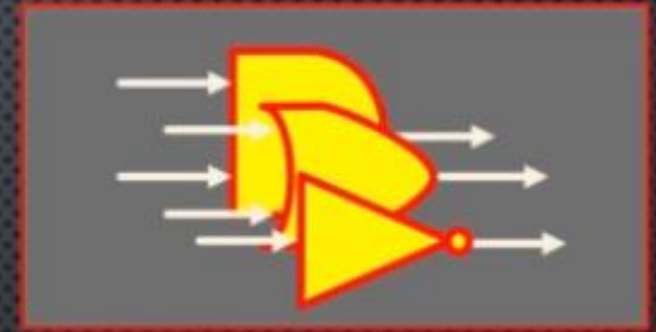


SEQUENTIAL CIRCUITS-DFF

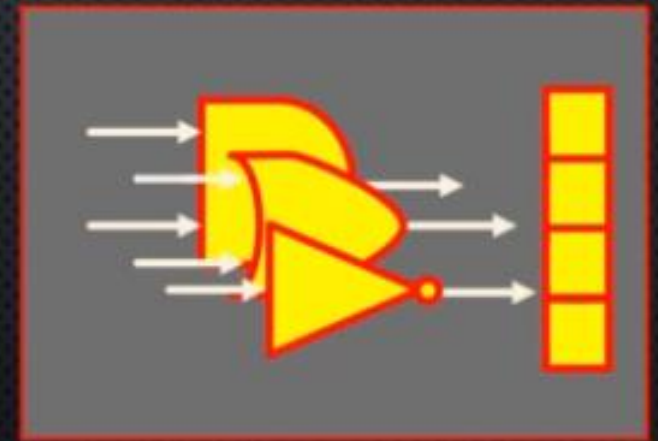
Lecture-8

COMBINATIONAL CIRCUITS

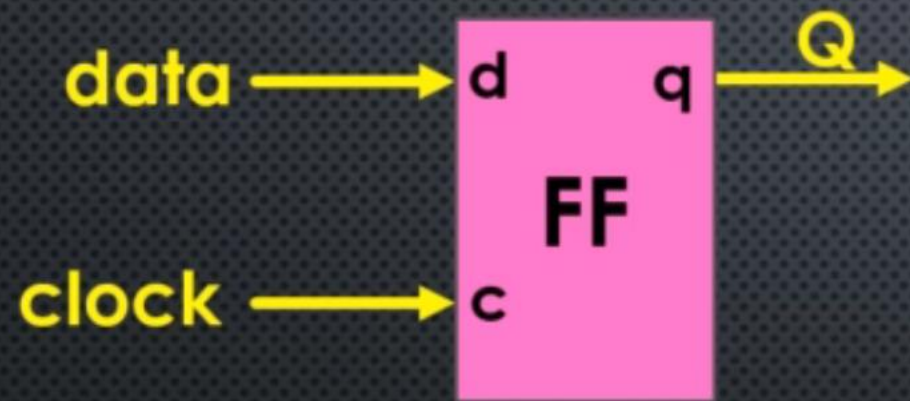
Combinational Circuits



Sequential Circuits

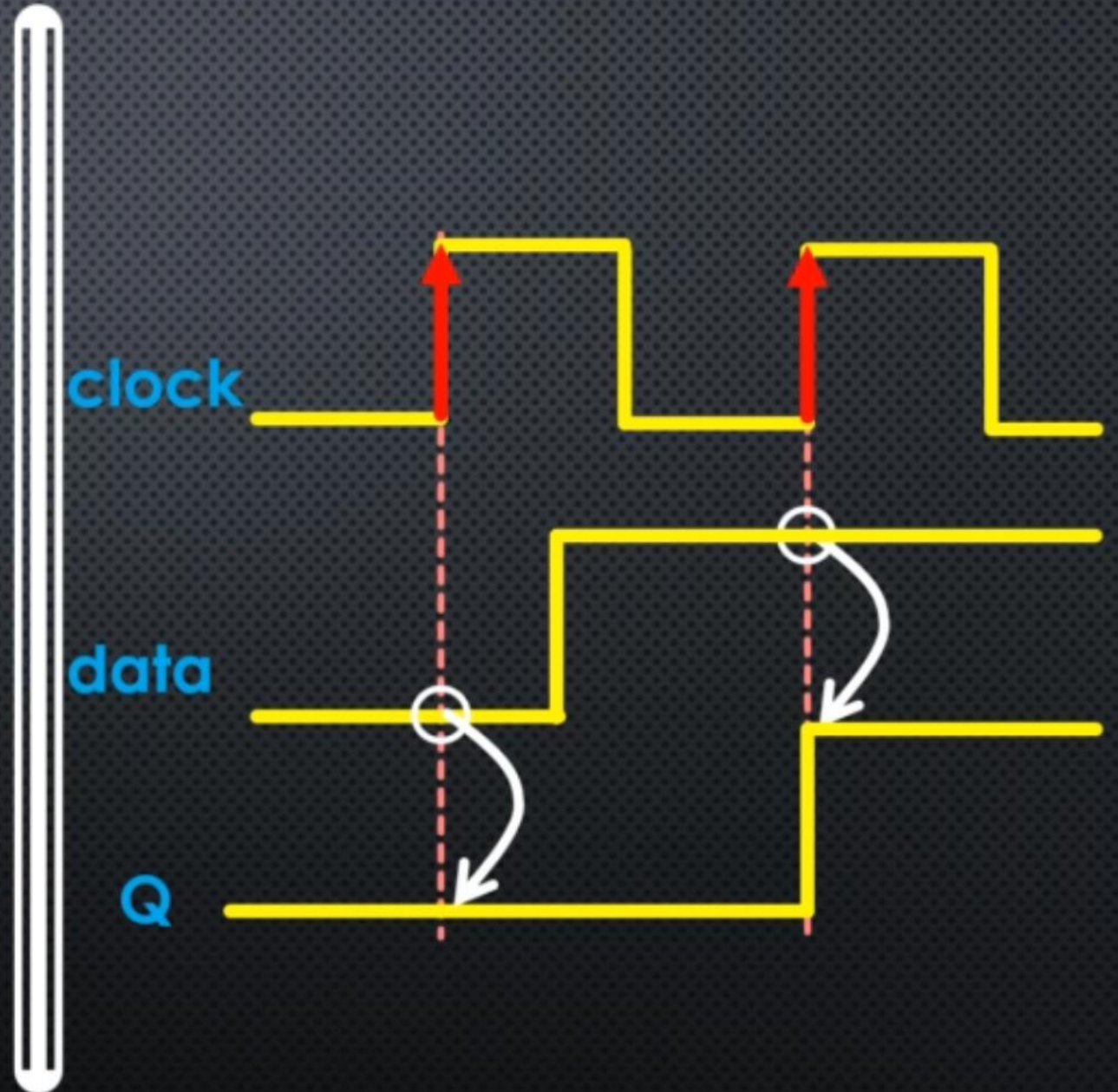


FLIP-FLOP

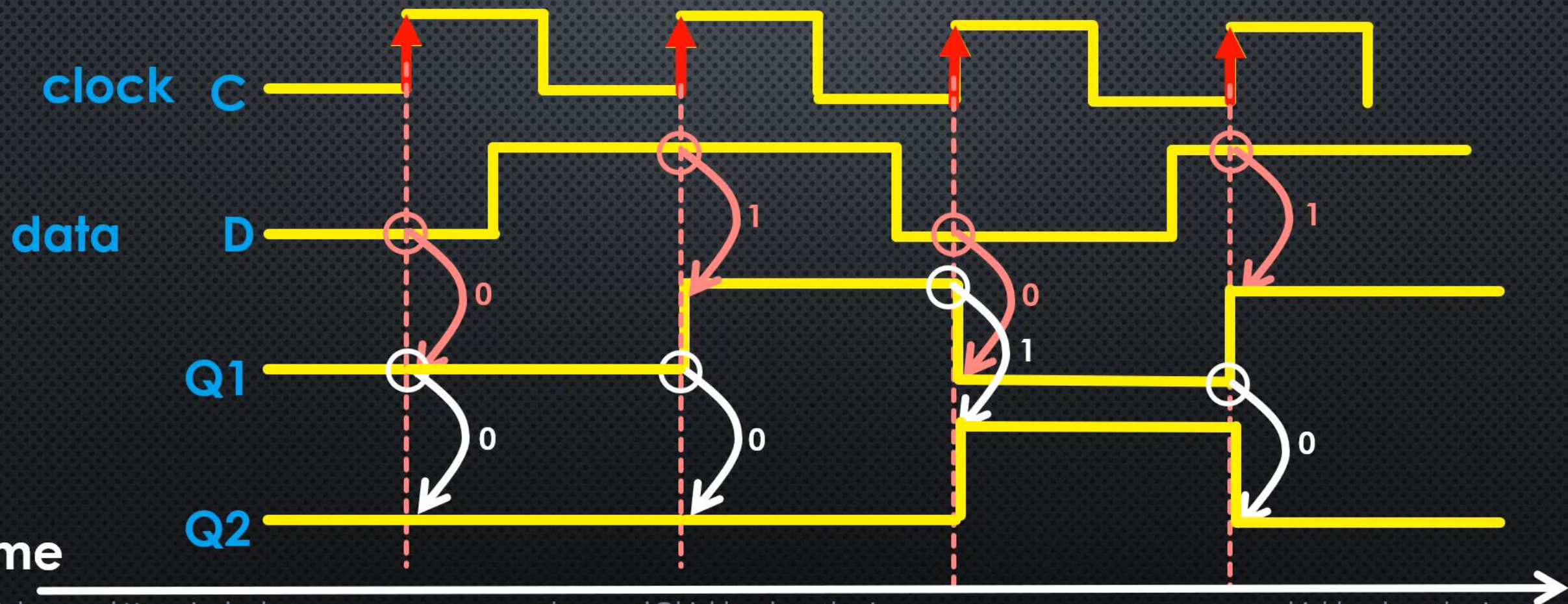
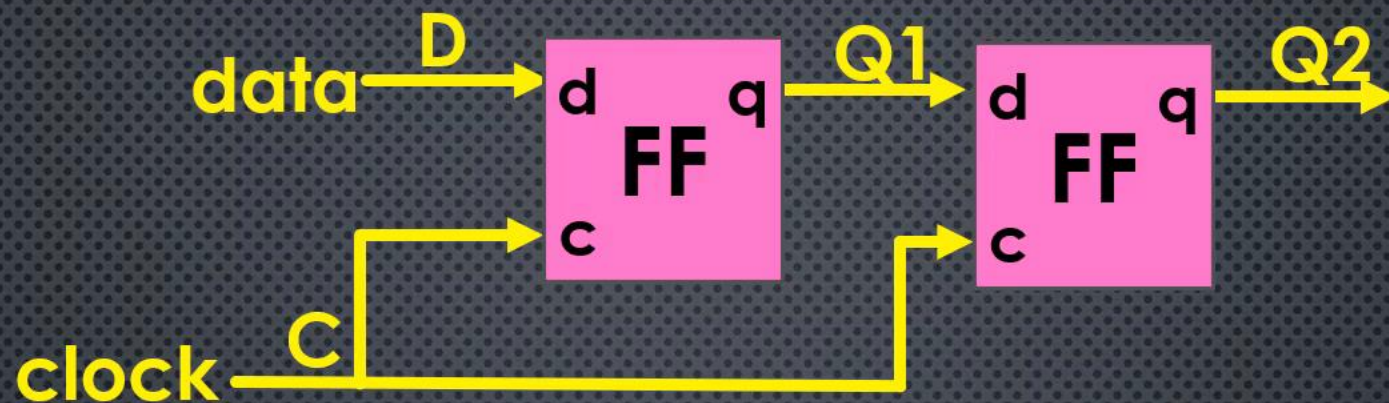


Characteristic Table

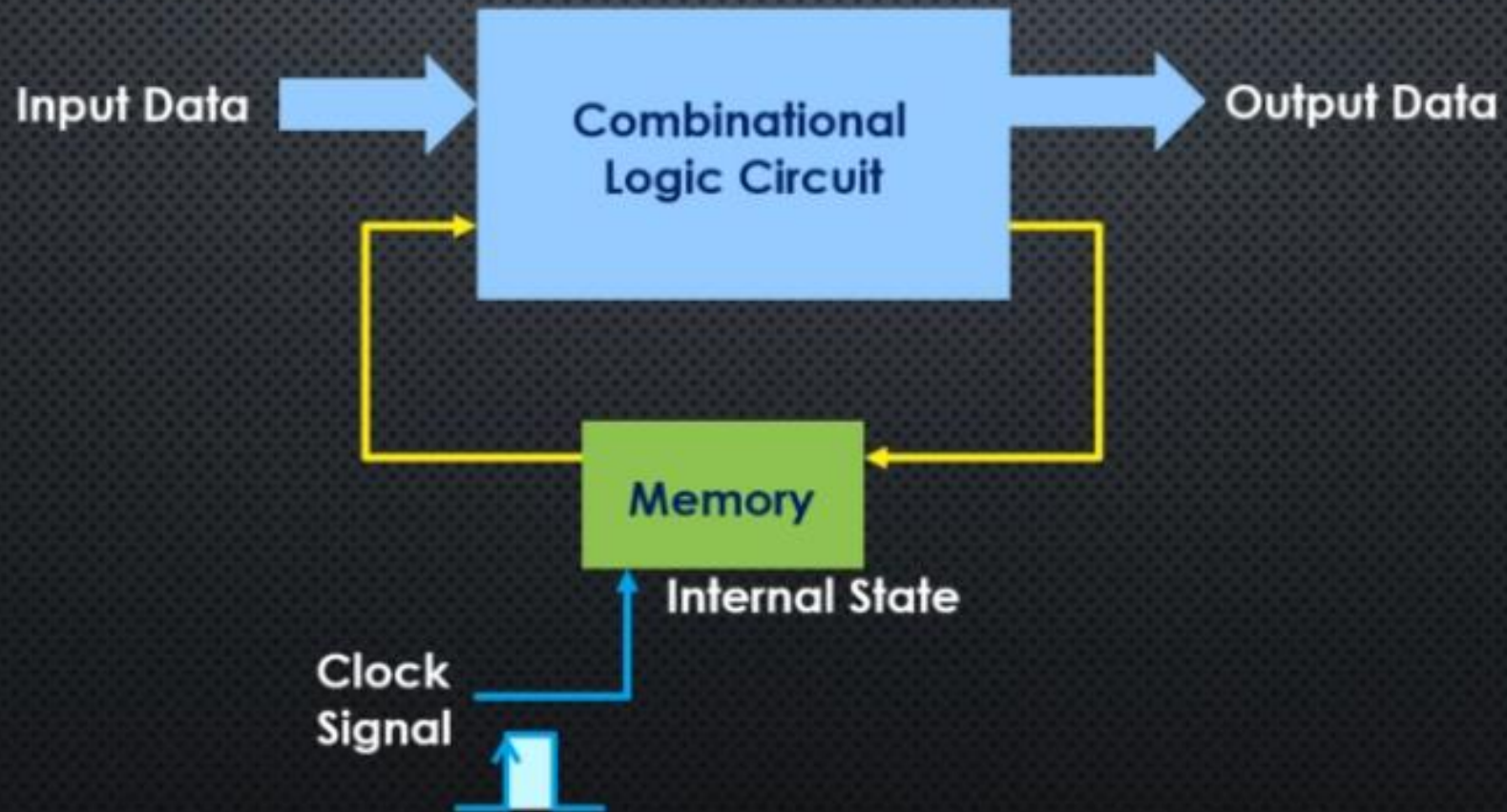
D	C	Q
1	↑	1
0	↑	0



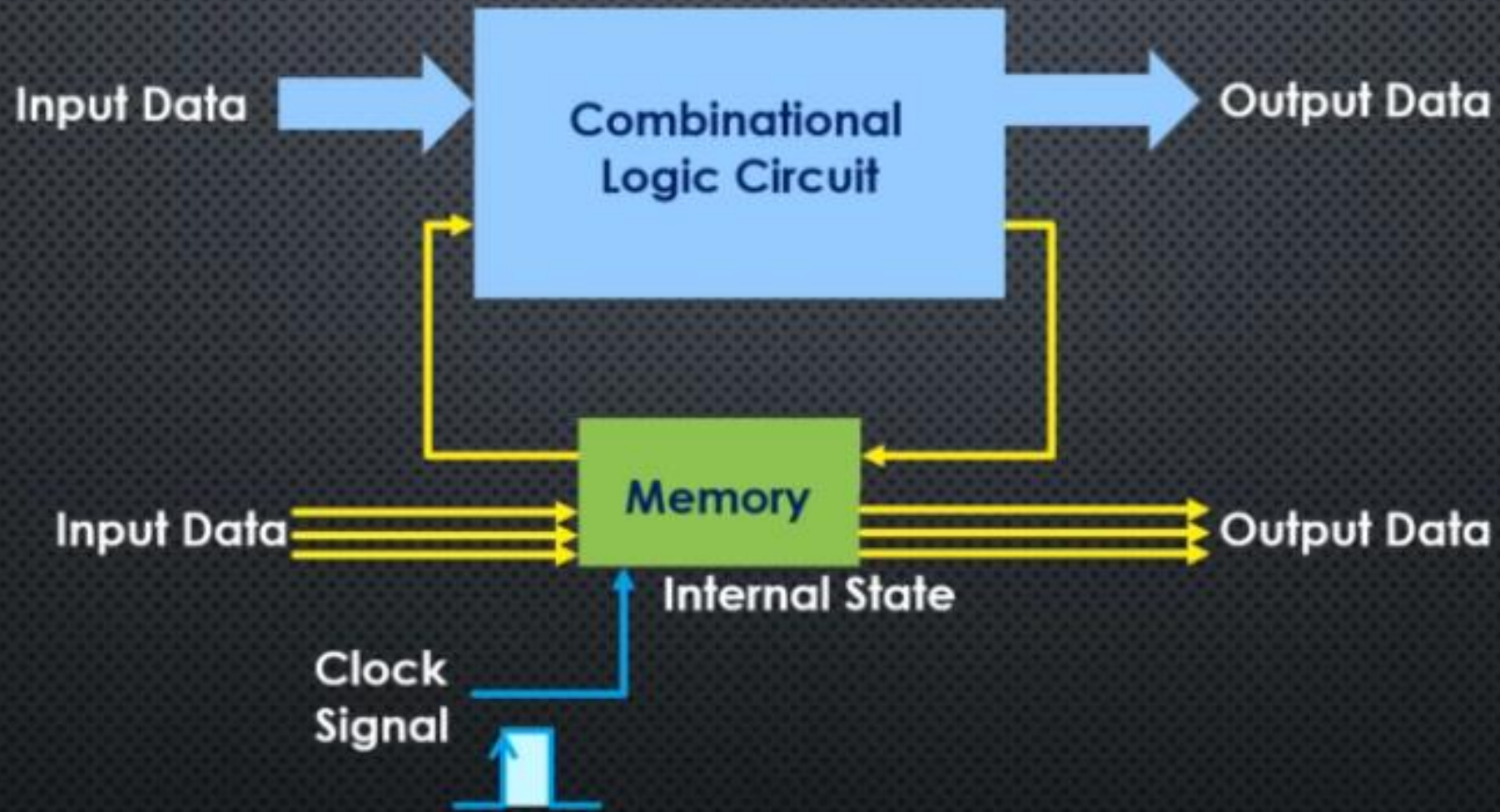
EXAMPLE



SEQUENTIAL CIRCUIT



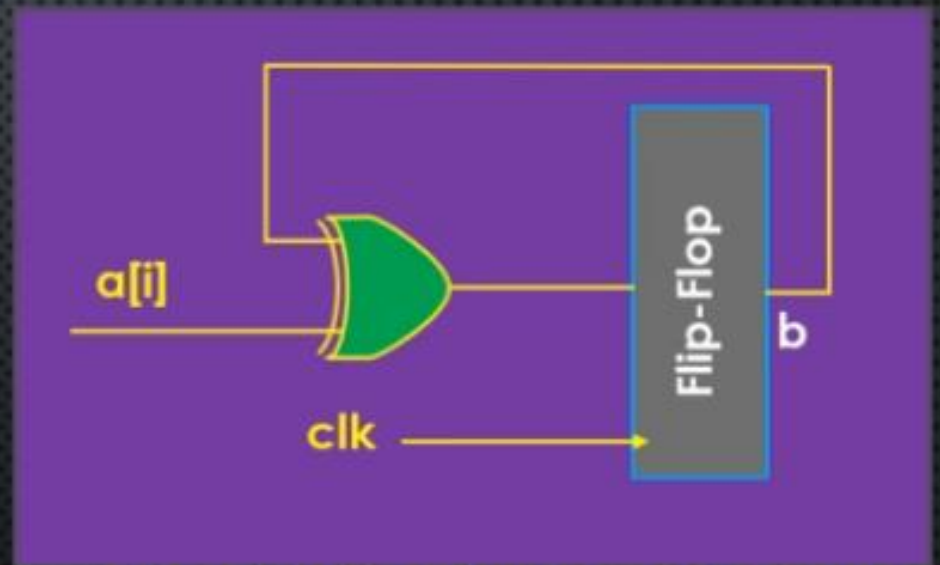
MORE DETAILS



EXAMPLE

```
bool sequential_parity(ap_uint<N> a)
{
    bool b=0;
    for (int i = 0; i < n; i++)
        b = b^a[i];
    return b;
}
```

Hardware design described in C

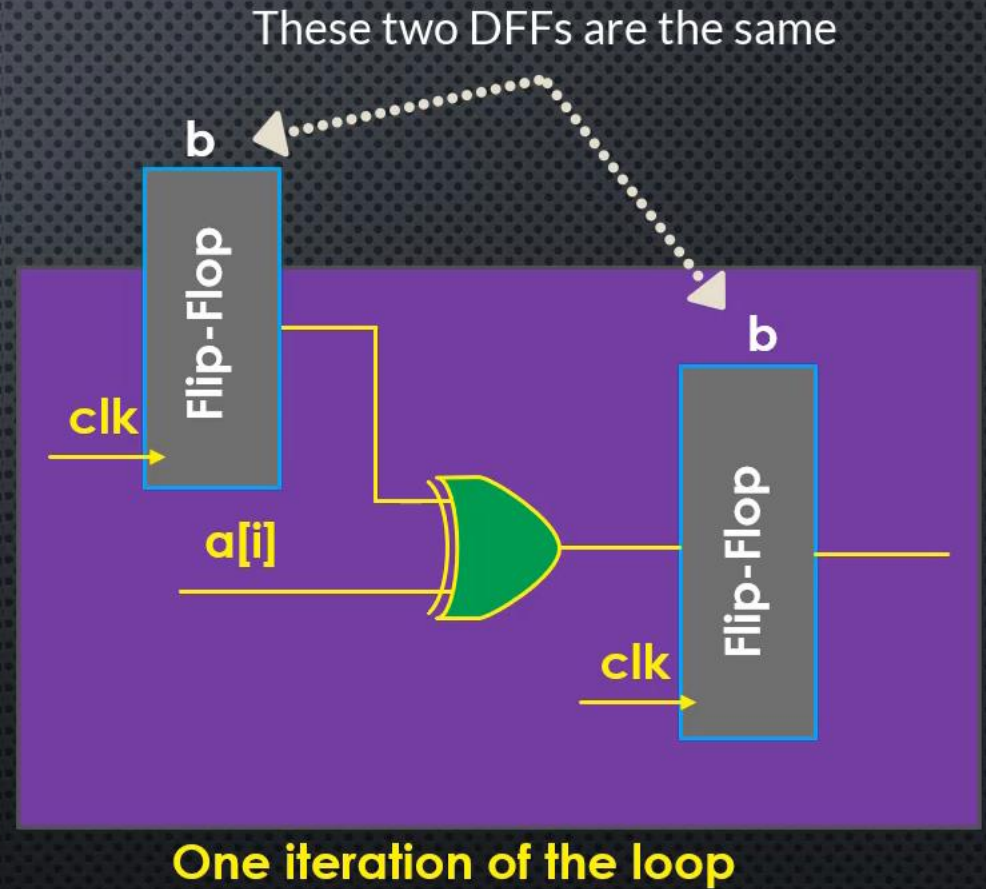


Simplified hardware architecture

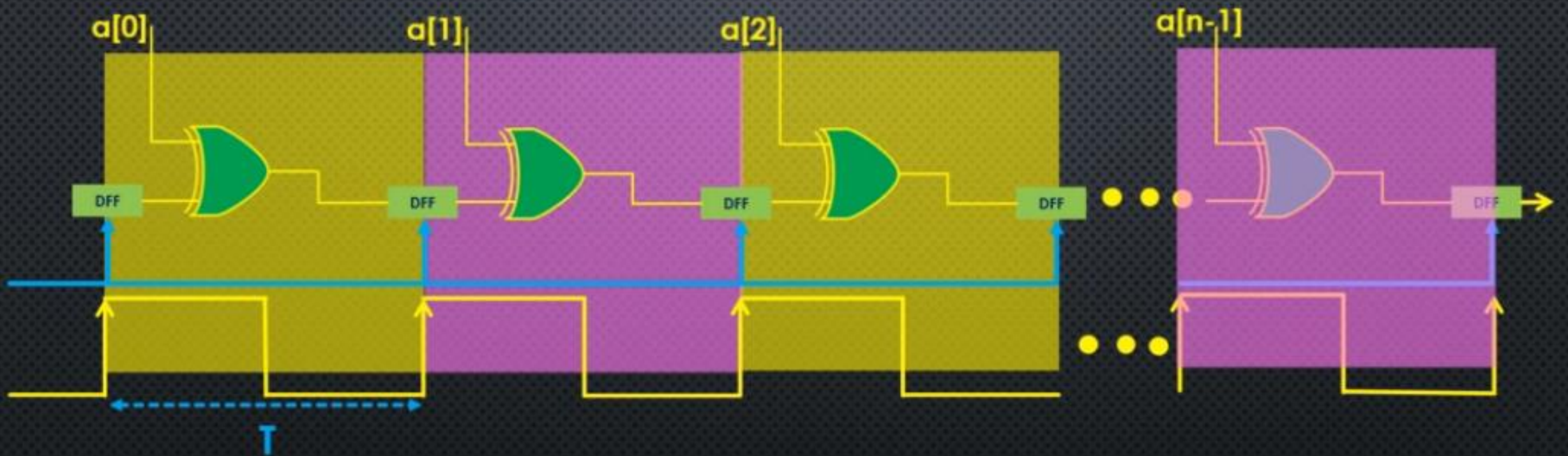
EXAMPLE: ONE ITERATION

```
bool sequential_parity(ap_uint<N> a)
{
    bool b=0;
    for (int i = 0; i < n; i++)
        b = b^a[i];
    return b;
}
```

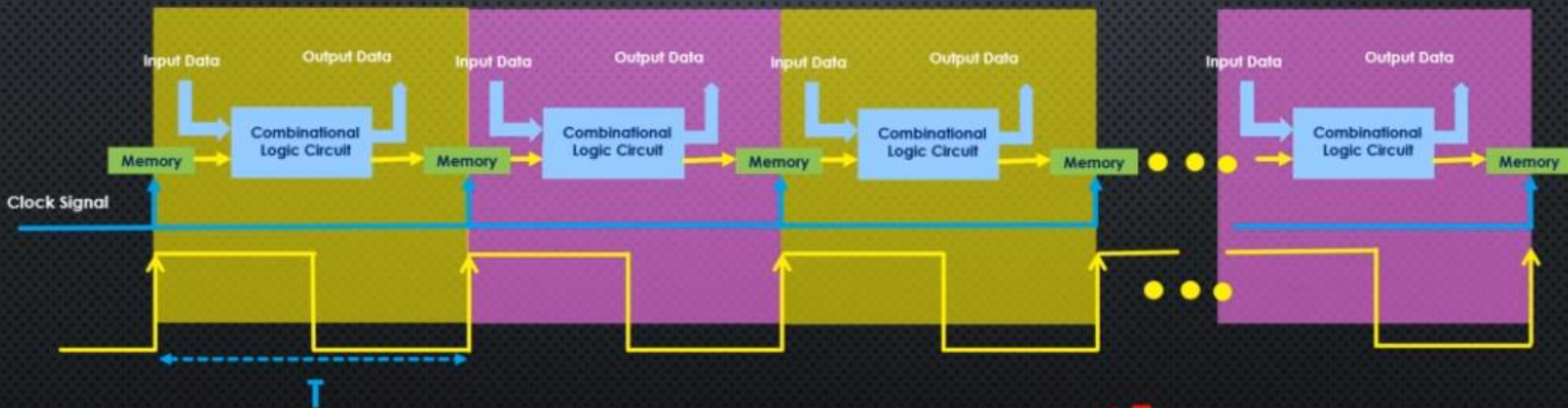
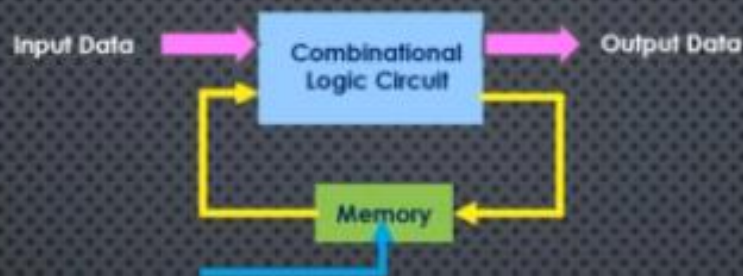
Hardware design in C



EXAMPLE: SEQUENCE OF EXECUTION

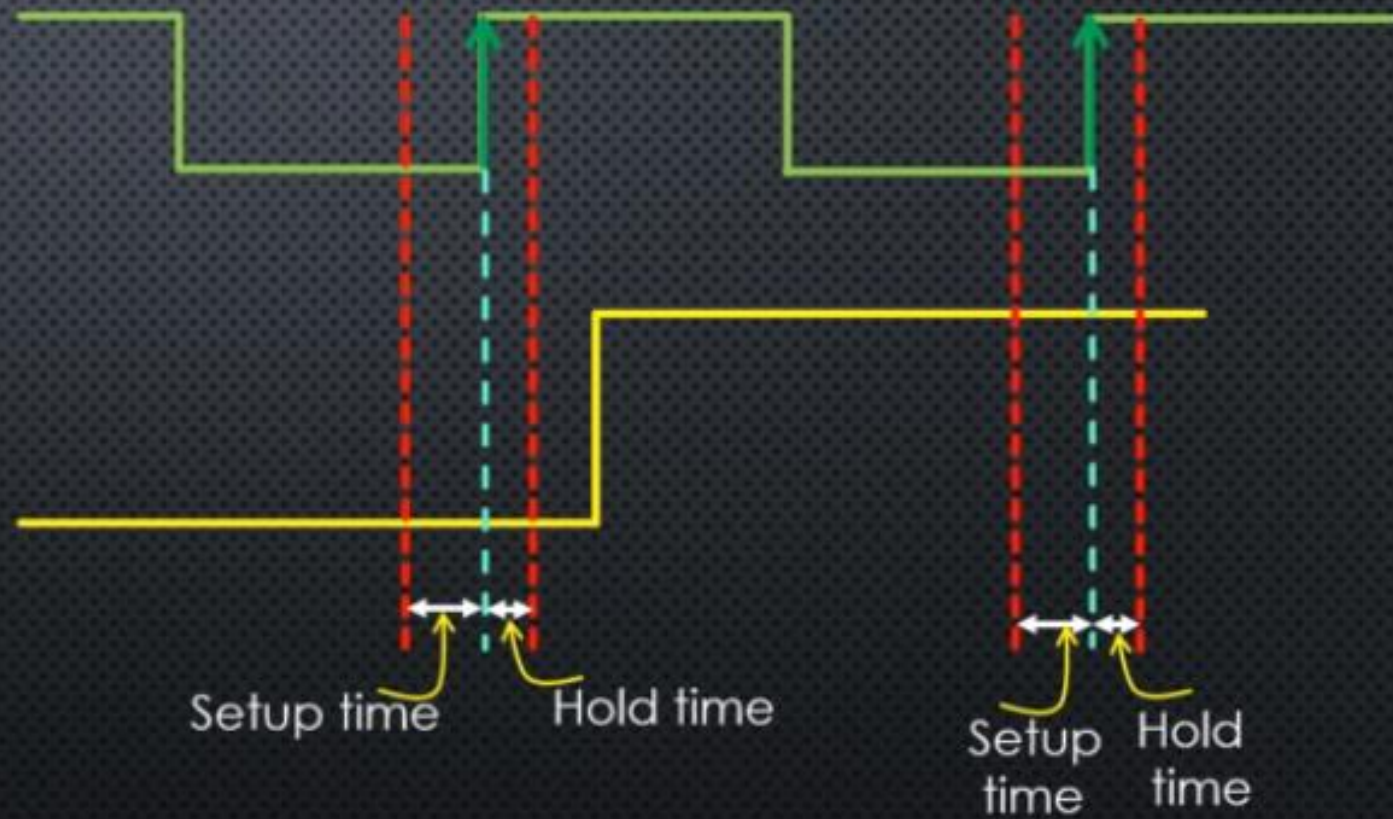
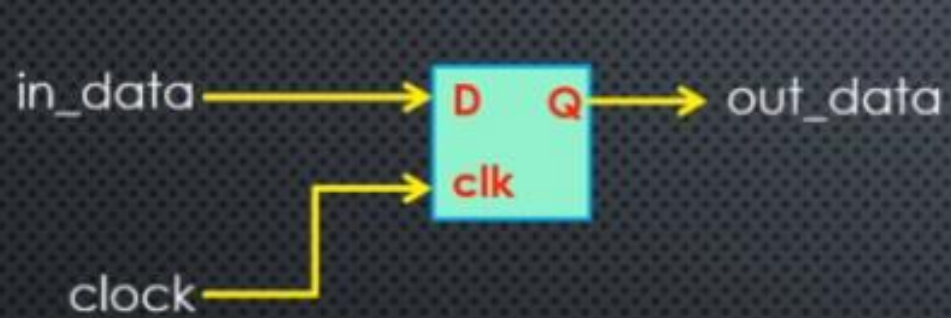


SEQUENTIAL CIRCUITS: SEQUENCE OF EXECUTION

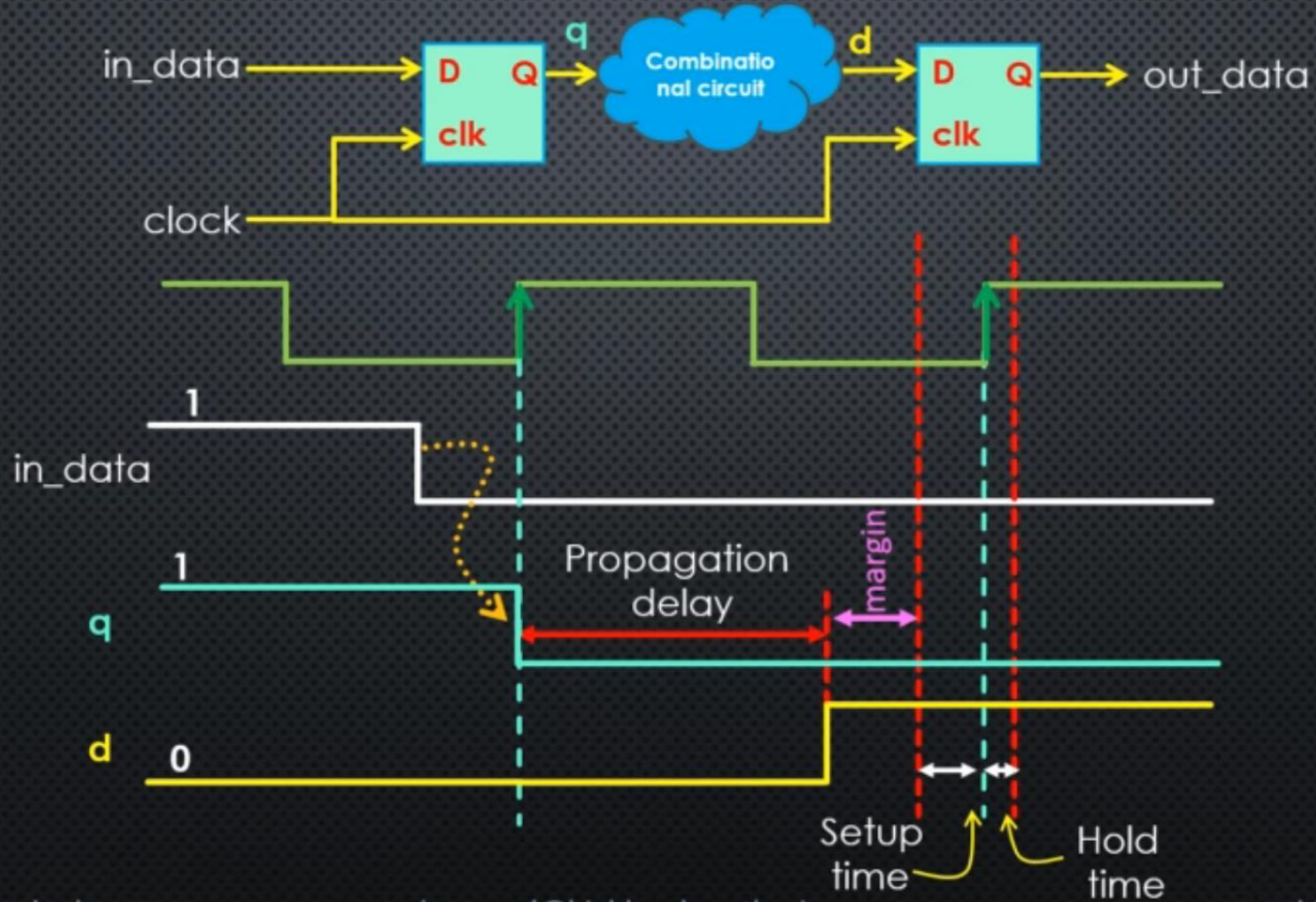


Combinational Circuit propagation delay $< T$

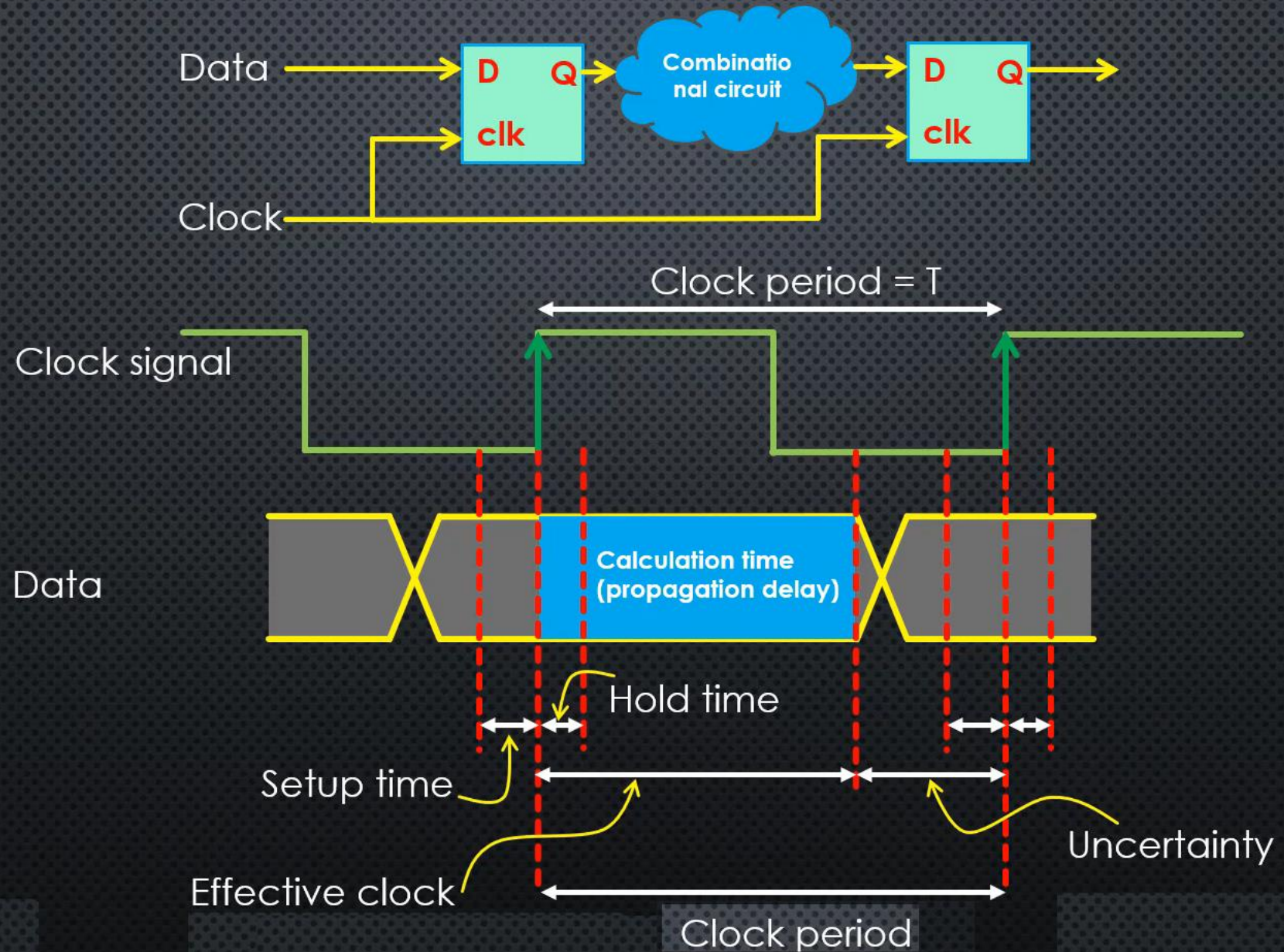
SETUP AND HOLD TIME



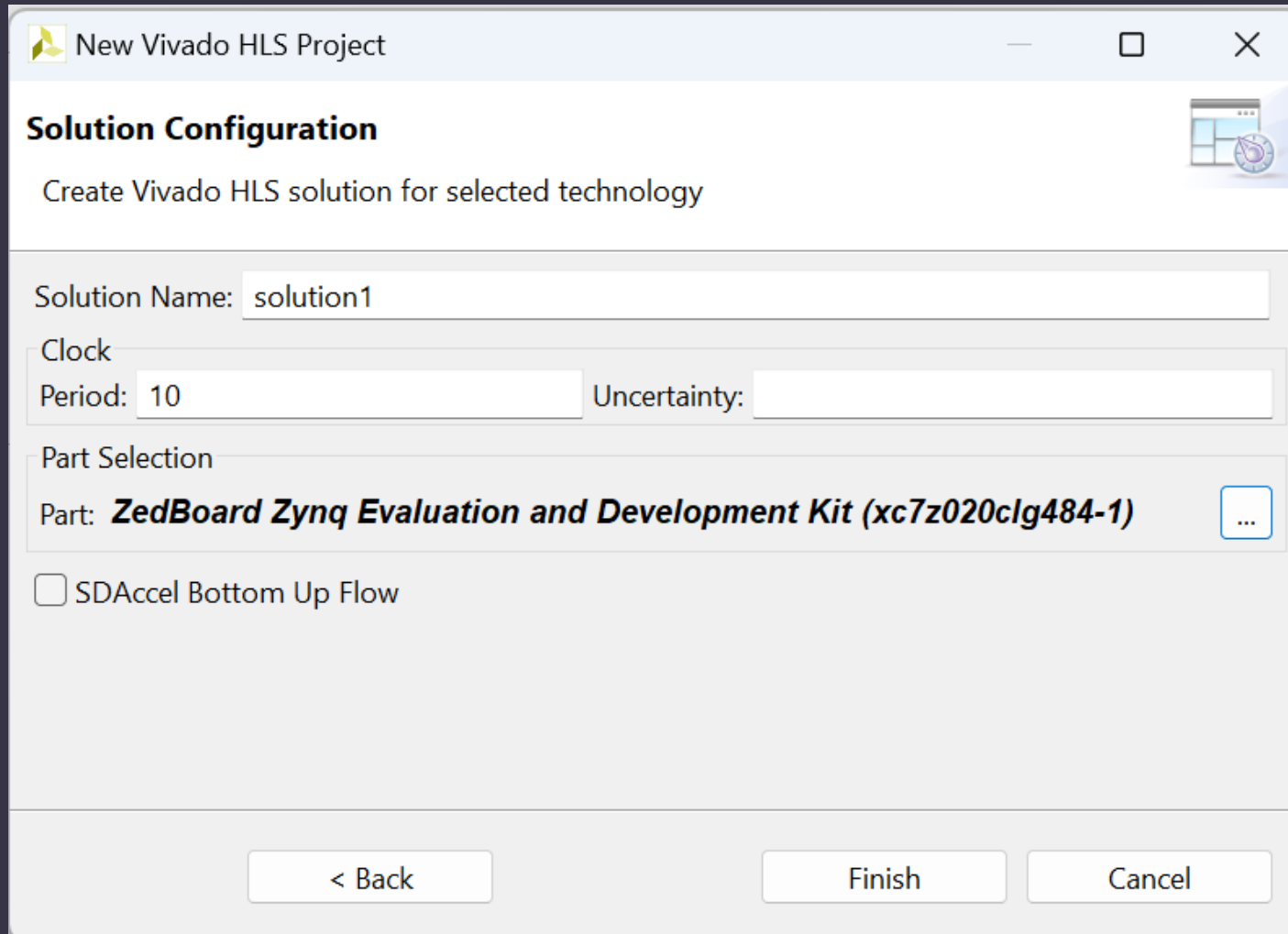
PROPAGATION DELAY & SETUP/HOLD TIME



CLOCK ANATOMY



CLOCK CONSTRAINT IN HLS



The screenshot shows the 'New Vivado HLS Project' dialog box. The title bar reads 'New Vivado HLS Project'. The main section is titled 'Solution Configuration' with a subtitle 'Create Vivado HLS solution for selected technology'. Below this, there are three main configuration areas: 'Solution Name' with a text field containing 'solution1'; 'Clock' with 'Period' set to '10' and an empty 'Uncertainty' field; and 'Part Selection' with the part name 'ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)' and a selection icon. At the bottom, there is an unchecked checkbox for 'SDAccel Bottom Up Flow' and three buttons: '< Back', 'Finish', and 'Cancel'.

New Vivado HLS Project

Solution Configuration

Create Vivado HLS solution for selected technology

Solution Name:

Clock

Period: Uncertainty:

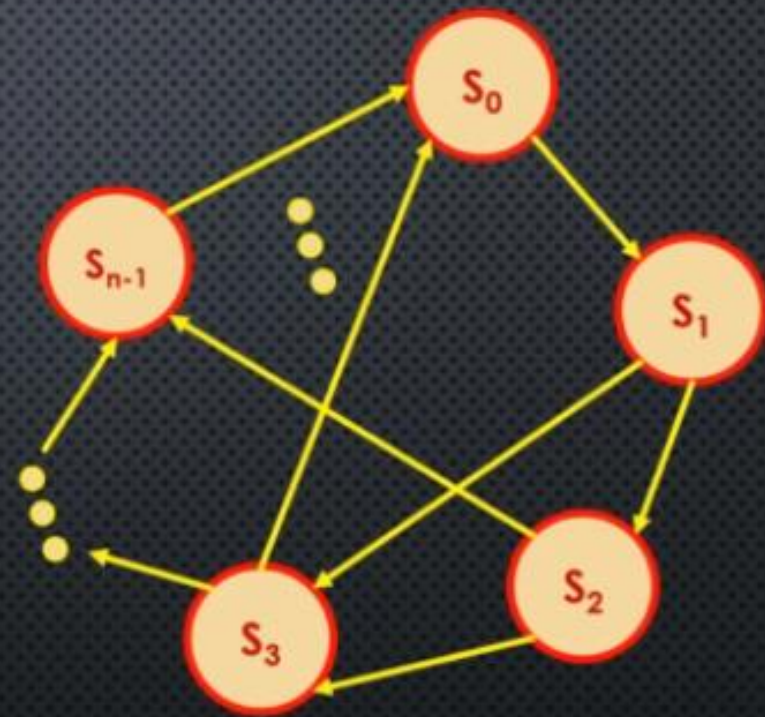
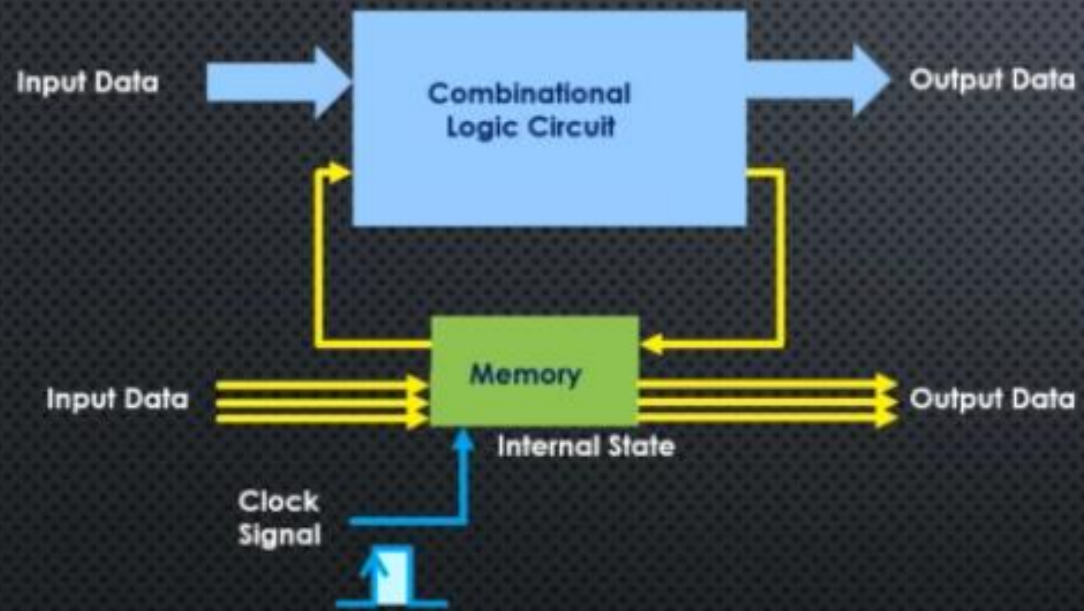
Part Selection

Part: **ZedBoard Zynq Evaluation and Development Kit (xc7z020clg484-1)**

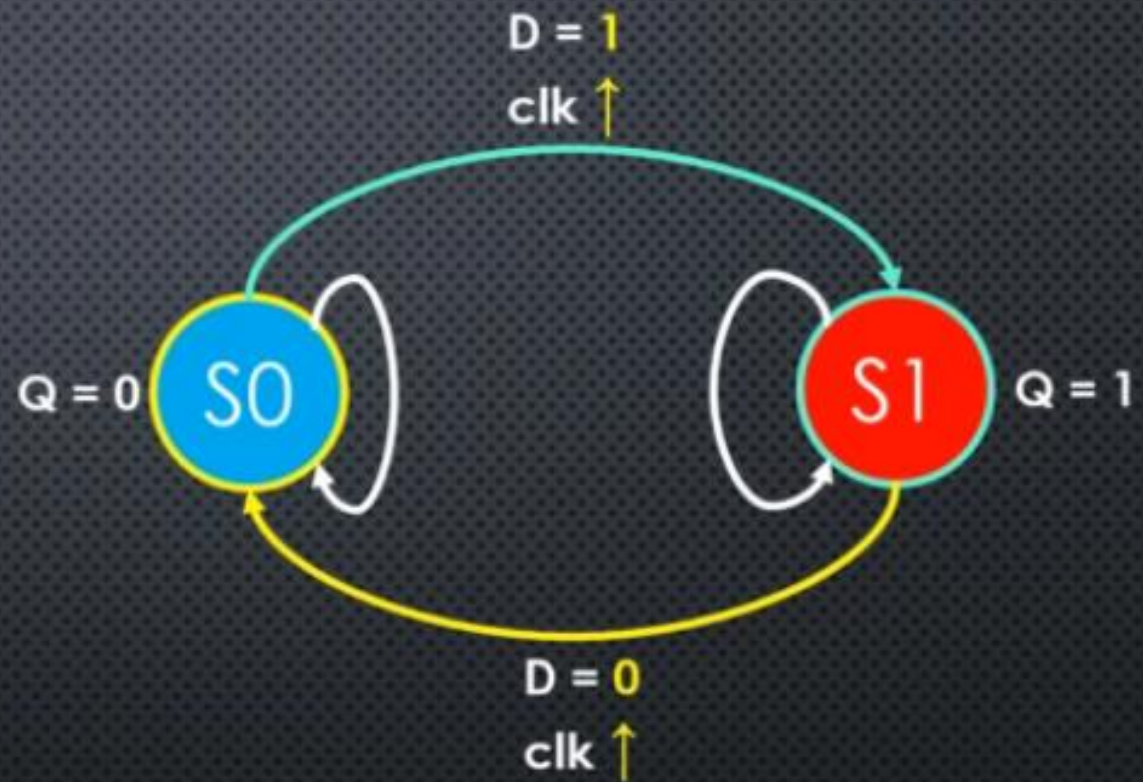
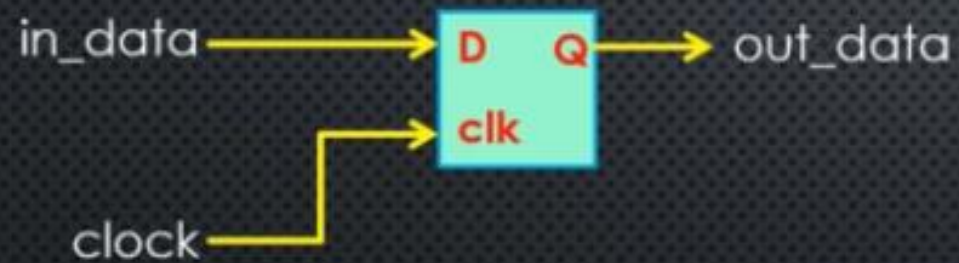
☐ SDAccel Bottom Up Flow

< Back Finish Cancel

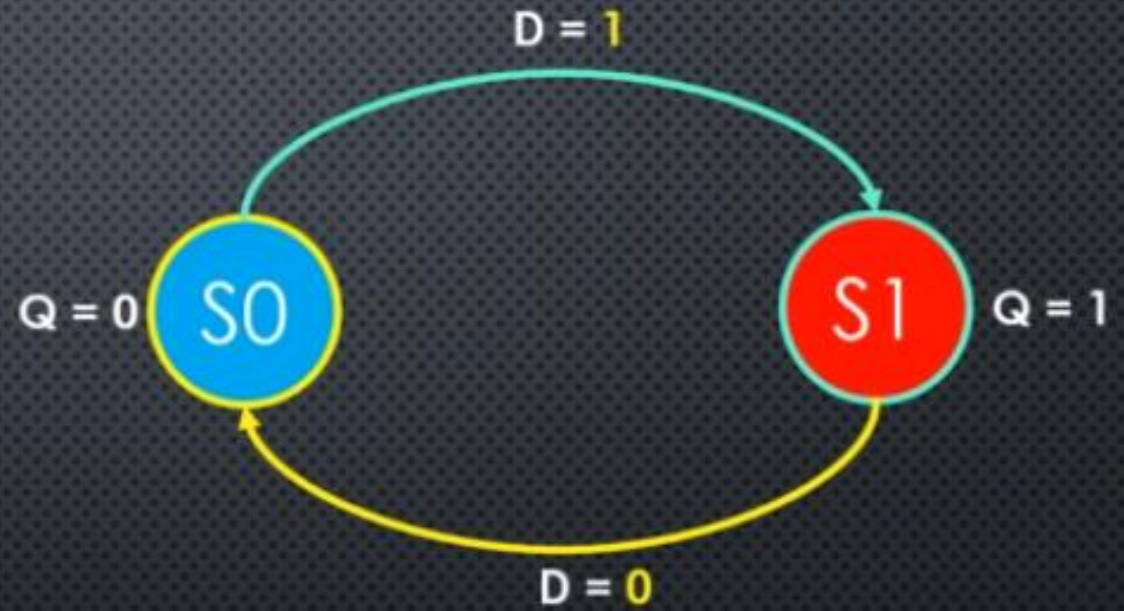
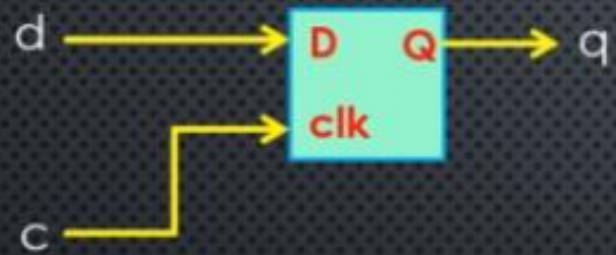
STATE



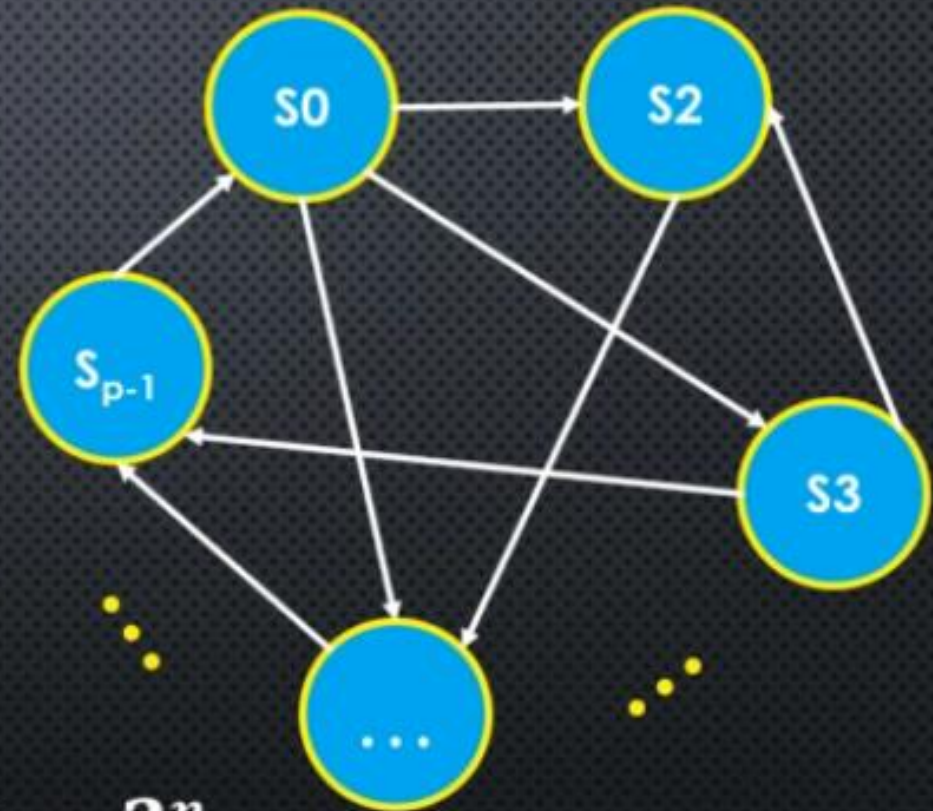
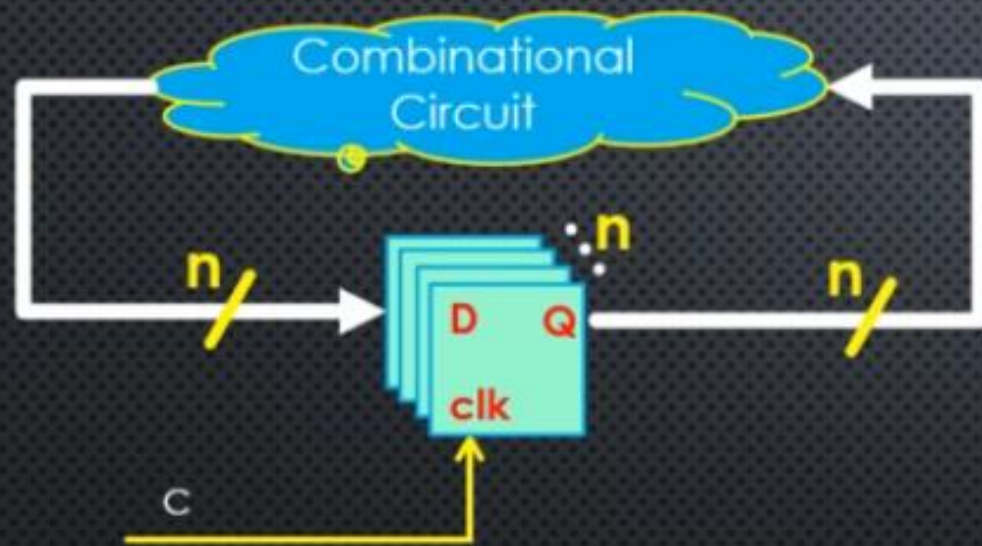
DFF STATE



STATE DIAGRAM

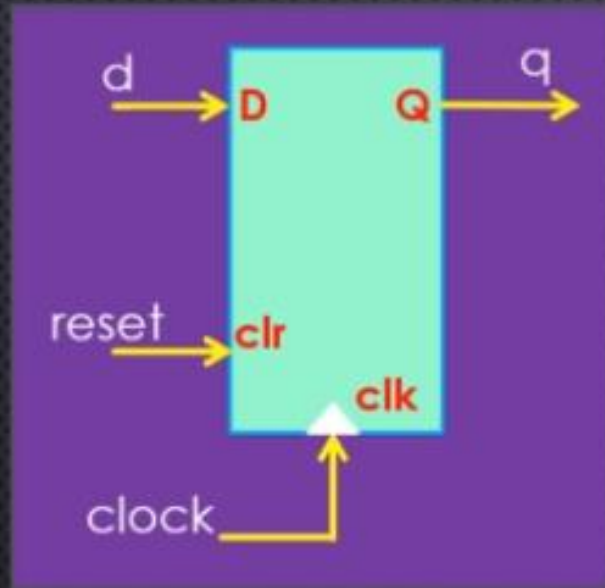


STATE DIAGRAM ATE



$$p = 2^n$$

RESET SIGNAL



❖ Active High

❖ Active Low

Active-High reset

reset

q

0 or 1

0 or 1

Active-Low reset

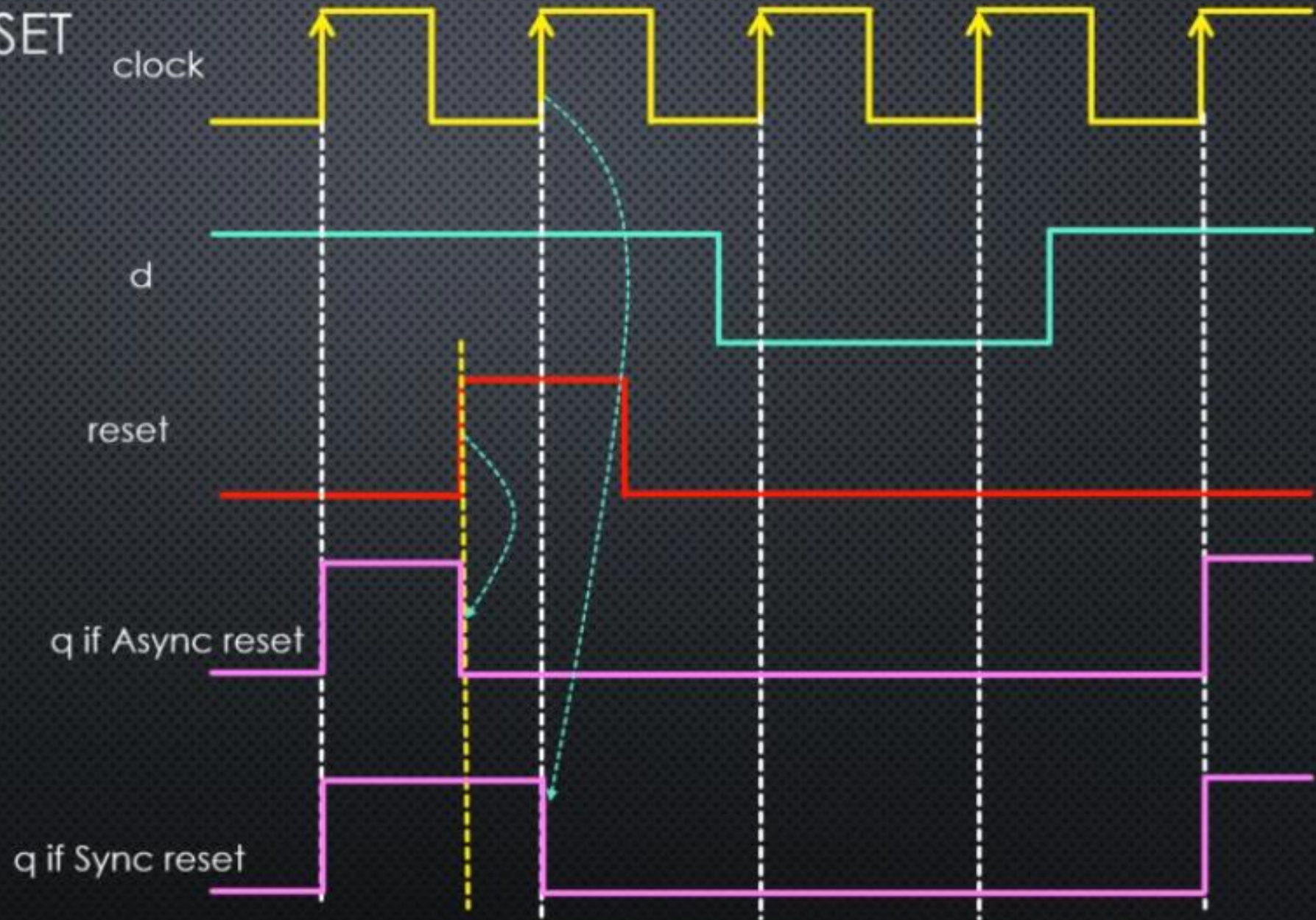
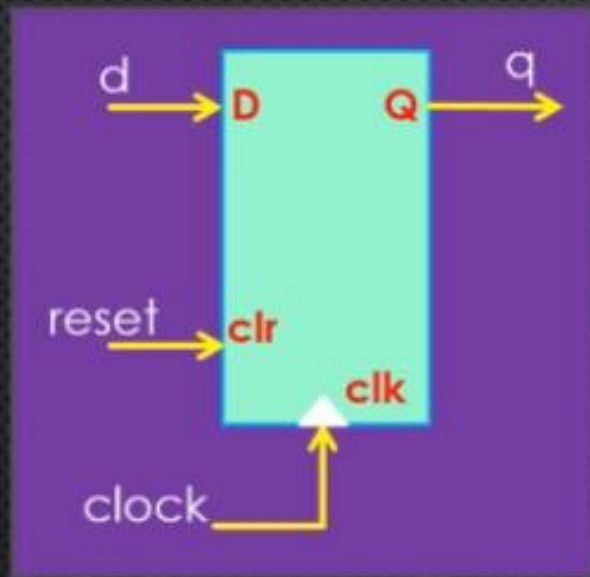
reset

q

0 or 1

0 or 1

ASYNC/SYNC RESET



RESET SIGNAL CONFIGURATION IN HLS

reset: none
control
state
all

reset async:

reset level: low
high

Solution Settings (solution1)

General
Synthesis
Cosimulation
Export

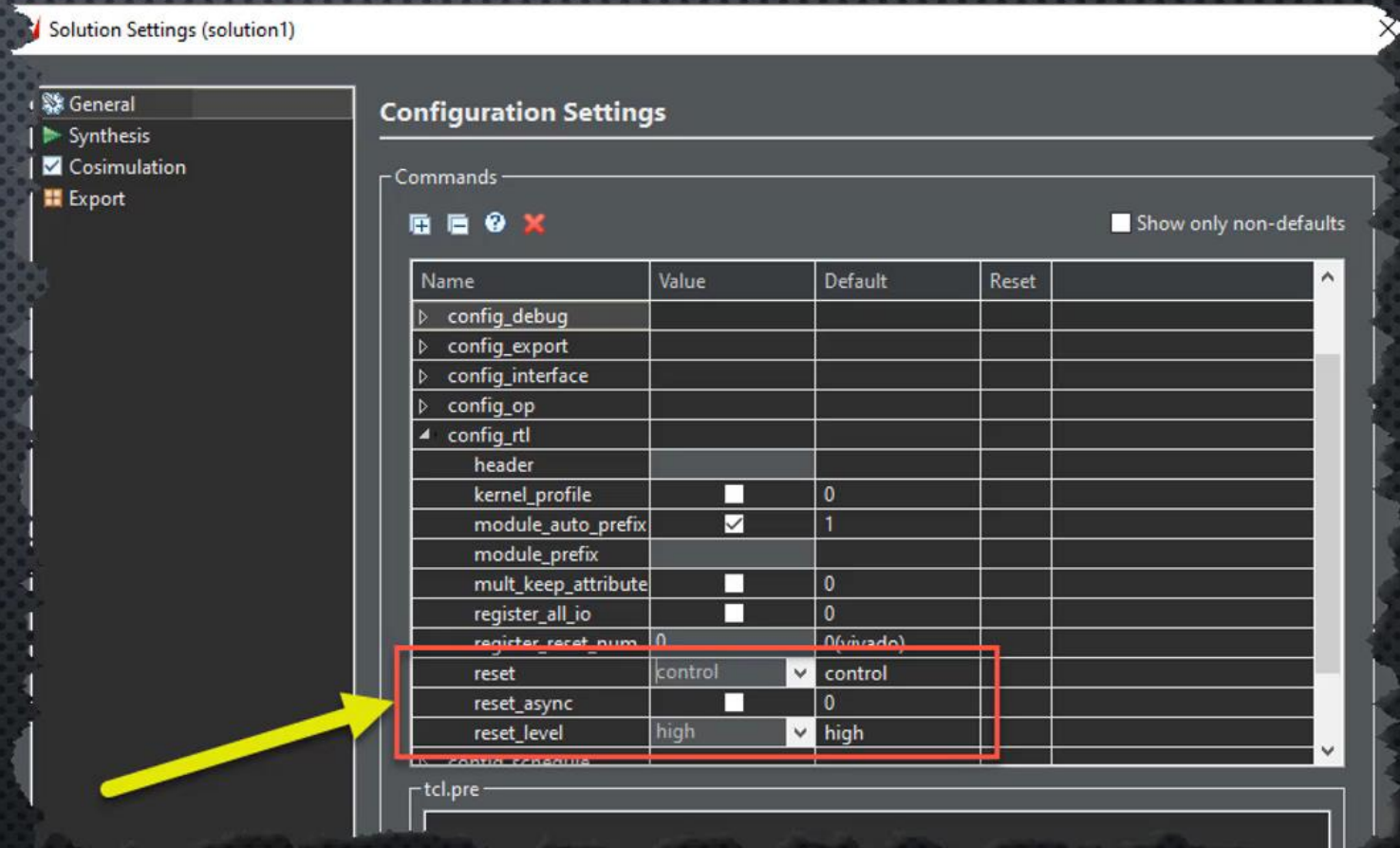
Configuration Settings

Commands

Show only non-defaults

Name	Value	Default	Reset
config_debug			
config_export			
config_interface			
config_op			
config_rtl			
header			
kernel_profile	<input type="checkbox"/>	0	
module_auto_prefix	<input checked="" type="checkbox"/>	1	
module_prefix			
mult_keep_attribute	<input type="checkbox"/>	0	
register_all_io	<input type="checkbox"/>	0	
register_reset_num	0	0 (invalid)	
reset	control	control	
reset_async	<input type="checkbox"/>	0	
reset_level	high	high	

tcl.pre



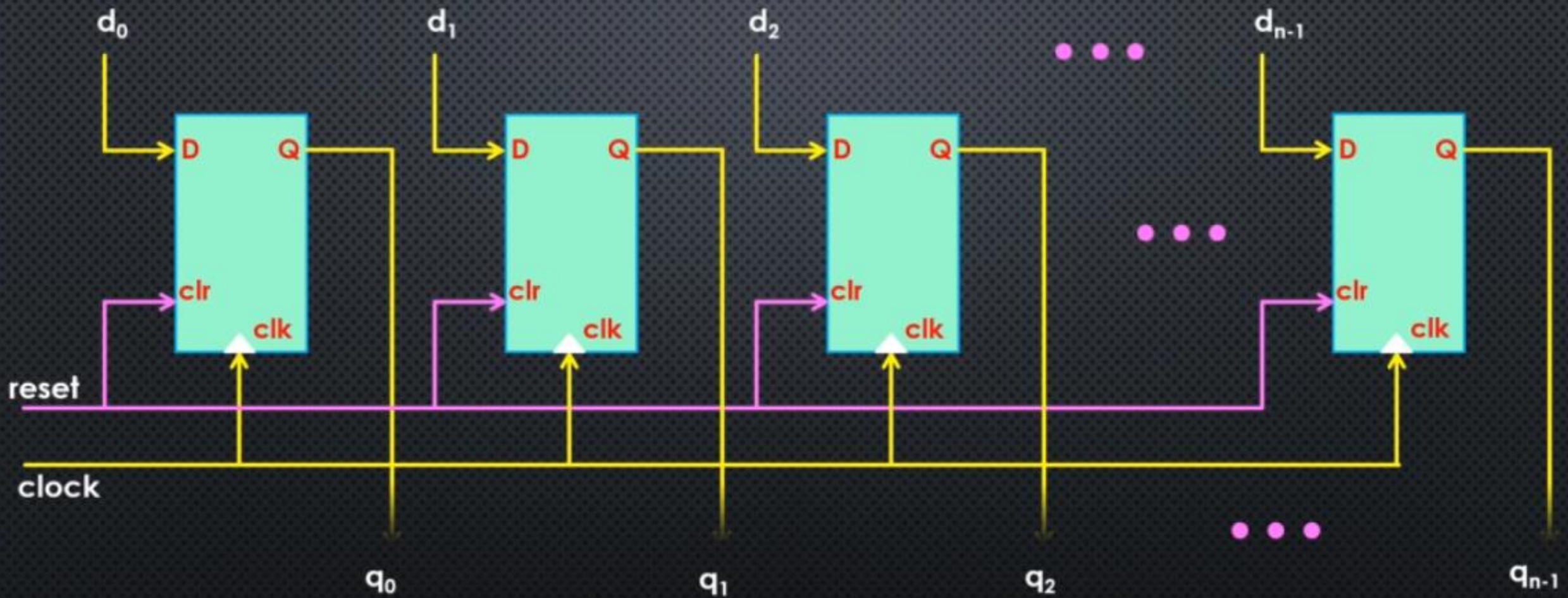
RESET SIGNAL IN HLS

```
void dffs_register(bool d, ap_uint<3> &q) {  
#pragma HLS INTERFACE ap_none port=d  
#pragma HLS INTERFACE ap_none port=q  
#pragma HLS INTERFACE ap_ctrl_none port=return
```

```
    static ap_uint<3> reg = 0b000;  
  
    reg = reg >> 1;  
    reg[2] = d;  
  
    q = reg;  
}
```

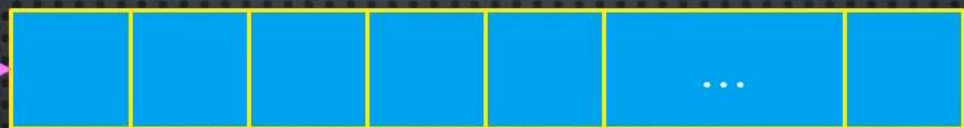


SIMPLE REGISTER STRUCTURE



REGISTERS IN HLS

```
void func(...) {  
    static int x;  
    ...  
    ...  
}
```



32 bits

clk
rst

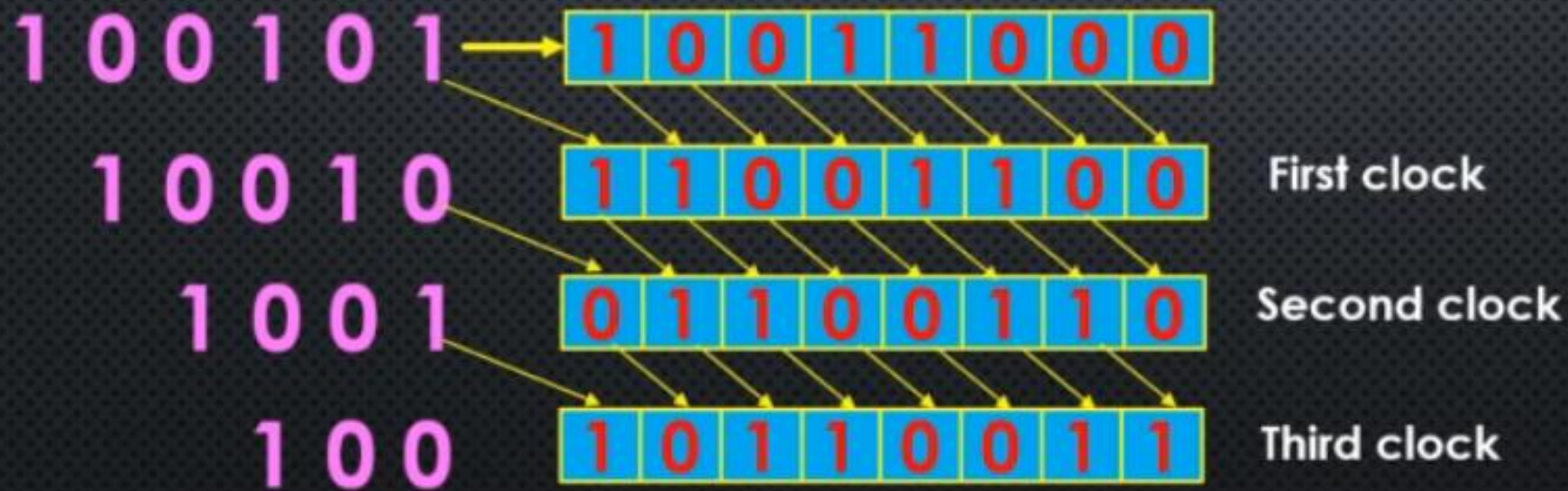
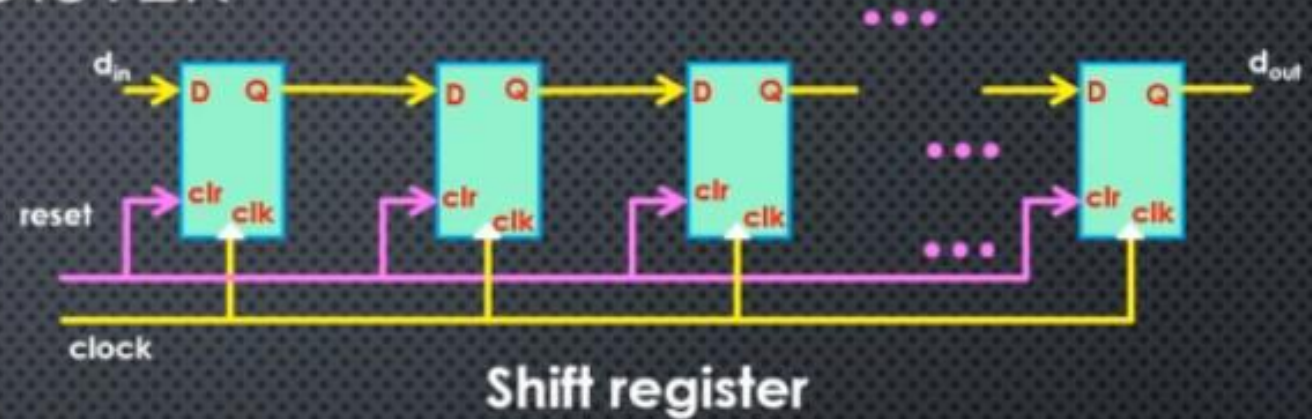


```
for (unsigned int i = 0; i < 1024; i++)  
{  
    ...  
}
```

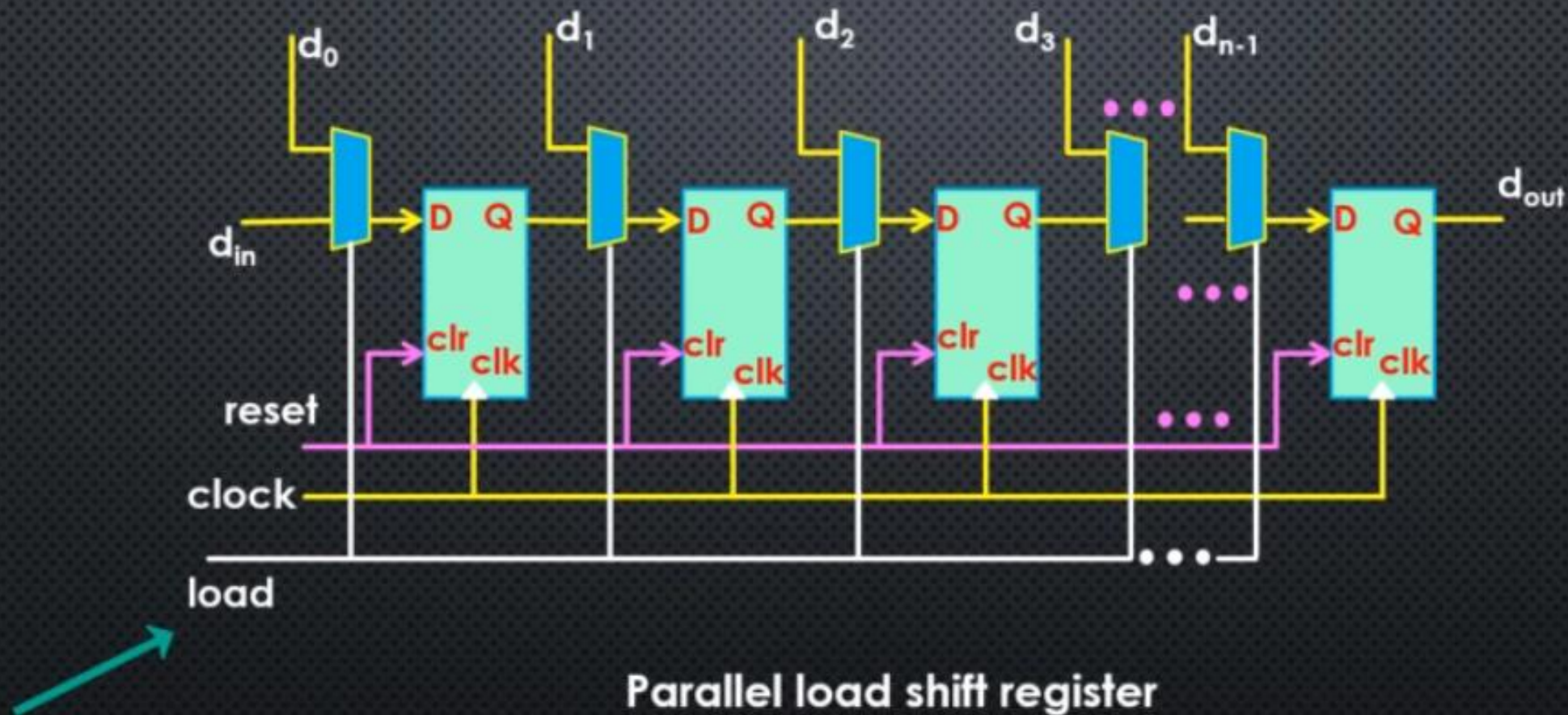
10-bit register



REGISTER TYPES - SHIFT REGISTER



REGISTER TYPES – PARALLEL LOAD SHIFT REGISTER



REGISTER TYPES IN HLS

```
void func(...) {  
    static ap_int<10> x;
```

```
    ...
```

```
    x = 2; —————→ Parallel load
```

```
    ...
```

```
    x = x >> 2; —————→ Shift right
```

```
    ...
```

```
    x = x << 4; —————→ Shift left
```

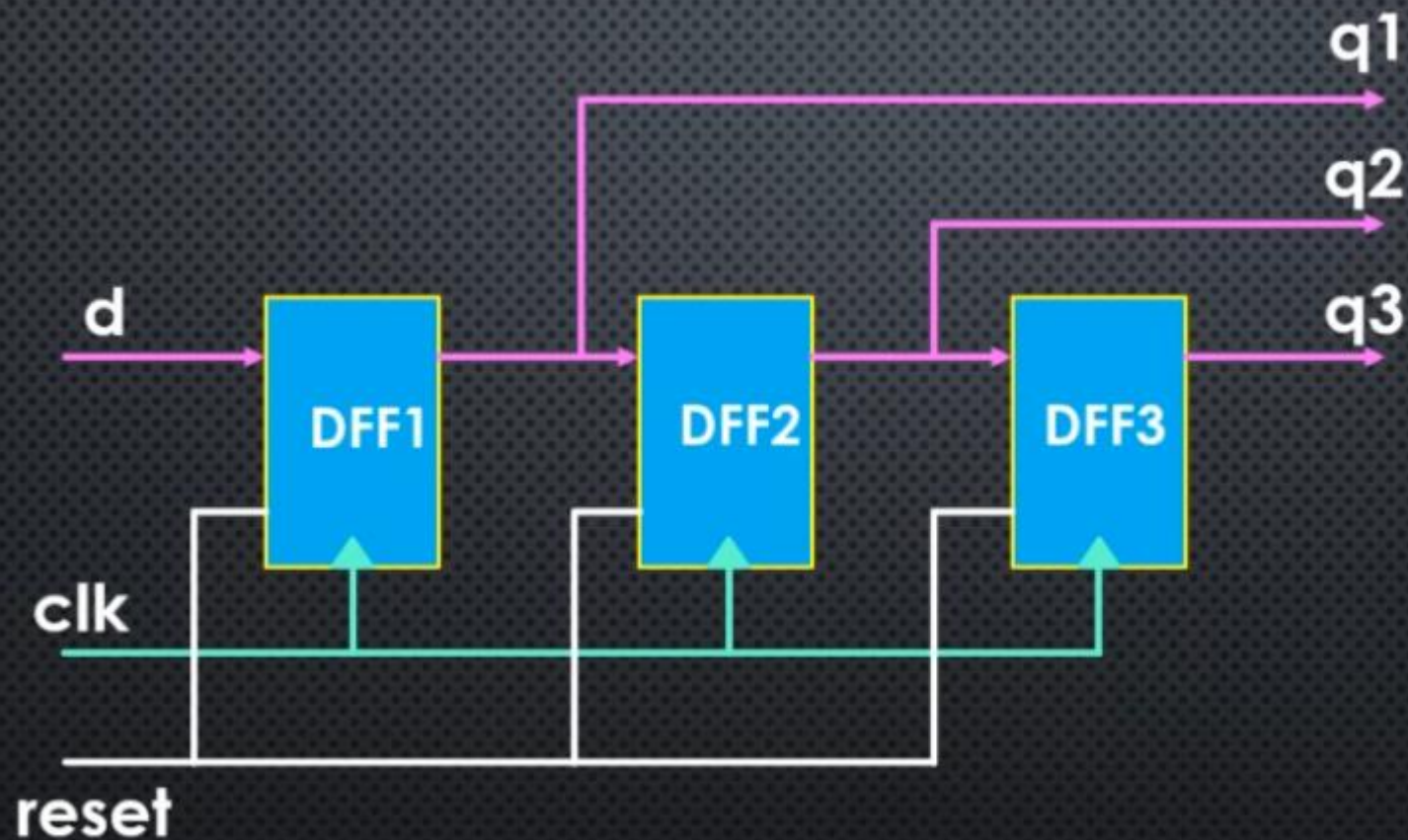
```
    ...
```

```
    x[5] = 10; —————→ Bit access
```

```
    ...
```

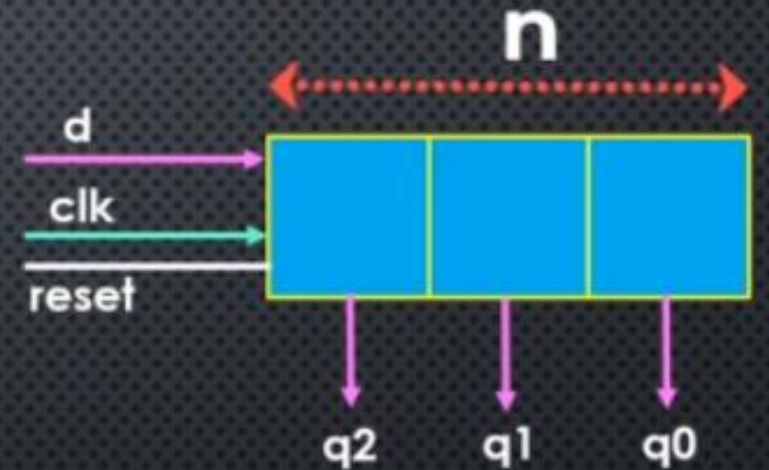
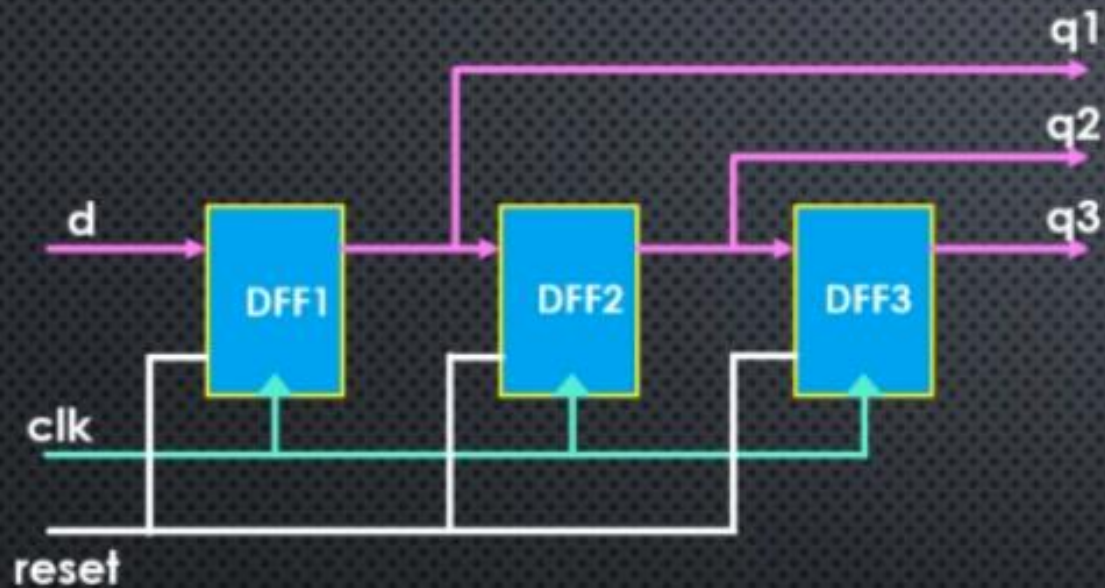
```
}
```

BASIC EXAMPLE WITH VITIS-1



BASIC EXAMPLE WITH VITIS-2

TARGET DESIGN



Any Question...

Thank you