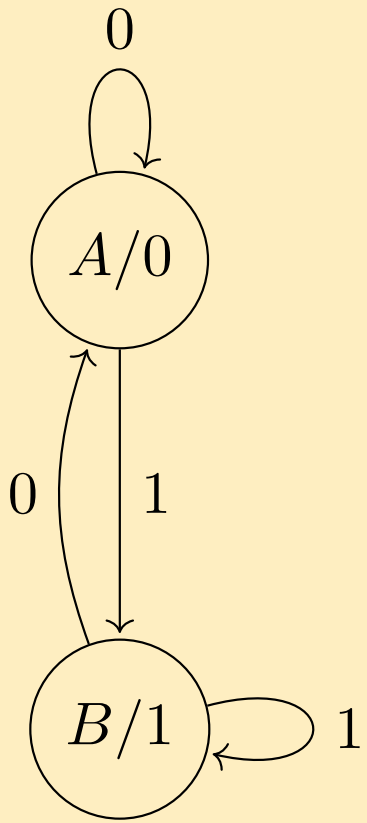




$$Q(t+1) = D$$

- On the raising edge of the clock, D is used.
- Then the flipflop freezes
- This is a very very short time in respect to the cycle time

D-Flipflop State Diagram



D-Flipflop Characteristic Table and Equation

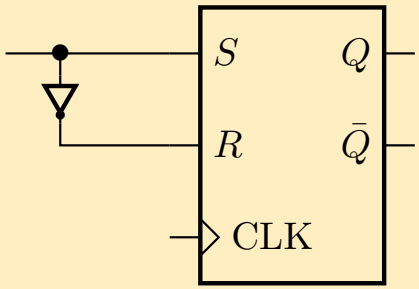
Characteristic Table

D	$Q(t+1)$
0	0
1	1

Characteristic Equation

$$Q(t+1) = D$$

D-Flipflop Implementation Example



D Flipflop I

Listing 8: demo/05_dFF/dFF.v

```
`timescale 1ns/1ns

module dFF(
  output q,
  output qn,
  input d,
  input clk
);


  wire dn;
  notGate n(dn, d);
```

D Flipflop II

```
  srFF sr(q,qn,d,dn,clk);

endmodule
```

D Flipflop I

Listing 9: demo/05_dFF/dFF_tb.v

```
`timescale 1ns/1ns

module dFF_tb;
  logic d;
  wire q, qn;

  wire clk;

  clock #(100) c(clk);

  dFF dff(q, qn, d, clk);

  logic x;
```

D Flipflop II

```
initial begin

  $dumpfile("dFF.vcd");
  $dumpvars;

   d = 0;

  #2000ns;
  $finish;
end

always @(posedge clk) begin
  d = d + 1;
end
```

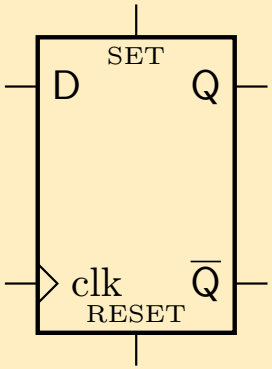
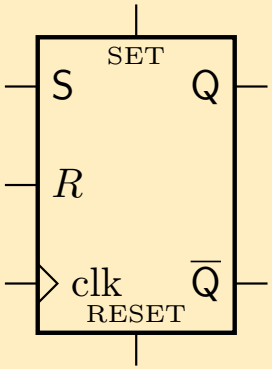
D Flipflop III

```
endmodule
```

(Carmi) Lecture 15 reached here

- Seems as if a lecture was given twice...
- Apparently there is Christmas vacation on Wed.

Library FFs

Component	Verilog
	`dffi(q,qn,d,set,reset,clk)`
	`srffi(q,qn,S,R,set,reset,clk)`