

Figura 1: Osciloscopio *Tektronix MSO 70404C*

1. Mediciones

1.1. Instrumental utilizado

Para las mediciones se utilizó un osciloscopio *Tektronix MSO 70404C*, en la figura 1 puede observarse el mismo. El instrumento posee 4 GHz de ancho de banda analógico, y una tasa de muestreo de 25 GS/s [1].

1.1.1. Seguridad del instrumento

La impedancia de entrada de entrada del instrumento es de 50Ω y cumplía el rol de carga para el generador de pulsos. Dado el cuantioso costo del instrumento, era fundamental garantizar que el circuito bajo ninguna condición fuera a entregar una potencia potencialmente peligrosa para el mismo.

El osciloscopio tiene especificada una tensión de entrada máxima de $5 V_{RMS}$ para una resolución $\geq 100 mV/div$ y $1 V_{RMS}$ para una resolución $< 100 mV/div$.

En condiciones normales de funcionamiento, la potencia disipada por la carga es mínima, ya que es la potencia que disipa el tren de pulsos en un carga de 50Ω . **REFERENCIAR ACÁ LA SECCIÓN DONDE SE DESARROLLA ESTA EXPRESIÓN.**

No solo es necesario analizar la disipación de potencia en condiciones normales de funcionamiento, sino también para el caso de una falla, ya que el principal objetivo es garantizar la integridad del instrumento en cualquier condición.

En caso de haber alguna falla con algún componente del circuito, el *stub* de salida provee una función de protección. Este componente, para señales con una variación temporal mucho mayor al largo del mismo, actúa como una puesta a tierra.

Entonces, la componente de continua a la salida del generador de pulsos tiene un valor esperado de $0 V$, tanto para condiciones normales de funcionamiento como en presencia de fallas.

En cuanto a la componente alterna de la salida, su valor esperado es extremadamente bajo, ya que únicamente señales de gran ancho de banda pueden ser filtradas y permanecer con una amplitud considerable a la salida del *stub*.

1.2. Banco de medición

En la figura 2 puede observarse el esquema del banco de medición. Consistió de los siguientes bloques



Figura 2: Banco de medición

- Placa FPGA: consistía de una placa de desarrollo *Nexys 4 DDR*. La función de este componente era la de generar el pulso cuadrado unipolar, que controla la *PRF* y el ciclo de trabajo del pulso del driver.
- Fuente de alimentación: provee DC para el driver. La amplitud de esta fuente determina la amplitud del pulso unipolar del driver.
- Driver: tiene tres funciones: actuar como buffer para la FPGA, de manera que la carga exigida sea baja, convertir la amplitud digital del pulso de la FPGA a la amplitud de la fuente de alimentación, y convertir el pulso unipolar en uno bipolar.
- *Pulser*: el *DUT*, genera los pulsos en base a la salida del driver.
- Osciloscopio: instrumento de medición del experimento. Actúa como carga con su impedancia de entrada de $50\ \Omega$.

1.2.1. Fuente de alimentación

La fuente de alimentación era una *Marconi Instruments TF2154*, en la figura 3 puede observarse la misma.

Presentaba limitación de corriente regulable e indicadores para la amplitud y la corriente entregada, lo que permitía trabajar de manera segura, dentro de los límites de consumo obtenidos en las simulaciones anteriores.

Como fuese explicado en la sección [REFERENCIAR SECCIÓN CON CALCULOS DE POTENCIA](#), la corriente máxima esperada en las condiciones de trabajo era menor a 200 mA, por lo que se monitoreó durante todo el experimento que la corriente entregada por la fuente no supere este máximo teórico.

1.2.2. FPGA

La *FPGA* cumplía el rol de excitación del sistema. Generaba el pulso unipolar cuadrado de entrada, que controla la *PRF* y el ciclo de trabajo del pulso del driver. La *FPGA* utilizada es una *Artix-7* de Xilinx, montada en una placa de desarrollo *Nexys-4 DDR* de Digilent [2]. En la figura 4 puede observarse la misma.

Las variables de ajuste del pulso unipolar eran las siguientes

- Frecuencia: la frecuencia de la señal cuadrada de entrada es igual a la frecuencia de repetición de pulsos (*PRF*) del sistema.
- Ciclo de trabajo: el ciclo de trabajo de la señal cuadrada unipolar determina los extremos de la señal cuadrada unipolar. A mayor ciclo de trabajo, valores más negativos.



Figura 3: Fuente de alimentación *Marconi Instruments TF2154*.



Figura 4: *FPGA* utilizada

Como fuera explicado anteriormente, a mayor tensión en la porción negativa del pulso bipolar, mayor amplitud de pulso a la salida. Por lo tanto, para poder medir las mayores amplitudes de pulso posibles era necesario para el pulso de entrada alcanzar el mayor ciclo de trabajo posible.

El circuito programado en la FPGA consistía en un generador de cuadrada con ciclo de trabajo variable. Mediante dos botones externos, disponibles en la placa, se podía subir y bajar el ciclo de trabajo.

Debido a la implementación del circuito, con un contador, el paso de estas variaciones era de un 10 %.

En el anexo A puede inspeccionarse el *HDL* del circuito implementado.

1.3. Mediciones realizadas

Las mediciones consistieron en mediciones en del dominio del pulso de salida. Se midieron tiempo de crecimiento, tiempo de decaimiento, amplitud máxima, y ancho a medio máximo (*FWHM* del inglés *Full Width at Half Maximum*).

Se realizaron distintas mediciones para distintas condiciones del circuito. Se barrió para el pulso digital de entrada, el ciclo de trabajo, y para la fuente de alimentación distintos valores de

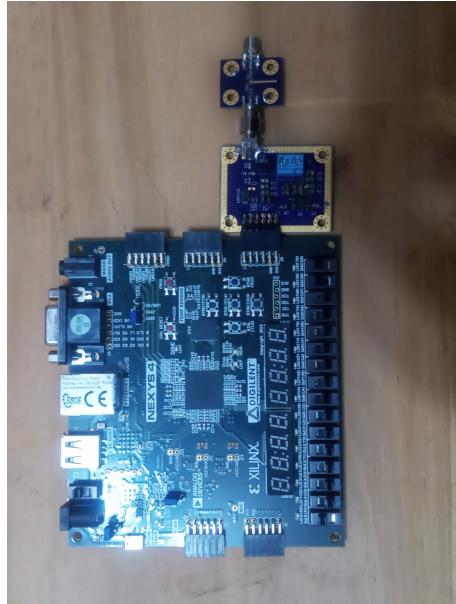


Figura 5: *FPGA, driver y pulser.*

tensión.

- Para la amplitud de la fuente, se utilizaron valores de 5 V y 7 V.
 - 5 V por ser un valor fácilmente obtenible en los sistemas *UWB* de referencia.
 - 7 V por ser la máxima amplitud tolerable por el circuito. Tensiones de alimentación mayores a estas resultan en corrientes de polarización mayores a las máximas admisibles dado los dimensionamientos de las pistas de los *PCBs*.
- El ciclo de trabajo se barrió entre 50 % y 70 %.
 - Se tomo 50 % como límite inferior por ser un valor fácilmente obtenible como división de un reloj digital.
 - Se tomo 70 % como límite superior ya que se observó que valores superiores a este resultaban en un pulso bipolar con amplitudes negativas decrecientes, y por lo tanto, amplitudes de pulso decrecientes.
 - La teoría no indicaba un límite superior para el ciclo de trabajo. Sin embargo, este se observó en la práctica debido a no idealidades en el pulso de salida del driver, que no era perfectamente cuadrado.

En la figura 5 puede observarse el *pulser* junto con el *driver* y la *FPGA*.

1.3.1. Mediciones previas

Previo a las mediciones principales, se realizó una medición de la salida del driver, con el objetivo de validar el pulso bipolar generado.

El motivo de esta medición previa, fue a la limitada disponibilidad del osciloscopio de gran ancho de banda utilizado para la medición final del pulso. Esta pre-medición del pulso bipolar, se realizó con un osciloscopio de bajo ancho de banda, ya que el objetivo era validar los niveles de tensión del pulso, y su correcta variación con la variación del ciclo de trabajo del pulso unipolar.

En la figura 6 puede observarse el banco de medición. Los resultados fueron los esperados y, por lo tanto, no se requirió ninguna iteración sobre el diseño.

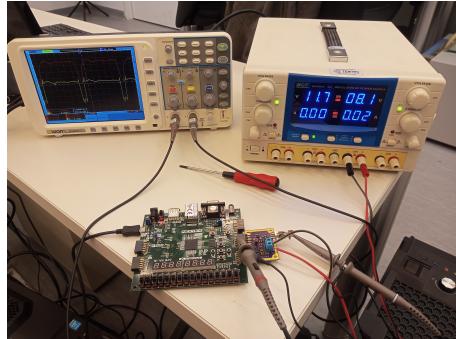


Figura 6: Banco de mediciones previas a la medición final del pulso.



Figura 7: Salida @ V_{cc} 5 V, D 50 %

1.4. Resultados

En las figuras 7, 8, 9, 10, 11 pueden observarse los resultados en diversas capturas de pantalla tomadas del osciloscopio.

Se observó en las mediciones una amplitud de pulso creciente con mayor ciclo de trabajo y mayor amplitud de pulso, como era esperado. La menor amplitud de pulso obtenida fue de 380 mV para un V_{cc} de 5 V y un D de 50 %, y la mayor fue de 1,12 V para un V_{cc} de 7 V y un D de 70 %

En cuanto al ancho de pulso, se mantuvo aproximadamente constante en 160 ps, al igual que los tiempos de crecimiento y decrecimiento, que se mantuvieron constantes en 90 ps.

En la tabla 1 pueden observarse los resultados obtenidos.

V_{cc} [V]	D [%]	A [V]	FWHM [ps]	3 dB B [GHz]	t_r [ps]	t_f [ps]
5	50	0.380	159	7.5	93	88
5	70	0.625	161	3.6	93	91
7	50	0.702	162	3	93	93
7	60	0.909	164	3	94	95
7	70	1.120	165	3	95	96

Cuadro 1: Resultados de mediciones.

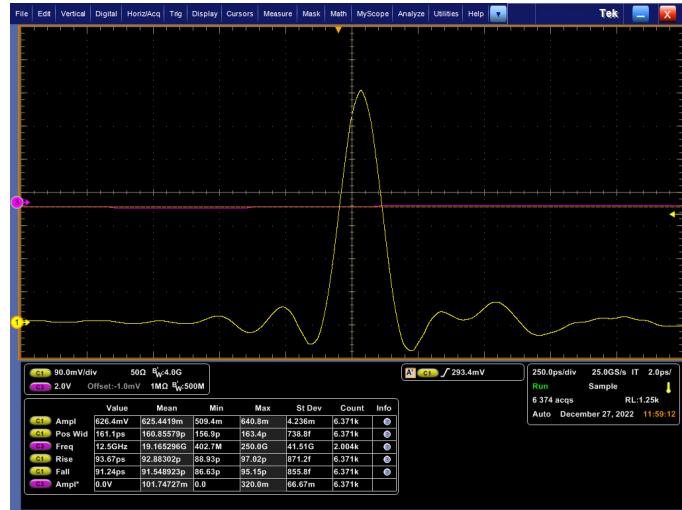


Figura 8: Salida @ V_{cc} 5 V, D 70 %



Figura 9: Salida @ V_{cc} 7 V, D 50 %

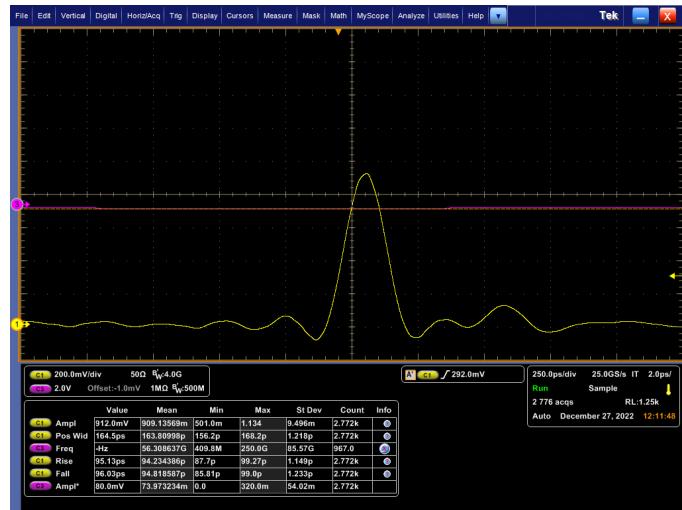


Figura 10: Salida @ V_{cc} 7 V, D 60 %

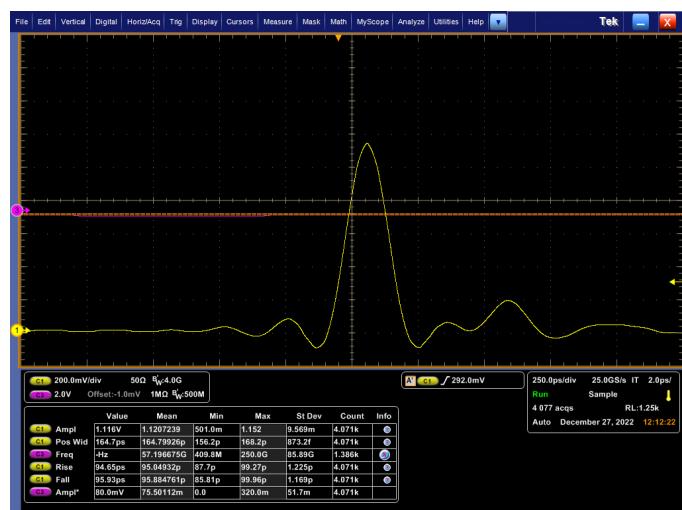


Figura 11: Salida @ V_{cc} 7 V, D 70 %

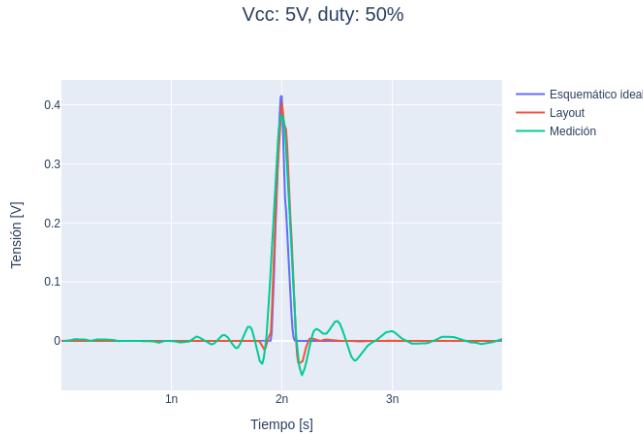


Figura 12: Pulso @ V_{cc} 5 V, D 50 %

1.4.1. Comparación contra simulación

En las figuras pueden observarse los resultados de las mediciones obtenidas superpuestos con los resultados de simulación para las mismas condiciones de trabajo (amplitud de alimentación y ciclo de trabajo).

Para las simulaciones, se toman dos resultados: el de una simulación “ideal”, indicada como “esquemático ideal” en las leyendas, que se corresponde a una simulación sin contemplar parásitos de ningún tipo. El otro resultado corresponde a una simulación mas completa, en las leyendas “Layout”, es una simulación en la que se extrajeron previamente los efectos parásitos del PCB, y se incorporaron en la simulación del pulso.

Se realizan las comparaciones en el dominio del tiempo y de la frecuencia.

En el dominio del tiempo, se observa una buena coincidencia entre la amplitud de los pulsos y el ancho. Se observa una diferencia en el *ringing* de ambos. Las simulaciones prácticamente no presentan oscilaciones alrededor del pulso, mientras que las mediciones las presentan tanto previa como posteriormente. También se observa un segundo pulso de menor amplitud siguiendo al primero.

Como causa de estas discrepancias, se descarta un efecto del PCB no modelado, ya que los parásitos de esta estructura fueron extraídos por una simulación electromagnética, y sus efectos contemplados en las simulaciones del *layout*.

Estas discrepancias sugieren una limitación en el modelado de alguno de los dispositivos, tanto el SRD como el Schottky. Las simulaciones predijeron correctamente la amplitud y el ancho de los pulsos resultantes, pero fallaron en predecir el ringing y el pulso secundario.

EN REALIDAD NO, SE VE QUE EL PULSO 5V@50D TIENE UN ANCHO DE BANDA *MUCHO* MAYOR A LAS SIMULACIONES Y A LAS DEMÁS MEDICIONES. HAY Q VER Q PASÓ AHÍ

1.4.2. Comparación con resultados de la literatura

En la tabla ?? se resumen resultados reportados para generadores de pulsos *UWB* en la literatura.

PSD, Vcc: 5V, duty: 50%

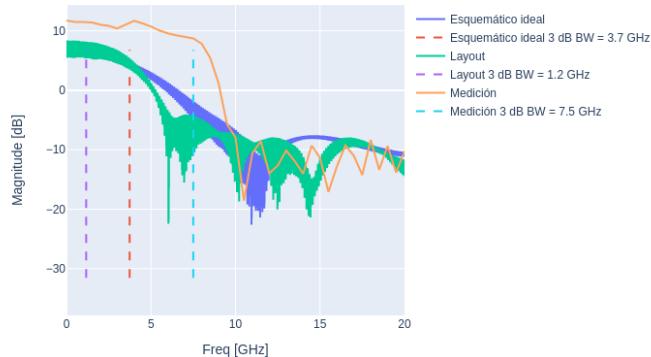


Figura 13: PSD @ V_{cc} 5 V, D 50 %

Vcc: 5V, duty: 70%

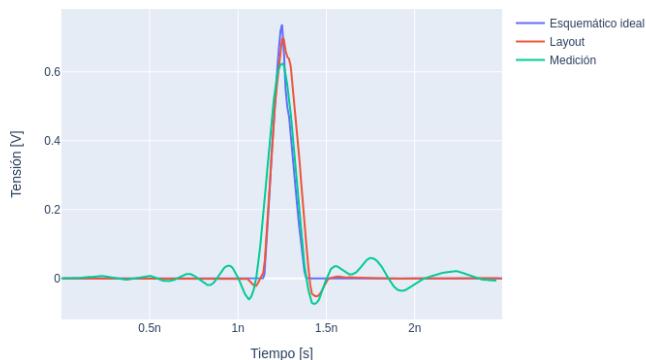


Figura 14: Pulso @ V_{cc} 5 V, D 70 %

PSD, Vcc: 5V, duty: 70%

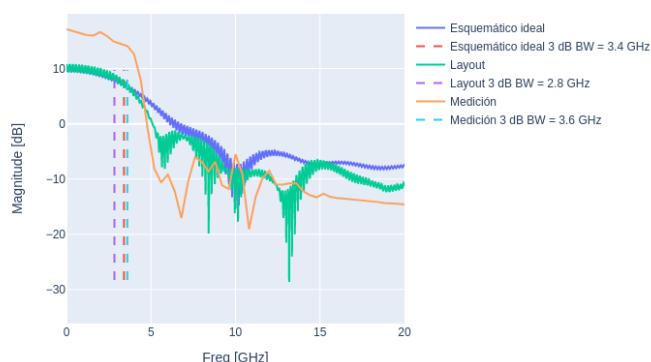


Figura 15: PSD @ V_{cc} 5 V, D 70 %



Figura 16: Pulso @ V_{cc} 7 V, D 50 %



Figura 17: PSD @ V_{cc} 7 V, D 50 %



Figura 18: Pulso @ V_{cc} 7 V, D 60 %



Figura 19: PSD @ V_{cc} 7 V, D 60 %

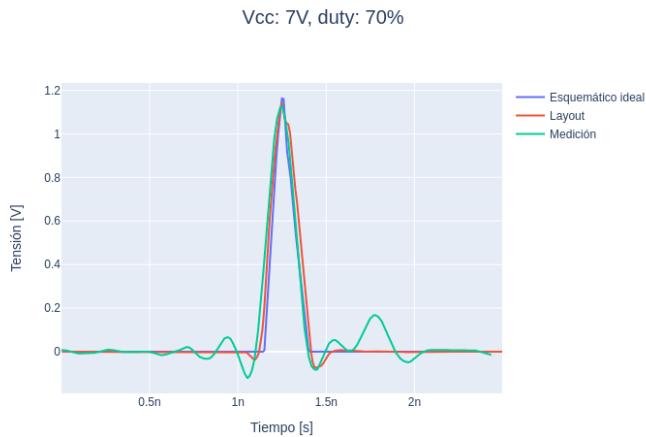


Figura 20: Pulso @ V_{cc} 7 V, D 70 %

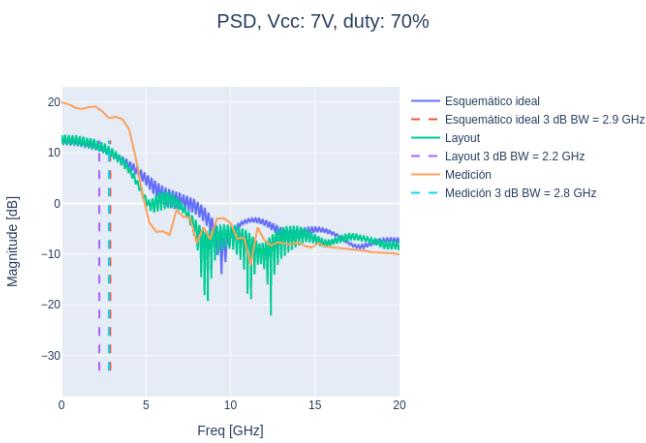


Figura 21: PSD @ V_{cc} 7 V, D 70 %

Cuadro 2: Resultados reportados en la literatura

Referencia	<i>A</i> [V]	<i>FWHM</i> [ps]	Bal ^a	Bias	Dispositivos	<i>V_{cc}</i> [V]	<i>V_{in}</i> [V]	<i>PRF</i> [MHz]
[3]	$\pm 0,896, \pm 1,6$ ^b	335, 511	Sí	Int	SRD	+5	TTL	50
[4]	7,5	110	No	Ext	SRD+3TBJ+SD	+12	TTL	5
[5]	0,8	170	No	Int	SRD	+4	4	10
[6]	200	300	No	Ext	SRD+2SD	?	?	10
[7]	6, 4	150	No	Int	SRD+L	?	5	12
[8]	1,27 ^c	286	No	Int	2SRD+L	10	10 ^d	?

^a Balanceado.

^b la publicación presenta dos resultados, correspondientes a circuitos con componentes concentrados y distribuidos.

^c la publicación presenta múltiples resultados, se muestran los mejores.

^d la señal de entrada es senoidal.

Referencias

- [1] Tektronix, Inc., “Osciloscopio MSO 70404C.” Datasheet, 2023. Disponible online en <https://download.tek.com/datasheet/DPO-DSA-MSO70000-DataSheet-EN-11Apr23.pdf>.
- [2] Digilent, “Nexys 4 DDR.” Página Oficial, 2023.
- [3] P. Rulikowski and J. Barrett, “Truly balanced step recovery diode pulse generator with single power supply,” in *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No.04TH8746)*, pp. 347–350, 2004.
- [4] P. Protiva, J. Mrkvica, and J. Macháč, “A compact step recovery diode subnanosecond pulse generator,” *Microwave and Optical Technology Letters*, vol. 52, no. 2, pp. 438–440, 2010.
- [5] A. Kamal, A. Bhattacharya, M. Tamrakar, and C. Roy, “Low-ringing and reduced-cost step recovery diode based uwb pulse generators for gpr applications,” *Microwave and Optical Technology Letters*, vol. 56, no. 10, pp. 2289–2294, 2014.
- [6] J. Han and C. Nguyen, “A new ultra-wideband, ultra-short monocycle pulse generator with reduced ringing,” *IEEE Microwave and Wireless Components Letters*, vol. 12, no. 6, pp. 206–208, 2002.
- [7] J. Han and C. Nguyen, “Coupled-slotline-hybrid sampling mixer integrated with step-recovery-diode pulse generator for uwb applications,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 6, pp. 1875–1882, 2005.
- [8] D. Oloumi and E. Fear, “A picosecond pulse generator using srd diodes: Design, analysis, and measurements,” in *2018 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium)*, pp. 159–160, 2018.

A. Código Fuente Verilog

```
'timescale 1ns / 1ps

module top(
    input clk ,
    input btnU , btnD ,
    input btnL , btnR ,
    input [1:0] sw ,
    output [1:0] led ,
```

```

output [0:0] JA
);

wire duty_inc_coarse , duty_inc_fine , duty_dec_coarse , duty_dec_fine;

localparam W = 8;
reg [W-1:0] div_value = 'd10;
reg [1:0] sw_old = 'd0;
reg initialized = 'b0;
reg [3:0] init_counter = 'b0;
reg srst = 'b0;

localparam real DUTY_CYCLE_NOMINAL = 0.5;
localparam real DUTY_CYCLE_COARSE = 0.1;
localparam real DUTY_CYCLE_FINE = 0.01;

localparam integer DIV_VALUE_10MHZ = 10;
localparam integer COUNTS_DUTY_NOMINAL_10MHZ
localparam integer COUNTS_DUTY_COARSE_10MHZ
localparam integer COUNTS_DUTY_FINE_10MHZ
*DIV_VALUE_10MHZ;

localparam integer DIV_VALUE_5MHZ = 20;
localparam integer COUNTS_DUTY_NOMINAL_5MHZ
localparam integer COUNTS_DUTY_COARSE_5MHZ
localparam integer COUNTS_DUTY_FINE_5MHZ

localparam integer DIV_VALUE_1MHZ = 100;
localparam integer COUNTS_DUTY_NOMINAL_1MHZ
localparam integer COUNTS_DUTY_COARSE_1MHZ
localparam integer COUNTS_DUTY_FINE_1MHZ
*DIV_VALUE_1MHZ;

reg [W-1:0] div_value_curr ,
counts_duty_nominal_curr ,
counts_duty_coarse_curr ,
counts_duty_fine_curr ;

always @(posedge clk) begin
if (sw[1]) begin
div_value_curr <= DIV_VALUE_1MHZ;
counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_1MHZ;
counts_duty_coarse_curr <= COUNTS_DUTY_COARSE_1MHZ;
counts_duty_fine_curr <= COUNTS_DUTY_FINE_1MHZ;
end
else if (sw[0]) begin
div_value_curr <= DIV_VALUE_5MHZ;
counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_5MHZ;
counts_duty_coarse_curr <= COUNTS_DUTY_COARSE_5MHZ;
counts_duty_fine_curr <= COUNTS_DUTY_FINE_5MHZ;
end
else begin
div_value_curr <= DIV_VALUE_10MHZ;
counts_duty_nominal_curr <= COUNTS_DUTY_NOMINAL_10MHZ;

```

```

        counts_duty_coarse_curr      <= COUNTS_DUTY_COARSE_10MHZ;
        counts_duty_fine_curr       <= COUNTS_DUTY_FINE_10MHZ;
    end
    sw_old <= sw;
end

always @(posedge clk) begin
    if (initialized == 'b0) begin
        if (init_counter < 'd14) begin
            init_counter <= init_counter+1;
        end
        else begin
            initialized <= 'b1;
            srst <= 'b1;
        end
    end
    else begin
        if (sw_old != sw) srst <= 'b1;
    end
    if (srst) srst <= 'b0;
end

// debounce of buttons
debounce u_debounce_inc_coarse(clk,btnU,duty_inc_coarse);
debounce u_debounce_dec_coarse(clk,btnD,duty_dec_coarse);
debounce u_debounce_inc_fine(clk,btnR,duty_inc_fine);
debounce u_debounce_dec_fine(clk,btnL,duty_dec_fine);

// PWM generator
adhoc_generator#.WIDTH(W) u_generator(
    .clk(clk),
    .srst(srst),
    .duty_coarse(counts_duty_coarse_curr),
    .duty_fine(counts_duty_fine_curr),
    .duty_nominal(counts_duty_nominal_curr),
    .div_value(div_value_curr),
    .duty_inc_coarse(duty_inc_coarse),
    .duty_inc_fine(duty_inc_fine),
    .duty_dec_coarse(duty_dec_coarse),
    .duty_dec_fine(duty_dec_fine),
    .PWM(JA[0])
);
assign led = sw;

endmodule

module debounce(
    input clk,
    input data_in,
    output data_out
);
    localparam DEBOUNCE_BITS = 23;

```

```

reg [DEBOUNCE_BITS-1:0] debounce_counter;
wire debounce_enable;
wire tmp_1 , tmp_2;

// debounce enable generation , has period T_clk/2**DEBOUNCE_BITS
always @(posedge clk) debounce_counter = debounce_counter + 'b1;
assign debounce_enable = debounce_counter == 2**DEBOUNCE_BITS-1 ? 'b1 : 'b0;

// debounce of buttons
DFF u_DFF_inc_coarse(clk , debounce_enable , data_in , tmp_1);
DFF u_DFF_dec_coarse(clk , debounce_enable , tmp_1 , tmp_2);

assign data_out = tmp_1 & (~ tmp_2) & debounce_enable;

endmodule

module DFF(
    input clk ,
    input en ,
    input D,
    output reg Q
);
    always @(posedge clk) begin
        if (en) Q <= D;
    end
endmodule

module pwm_generator(
    input clk ,           // 100MHz clock input
    input increase_duty , // input to increase 10% duty cycle
    input decrease_duty , // input to decrease 10% duty cycle
    output PWMOUT
);
    localparam DEBOUNCE_COUNT = 25_000_000;

    wire slow_clk_enable;           // slow clock enable signal for debouncing FFs
    reg[27:0] counter_debounce=0;   // counter for creating slow clock enable signals
    wire tmp1,tmp2,duty_inc;       // temporary flip-flop signals for debouncing the
    wire tmp3,tmp4,duty_dec;       // temporary flip-flop signals for debouncing the
    reg[3:0] counter_PWM=0;        // counter for creating 10Mhz PWM signal
    reg[3:0] DUTY_CYCLE=5;         // initial duty cycle is 50%

    always @(posedge clk) begin
        counter_debounce <= counter_debounce + 1;
        if(counter_debounce>=DEBOUNCE_COUNT) counter_debounce <= 0;
    end

    assign slow_clk_enable = counter_debounce == DEBOUNCE_COUNT ?1:0;

// debouncing FFs for increasing button
DFF_PWM PWM_DFF1(clk , slow_clk_enable , increase_duty ,tmp1);
DFF_PWM PWM_DFF2(clk , slow_clk_enable ,tmp1 , tmp2);

```

```

assign duty_inc = tmp1 & (~ tmp2) & slow_clk_enable;

// debouncing FFs for decreasing button
DFFPWM PWM_DFF3(clk, slow_clk_enable, decrease_duty, tmp3);
DFFPWM PWM_DFF4(clk, slow_clk_enable, tmp3, tmp4);
assign duty_dec = tmp3 & (~ tmp4) & slow_clk_enable;

// vary the duty cycle using the debounced buttons above
always @(posedge clk) begin
    if (duty_inc==1 && DUTY_CYCLE <= 9) DUTY_CYCLE <= DUTY_CYCLE + 1;
    else if (duty_dec==1 && DUTY_CYCLE>=1) DUTY_CYCLE <= DUTY_CYCLE - 1;
end

// Create 10MHz PWM signal with variable duty cycle controlled by 2 buttons
always @(posedge clk) begin
    counter_PWM <= counter_PWM + 1;
    if (counter_PWM>=9) counter_PWM <= 0;
end

assign PWMOUT = counter_PWM < DUTY_CYCLE ? 1 : 0;

endmodule

// Debouncing DFFs for push buttons on FPGA
module DFF_PWM(
    input clk,
    input en,
    input D,
    output reg Q
);
    always @(posedge clk) begin
        if (en) Q <= D;
    end
endmodule

```