1- The following code solved the problem

```
#include "clock_generator.h"



typedef enum{zero, one} clock_state_type;

void clock_generator(ap_uint<16> divisor, bool start, bool &slow_clock_signal)
{
#pragma HLS INTERFACE ap_none port=slow_clock_signal
#pragma HLS INTERFACE ap_none port=divisor
#pragma HLS INTERFACE ap_none port=start
#pragma HLS INTERFACE ap_ctrl_none port=return


   unsigned int CLOCK_FREQUENCY_COUNTER = (divisor>>1) - 1;
   static clock_state_type state = zero;
   static unsigned int     counter = 0;

   clock_state_type next_state;
   unsigned int     next_counter;




   bool slow_clock_signal_local;

   switch(state) {
   case zero:
         if (counter == 0) {
               next_counter       = CLOCK_FREQUENCY_COUNTER;
               next_state         = one;
         } else {
               next_counter       = counter-1;
               next_state         = zero;
         }
         slow_clock_signal_local = 0;
         break;
   case one:
         if (counter == 0) {
               next_counter       = CLOCK_FREQUENCY_COUNTER;
               next_state         = zero;
         } else {
               next_counter       = counter-1;
               next_state         = one;
         }
         slow_clock_signal_local = 1;
         break;
   default:
         break;
   }
```

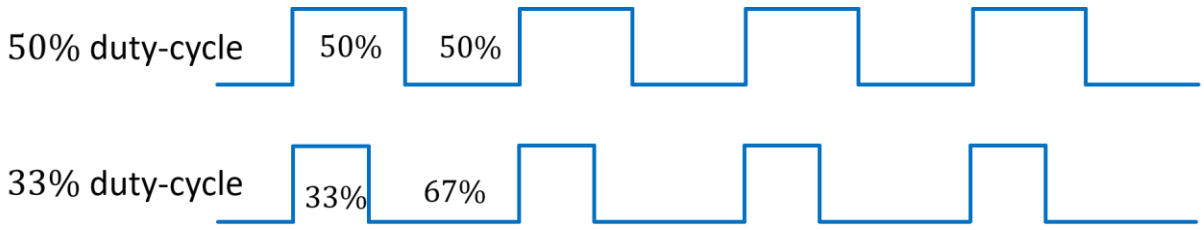
```
    state = next_state;

    if (start == 1 ) {
            next_counter = CLOCK_FREQUENCY_COUNTER;
    } else {
            counter = next_counter;
    }

    counter = next_counter;
    slow_clock_signal = slow_clock_signal_local;

}
```

2- PWM. The definition of the clock duty cycle is important in understanding PWM.

The clock's duty cycle is defined as the percentage of the total period that the clock is in the high state.

$$duty\_cycle = \frac{t_h}{T}\%100$$

For example:



The following code solves the problem. In this code, the $divisor$ input represents the number of the design clock-cycle in a PWM signal period, and $duty\_cycle$ represents the number of the design clock-cycle when the PWM signal is in the high-state.

```
#include "pwm.h"



typedef enum{zero, one} clock_state_type;

void pwm(ap_uint<16> divisor, ap_uint<16> duty_cycle, bool start, bool
&slow_clock_signal) {
#pragma HLS INTERFACE ap_none port=slow_clock_signal
#pragma HLS INTERFACE ap_none port=divisor
#pragma HLS INTERFACE ap_none port=duty_cycle
#pragma HLS INTERFACE ap_none port=start
#pragma HLS INTERFACE ap_ctrl_none port=return
```

```
	static clock_state_type state = zero;
	static unsigned int     one_counter  = 0;
	static unsigned int     zero_counter = 0;

	clock_state_type next_state;
	unsigned int     next_one_counter;
	unsigned int     next_zero_counter;




	bool slow_clock_signal_local;

	switch(state) {
	case zero:
		if (zero_counter == 0) {
			next_zero_counter  = divisor - duty_cycle-1;
			next_state         = one;
		} else {
			next_zero_counter  = zero_counter-1;
			next_state         = zero;
		}
		slow_clock_signal_local = 0;
		next_one_counter        = duty_cycle-1;
		break;
	case one:
		if (one_counter == 0) {
			next_one_counter   = duty_cycle-1;
			next_state         = zero;
		} else {
			next_one_counter   = one_counter-1;
			next_state         = one;
		}
		next_zero_counter  = divisor - duty_cycle-1;
		slow_clock_signal_local = 1;
		break;
	default:
		break;
	}

	state = next_state;
	if (start == 1 ) {
		one_counter  =  (duty_cycle-1);
		zero_counter =  (divisor - duty_cycle-1);
	} else {
		one_counter  =  next_one_counter;
		zero_counter =  next_zero_counter;
	}


	slow_clock_signal = slow_clock_signal_local;

}
```