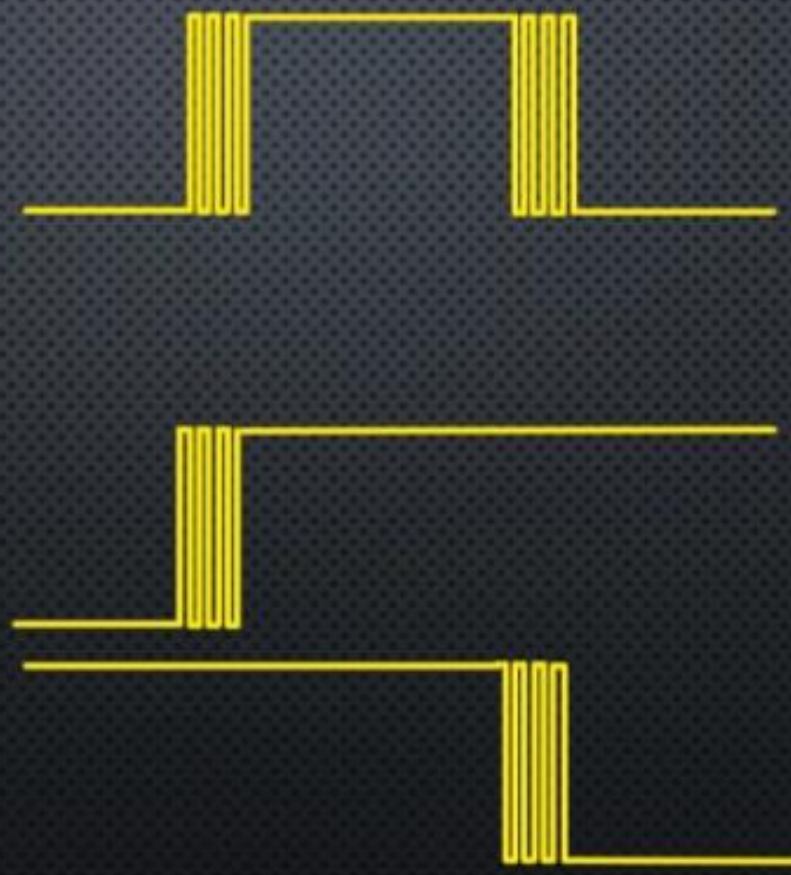
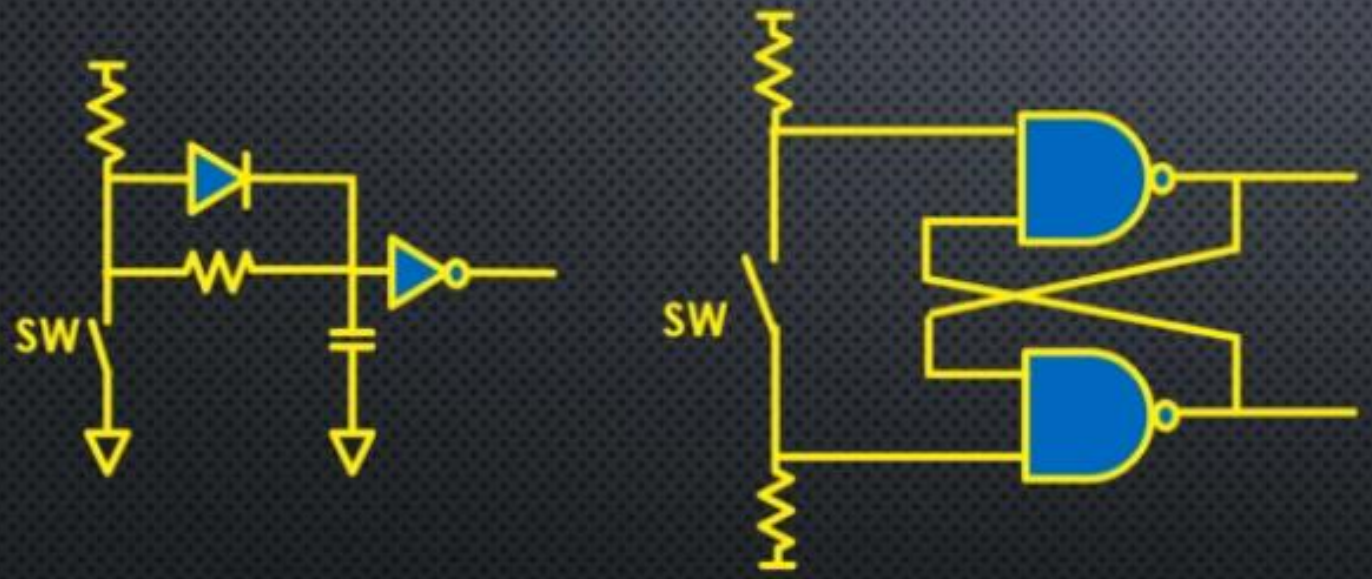
DEBOUNCER

Lecture-10

SWITCH BOUNCING



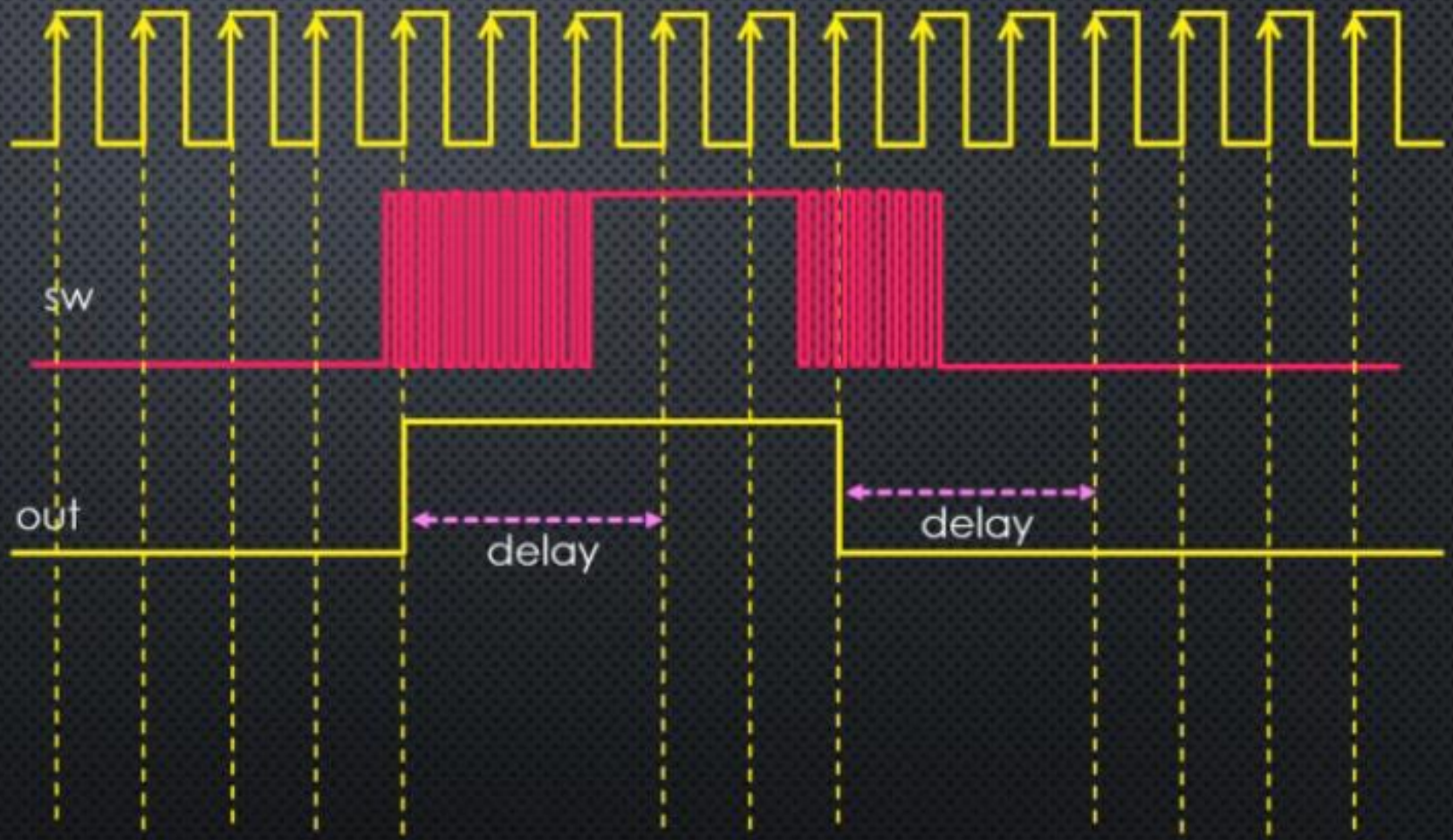
SWITCH DEBOUNCING –SOLUTION




```
...

currentBtnState = read_button(btnPin);

if (currentBtnState != previousBtnState) {
        lastDbncTime = millis();
}

if ((millis() - lastDbncTime) > dbnc_delay) {
        currentLedState = currentBtnState;
}
....
```

SWITCH DEBOUNCING – HLS SOLUTION - IDEA

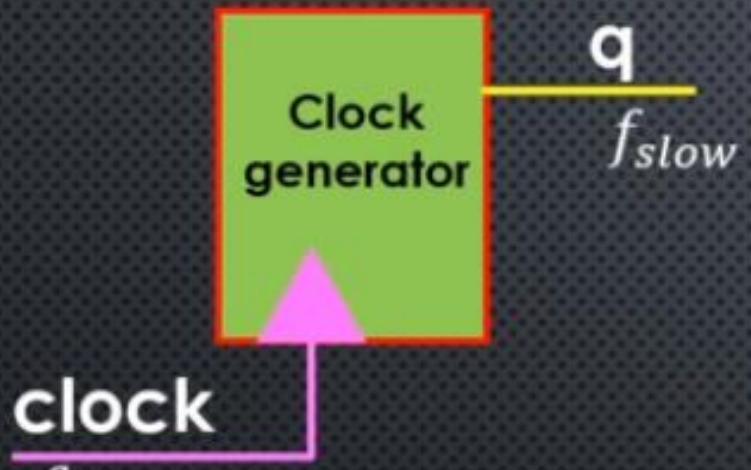
Slow Sampling DFF



CLOCK GENERATOR

CLOCK DIVIDER

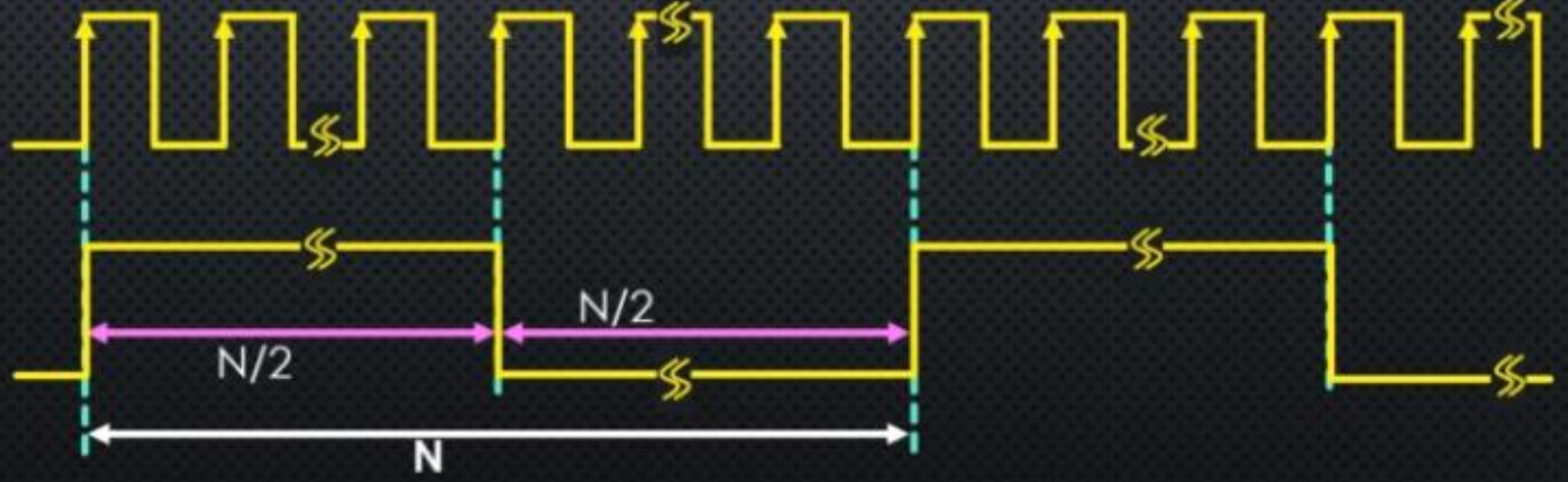
$$f_{slow} = \frac{1}{N} f_{fast}$$



$f_{fast} = 100MHz$

$N = 1000$

$f_{slow} = 100\ KHz$



CLOCK DIVIDER

$$N = 20$$

$$f_{fast} = 100MHz \Rightarrow f_{slow} = 5\ MHz$$

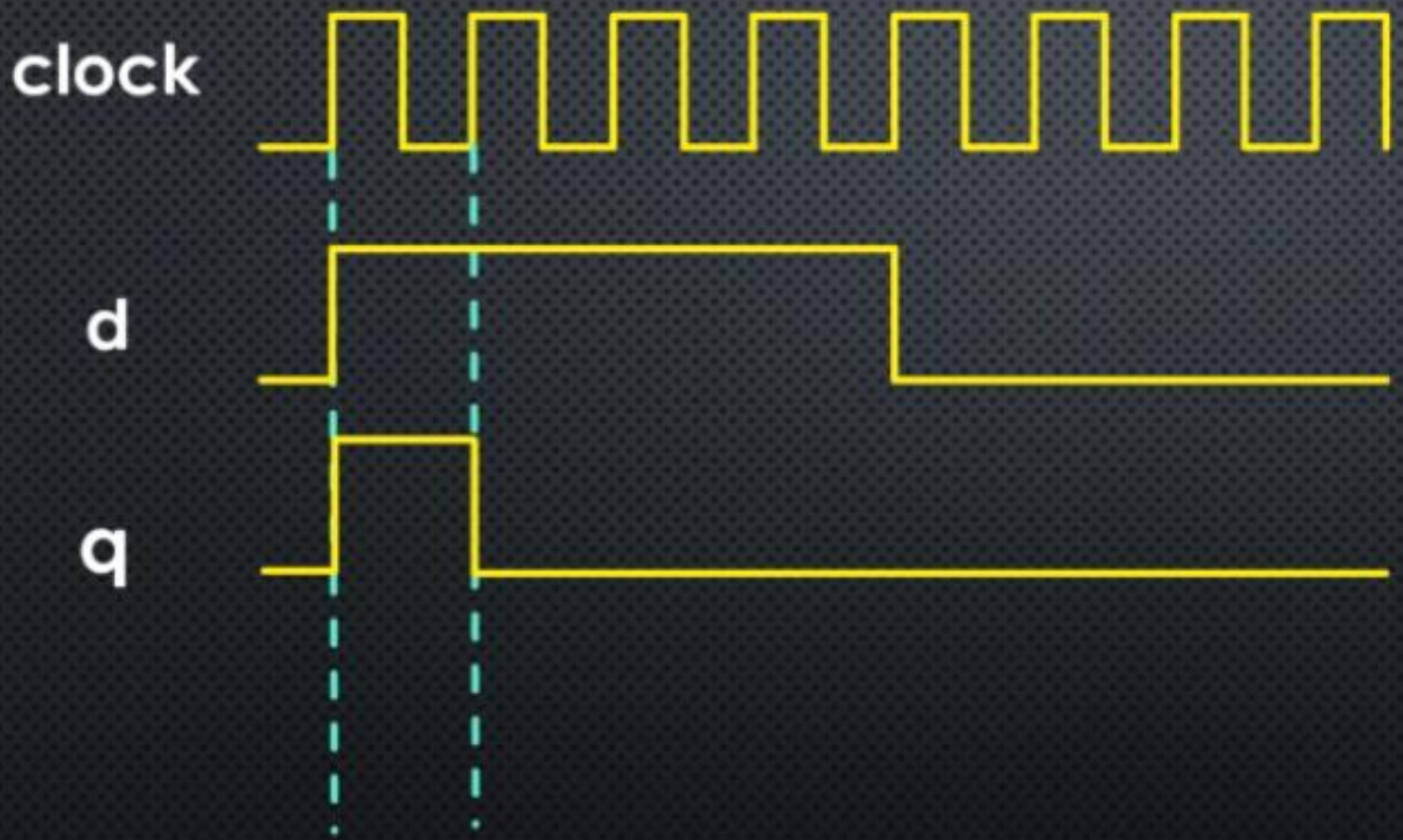
$$T_{fast} = \frac{1}{100} = 10ns$$

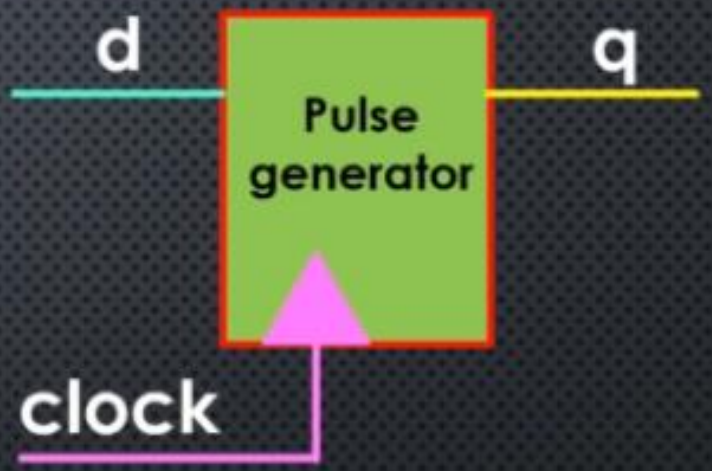
$$T_{slow} = \frac{1}{5} = 200ns$$

duty cycle =(10/20)%=50%

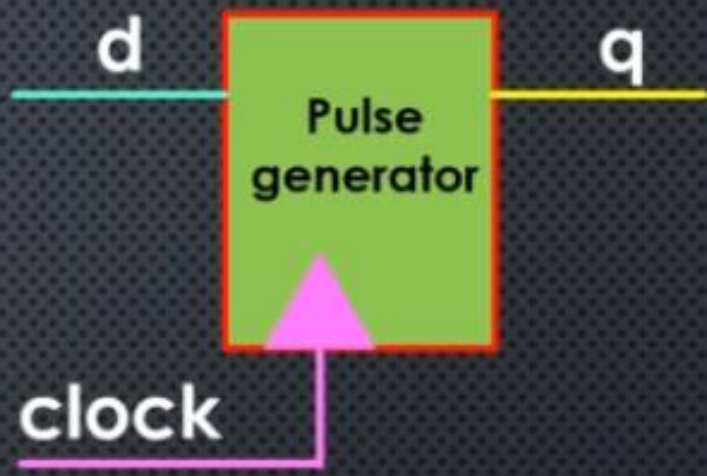
SINGLE-CYCLE PULSE GENERATOR

SINGLE CYCLE PULSE GENERATOR

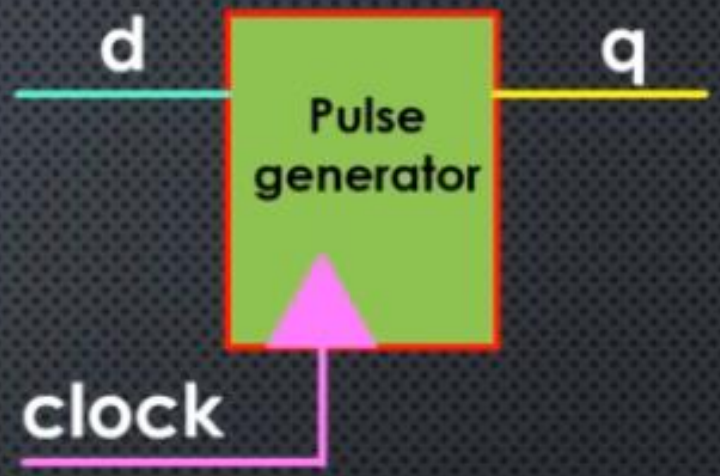




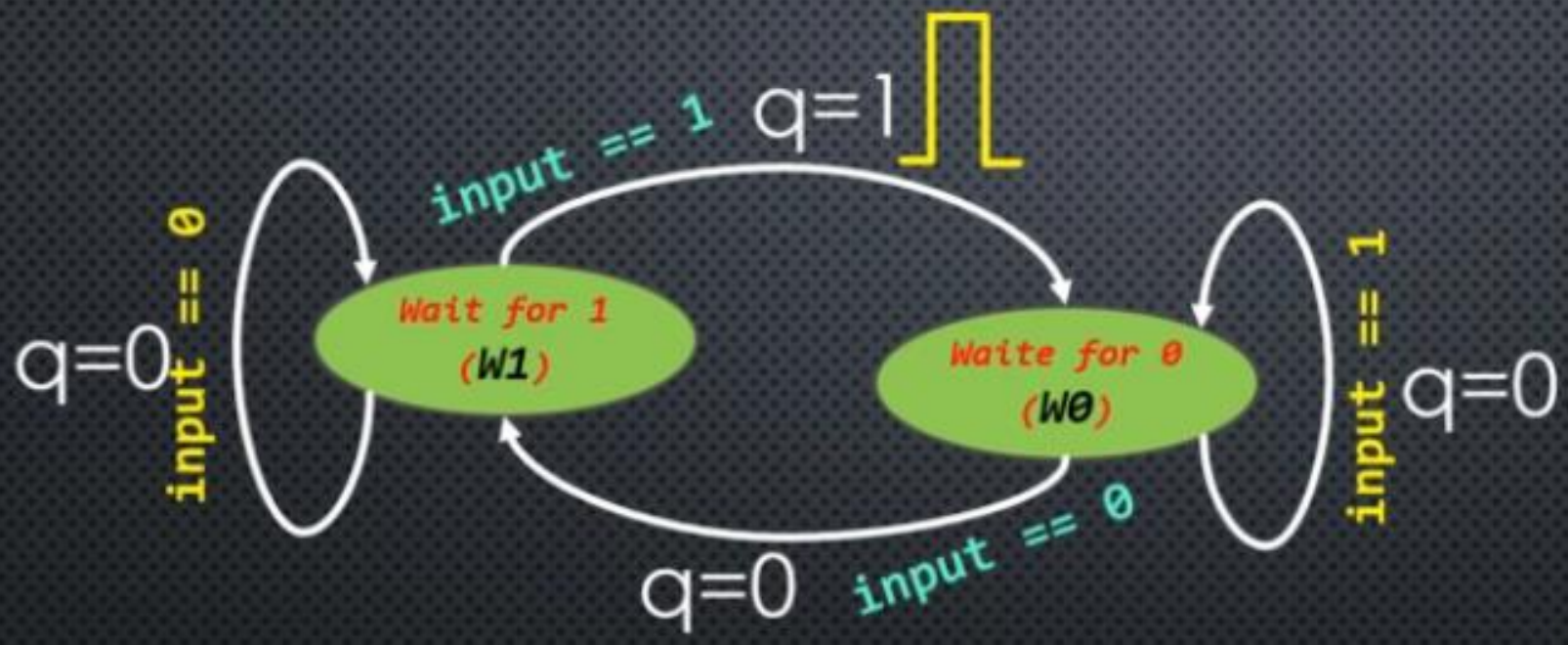
PULSE GENERATOR - FSM

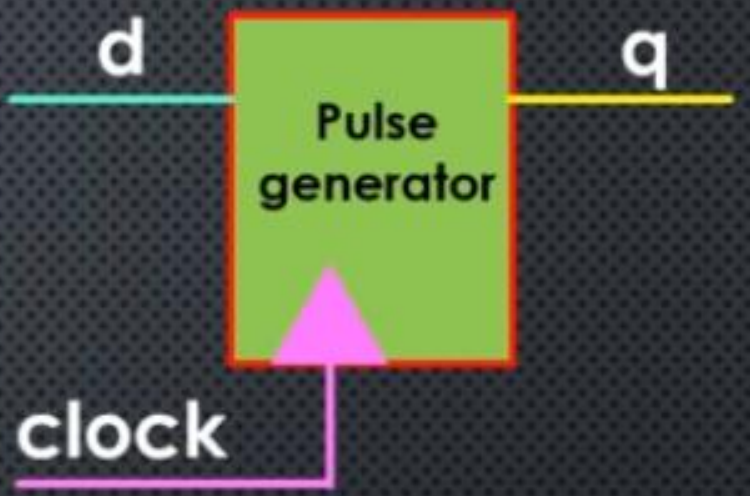


PULSE GENERATOR - FSM



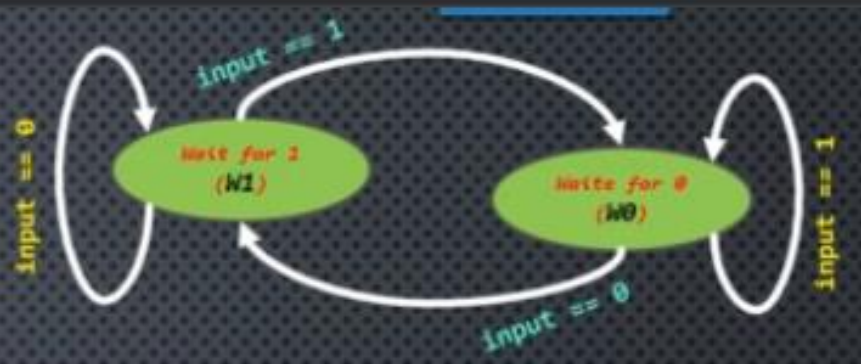
PULSE GENERATOR - FSM





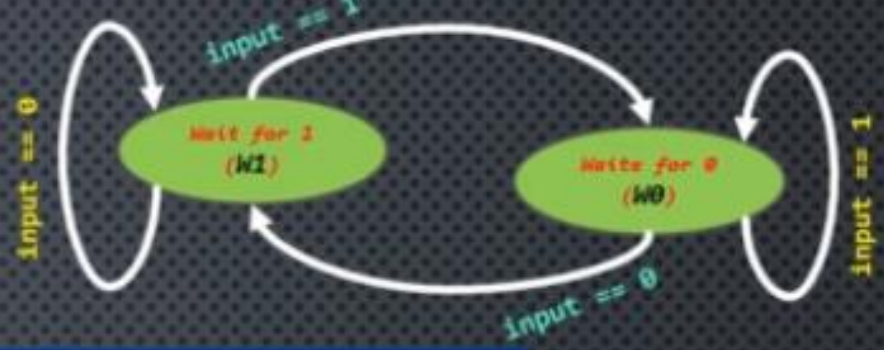
PULSE GENERATOR - HLS

```
typedef enum{W1, W0} pulse_gen_states_type;

void pulse_generator(bool input, bool &pulse) {
#pragma HLS INTERFACE ap_none port=input
#pragma HLS INTERFACE ap_none port=pulse
#pragma HLS INTERFACE ap_ctrl_none port=return

  static pulse_gen_states_type state = idle;

  pulse_gen_states_type next_state;
  bool next_pulse;

  switch(state) {
  case W1:
    ...
    break;
  case W0:
    ...
    break;
  default:
    break;
  }
  state = next_state;
  pulse = next_pulse;
}
```



PULSE GENERATOR - HLS

```
typedef enum{W1, W0} pulse_gen_states_type;

void pulse_generator(bool input, bool &pulse) {
#pragma HLS INTERFACE ap_none port=input
#pragma HLS INTERFACE ap_none port=pulse
#pragma HLS INTERFACE ap_ctrl_none port=return

  static pulse_gen_states_type state = idle;

  pulse_gen_states_type next_state;
  bool next_pulse;

  switch(state) {
  case W1:
    ...
    break;
  case W0:
    ...
    break;
  default:
    break;
  }
  state = next_state;
  pulse = next_pulse;
}
```



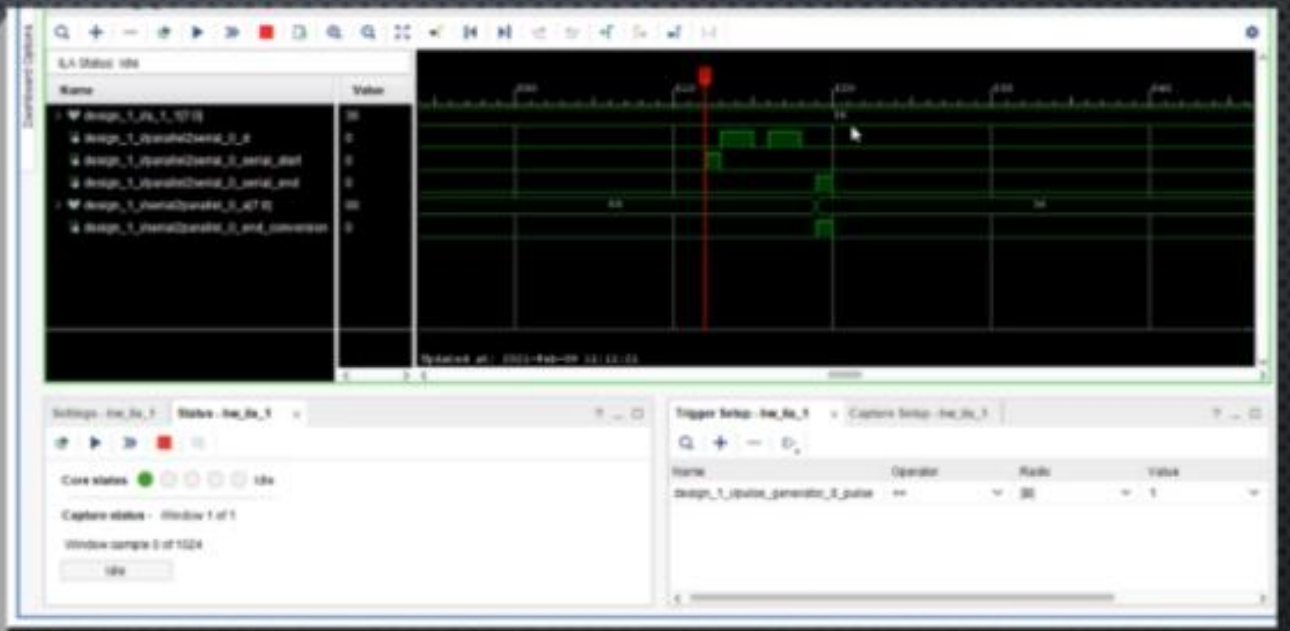

```
switch(state) {
case W1:
  if (input == 1) {
    next_state  = W0;
    next_pulse = 1;
  } else {
    next_state  = W1;
    next_pulse = 0;
  }
  break;
case W0:
  if (input == 1) {
    next_state = W0;
    next_pulse = 0;
  } else {
    next_state = W1;
    next_pulse = 0;
  }
  break;
default:
  break;
}
```

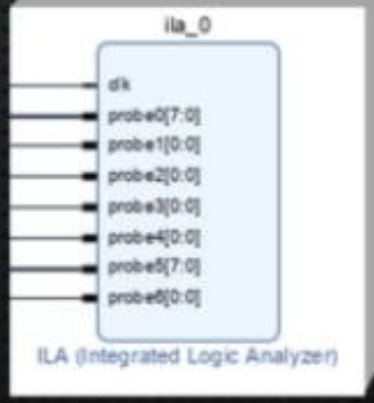
ILA

OBJECTIVES

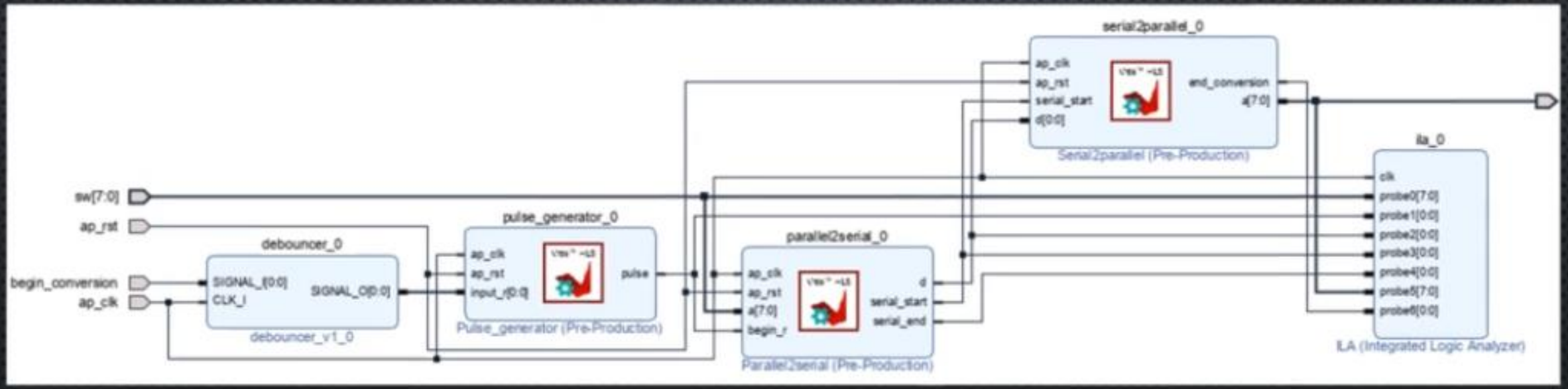
- Understanding the concept behind the ILA IP
- Customising the ILA IP
- Using the ILA IP to monitor signals in an HLS design

INTEGRATED LOGIC ANALYSER (ILA) IP

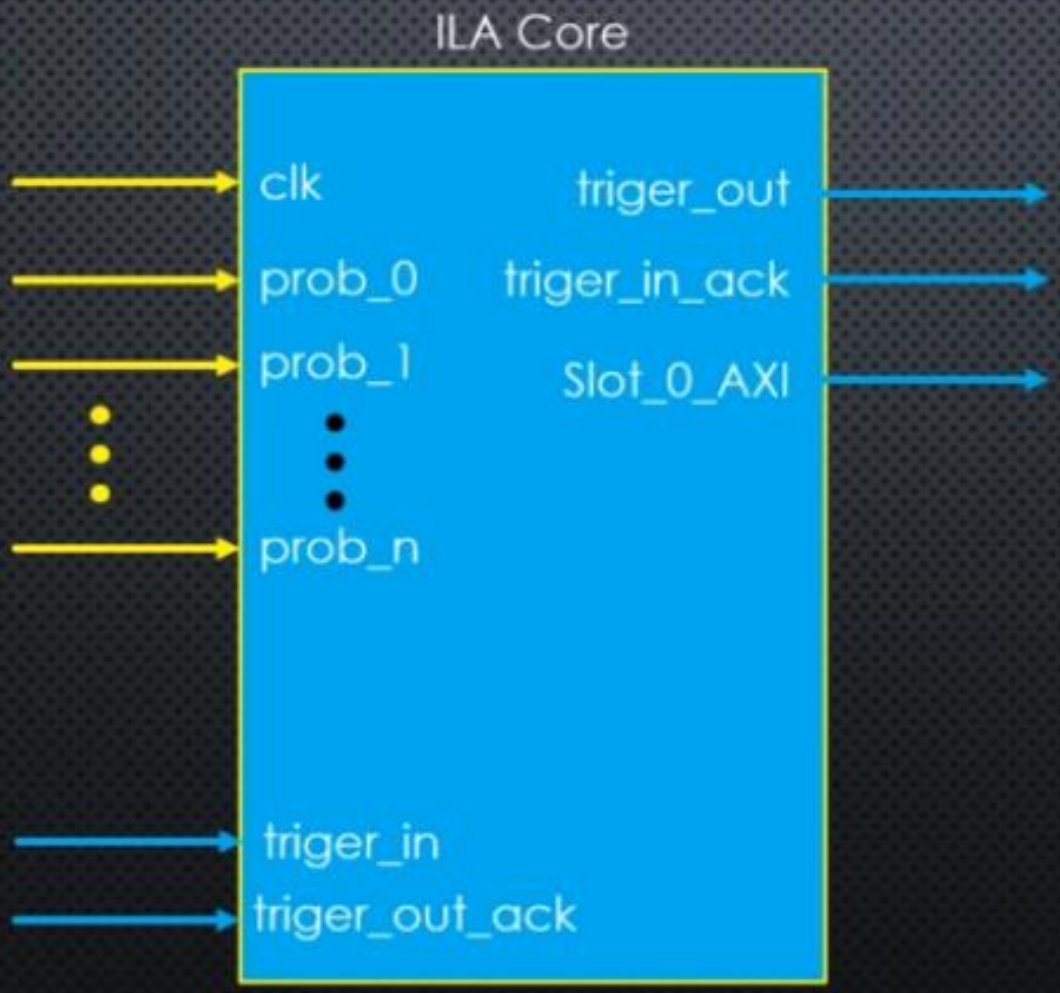




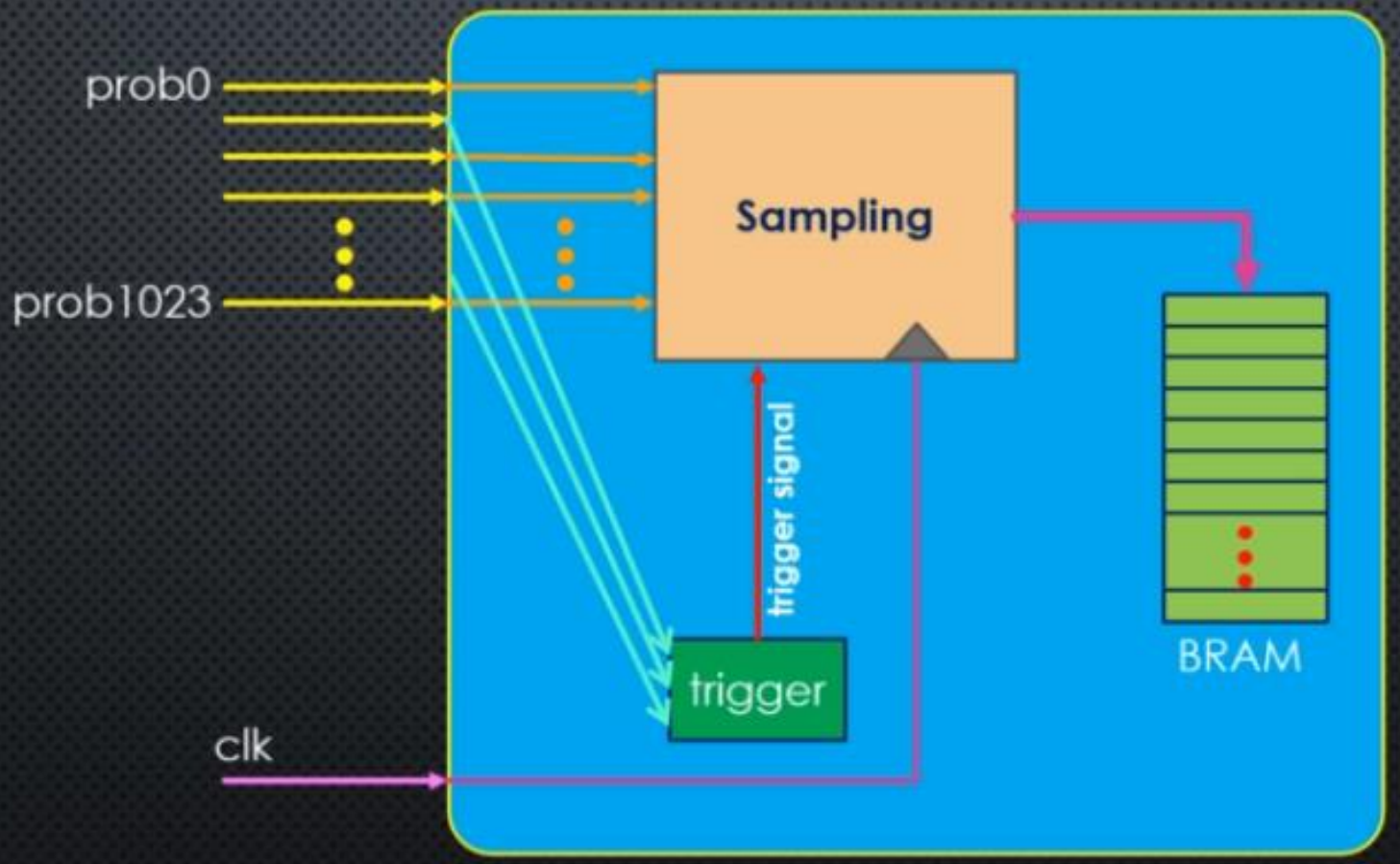
ILA IN A DESIGN



ILA CORE SYMBOL



HOW DOES ILA WORK?



ILA DEBUG FLOW

Design Time

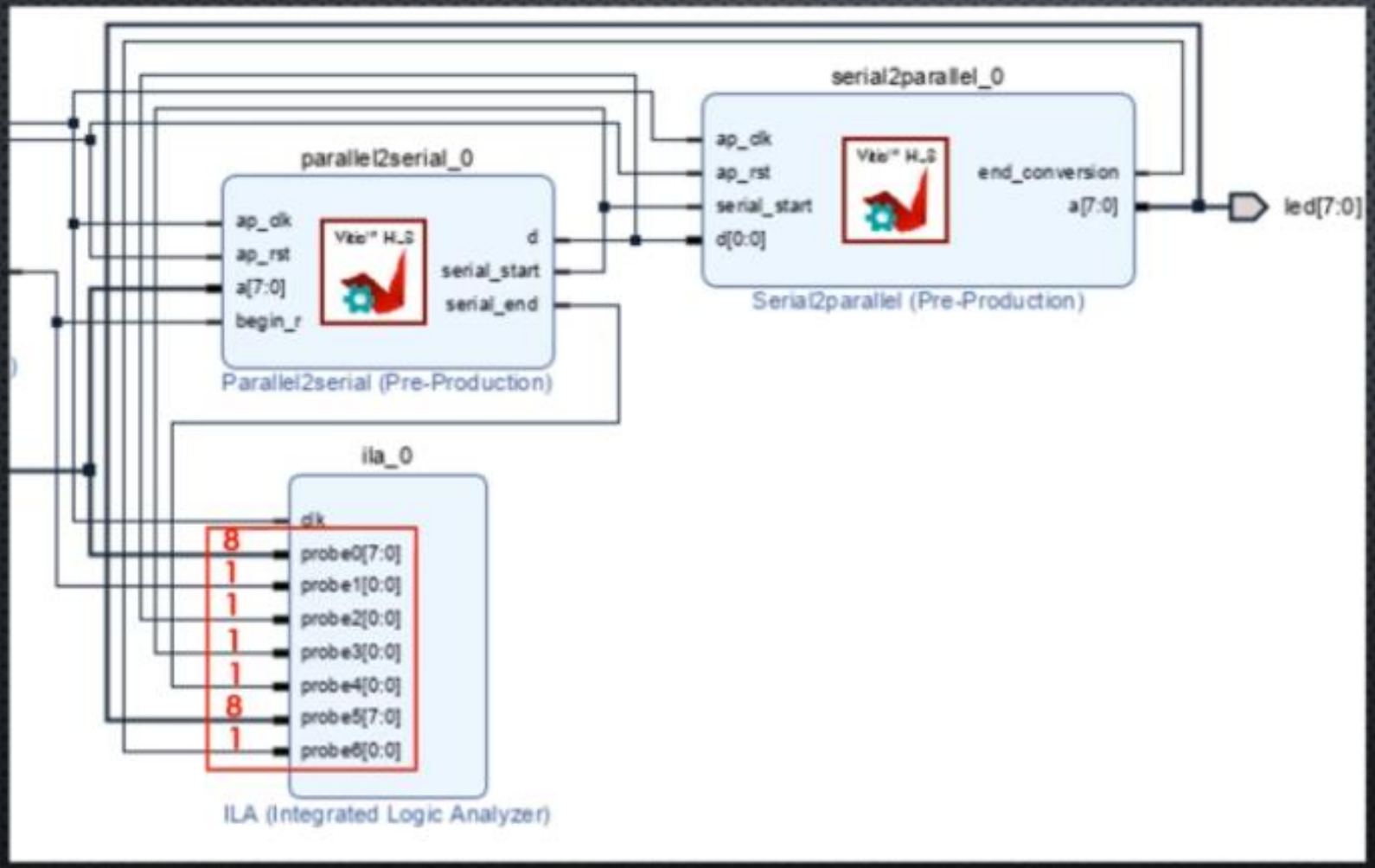
- Instantiation
- Customisation
- Connection

FPGA Programming

Runtime

- Define Trigger
- Run
- Inspection

ILA IN PARALLEL-TO-SERIAL-TO-PARALLEL



probe0	8	a (sw)
probe1	1	begin_r
probe2	1	d
probe3	1	serial_start
probe4	1	serial_end
probe5	8	a (led)
probe6	1	end_conversion

Any Question...

Thank you