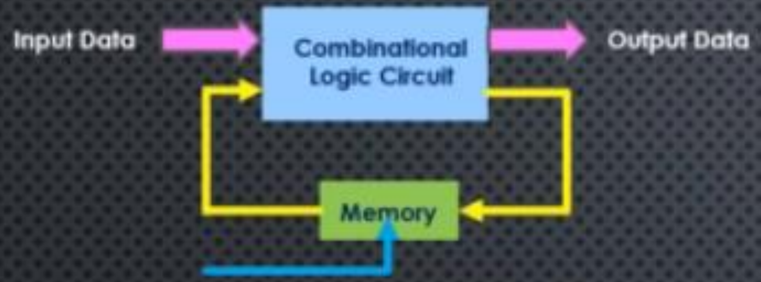
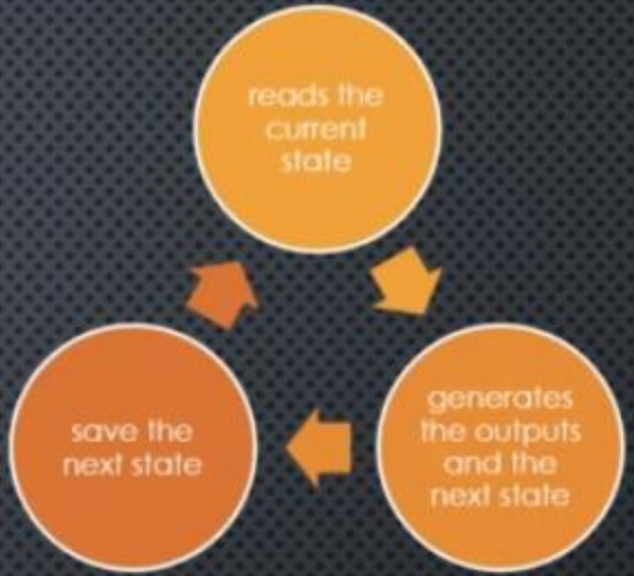
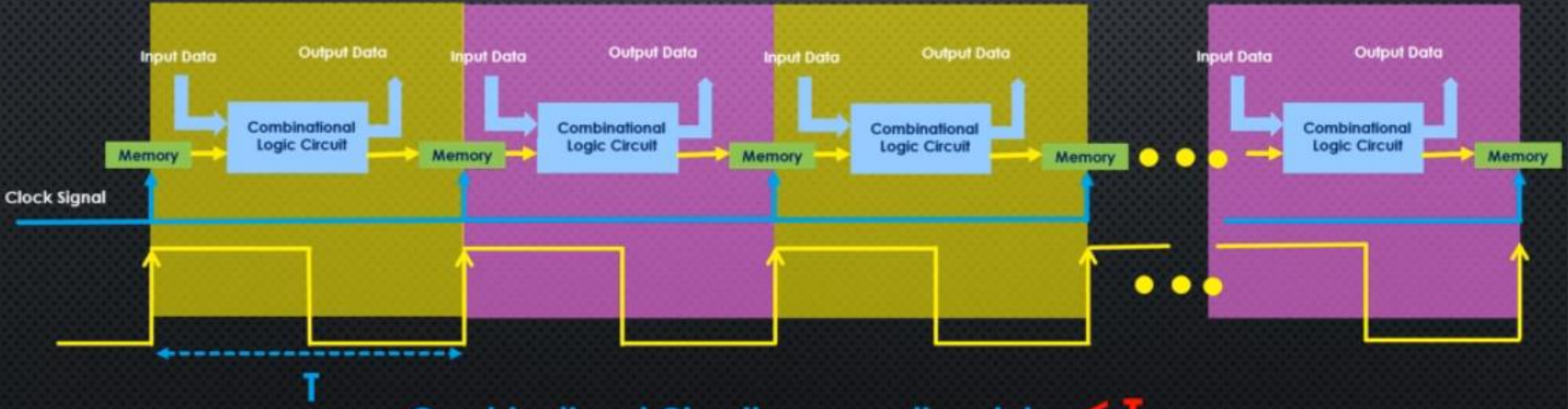
SINGLE CYCLE DESIGN FLOW

Lecture-9

SEQUENTIAL CIRCUIT MODEL

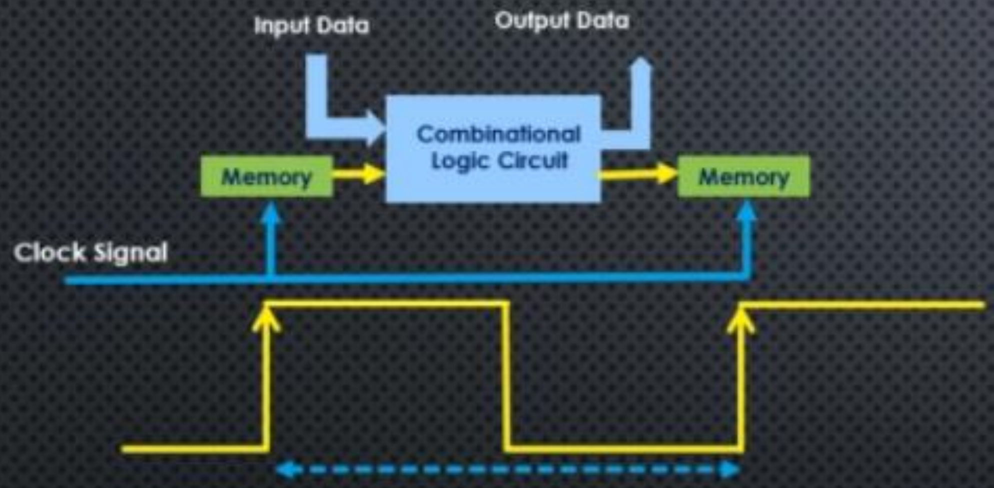






Combinational Circuit propagation delay < T

SEQUENTIAL CIRCUIT DESIGN IDEA



```
void func (arguments) {
      define registers;

      read register values
      perform tasks
      modify register values

}
```

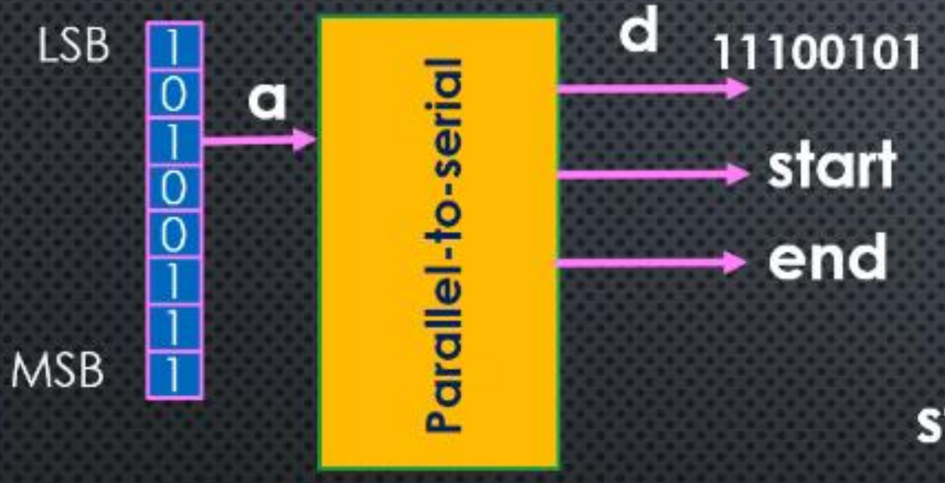
We use static variable definition in C/C++ to declare registers in HLS.

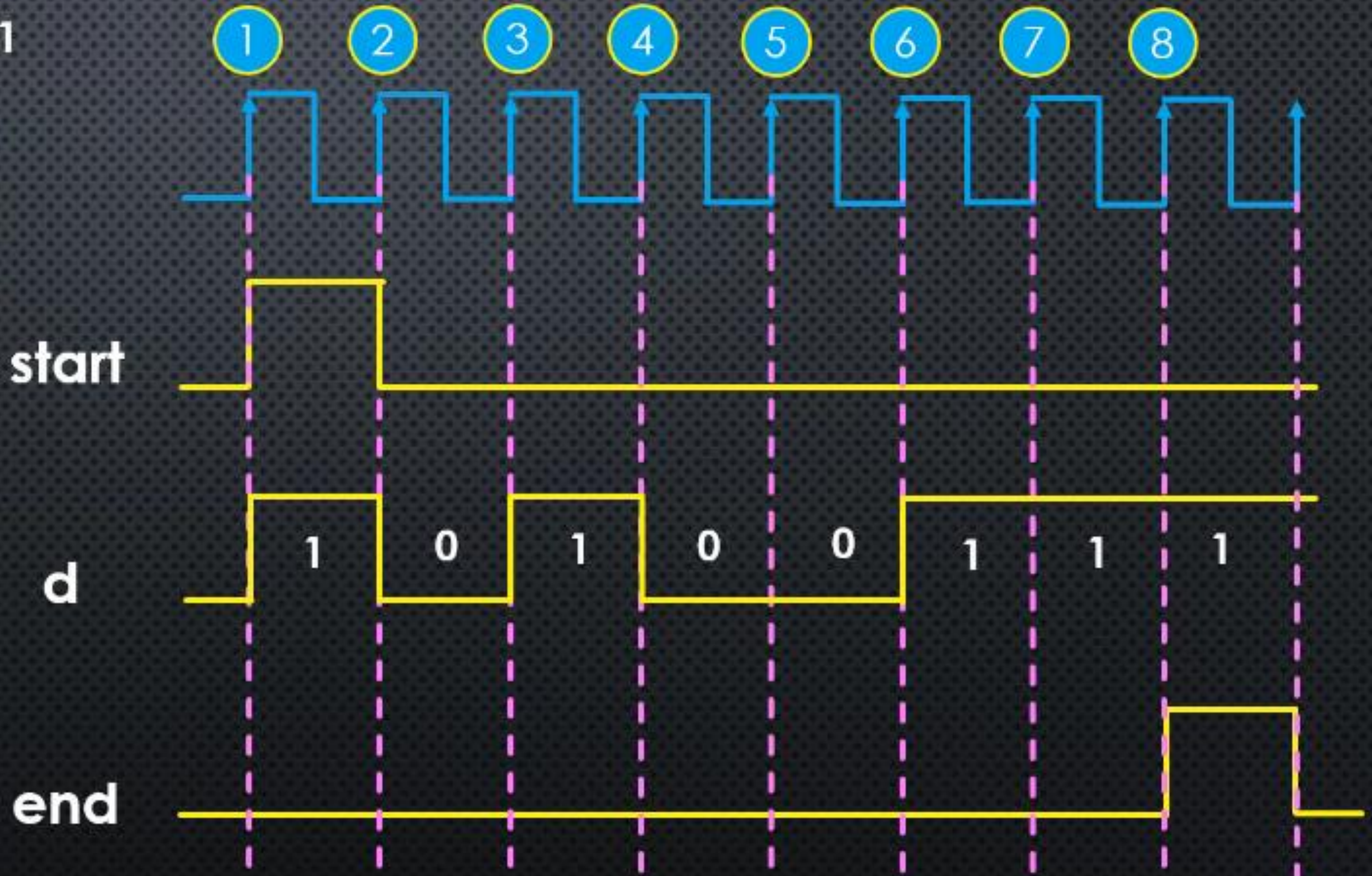
SINGLE-CYCLE DESIGN

The key idea in this section is developing a one-clock-cycle design

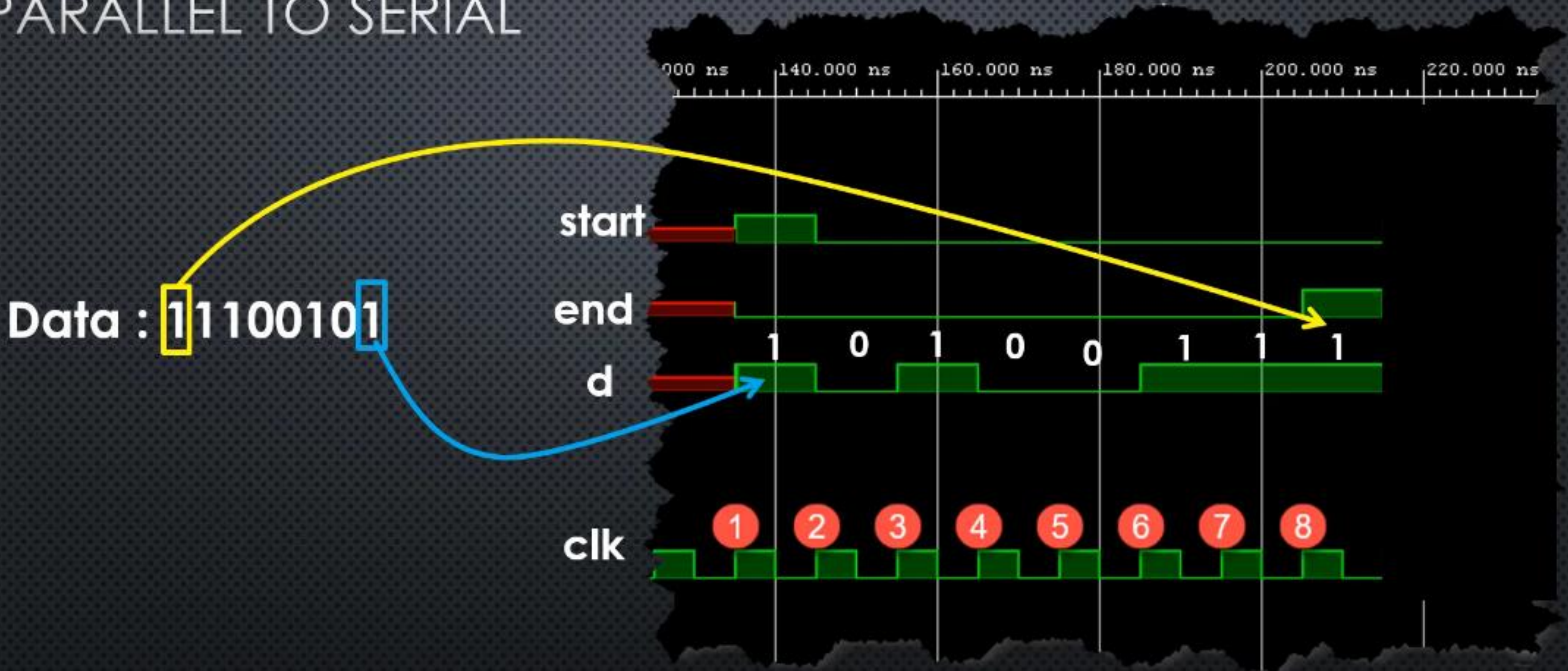
The single-cycle design approach can provide a high-performance hardware comparable with the HDL descriptions.

PARALLEL TO SERIAL

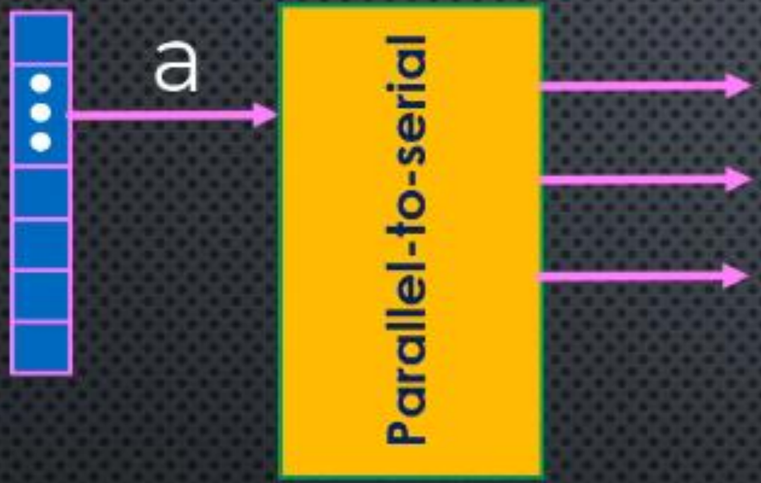


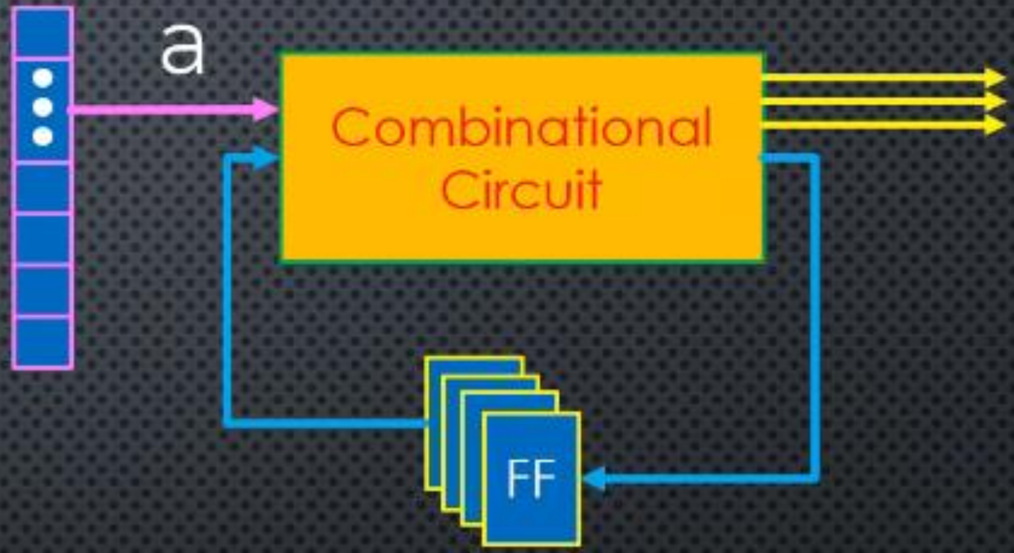


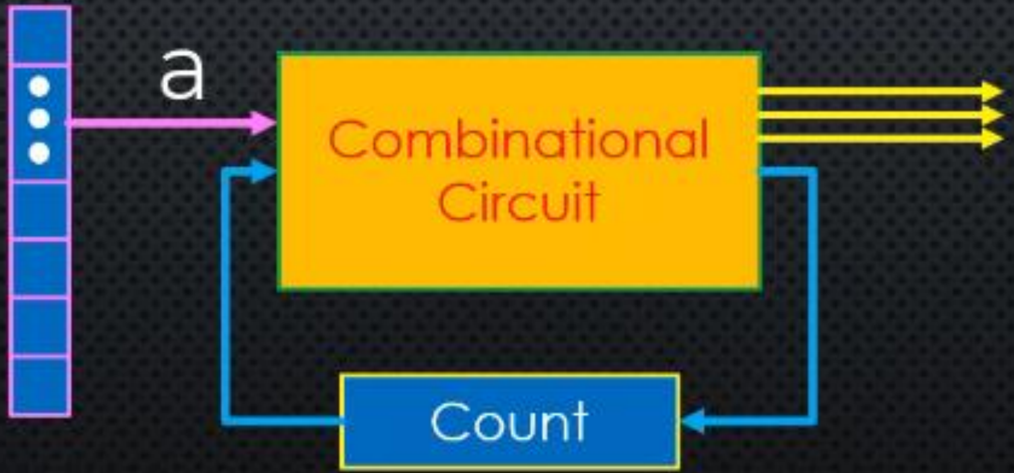
PARALLEL TO SERIAL



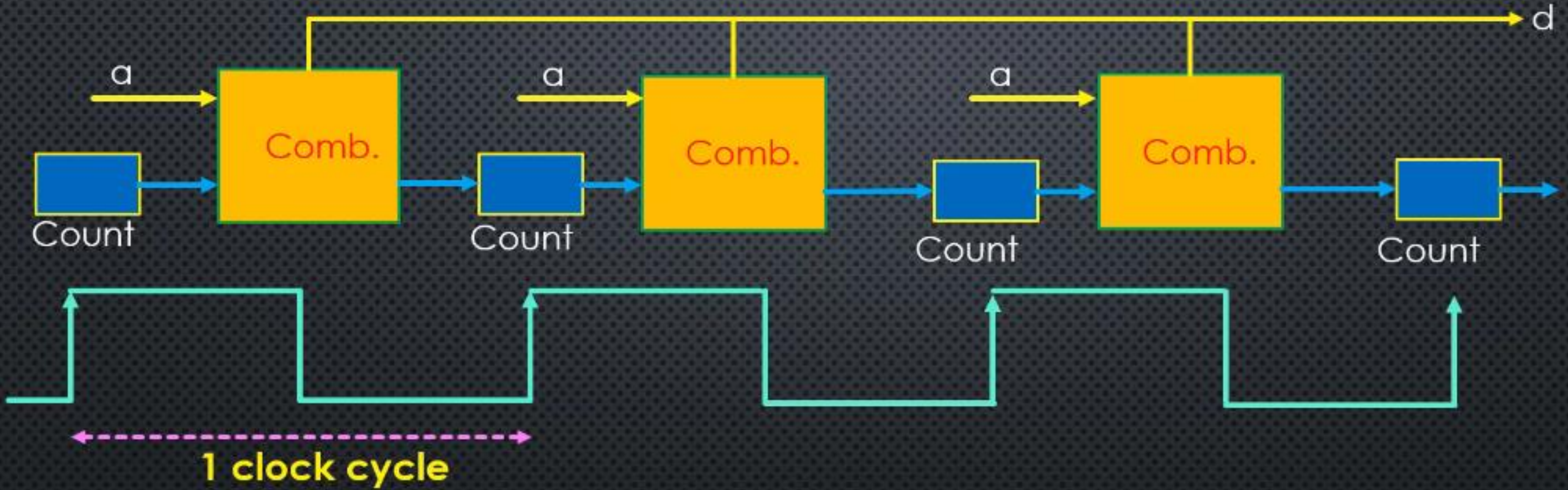
DESIGN


a
Parallel-to-serial

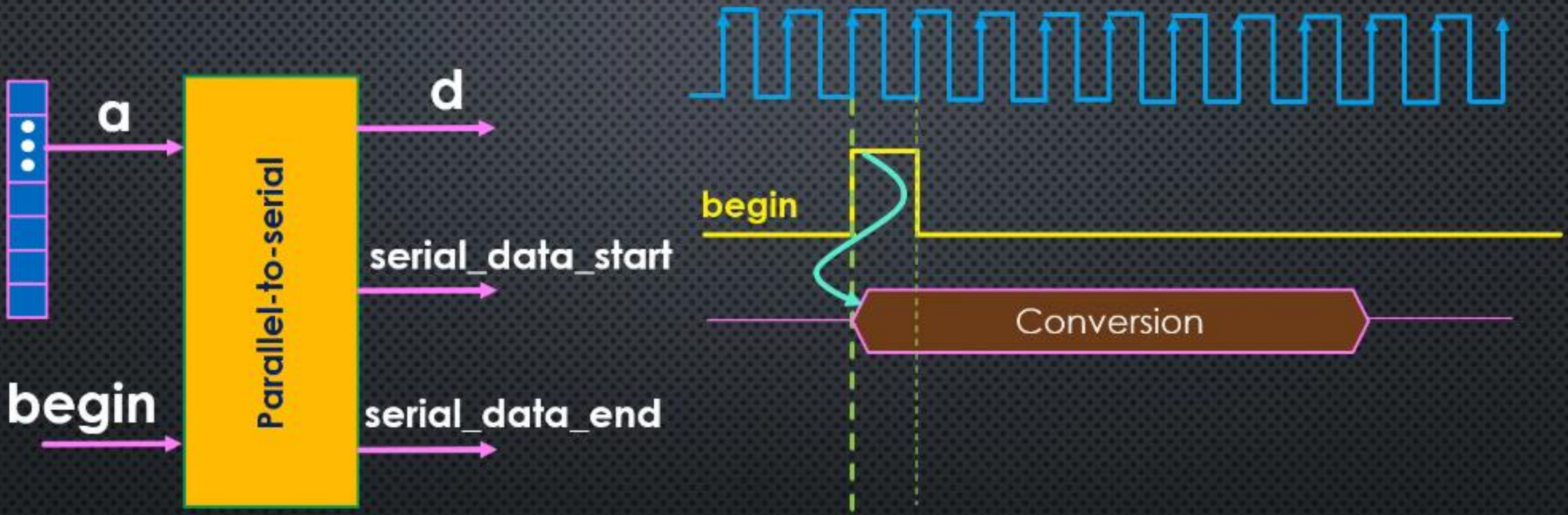

a
Combinational Circuit
FF


a
Combinational Circuit
Count

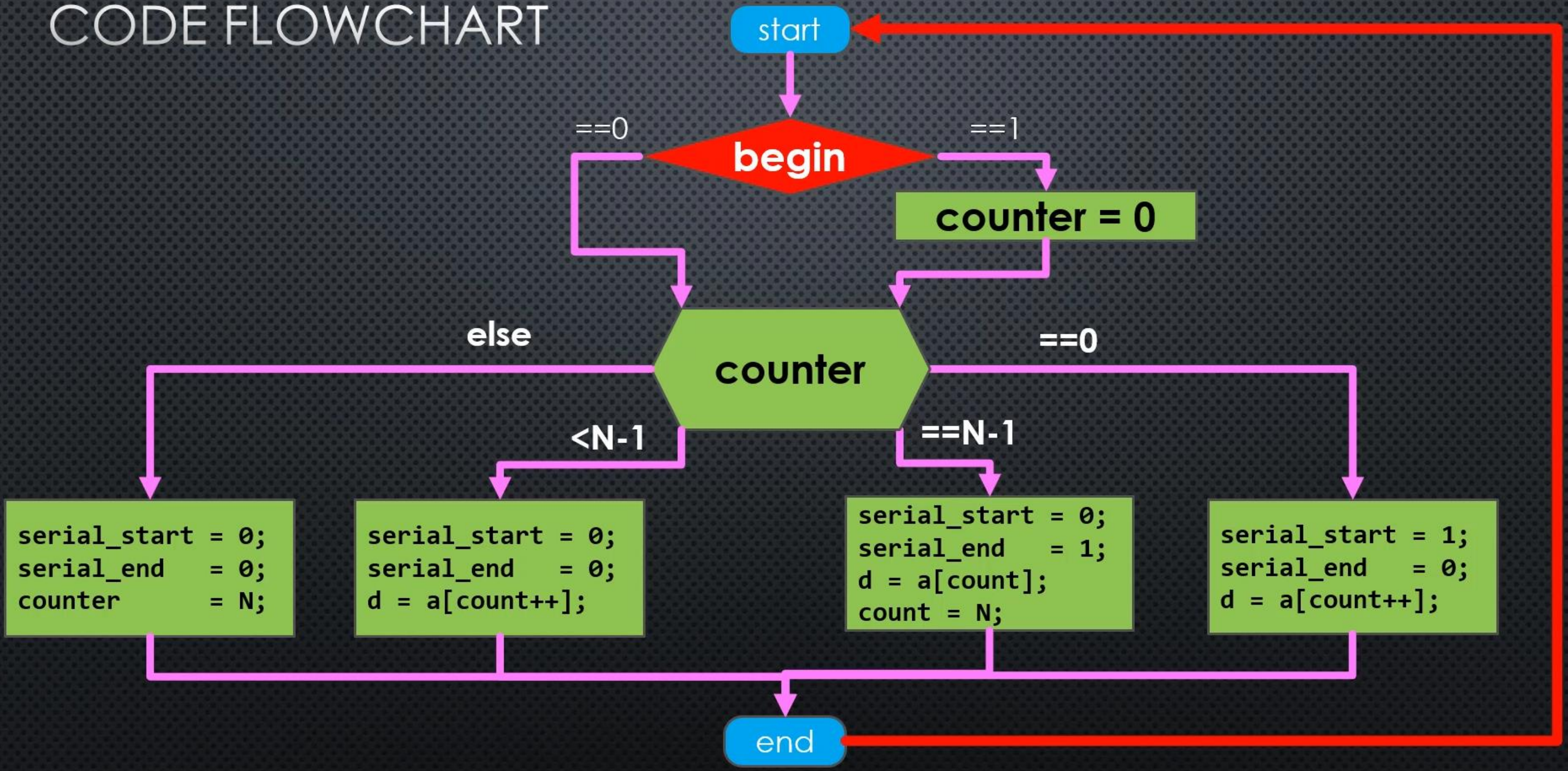
SINGLE-CYCLE DESIGN



SINGLE-CYCLE DESIGN



CODE FLOWCHART



```
serial_start = 0;
serial_end   = 0;
counter      = N;
```

```
serial_start = 0;
serial_end   = 0;
d = a[count++];
```

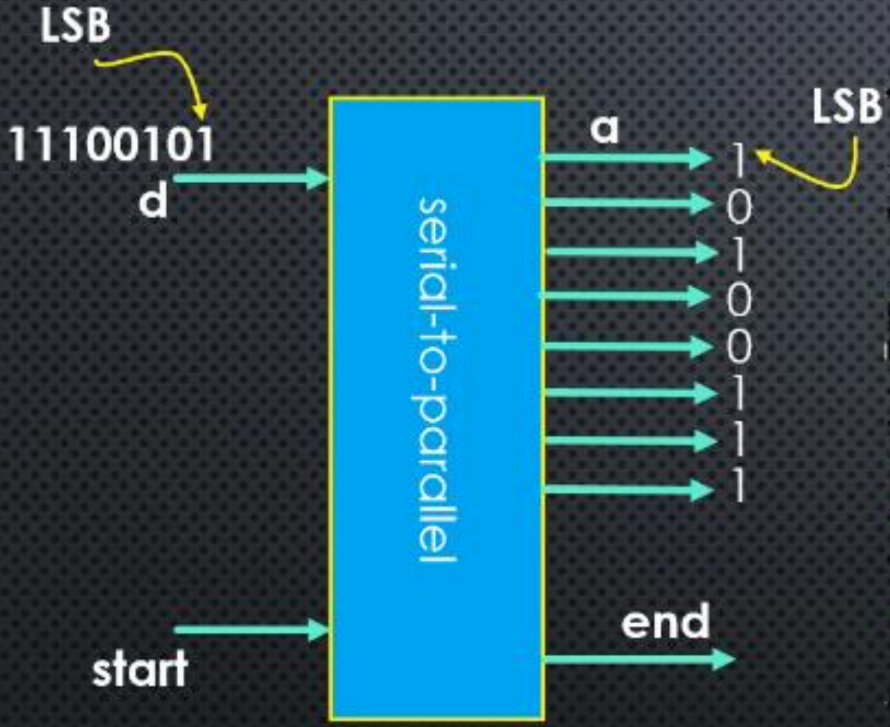
```
serial_start = 0;
serial_end   = 1;
d = a[count];
count = N;
```

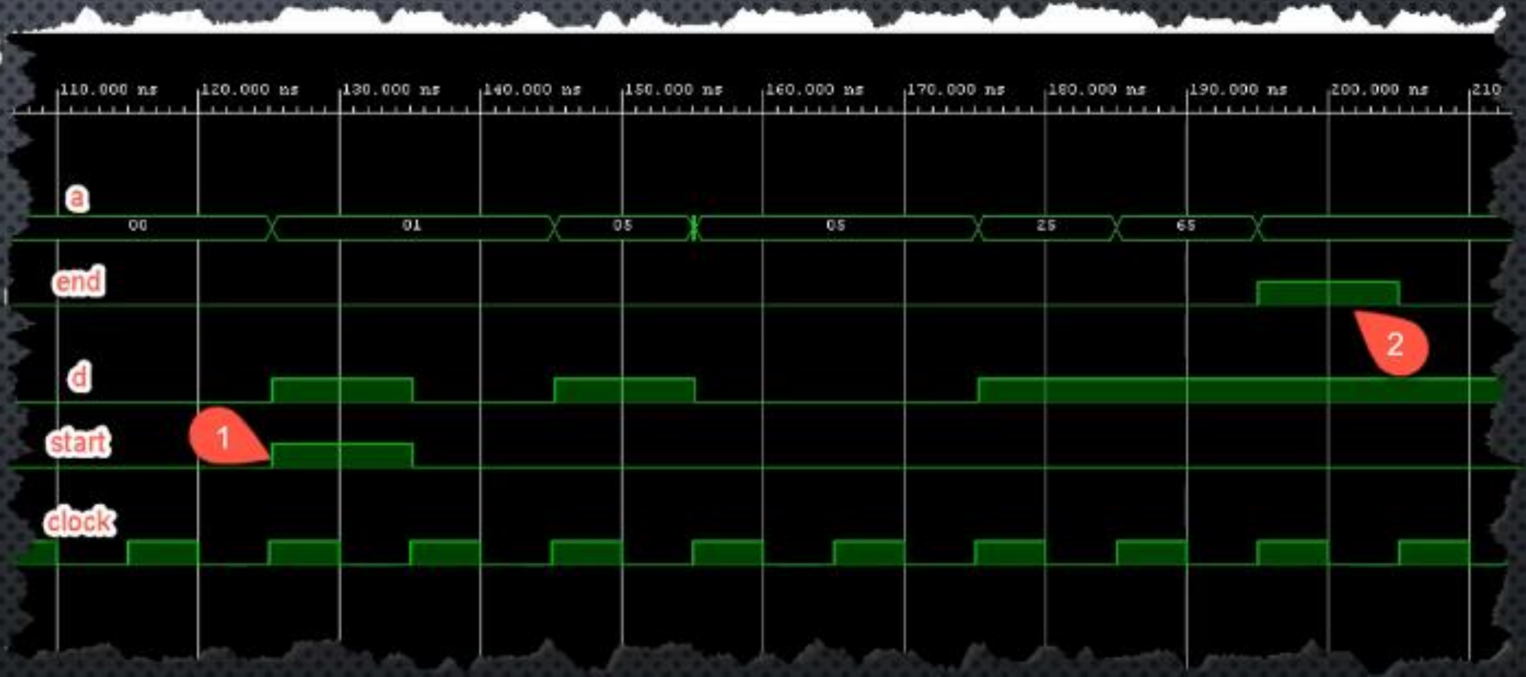
```
serial_start = 1;
serial_end   = 0;
d = a[count++];
```

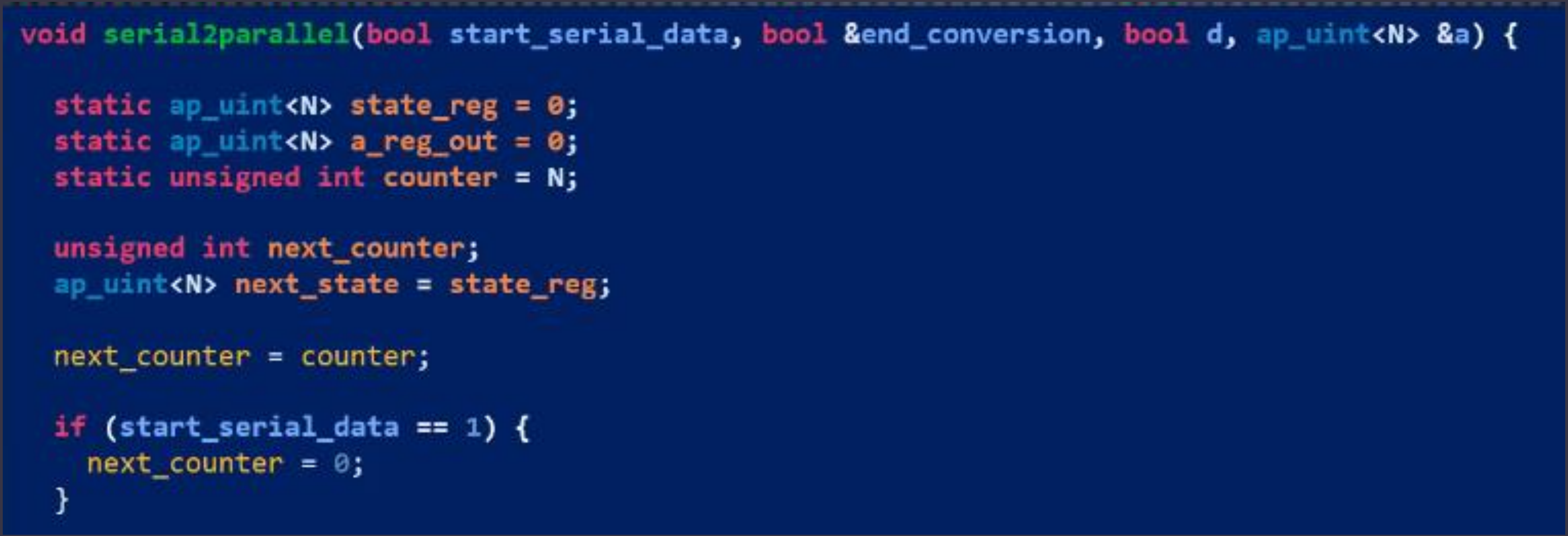
PARALLEL TO SERIAL CODE

```
void parallel2serial(
    ap_uint<N> a,
    bool       &d,
    bool       &start_serial_data,
    bool       &end_serial_data,
    bool       begin) {

  static int count = N;
  if (begin == 1) {
    count = 0;
  }
  if (count == 0) {
    start_serial_data = 1;
    end_serial_data   = 0;
    d = a[count++];
  } else if (count == N-1) {
    start_serial_data = 0;
    end_serial_data   = 1;
    d = a[count];
    count = N;
  } else if (count < N-1) {
    start_serial_data = 0;
    end_serial_data   = 0;
    d = a[count++];
  } else  {
    start_serial_data = 0;
    end_serial_data   = 0;
  }
}
```

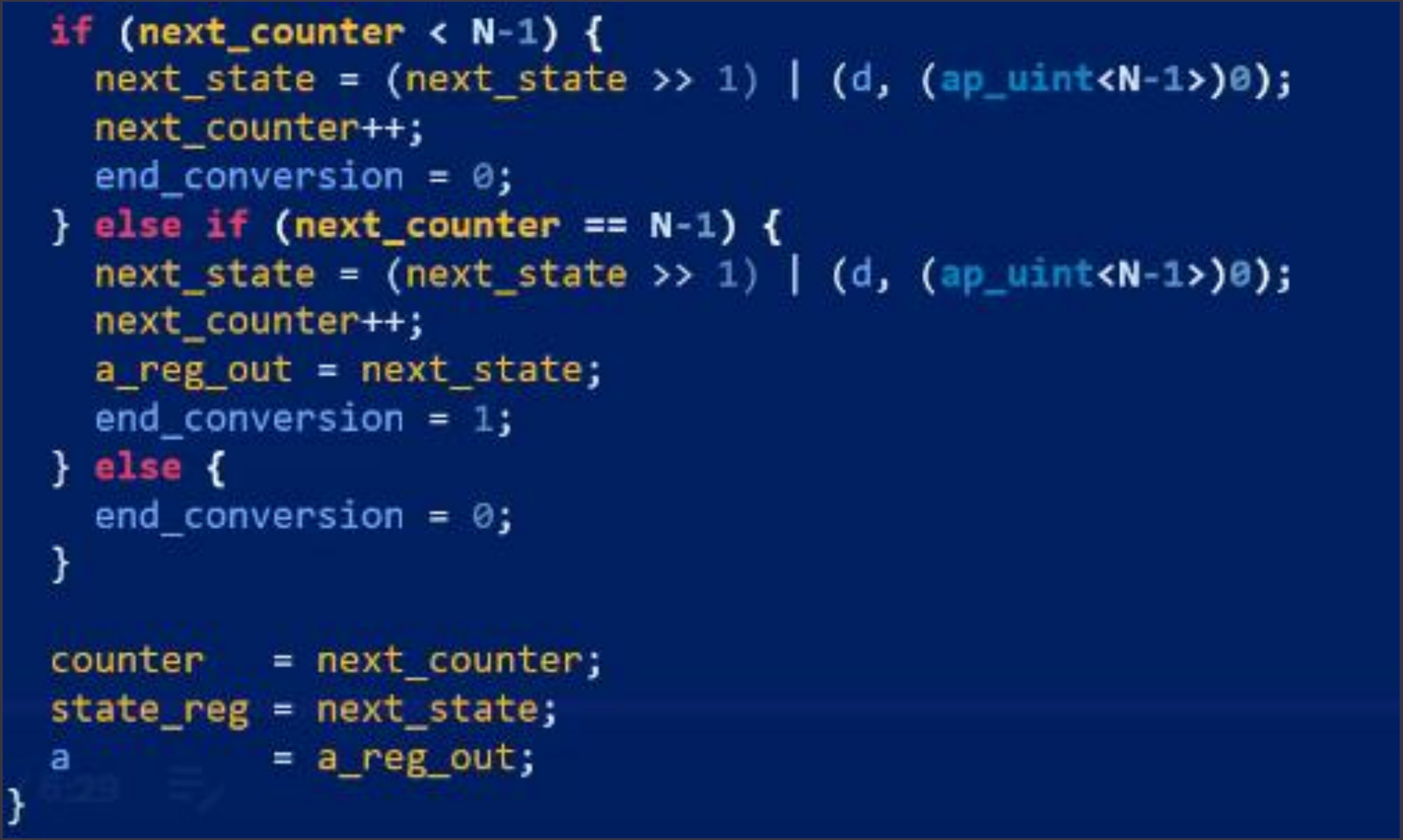
SERIAL TO PARALLEL



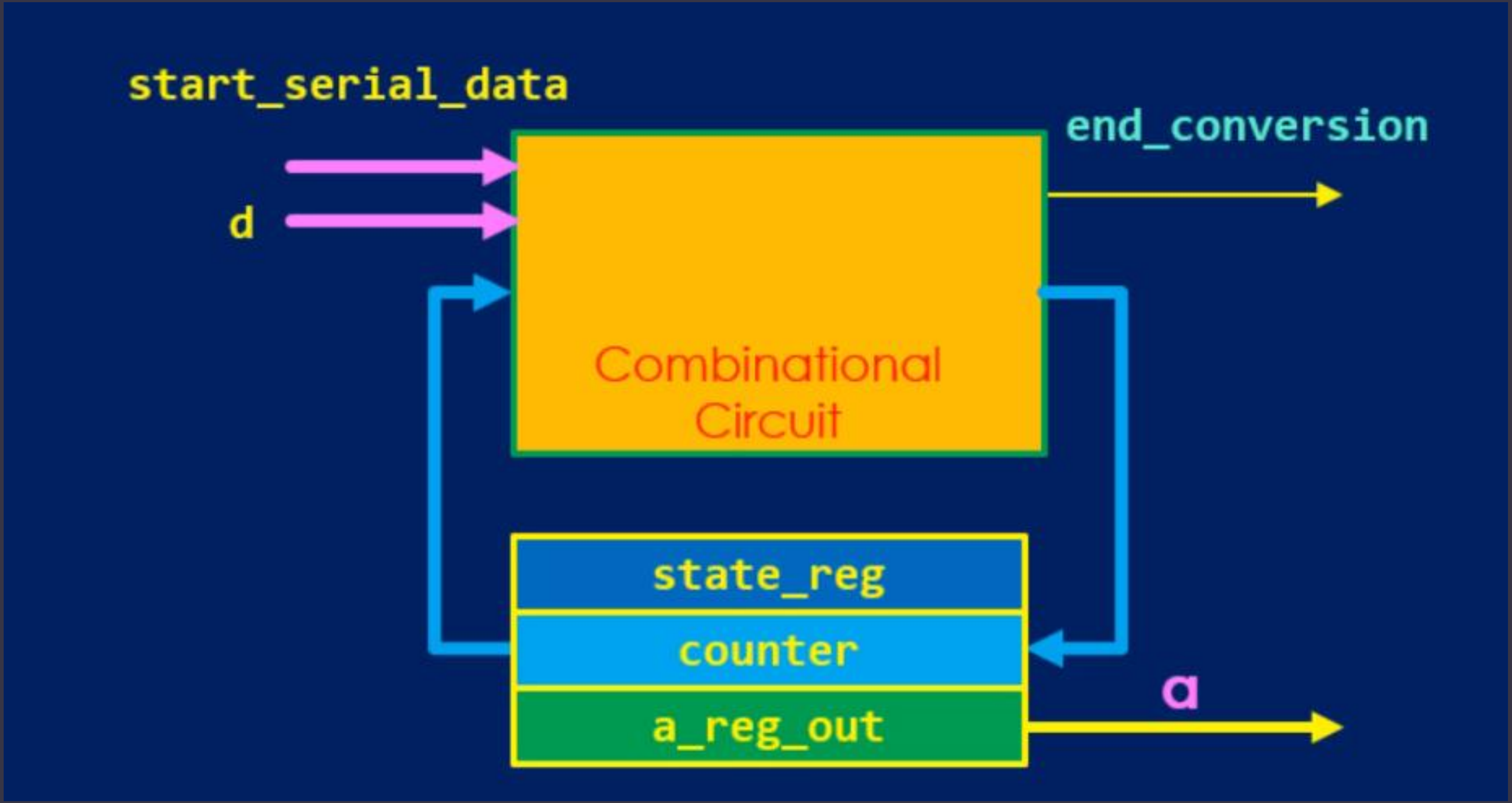




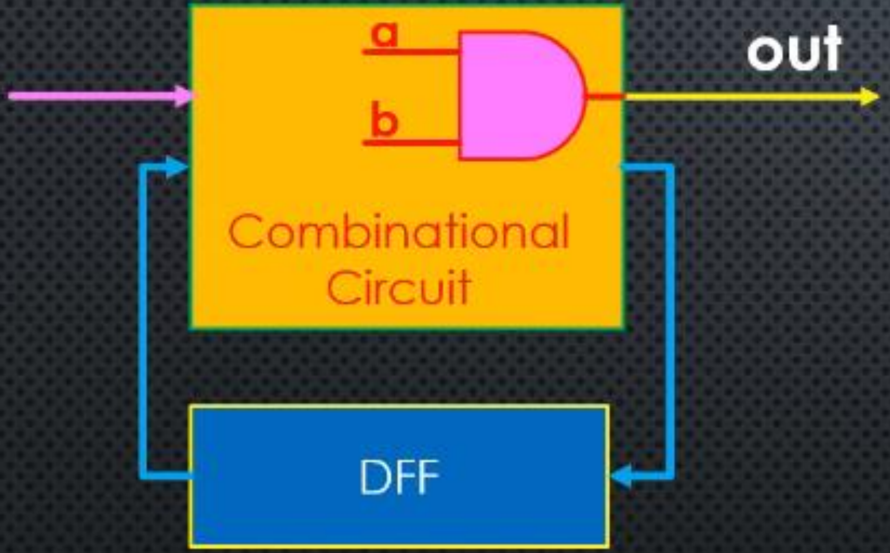
```
void serial2parallel(bool start_serial_data, bool &end_conversion, bool d, ap_uint<N> &a) {

  static ap_uint<N> state_reg = 0;
  static ap_uint<N> a_reg_out = 0;
  static unsigned int counter = N;

  unsigned int next_counter;
  ap_uint<N> next_state = state_reg;

  next_counter = counter;

  if (start_serial_data == 1) {
    next_counter = 0;
  }
```

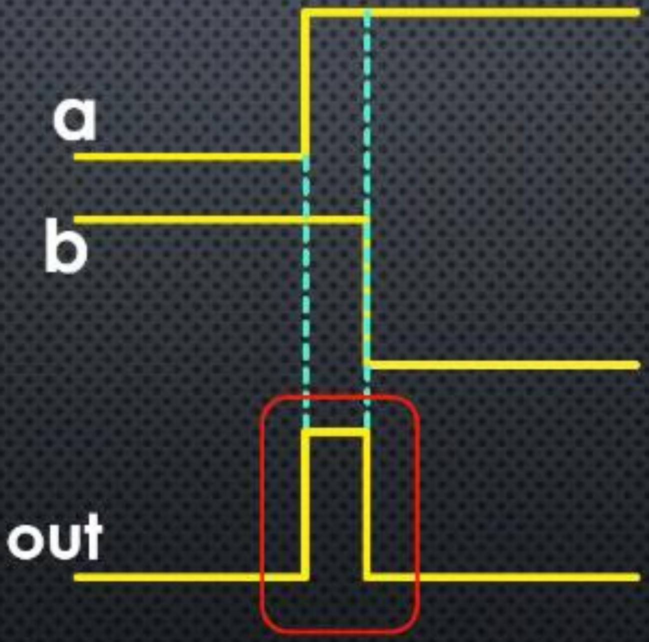


```
  if (next_counter < N-1) {
    next_state = (next_state >> 1) | (d, (ap_uint<N-1>)0);
    next_counter++;
    end_conversion = 0;
  } else if (next_counter == N-1) {
    next_state = (next_state >> 1) | (d, (ap_uint<N-1>)0);
    next_counter++;
    a_reg_out = next_state;
    end_conversion = 1;
  } else {
    end_conversion = 0;
  }

  counter   = next_counter;
  state_reg = next_state;
  a         = a_reg_out;
}
```

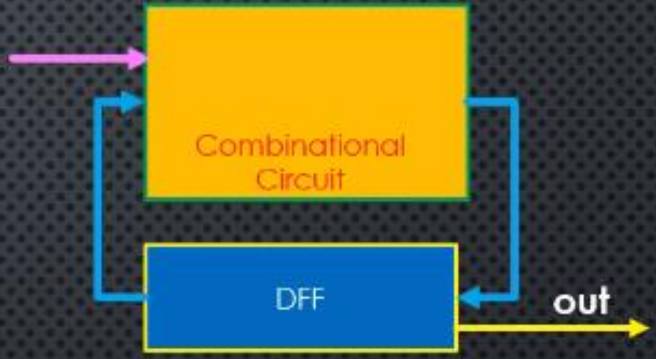

start_serial_data
d
end_conversion
Combinational Circuit
state_reg
counter
a_reg_out
a

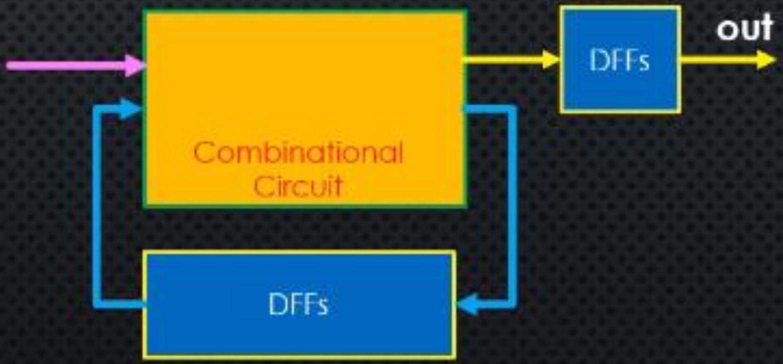
COMBINATIONAL CIRCUIT OUTPUT



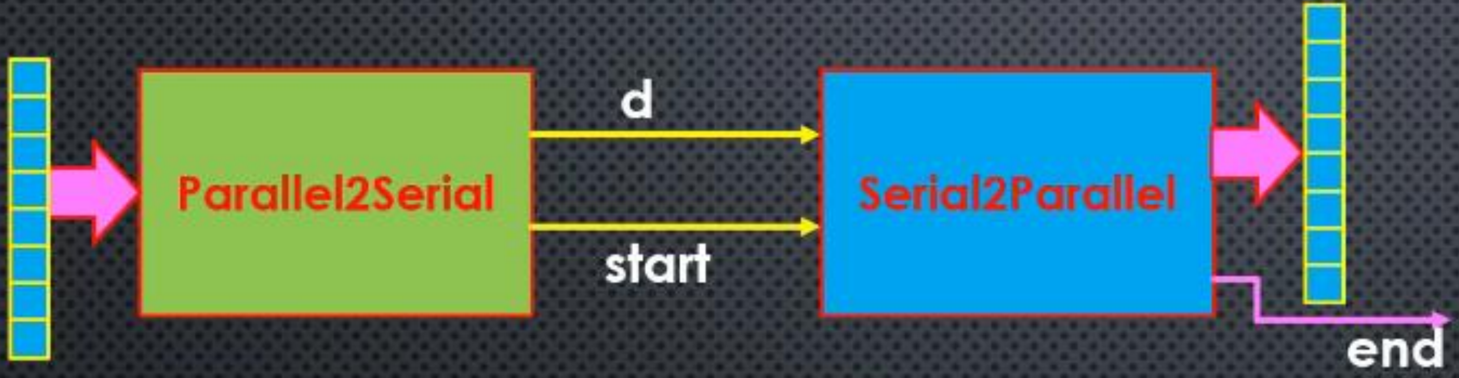


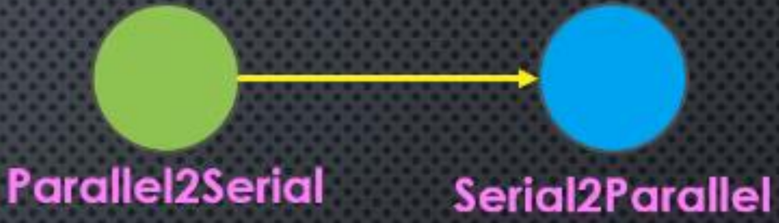
Undesired glitch

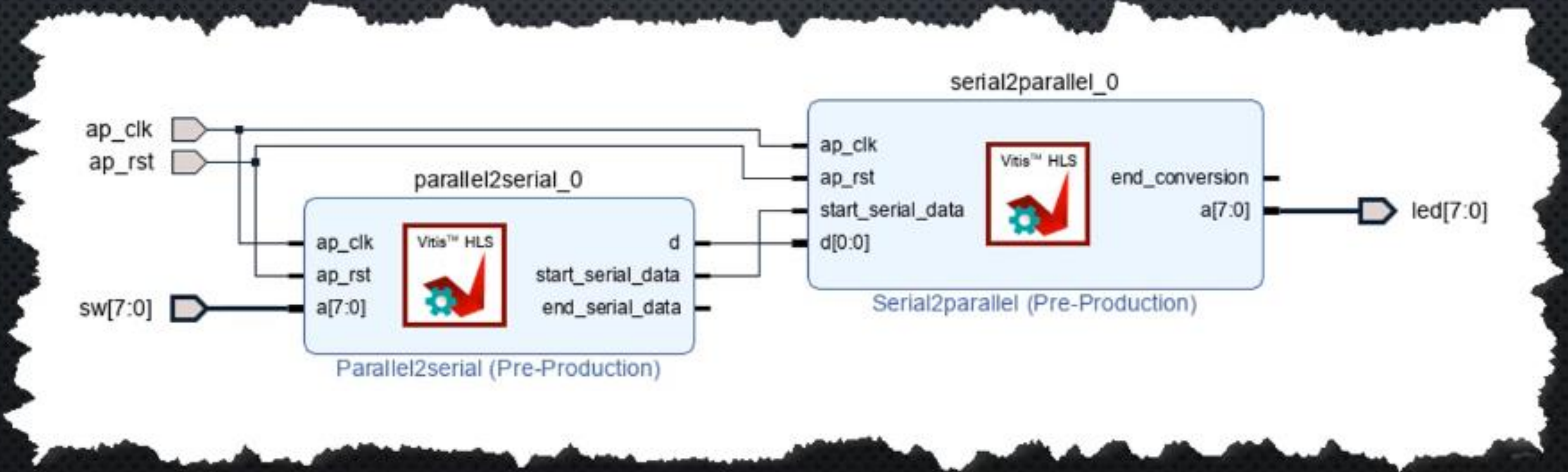




PARALLEL-SERIAL-PARALLEL LAB







HLS IMPLEMENTATION

Lecture 9

Any Question…

Thank you