Cómo configurar el clocking wizard de forma dinámica (experimental)

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/11/17/como-configurar-el-clocking-wizard-de-forma-dinamica-experimental/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

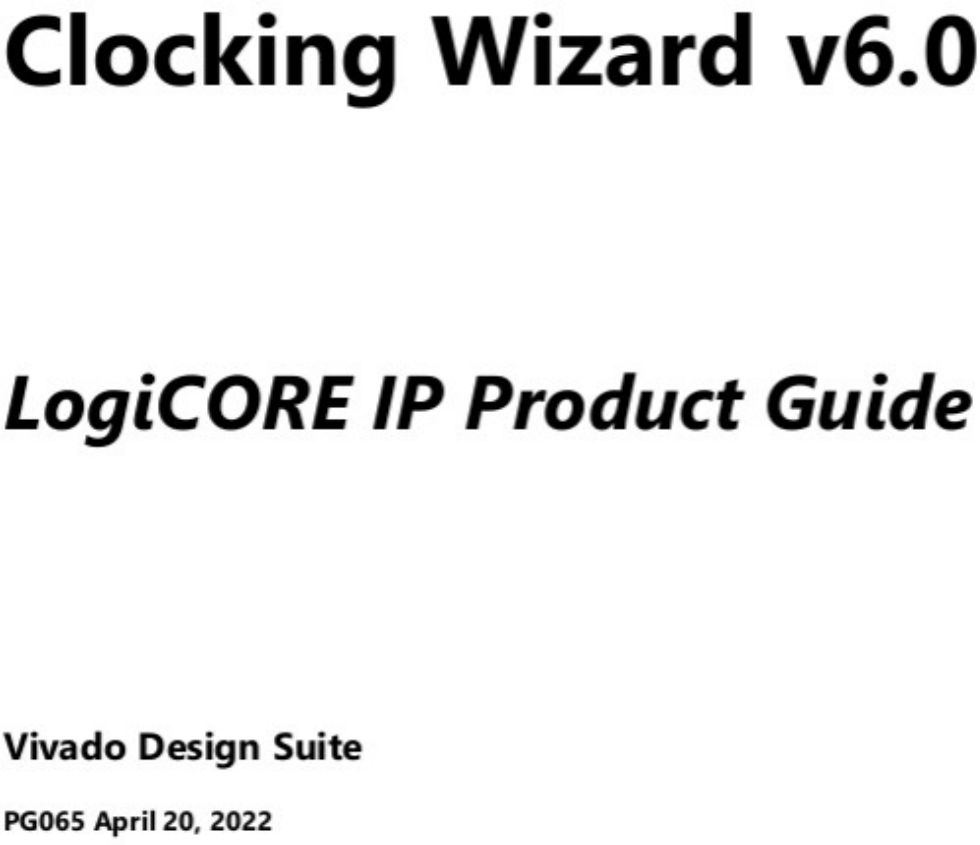
Fecha última modificación: 23/02/2025

NOTA: *todo lo que voy a contar ahora es experimental, por lo que hay un alto riesgo que las cosas salgan mal. Pero aún así te lo cuento.*

Una cosa que he descubierto hace poco, y que no es muy conocida, es que se puede configurar el *clocking wizard* por AXI mediante registros o por DRP. Eso significa que se puede reconfigurar de forma dinámica el *clk_wizard*. Para ello se tiene que habilitar la opción de *Dynamic Reconfig* del bloque.

Documentación

Si accedes a la documentación oficial de Xilinx (PG065) del Clocking Wizard…


Clocking Wizard v6.0

LogiCORE IP Product Guide

Vivado Design Suite

PG065 April 20, 2022

Hay un apartado llamado *Register Space*.

Register Space

Table 2-2 shows the set of registers applicable when the Dynamic Reconfiguration mode is selected. All registers are accessed as 32-bit.

Este apartado tiene un sistema de registros, que dejo en el Anexo, pues este sistema de registros permite modificar los parámetros de frecuencia del Clocking Wizard siguiendo una fórmula como la siguiente:

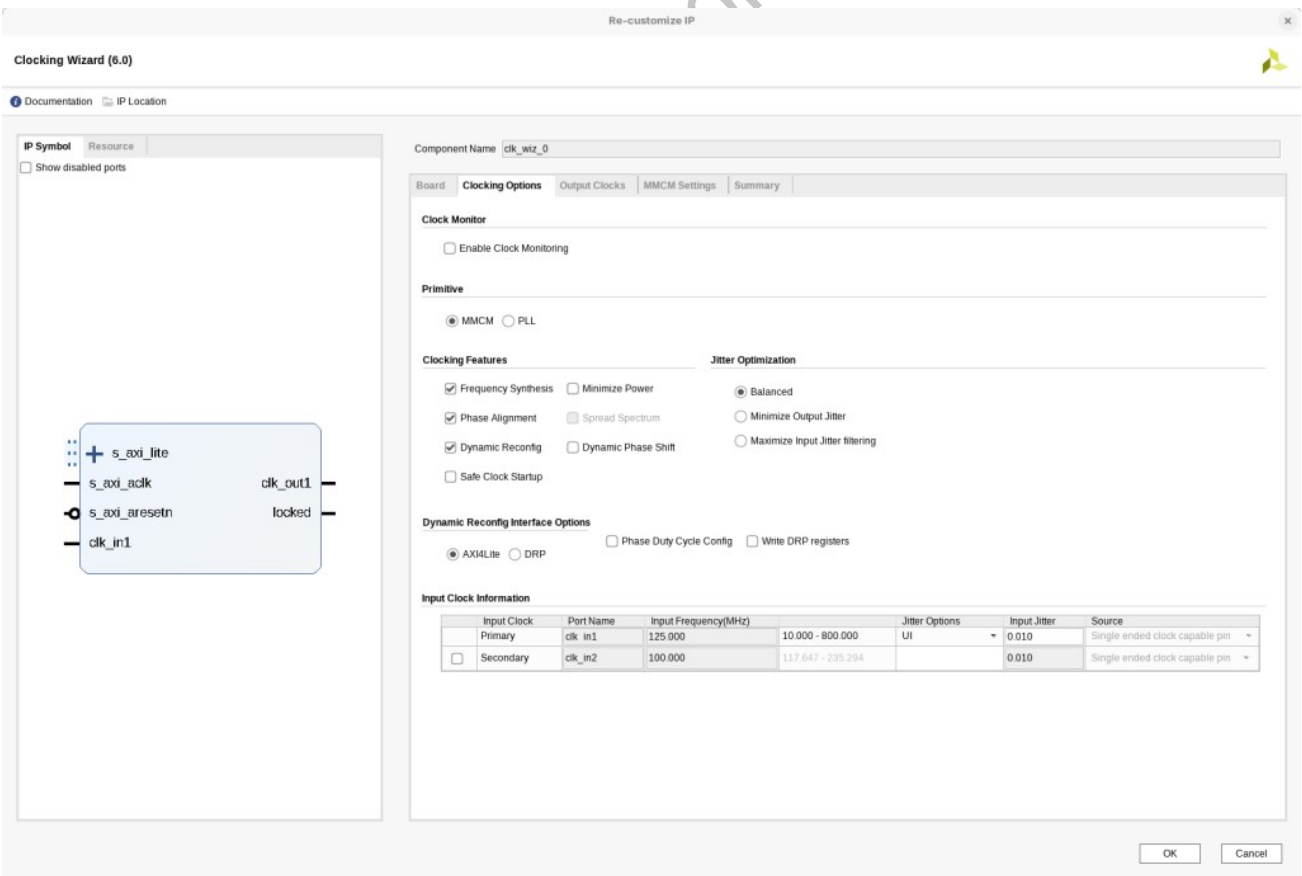
VCO Frequency = (Input Clock Frequency) * (CLKFBOUT_MULT)/DIVCLK_DIVIDE

Nota: No hay que fiarse mucho de esta fórmula debido a que puede no generar la frecuencia que se desea, o ni siquiera modificar la frecuencia.

Configuración

Para configurarlo se tiene que seguir los siguientes pasos:

- marcar la casilla *Dynamic Reconfig* en el clocking wizard.



- Atacar los registro de la siguiente forma

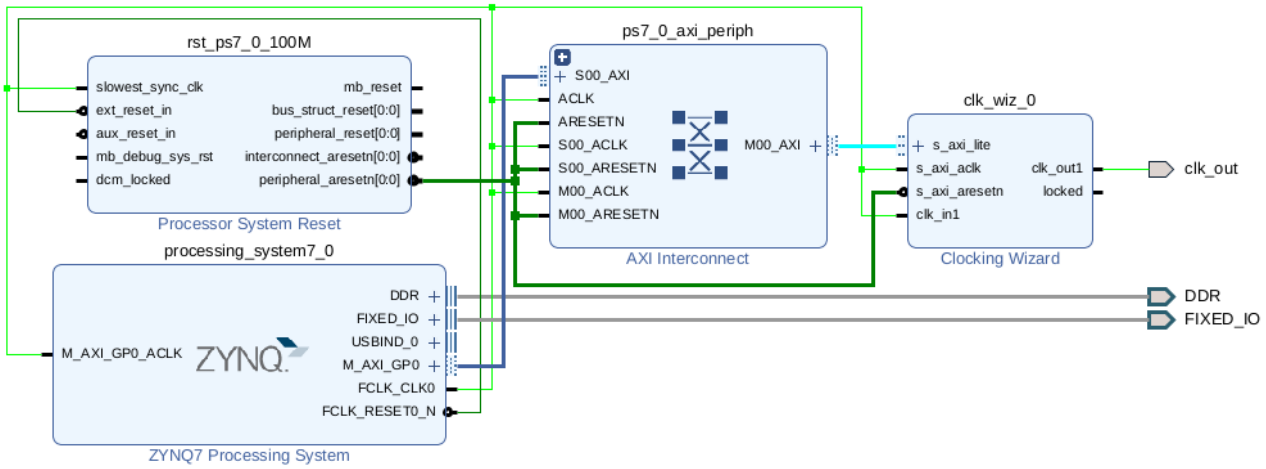
1. Modificar el/los parámetros que quieres tocar, divisor de frecuencia, ciclo de trabajo, etc.
2. Atacar el registro 23 (+0x25C), este registro es el que actualiza el clocking wizard. De este registro solo es necesario toca los dos bits de menor peso, con un 0x3, vale. Después de escribir este valor y reconfigurarse, el valor del primer bit se pone a ‘0’.

C_BASEADDR + 0x25C	Clock Configuration Register 23	0x00000000	R/W	Bit[0] = LOAD / SEN Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration and initiates reconfiguration state machine. This bit should be asserted when the required settings are already written into Clock Configuration Registers. This bit retains to 0, when the dynamic reconfiguration is done and the clock is locked. Bit[1] = SADDR When written 0, default configuration done in the Clocking Wizard GUI is loaded for dynamic reconfiguration. When written 1, setting provided in the Clock Configuration Registers are used for dynamic reconfiguration.

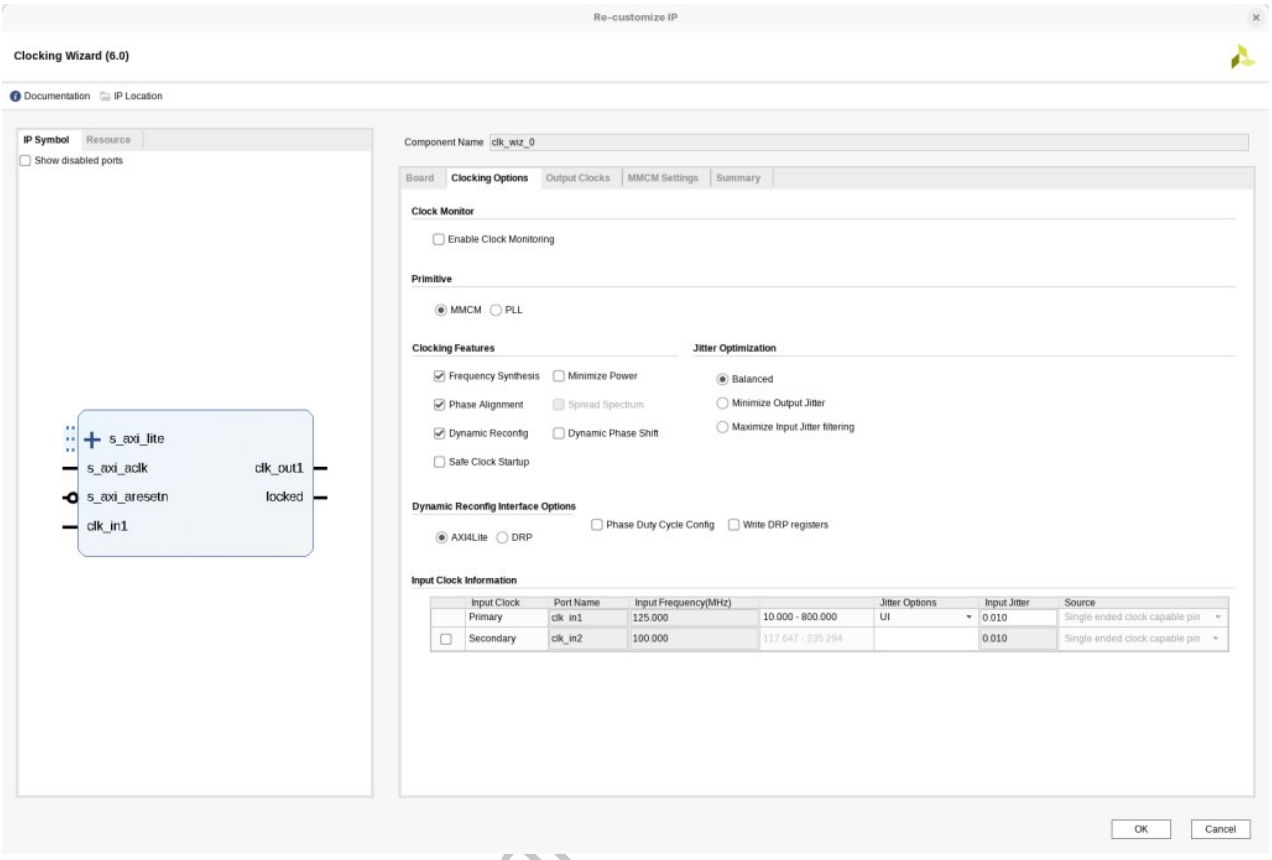
Ejemplo

La mejor forma de explicar cómo se puede modificar la frecuencia es con un pequeño ejemplo. Este ejemplo lo único que hace es sacar la frecuencia por un pin de la placa para verlo en un osciloscopio.

En nuestro caso como se puede configurar mediante AXI, también por DRP, se utiliza una Zynq. Esta Zynq tiene un clocking wizard conectado con un reloj de entrada de 100MHz.

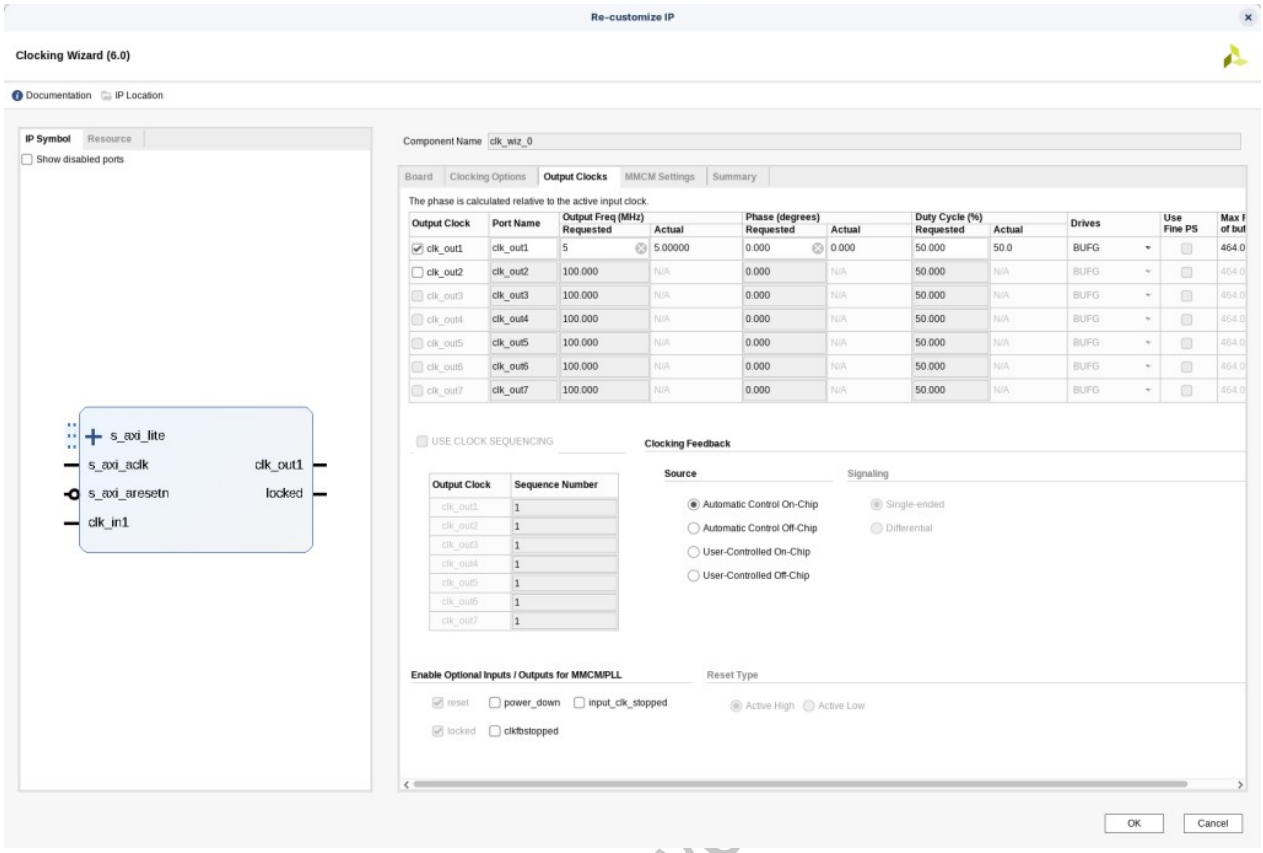


Ahora configuramos el clocking wizard. Para poder configurarlo dinámicamente se tiene que marca la opción de *Dynamic Reconfig*. Esta opción hacer que aparezca un conector AXI para conectarlo al AXI Interconnect.



Por lo demás, se configura una frecuencia base de 5MHz.

https://soceame.wordpress.com/2024/11/17/como-configurar-el-clocking-wizard-de-forma-dinamica-experimental/

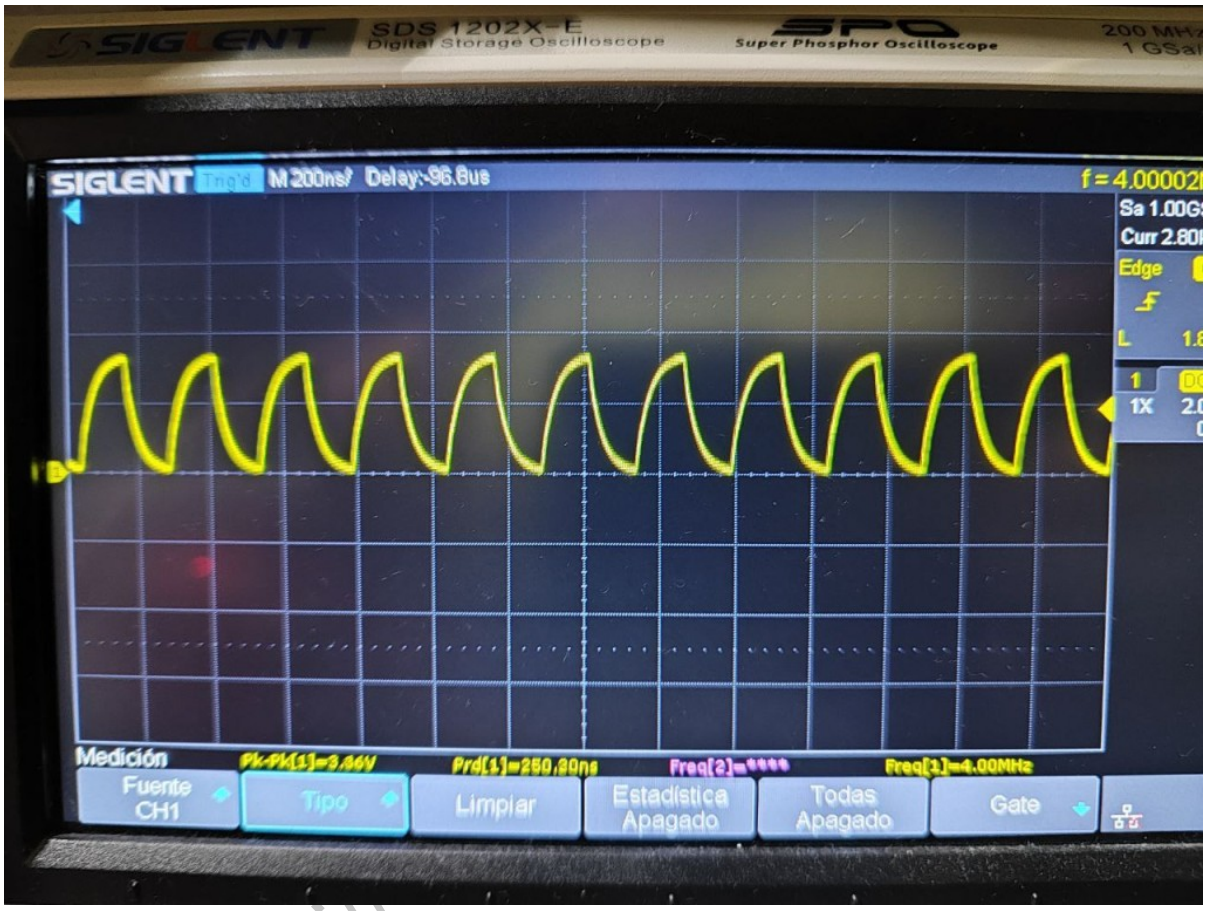


Y ahora exportamos el modelo a Vitis. El código de ejemplo es el siguiente:

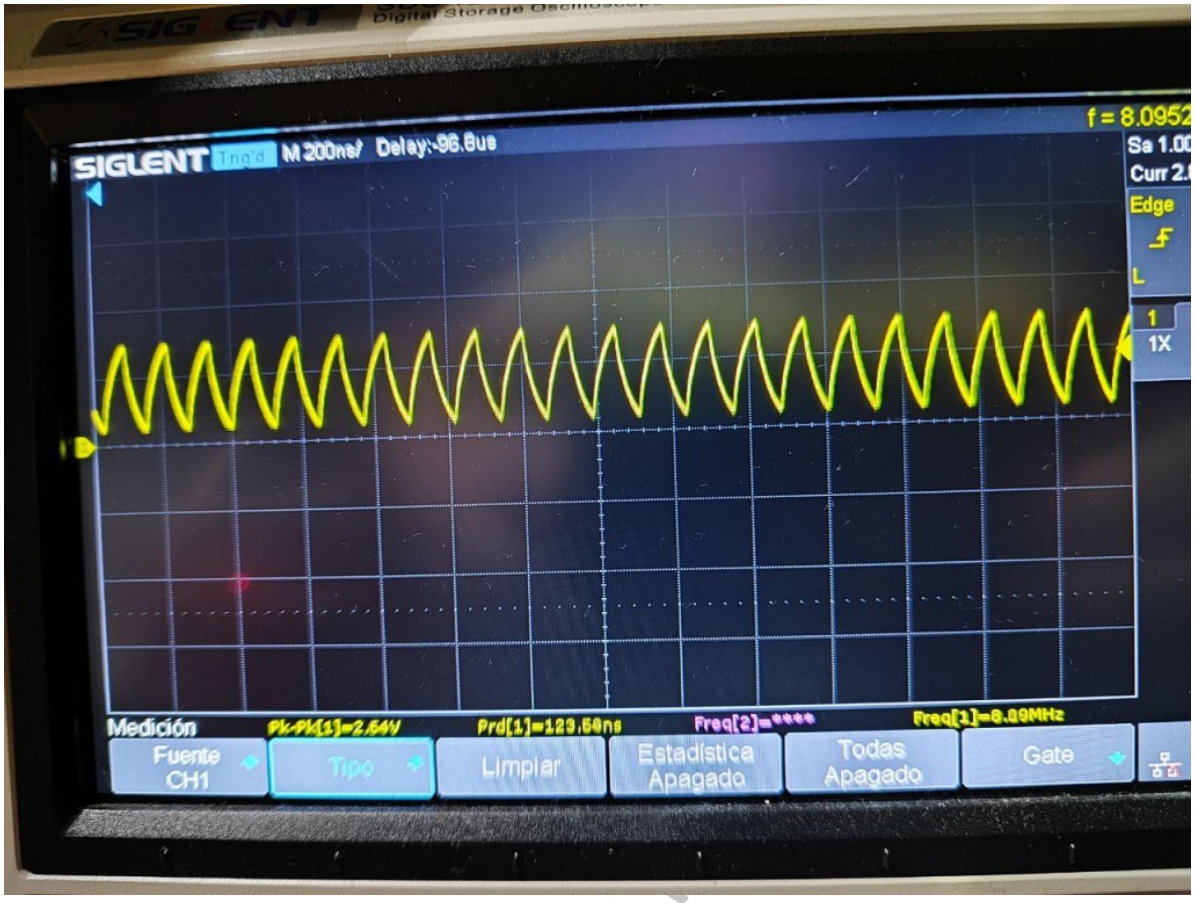
```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xil_io.h"
#include "xparameters.h"

#define CLK_BASE_ADDR XPAR_CLK_WIZ_0_BASEADDR

int main()
{
    init_platform();

    while(1){
            for(int i=0; i< 60000000; i++);
            Xil_Out32(CLK_BASE_ADDR+0x208, 0x2FA3F*2);
            Xil_Out32(CLK_BASE_ADDR+0x25C, 0x3);
            for(int i=0; i< 60000000; i++);
            Xil_Out32(CLK_BASE_ADDR+0x208, 0x17D1F);
            Xil_Out32(CLK_BASE_ADDR+0x25C, 0x3);
            for(int i=0; i< 60000000; i++);
            Xil_Out32(CLK_BASE_ADDR+0x208, 0x2FA3F);
            Xil_Out32(CLK_BASE_ADDR+0x25C, 0x3);
            for(int i=0; i< 60000000; i++);
        }
    cleanup_platform();
    return 0;
}
```

Con el modelo actual, si lo cargamos al SoC y miramos la frecuencia que emite el pin, se puede ver que genera la frecuencia base.

NOTA: en la imagen se puede ver que la frecuencia es de 4MHz en vez de 5MHz, eso no es un fallo a la hora de tomar la foto, la foto es de la frecuencia real, yo lo achaco a que no puede generar una frecuencia de 5MHz con lo parámetros que tiene configurados por defecto, ahí el usuario tiene poco que hacer.

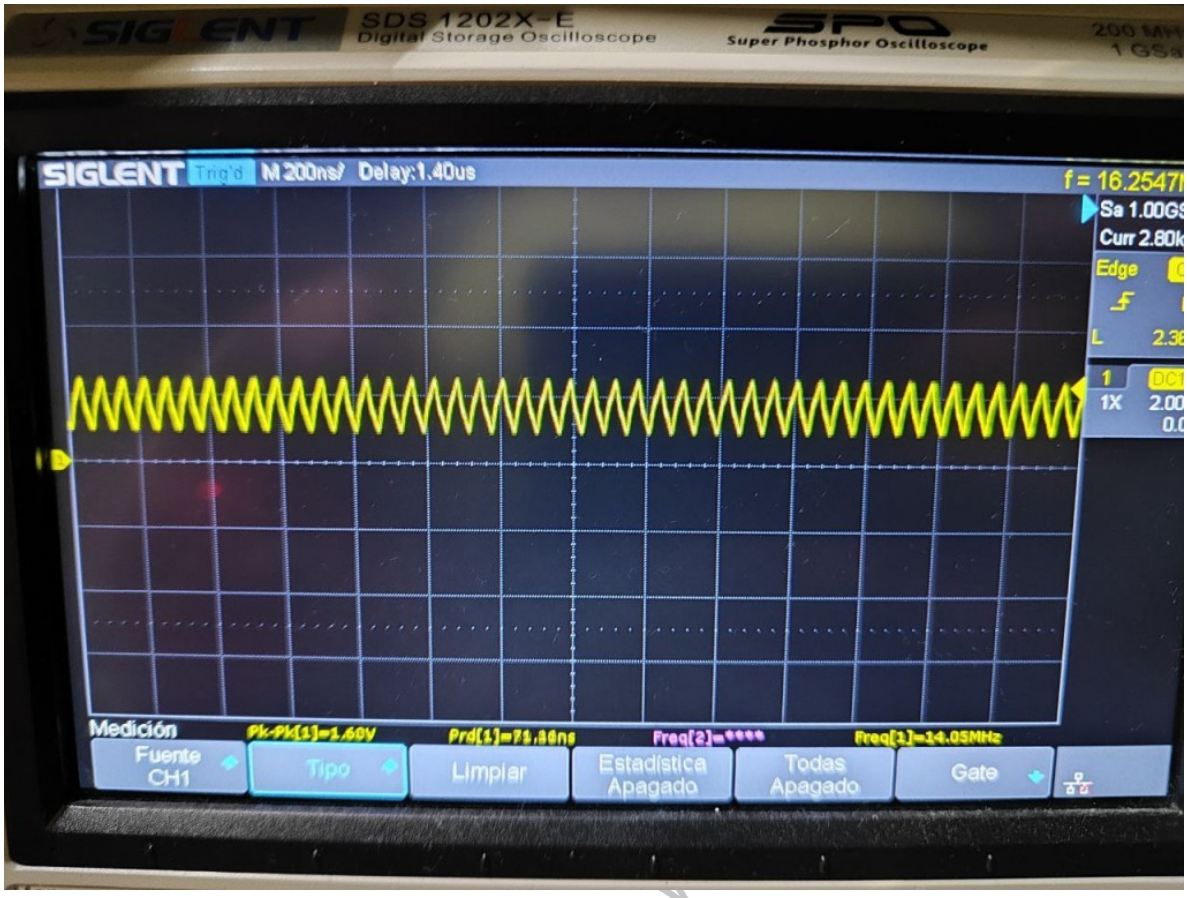


Ahora pasamos a reconfigurarlo. Si por ejemplo modificamos el valor que viene de serie en el parámetro de dividir la frecuencia, si pasamos de 0x5F4FE a un valor de la mitad 0x2FA3F. Se puede ver cómo la frecuencia se dobla, llegando a 8MHz.



Si ahora volvemos a hacer lo mismo y cambiamos el valor a la mitad del anterior, 0x17D1F. Pasamos a una frecuencia de 14MHz, que no es el doble de 8MHz, por lo que estos parámetros son muy caóticos, y muchas veces no cumplen con los requisitos deseados.

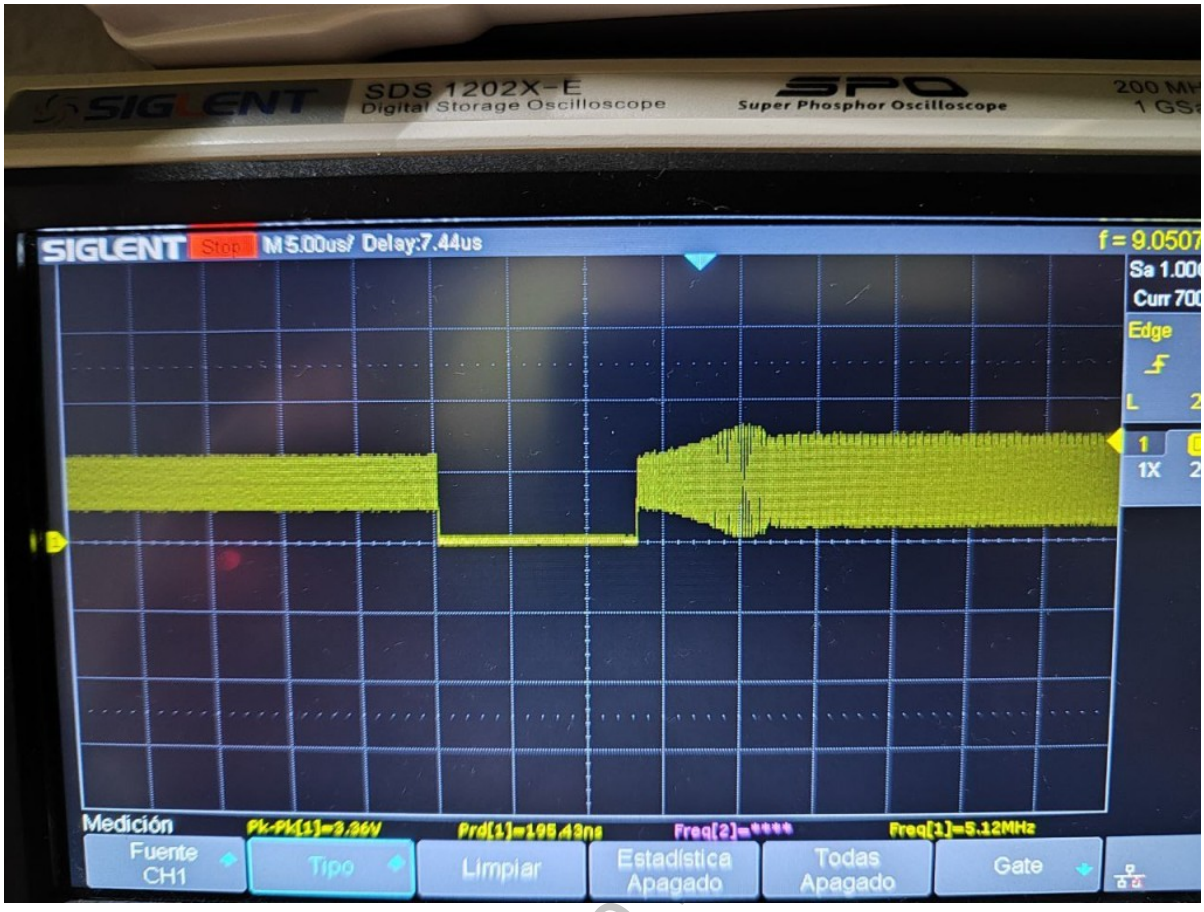
Y cualquier mínimo desfase, puede hacer aparecer un Jitter tan grande que ni siquiera se puede tener una señal periódica.



Del resto de parámetros no comento nada porque, o son difíciles de mostrar como el cambio de fase *(que sí que funciona)* o son parámetros que responden como les da la gana *(el multiplicador de frecuencia)*, o directamente no responden *(como el ciclo de trabajo)*.

Por último, dos anotaciones.

- El cambio de parámetro se hace anulando el reloj durante un tiempo, y con un transitorio.



- Y por último, como también se puede ver en la imagen anterior, el reloj tiene un offset que se agranda a más rápido va, y los Vp-p se modifican según aumenta la frecuencia, haciéndose más pequeños según aumenta la frecuencia. Esto influye en que cuanto más rápida es la frecuencia, llega un momento en que el reloj se puede convertir en ruido del osciloscopio y no ser fácilmente detectado.

Anexo

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x00	Software Reset Register (SRR)	N/A	W[1]	Software Reset Register To activate software reset, the value 0x0000_000A must be written to the register. Any other access, read or write, has undefined results.
C_BASEADDR + 0x04	Status Register (SR)	0x00000000	R	Status Register Bit[0] = Locked When 1 MMCM/PLL is Locked and ready for reconfiguration. The status of this bit is 0 during reconfiguration.
C_BASEADDR +0x08	Clock Monitor Error Status Register	0x00000000	R	This register gives the error status bits of the clock monitor feature.
C_BASEADDR +0x0C	Interrupt Status	0x00000000	R/W	Interrupt Status for Clock Stop, Clock Overrun, and Clock Underrun. These bits are gated by Interrupt enable bits. Interrupts corresponding to the enabled bits in Interrupt enable register would be updated in this register.
C_BASEADDR +0x10	Interrupt Enable	0x00000000	R/W	Interrupt Enable for Clock Stop, Clock Overrun, and Clock Underrun bits in the Interrupt status register.

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
Dynamic Reconfiguration Registers				
C_BASEADDR + 0x200	Clock Configuration Register 0	Default[2]: 0x01010A00	R/W	Bit[7:0] = DIVCLK_DIVIDE Eight bit divide value applied to all output clocks. Bit[15:8] = CLKFBOUT_MULT Integer part of multiplier value i.e. For 8.125, this value is 8 = 0x8. Bit[25:16] = CLKFBOUT_FRAC Multiply[3] Fractional part of multiplier value i.e. For 8.125, this value is 125 = 0x7D. ***Note:*** You need not set any bit for specifying that the multiplier value is fractional. Just mention the fractional value in the register space. The value of CLKFBOUT fractional divide can be from 0 to 875 representing the fractional multiplied by 1000.
C_BASEADDR + 0x204	Clock Configuration Register 1	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKFBOUT_PHASE Phase values entered are Signed Number for +/- phase.
C_BASEADDR + 0x208	Clock Configuration Register 2	Default[2]: 0x0004000a	R/W	Bit[7:0] = CLKOUT0_DIVIDE Integer part of clkout0 divide value For example, for 2.250, this value is 2 = 0x2 Bit[17:8] = CLKOUT0_FRAC Divide[3] Fractional part of clkout0 divide value For example, for 2.250, this value is 250 = 0xFA ***Note:*** You need not set any bit for specifying that the multiplier value is fractional. Just mention the fractional value in the register space.
C_BASEADDR + 0x20C	Clock Configuration Register 3	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKOUT0_PHASE[5]

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x210	Clock Configuration Register 4	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT0_DUTY Duty cycle value = (Duty Cycle in %) * 1000 For example, for 50% duty cycle, value is 50000 = 0xC350
C_BASEADDR + 0x214	Clock Configuration Register 5	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT1_DIVIDE(4) Eight bit clkout1 divide value
C_BASEADDR + 0x218	Clock Configuration Register 6	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT1_PHASE(5) Phase values entered are Signed Number for +/- phase
C_BASEADDR + 0x21C	Clock Configuration Register 7	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT1_DUTY(6)
C_BASEADDR + 0x220	Clock Configuration Register 8	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT2_DIVIDE(4)
C_BASEADDR + 0x224	Clock Configuration Register 9	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT2_PHASE(5)
C_BASEADDR + 0x228	Clock Configuration Register 10	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT2_DUTY(6)
C_BASEADDR + 0x22C	Clock Configuration Register 11	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT3_DIVIDE(4)
C_BASEADDR + 0x230	Clock Configuration Register 12	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT3_PHASE(5)
C_BASEADDR + 0x234	Clock Configuration Register 13	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT3_DUTY(6)
C_BASEADDR + 0x238	Clock Configuration Register 14	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT4_DIVIDE(4)
C_BASEADDR + 0x23C	Clock Configuration Register 15	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT4_PHASE(5)
C_BASEADDR + 0x240	Clock Configuration Register 16	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT4_DUTY(6)
C_BASEADDR + 0x244	Clock Configuration Register 17	Default(2): 0x0000000A	R/W	Bit[7:0] = CLKOUT5_DIVIDE(4)
C_BASEADDR + 0x248	Clock Configuration Register 18	Default(2): 0x00000000	R/W	Bit[31:0] = CLKOUT5_PHASE(5)
C_BASEADDR + 0x24C	Clock Configuration Register 19	Default(2): 0x0000C350	R/W	Bit[31:0] = CLKOUT5_DUTY(6)

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR + 0x250[3]	Clock Configuration Register 20	Default[2]: 0x0000000A	R/W	Bit[7:0] = CLKOUT6_DIVIDE[4]
C_BASEADDR + 0x254[3]	Clock Configuration Register 21	Default[2]: 0x00000000	R/W	Bit[31:0] = CLKOUT6_PHASE[5]
C_BASEADDR + 0x258[3]	Clock Configuration Register 22	Default[2]: 0x0000C350	R/W	Bit[31:0] = CLKOUT6_DUTY[6]
C_BASEADDR + 0x25C	Clock Configuration Register 23	0x00000000	R/W	Bit[0] = LOAD / SEN Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration and initiates reconfiguration state machine. This bit should be asserted when the required settings are already written into Clock Configuration Registers. This bit retains to 0, when the dynamic reconfiguration is done and the clock is locked. Bit[1] = SADDR When written 0, default configuration done in the Clocking Wizard GUI is loaded for dynamic reconfiguration. When written 1, setting provided in the Clock Configuration Registers are used for dynamic reconfiguration.
C_BASEADDR + 0x260 to C_BASEADDR + 0x7FC	Undefined	Undefined	N/A (1)	Do not read/write these registers.
C_BASEADDR + 0x260 to C_BASEADDR + 0x2FC	Undefined	Undefined	N/A	

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
Dynamic Reconfiguration Registers (when the Write DRP feature is enabled)				
C_BASEADDR +0x300	Power Register	FFFF	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x304	CLKOUT0 Register 1	1145	R/W	
C_BASEADDR +0x308	CLKOUT0 Register 2	0000	R/W	
C_BASEADDR +0x30C	CLKOUT1 Register 1	1145	R/W	
C_BASEADDR +0x310	CLKOUT1 Register 2	00C0	R/W	
C_BASEADDR +0x314	CLKOUT2 Register 1 (Not available for PLLE3)	1145	R/W	

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR +0x318	CLKOUT2 Register 2 (Not available for PLLE3)	00C0	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x31C	CLKOUT3 Register 1 (Not available for PLLE3)	1145	R/W	
C_BASEADDR +0x320	CLKOUT3 Register 2 (Not available for PLLE3)	00C0	R/W	
C_BASEADDR +0x324	CLKOUT4 Register 1 (Not available for PLLE3)	1145	R/W	
C_BASEADDR +0x328	CLKOUT4 Register 2 (Not available for PLLE3)	00C0	R/W	
C_BASEADDR +0x32C	CLKOUT5 Register 1	1145	R/W	
C_BASEADDR +0x330	CLKOUT5 Register 2	00C0	R/W	
C_BASEADDR +0x334	CLKOUT6 Register 1 (Not available for PLLE2 or PLLE3)	1145	R/W	
C_BASEADDR +0x338	CLKOUT6 Register 2 (Not available for PLLE2 or PLLE3)	00C0	R/W	
C_BASEADDR +0x33C	DIVCLK Register	1041	R/W	
C_BASEADDR +0x340	CLKFBOUT Register 1	1145	R/W	
C_BASEADDR +0x344	CLKFBOUT Register 2	0000	R/W	
C_BASEADDR +0x348	Lock Register 1	03e8	R/W	
C_BASEADDR +0x34C	Lock Register 2	7001	R/W	
C_BASEADDR +0x350	Lock Register 3	73E9	R/W	
C_BASEADDR +0x354	Filter Register 1	800	R/W	For more information, see *MMCM and PLL Dynamic Reconfiguration* (XAPP888) [Ref 6].
C_BASEADDR +0x358	Filter Register 2	9190	R/W	

Base Address + Offset (hex)	Register Name	Reset Value (hex)	Access Type	Description
C_BASEADDR +0x35C	Clock Configuration Register 24s	0000	R/W	Bit[0] = LOAD / SEN Loads Clock Configuration Register values to the internal register used for dynamic reconfiguration and initiates reconfiguration state machine. This bit should be asserted when the required settings are already written into Clock Configuration Registers. This bit retains to 0, when the dynamic reconfiguration is done and the clock is locked. Bit[1] = SADDR When written 0, default configuration done in the Clocking Wizard GUI is loaded for dynamic reconfiguration. When written 1, setting provided in the Clock Configuration Registers are used for dynamic reconfiguration.
C_BASEADDR + 0x360 to C_BASEADDR + 0x7FC	Undefined	Undefined	N/A	