UPDuino

Lattice iCE 40 UltraPlus 5K

UPDuino

- Lattice UltraPlus **ICE40UP5K FPGA** with 5.3K LUTs, 1Mb SPRAM, 120Kb DPRAM, 8 Multipliers
- FTDI FT232H USB to SPI Device
- _ALL_ 32 FPGA GPIO on 0.1” headers
- _ALL_ FTDI pins brought to test points
- 4MB SPI Flash
- RGB LED
- On board 3.3V and 1.2V Regulators, can supply 3.3V to your project
- Open source schematic and layout using KiCAD design tools
- Integrated into the open source APIO toolchain

iCE 40

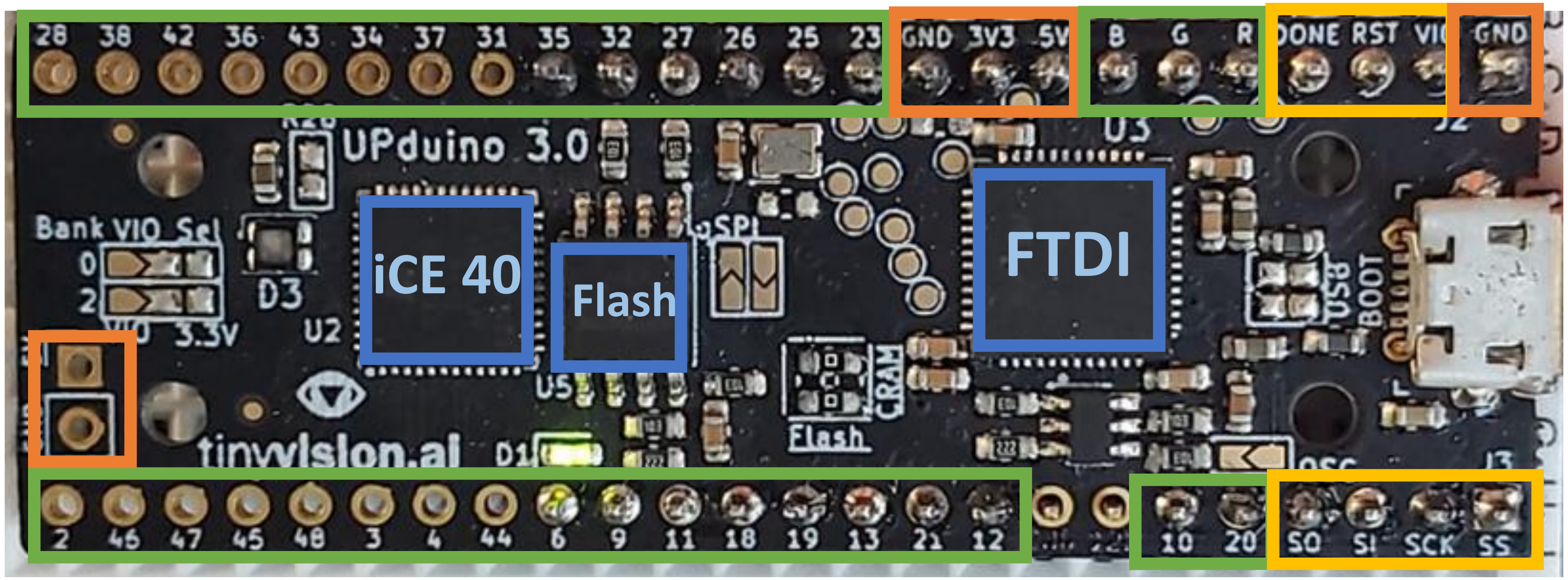
- https://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus

iCE40 UltraPLus

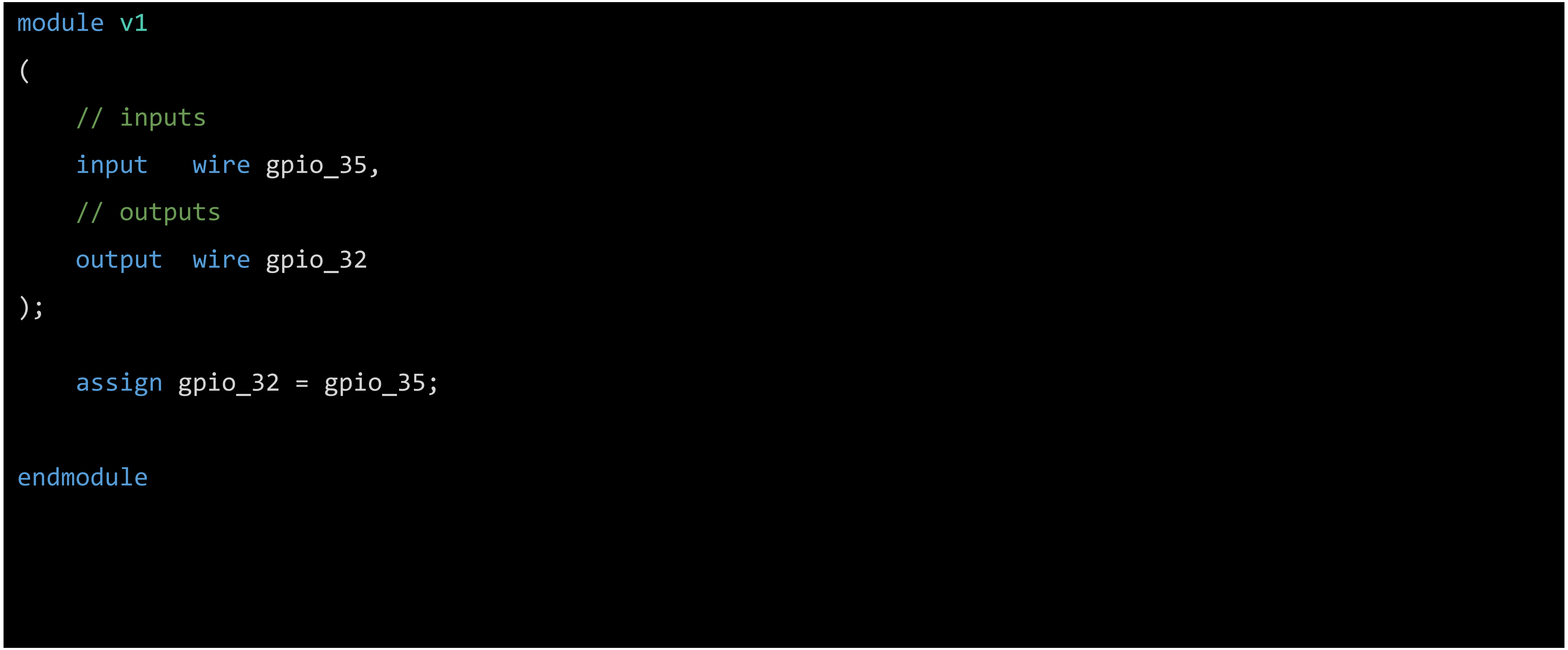
Parameter	UP3K	**UP5K**
Density LUTs	2800	**5280**
NVCM	Yes	**Yes**
Static Current (uA)	75	**75**
EBR RAM (kbits)	80	**120**
SPRAM (kbits)	1024	**1024**
PLL	1	**1**
I2C Core	2	**2**
SPI Core	2	**2**
Oscillator (10 kHz)	1	**1**
Oscillator (48 MHz)	1	**1**
24 mA Drive	3	**3**
500 mA Drive	-	**-**
16 x 16 Multiply & 32 bit Accumulator Blocks	4	**8**
PWM	Yes	**Yes**

UPDuino

- 5V/3.3V/GND
- GPIO/RGB
  - **3.3V!**
- Misc
  - SPI
  - Reset
  - Programming

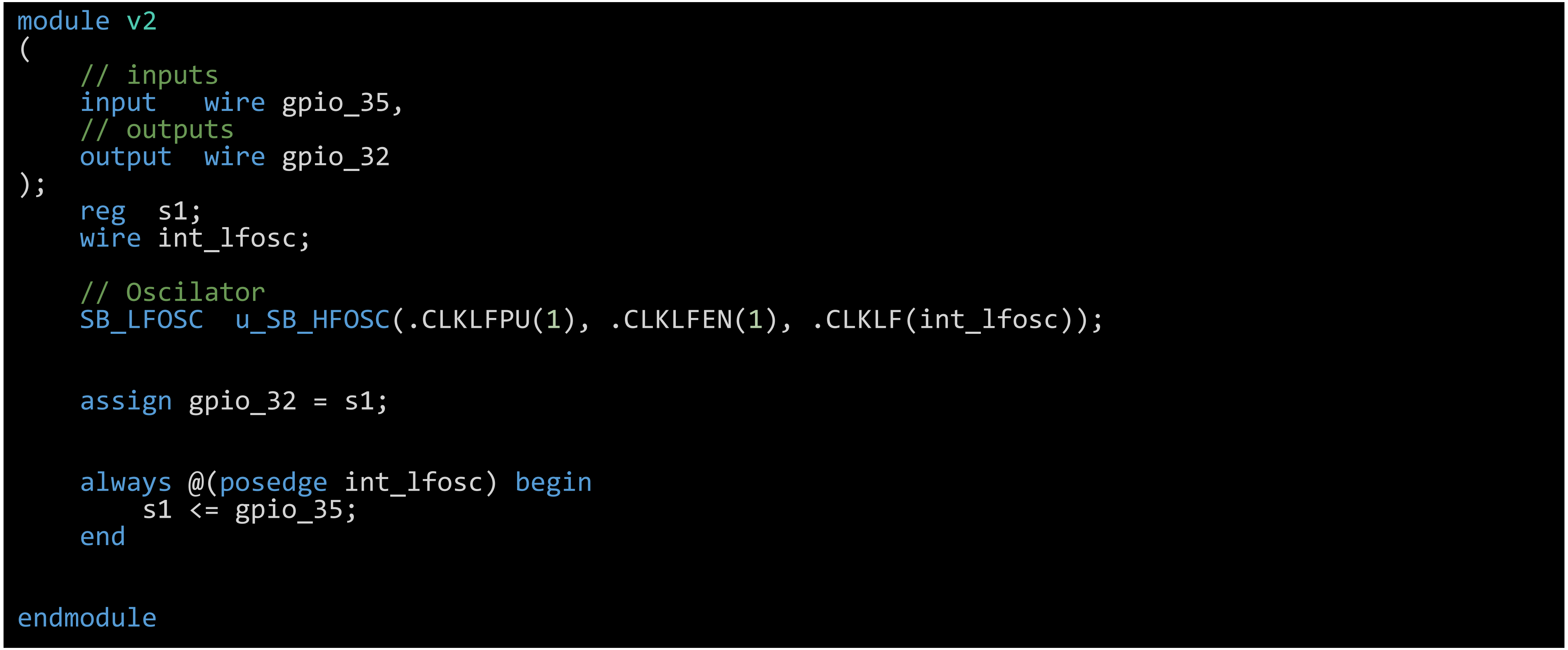


Verilog combinacional



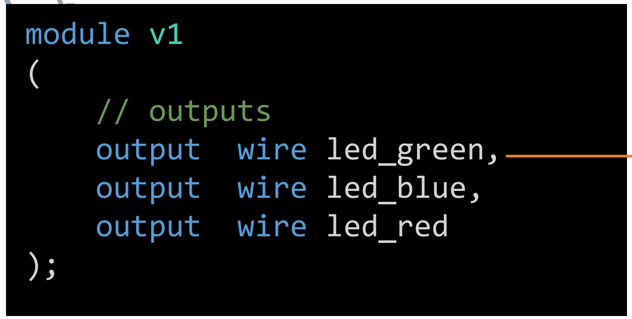
```
module v1
(
    // inputs
    input   wire gpio_35,
    // outputs
    output  wire gpio_32
);

    assign gpio_32 = gpio_35;


endmodule
```

Verilog secuencial + clock

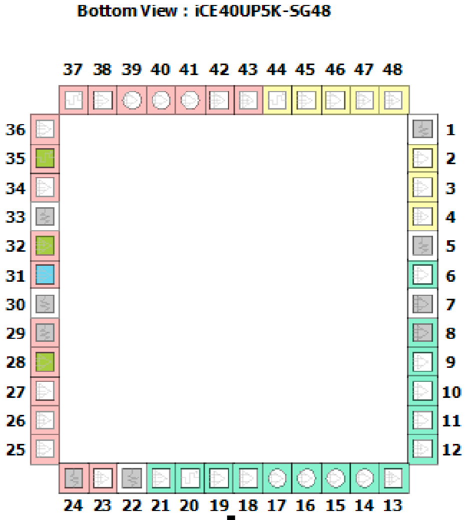


```
module v2
(
    // inputs
    input   wire gpio_35,
    // outputs
    output  wire gpio_32
);
    reg  s1;
    wire int_lfosc;

    // Oscilator
    SB_LFOSC  u_SB_HFOSC(.CLKLFPU(1), .CLKLFEN(1), .CLKLF(int_lfosc));


    assign gpio_32 = s1;


    always @(posedge int_lfosc) begin
        s1 <= gpio_35;
    end


endmodule
```

I/O

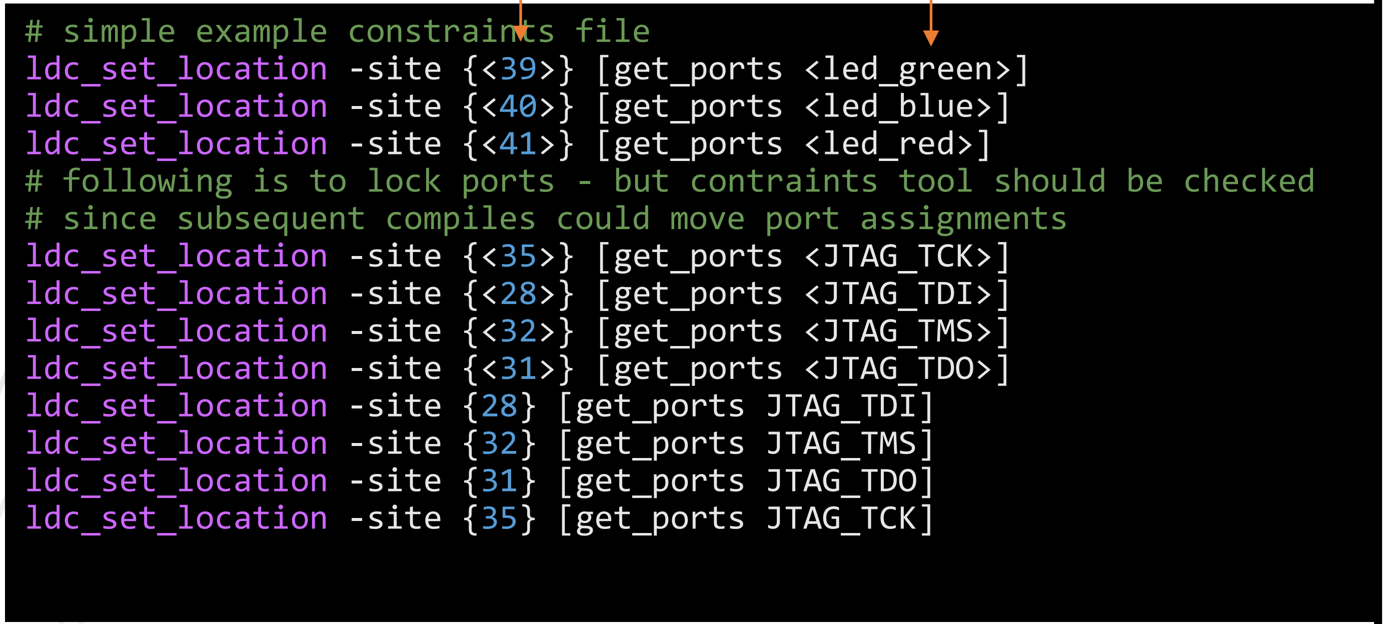


```
module v1
(
    // outputs
    output  wire led_green,
    output  wire led_blue,
    output  wire led_red
);
```

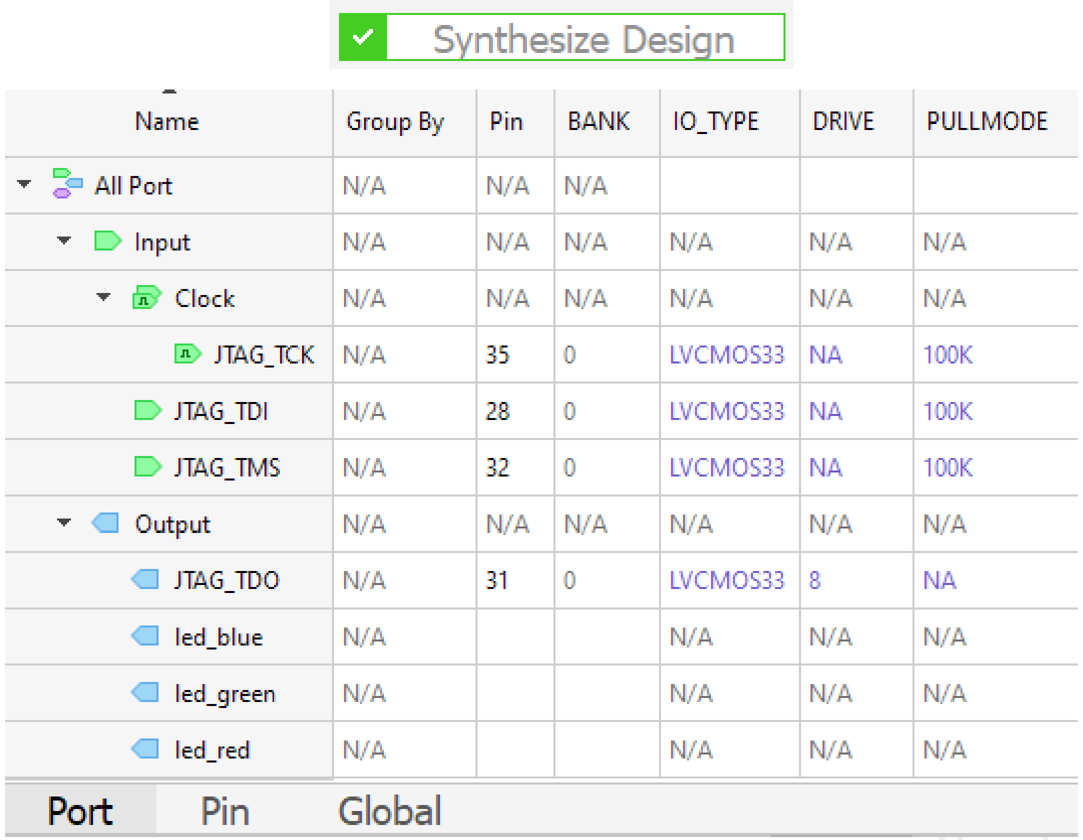


Device Constrain Editor

Archivo .PDC　　PIN FPGA　　‘Puerto’ logico



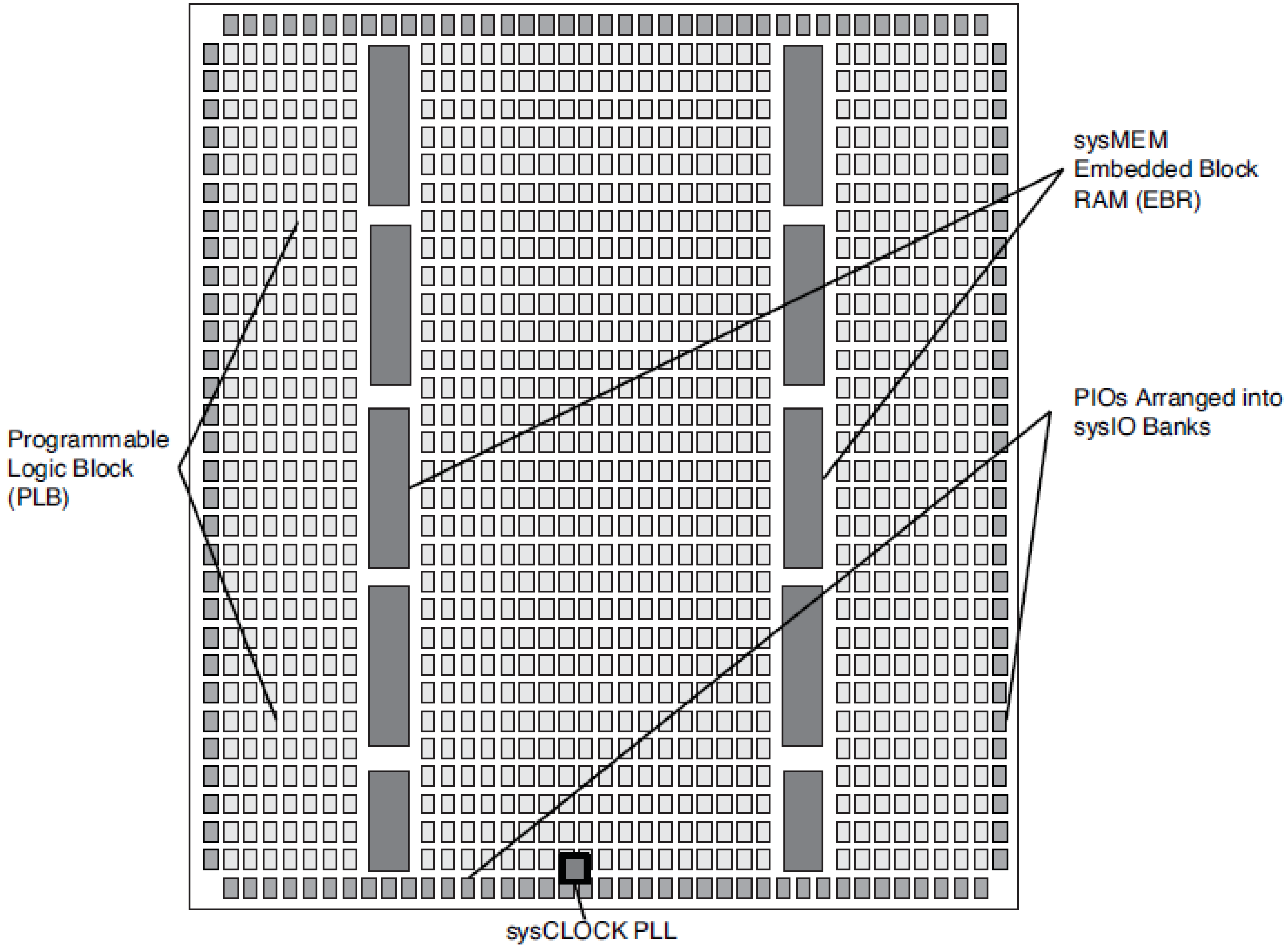
```
# simple example constraints file
ldc_set_location -site {<39>} [get_ports <led_green>]
ldc_set_location -site {<40>} [get_ports <led_blue>]
ldc_set_location -site {<41>} [get_ports <led_red>]
# following is to lock ports - but contraints tool should be checked
# since subsequent compiles could move port assignments
ldc_set_location -site {<35>} [get_ports <JTAG_TCK>]
ldc_set_location -site {<28>} [get_ports <JTAG_TDI>]
ldc_set_location -site {<32>} [get_ports <JTAG_TMS>]
ldc_set_location -site {<31>} [get_ports <JTAG_TDO>]
ldc_set_location -site {28} [get_ports JTAG_TDI]
ldc_set_location -site {32} [get_ports JTAG_TMS]
ldc_set_location -site {31} [get_ports JTAG_TDO]
ldc_set_location -site {35} [get_ports JTAG_TCK]
```



☑ Synthesize Design

Name	Group By	Pin	BANK	IO_TYPE	DRIVE	PULLMODE
All Port	N/A	N/A	N/A			
Input	N/A	N/A	N/A	N/A	N/A	N/A
Clock	N/A	N/A	N/A	N/A	N/A	N/A
JTAG_TCK	N/A	35	0	LVCMOS33	NA	100K
JTAG_TDI	N/A	28	0	LVCMOS33	NA	100K
JTAG_TMS	N/A	32	0	LVCMOS33	NA	100K
Output	N/A	N/A	N/A	N/A	N/A	N/A
JTAG_TDO	N/A	31	0	LVCMOS33	8	NA
led_blue	N/A			N/A	N/A	N/A
led_green	N/A			N/A	N/A	N/A
led_red	N/A			N/A	N/A	N/A

Modulos iCE40UP



Oscillator

- http://www.latticesemi.com/~/media/LatticeSemi/Documents/ApplicationNotes/IK/iCE40OscillatorUsageGuide.pdf?document_id=50670

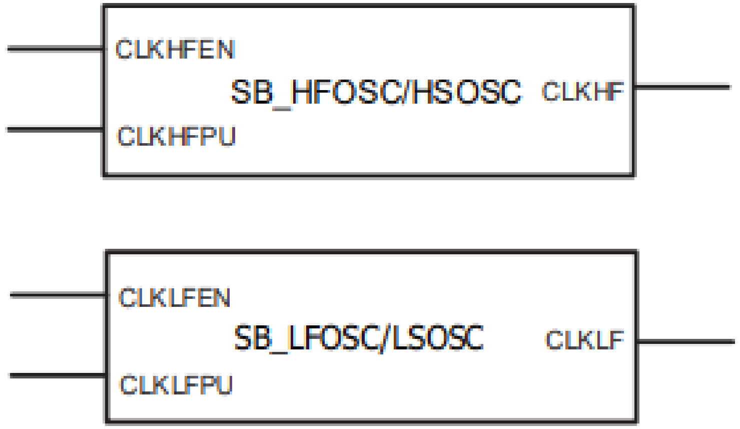
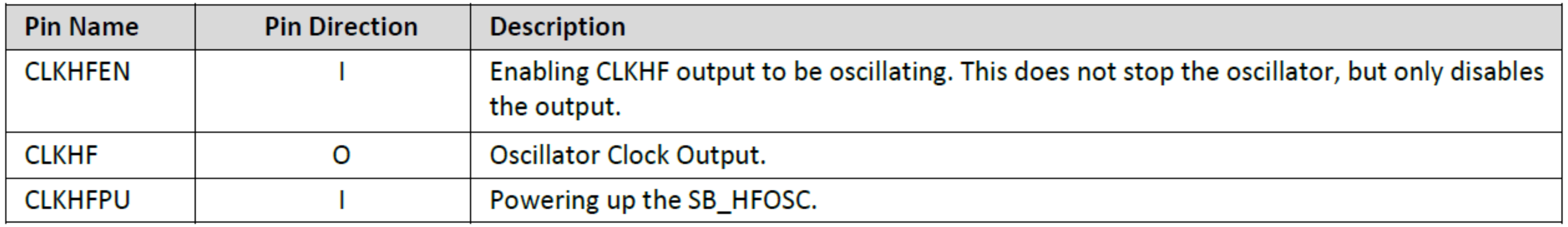


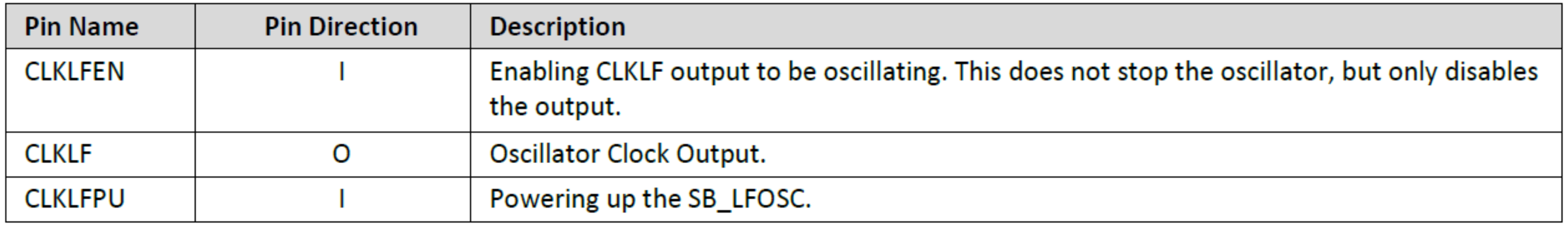
Figure 2.1. On-Chip Oscillator

Table 3.1. SB_HFOSC I/O



Pin Name	Pin Direction	Description
CLKHFEN	I	Enabling CLKHF output to be oscillating. This does not stop the oscillator, but only disables the output.
CLKHF	O	Oscillator Clock Output.
CLKHFPU	I	Powering up the SB_HFOSC.

Table 3.2. SB_LFOSC



Pin Name	Pin Direction	Description
CLKLFEN	I	Enabling CLKLF output to be oscillating. This does not stop the oscillator, but only disables the output.
CLKLF	O	Oscillator Clock Output.
CLKLFPU	I	Powering up the SB_LFOSC.

Low Frequency Oscillator **LSOSC – 10kHz**

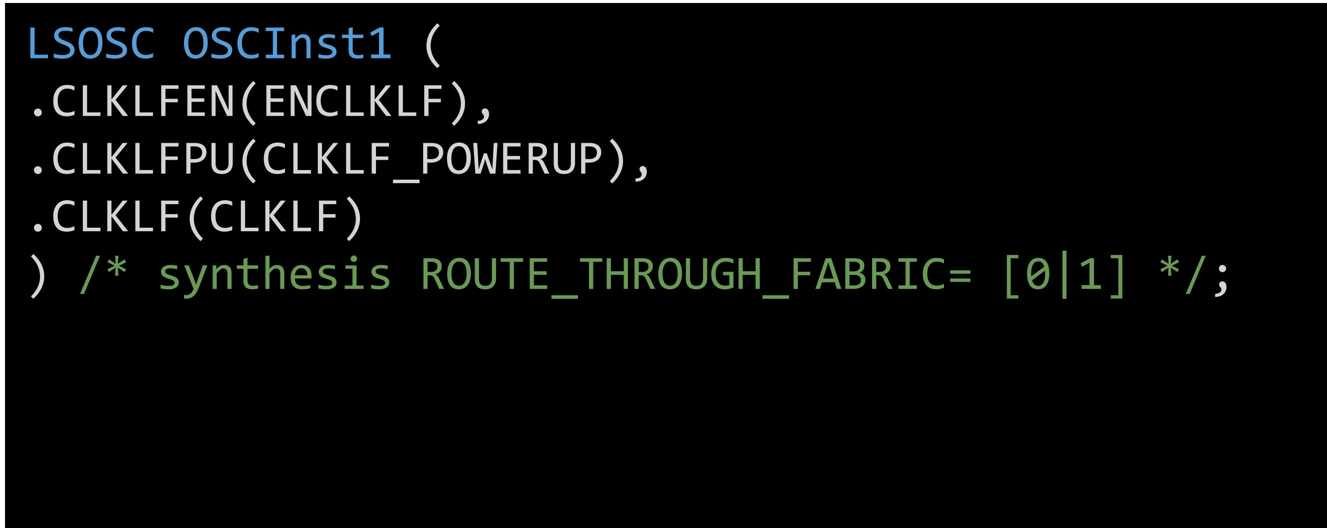
Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value>*/

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.



```
LSOSC OSCInst1 (
.CLKLFEN(ENCLKLF),
.CLKLFPU(CLKLF_POWERUP),
.CLKLF(CLKLF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
```

High Frequency Oscillator **HSOSC – 48 MHz**

Synthesis Attributes

/* synthesis ROUTE_THROUGH_FABRIC = <value> */

Value:

0: Use dedicated clock network. Default option.

1: Use fabric routes.

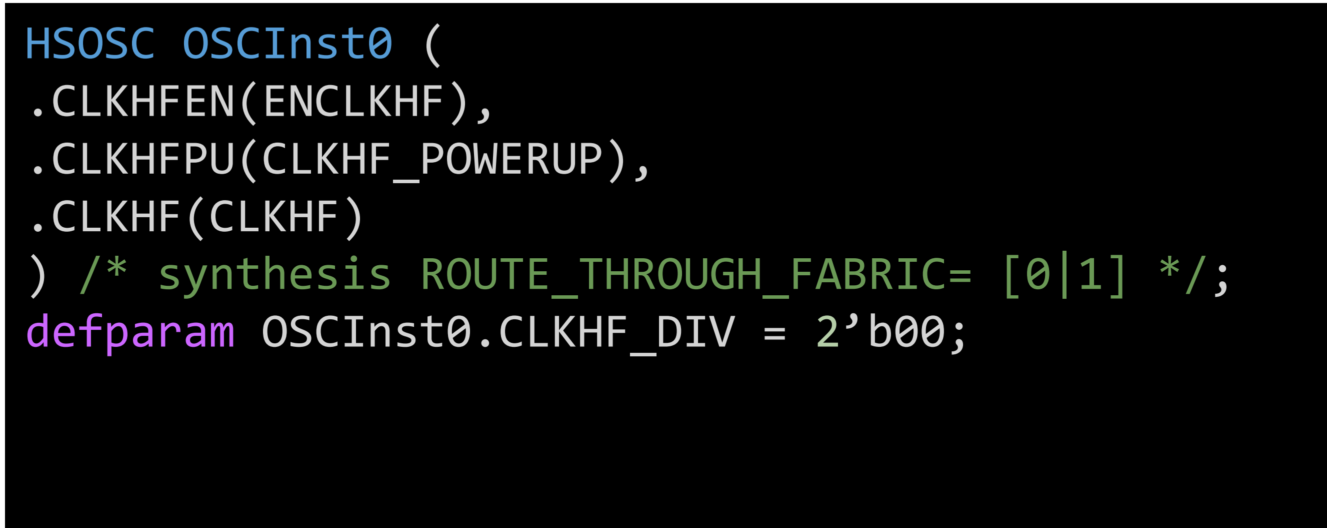
Parameter Values

The SB_HFOSC primitive contains the following parameter and their default values:

Parameter **CLKHF_DIV** = 2’b00 :

00 = div1, 01 = div2, 10 = div4, 11 = div8 ;

Default = “00”



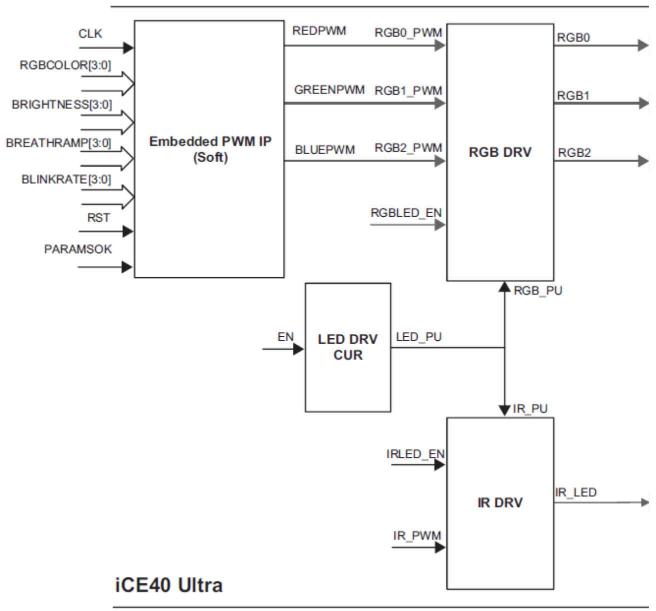
```
HSOSC OSCInst0 (
.CLKHFEN(ENCLKHF),
.CLKHFPU(CLKHF_POWERUP),
.CLKHF(CLKHF)
) /* synthesis ROUTE_THROUGH_FABRIC= [0|1] */;
defparam OSCInst0.CLKHF_DIV = 2’b00;
```

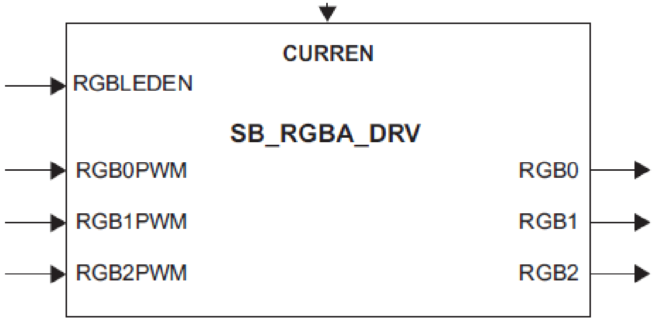
RGB driver

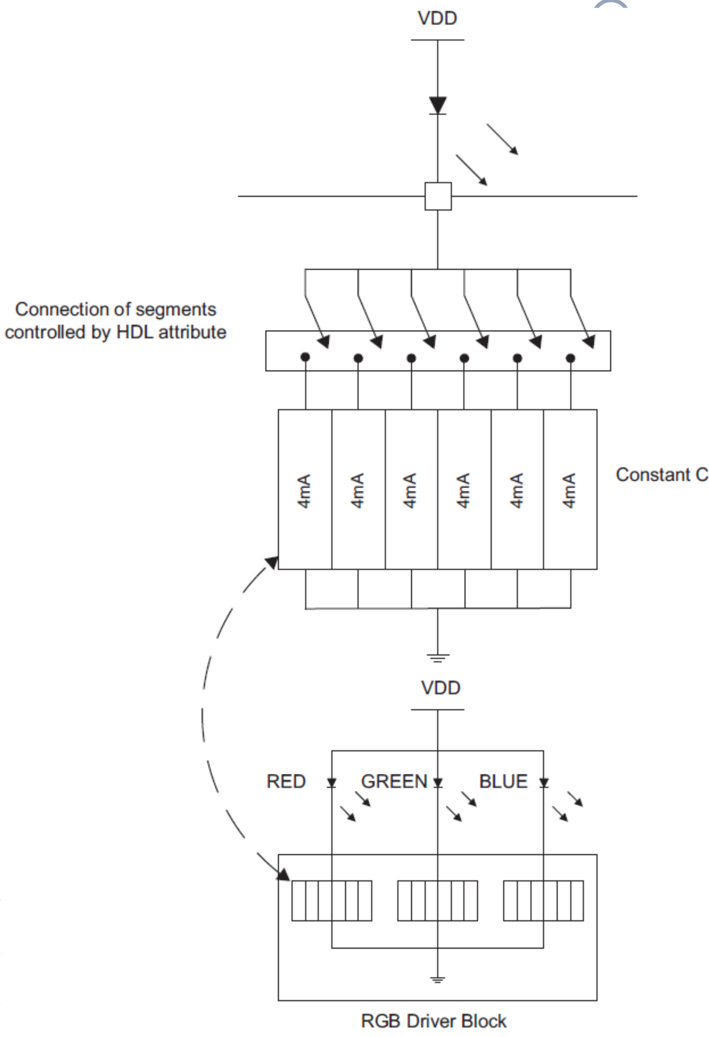
- https://www.latticesemi.com/-/media/LatticeSemi/Documents/ApplicationNotes/IK/ICE40LEDDriverUsageGuide.ashx?document_id=50668

Key features of the RGB Driver:

- Supports three Service LEDs (RGB) with sink current between 4 mA and 24 mA in steps of 4 mA or 2 mA and 12 mA in steps of 2 mA per device ball.
- Supports pins being independently configured as either a high-current sink or an OD GPIO.
- Accuracy of up to ± 10% of the amount of current being sunk at all steps when the voltage at the device pin is at least 0.5 V. Current matching within ± 5% across all three Service LEDs for the same current sink setting
- Wakeup time (from off to on -- fully functional) ≤ 100 µsec.







RGB Driver

```
SB_RGBA_DRV RGB_DRIVER (
.RGBLEDEN(ENABLE_LED),
.RGB0PWM(RGB0),
.RGB1PWM(RGB1),
.RGB2PWM(RGB2),
.CURREN(led_power_up),
.RGB0(LED0),
.RGB1(LED1),
.RGB2(LED2)
),
defparam RGB_DRIVER.CURRENT_MODE = “0”,
defparam RGB_DRIVER.RGB0_CURRENT = “111111”
defparam RGB_DRIVER.RGB1_CURRENT = “111111”
defparam RGB_DRIVER.RGB2_CURRENT = “111111”
```

PLL

http://www.latticesemi.com/~/media/LatticeSemi/Documents/ApplicationNotes/IK/iCE40sysCLOCKPLLDesignandUsageGuide.pdf?document_id=47778

- The PLL provides the following functions in iCE40 UltraPlus applications:
- Generates a new output clock frequency
  - Clock multiplication
  - Clock division
- De-skews or phase-aligns an output clock to the input reference clock
  - Faster input set-up time
  - Faster clock-to-output time
- Corrects output clock to have nearly a 50% duty cycle, which is important for Double Data Rate (DDR) applications
- Optionally phase shifts the output clock relative to the input reference clock
  - Optimal data sampling within the available bit period
  - Fixed quadrant phase shifting at 0°, 90°
  - Optional fine delay adjustments of up to 2.5 ns (typical) in 150 ps increments (typical)

PLL

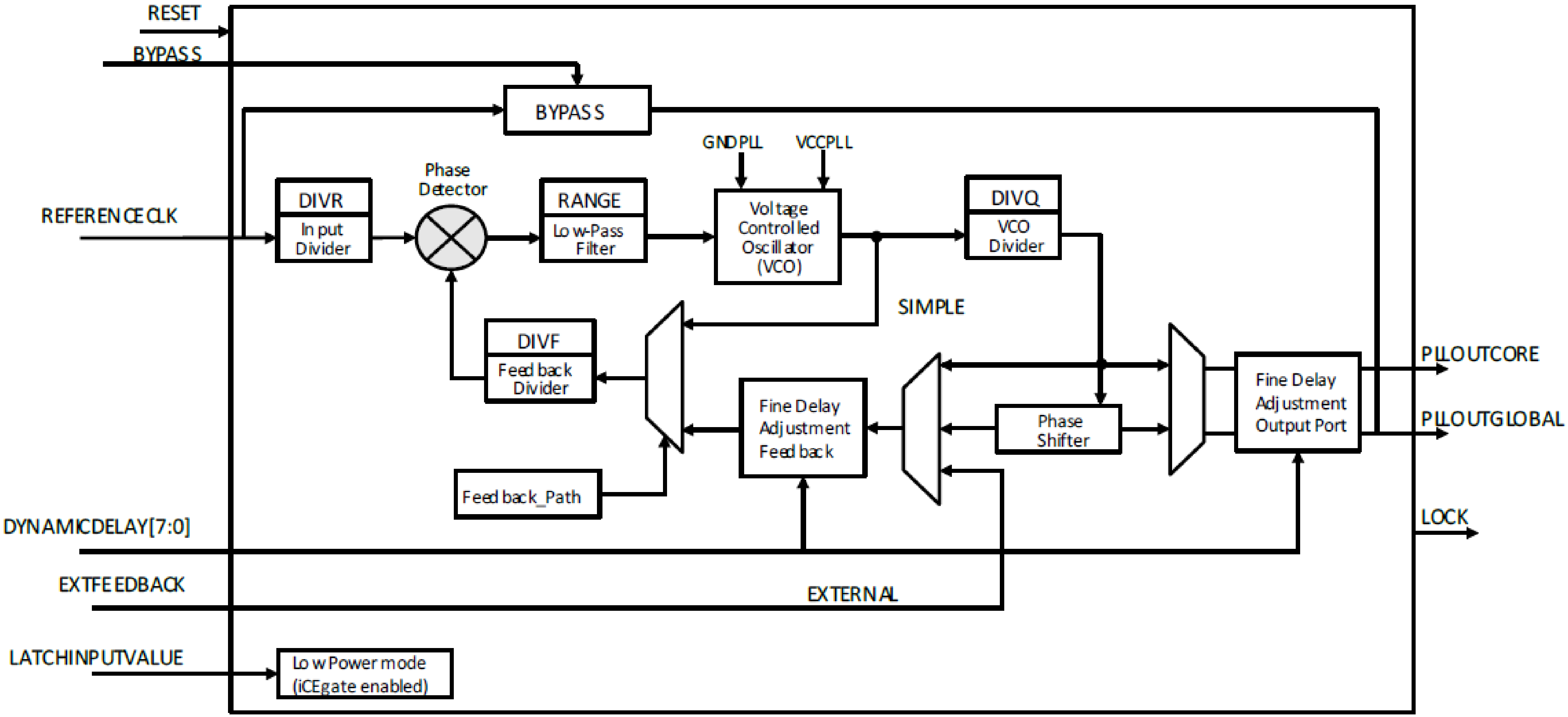
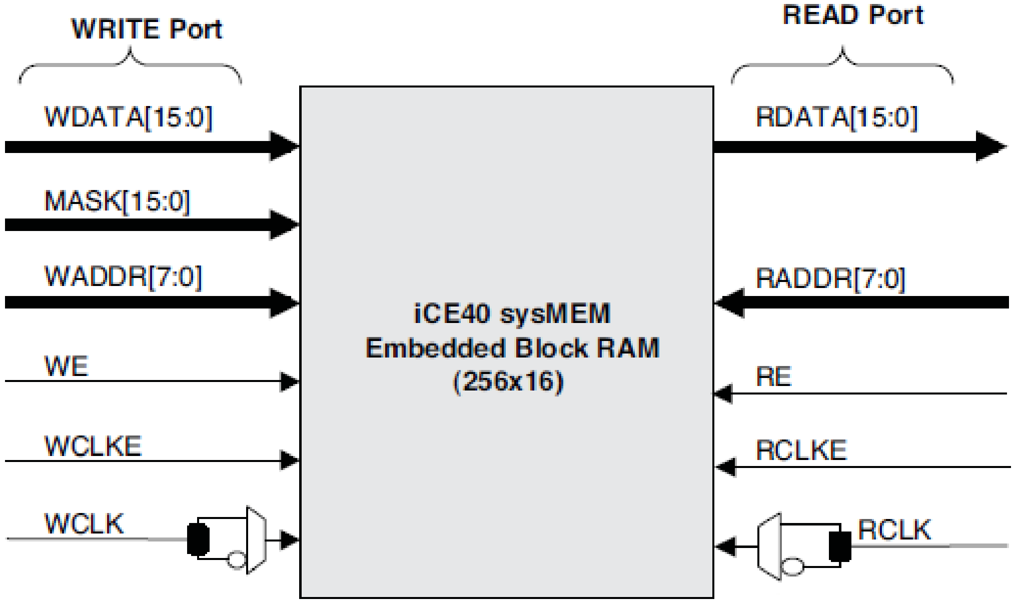


Figure 3.1. iCE40 UltraPlus Phase-Locked Loop (sysCLOCK PLL) Block Diagram

RAM

https://www.latticesemi.com/-/media/LatticeSemi/Documents/ApplicationNotes/MO/MemoryUsageGuideforiCE40Devices.ashx?document_id=47775



Signal Name	Direction	Description
WDATA[15:0]	Input	Write Data input.
MASK[15:0]	Input	Masks write operations for individual data bit-lines. 0 = write bit; 1 = don't write bit
WADDR[7:0]	Input	Write Address input. Selects one of 256 possible RAM locations.
WE	Input	Write Enable input.
WCLKE	Input	Write Clock Enable input.
WCLK	Input	Write Clock input. Default rising-edge, but with falling-edge option.
RDATA[15:0]	Output	Read Data output.
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible RAM locations.
RE[1]	Input	Read Enable input. Only available for SB_RAM256x16 configurations.
RCLKE	Input	Read Clock Enable input.
RCLK	Input	Read Clock input. Default rising-edge, but with falling-edge option.

Mas info

Resumen de componentes iCE40 Technology Library

Referencia de programación iCE40 Programming and Configuration

Datasheet iCE40 UltraPlus Family Data Sheet

Resumen de la familia de productos iCE40 iCE40 Family Handbook

Guía de uso de módulos

- iCE40 I2C and SPI Hardened IP Usage Guide
- iCE40 LED Driver Usage Guide
- iCE40 Oscillator Usage Guide
- iCE40 SPRAM Usage Guide
- iCE40 sysCLOCK PLL Design and Usage Guide

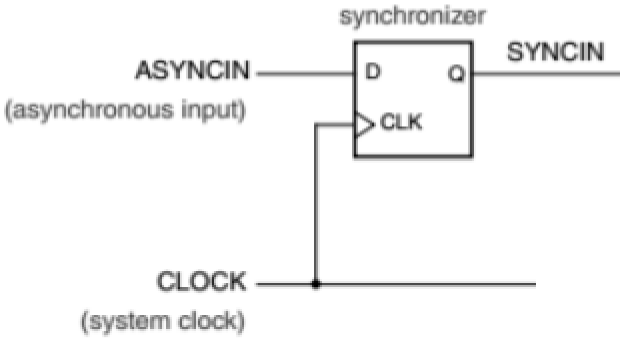
Ejercicios

Ejercicios

1. Implemente <u>uno</u> de los siguientes módulos combinacionales:
   a. AND de 3 entradas
   b. Decoder 8 bits
   c. Encoder 8 bits
   d. Sumador de 4 bits (A + B = C + Cy, A, B y C de 4 bits)

2. Implemente un sincronizador de entrada como los vistos en clase, de al menos un Flip Flop

   Utilice un clock interno generado por el modulo oscilador de la FPGA
   Entradas FPGA: Entrada async (pulsador externo)
   Salidas FPGA: Señal sincronizada, CLK



- Conectando la placa a el Digilent, use la entrada y salida Digital para mostrar el funcionamiento de los módulos anteriores

Ejercicios

3. Basado en el codigo de rgb_blink.v, modifiquelo para utilizar 4 entradas: IN_R, IN_G, IN_B, BLINK
   - Un valor de ‘1’ en cada una entrada corresponde a cada led encendido, un 0 siempre apagado
   - BLINK:
     - ‘1’ los colores titilan juntos siguiendo una frecuencia de clock dada (solo encienden los que tienen un ‘1’ en su entrada IN_x)
     - ‘0’ los leds se mantiene encendido permanentes
   - Bonus:
     - Entrada BRIGHT: ‘1’ alta intensidad, ‘0’ baja intensidad (use los parametros del modulo RGB)