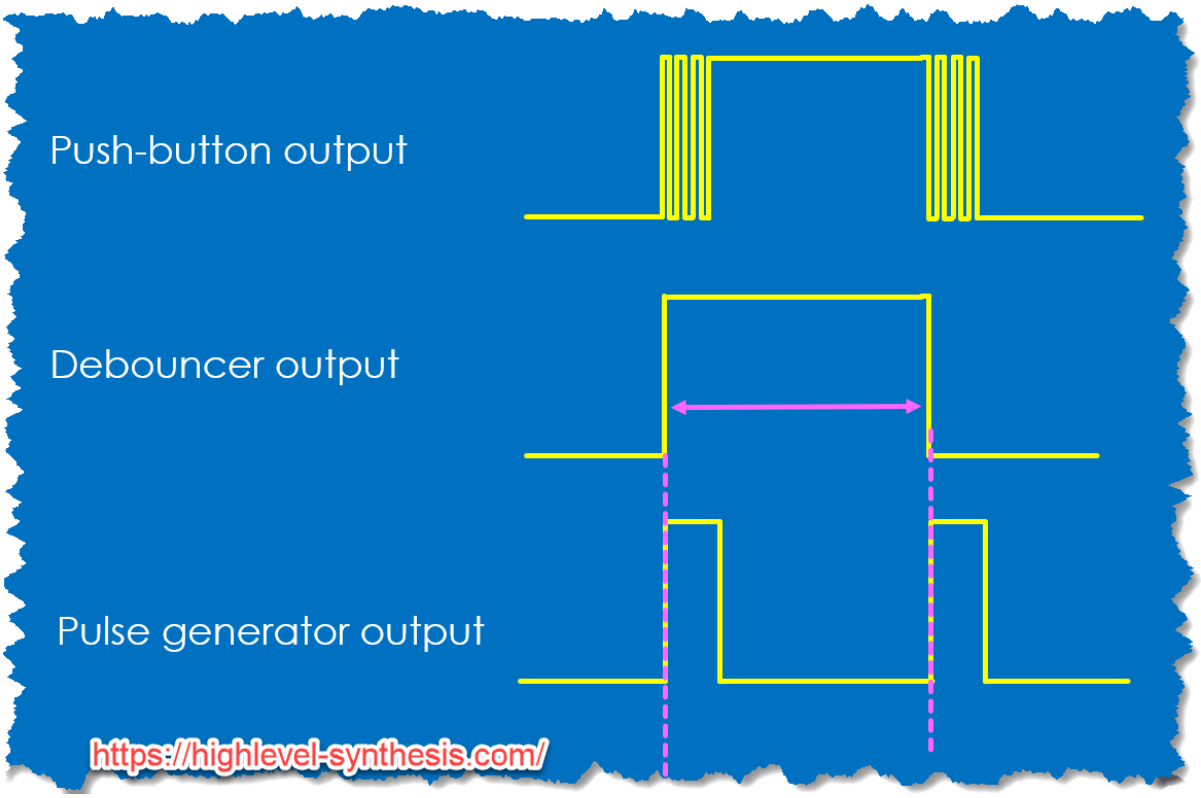
1- The idea is to use both rising edge and falling edge resulted from the pressed button, as shown in the following figure. In this case the pulse generator should generate two pulses. Note that the distance between two pulses is not deterministic.




The following code shows the changes in pulse generator and the dice roller IPs.

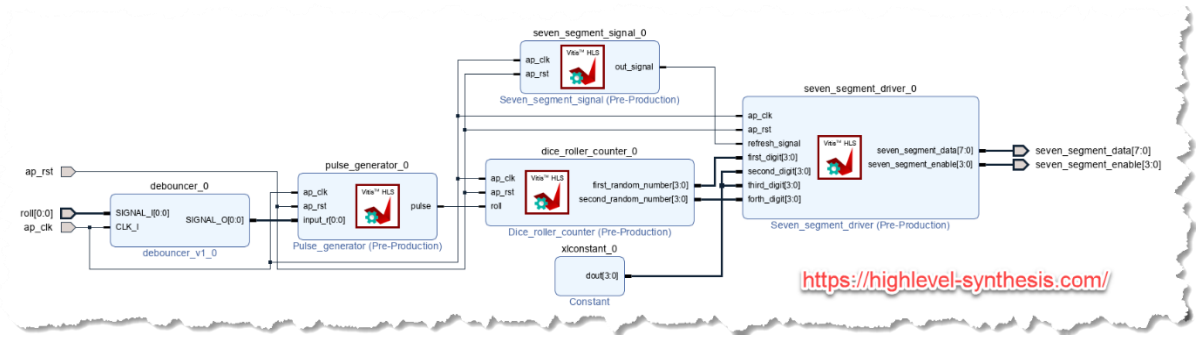
```
typedef enum{w1, w0} pulse_gen_states_type;

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 = w1;


      pulse_gen_states_type next_state;
      bool 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 = 1;
		}
		break;
	default:

		break;
	}

	state = next_state;
	pulse = next_pulse;

}
```

```
unsigned int get_rand()
{


	static unsigned int count = 0;

	unsigned int r;
	count++;
	r = count%6+1;

	return r;
}


void dice_roller_counter(
		bool         roll,
		ap_uint<4>  &first_random_number,
		ap_uint<4>  &second_random_number) {
#pragma HLS INTERFACE ap_none port=roll
#pragma HLS INTERFACE ap_none port=first_random_number
#pragma HLS INTERFACE ap_none port=second_random_number
#pragma HLS INTERFACE ap_ctrl_none port=return
```

```
#pragma HLS PIPELINE

	static bool random_number_index = 0;
	static unsigned int rand_number_1 = 0;
	static unsigned int rand_number_2 = 0;

	unsigned int r = get_rand();
	if (roll == 1) {
		if (random_number_index == 0) {
			rand_number_1 = r;
			random_number_index = 1;
		} else {
			random_number_index = 0;
			rand_number_2 = r;
		}
	}


	first_random_number  = rand_number_1;
	second_random_number = rand_number_2;

}
```

And this is the Vivado design




2- To solve this problem, we can simply use two LFSR with two different equations.