

Diseño Digital Avanzado

Unidad 3 - Arquitectura Multi-Rate

Dr. Ariel L. Pola

apola@fundacionfulgor.org.ar

October 8, 2021

Tabla de Contenidos

1. Arquitectura Multi-Rate

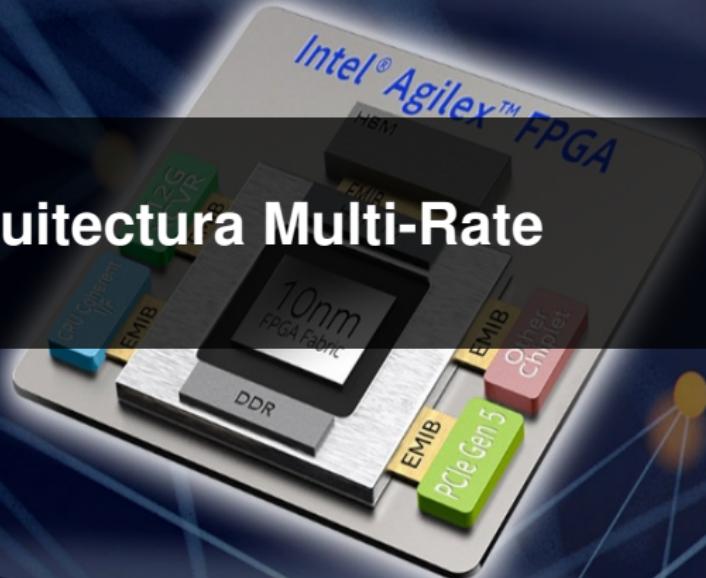
2. Simulación

3. Síntesis

4. Implementación

5. Trabajo de Laboratorio

Arquitectura Multi-Rate



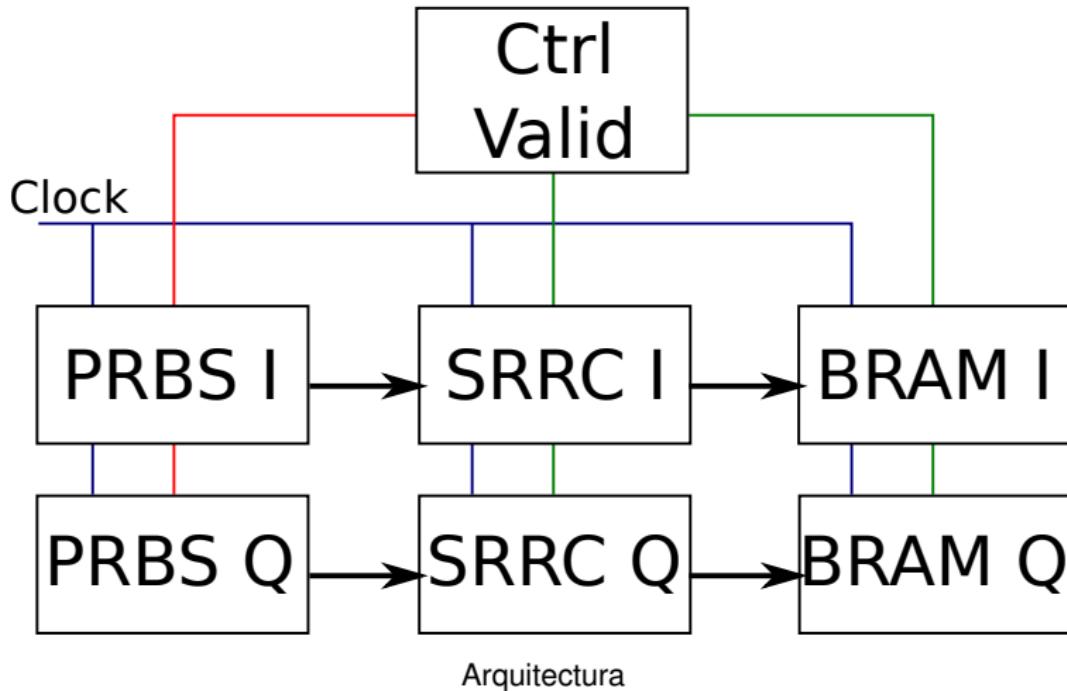
Objetivo del laboratorio

- Obtener elementos de análisis de eficiencia de la arquitectura implementada.

Ejemplo - Esquema general

- La arquitectura consiste en:
 - Un solo reloj que se distribuye a todos los módulos.
 - Un bloque de control que genera las señales de Valid.
 - Dos generadores de PRBS9.
 - Dos filtros SRRC - OverSample: 8.
 - Memorias BRAM para guardar las salidas de los filtros.
 - El modelo se sintetizó a las frecuencias de reloj 50MHz, 100MHz y 200MHz con el objetivo de analizar el efecto que causa sobre los caminos críticos el aumento de la frecuencia de trabajo.

Arquitectura Multi-Rate



Simulación



Arquitectura Multi-Rate

Simulación

Configuración para Simulación

- Antes de simular el sistema, verificar que se tengan los puertos habilitados (Fig. 1a) y las variables (Fig. 1b) e instancias de los módulos VIO e ILA (Fig. 1c) comentados.
- Ejecutar la simulación como **Run Behavioral Simulation**.

Utilizando formas de onda analógicas

- En muchos casos se puede interpretar mejor el comportamiento de un bloque observando la forma de onda de la misma forma que en el osciloscopio.
- Para configurar el simulador necesitamos configurar las características de la señal como **Signed Decimal** y luego cambiar el estilo de la señal como **analógica**.
- Las señales que se deben configurar son connect_srrc_to_dac_I, connect_srrc_to_dac_Q, o_data_from_ram_I y o_data_from_ram_Q. (Fig. 2 y 3)

Arquitectura Multi-Rate

Simulación

```
////////////////////////////////////////////////////////////////
// COMMENT TO IMPLEMENT IN FPGA
////////////////////////////////////////////////////////////////
output [NB_DATA_RAM_LOG*2 - 1 : 0] o_data_from_ram,
output reg   o_full_from_ram,           /// RAM Full signal
input        i_prbs_enable,           /// Enable PRBS
input        i_enable_src,            /// Enable SRRC
input [NB_ADDR_RAM_LOG - 1 : 0] i_ram_addr_from_micro,    /// Address for Read Memory
input        i_ram_run_from_micro,    /// Enable Load Memory
input        i_soft_reset,             /// Reset
input        i_ram_read_from_counter, /// RAM Read from counter
input        clock                   /// Clock
);
```

a)

```
////////////////////////////////////////////////////////////////
// UNCOMMENT TO IMPLEMENT IN FPGA
////////////////////////////////////////////////////////////////
// wire [NB_DATA_RAM_LOG*2 - 1 : 0] o_data_from_ram;
// reg   o_full_from_ram;           /// RAM Full signal
// wire   i_prbs_enable;           /// Enable PRBS
// wire   i_enable_src;            /// Enable SRRC
// wire [NB_ADDR_RAM_LOG - 1 : 0] i_ram_addr_from_micro;    /// Address for Read Memory
// wire   i_ram_run_from_micro;    /// Enable load Memory
// wire   i_soft_reset;             /// Reset
// wire   i_ram_read_from_counter; /// RAM Read from counter
////////////////////////////////////////////////////////////////
```

b)

```
////////////////////////////////////////////////////////////////
// UNCOMMENT TO IMPLEMENT IN FPGA
////////////////////////////////////////////////////////////////
// vio
//   u_vio
//     .probe_out0_0 (i_soft_reset      ), /// Reset
//     .probe_out1_0 (i_prbs_enable    ), /// PRBS Enable
//     .probe_out2_0 (i_enable_srcrc  ), /// SRRC Enable
//     .probe_out3_0 (i_ram_run_from_micro), /// RAM Init
//     .probe_out4_0 (i_ram_addr_from_micro), /// RAM Address to read
//     .probe_out5_0 (i_ram_read_from_counter), /// RAM read from counter
//     .probe_in0_0 (o_full_from_ram   ), /// RAM Full
//     .probe_in1_0 (o_data_from_ram   ), /// RAM Data
//     .clk_0       (clock              ) /// Clock
//   );
// 
```

```
// // ILA Instance
//   ila
//     u_ilal
//     (
//       .probe0_0 (data_from_ram_I), /// Data from RAM
//       .probe1_0 (data_from_ram_Q), /// Data from RAM
//       .probe2_0 (ram_address     ), /// RAM Address
//       .probe3_0 (o_full_from_ram ), /// RAM Full
//       .clk_0   (clock              ) /// Clock
//     );
// 
```

c)

Figure: Acomodar los comentarios para simulaciones o síntesis.

Arquitectura Multi-Rate

Simulación

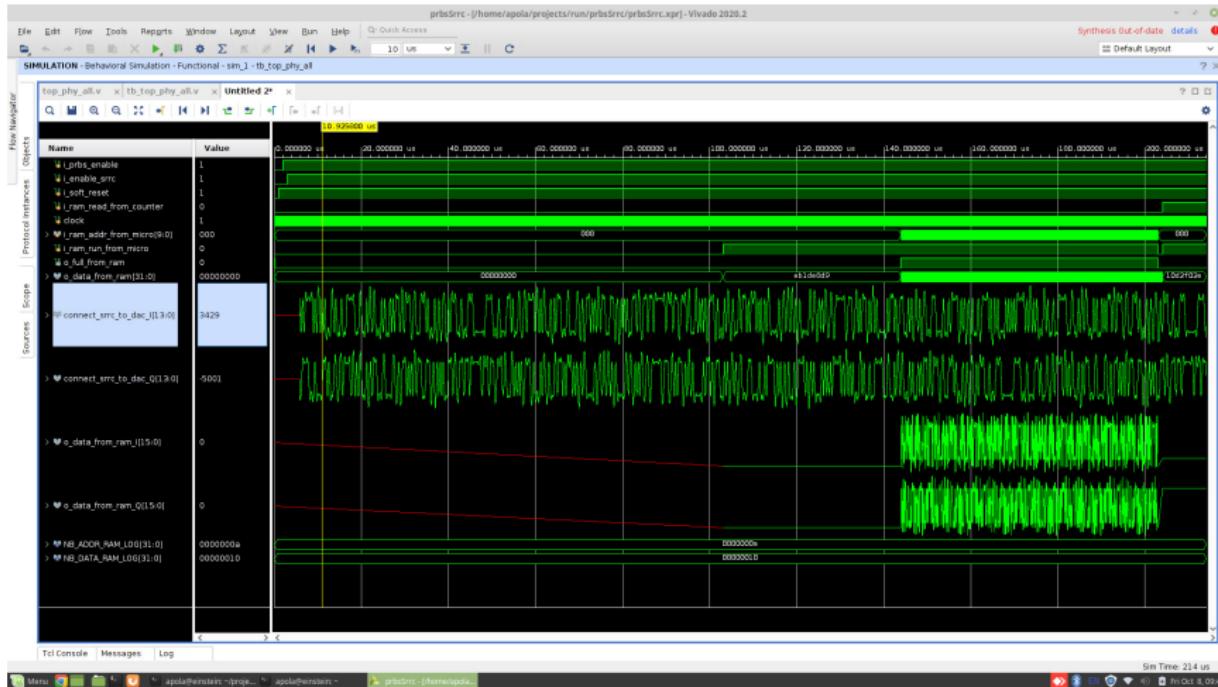


Figure: Formas de ondas analógicas en las simulaciones.

Arquitectura Multi-Rate

Simulación

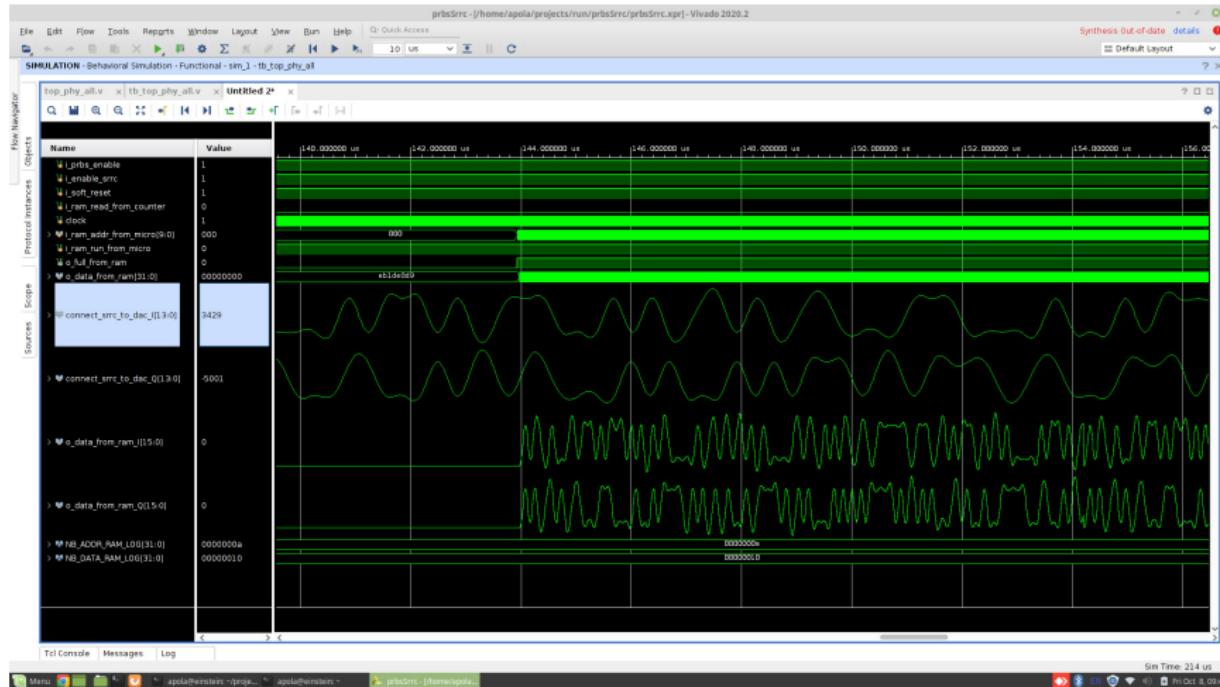


Figure: Formas de ondas analógicas en las simulaciones.

Síntesis



Arquitectura Multi-Rate

Síntesis

Preparar el diseño

- Antes de sintetizar, necesitamos agregar a nuestro diseño los módulos VIO e ILA.
- Comentar los puertos (Fig. 1a) y descomentar las variables (Fig. 1b) e instancias de los módulos VIO e ILA (Fig. 1c).
- Generar los módulos VIO e ILA según especificaciones que se observan en las Fig. 4 y Fig. 5.

Arquitectura Multi-Rate

Síntesis

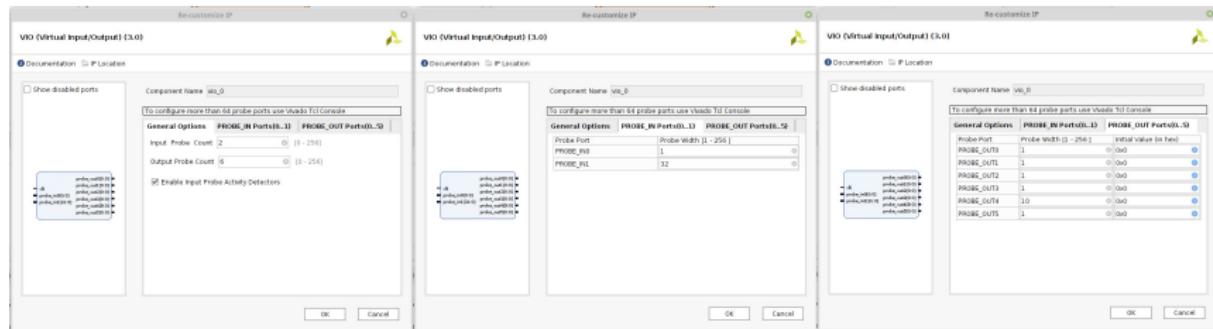


Figure: Configuración VIO.

Arquitectura Multi-Rate

Síntesis

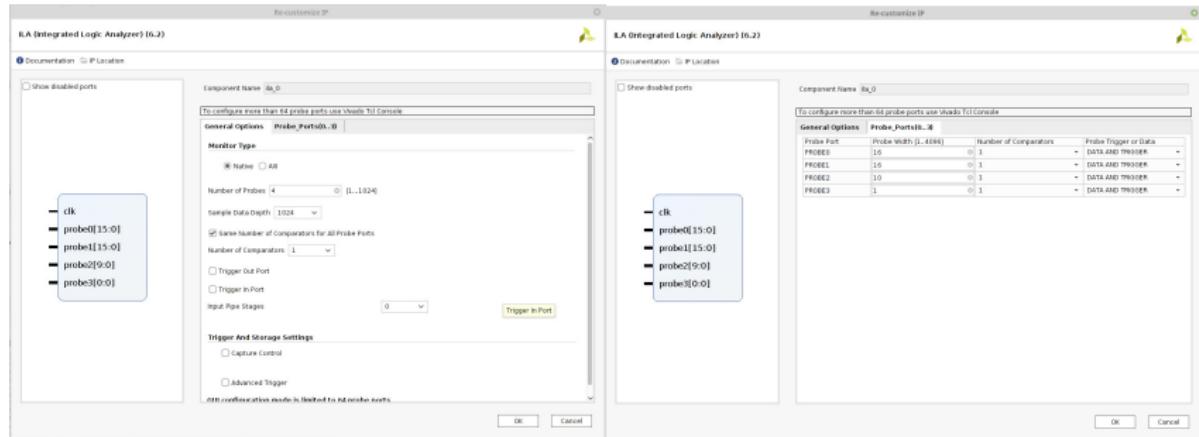


Figure: Configuración ILA.

Observando la complejidad

- En la etapa de síntesis vamos a poder observar las diferentes optimizaciones del diseño que la herramienta va generando a medida que procesa el código Verilog.
- Genera una primera versión de la complejidad mostrando que componentes utiliza.
- En la ventana **Log** vamos a encontrar el detalle de cada paso de la síntesis.
- Es importante poder observar que se hayan generado las jerarquías y utilizado los componentes que mejor reducción de complejidad brinde al diseño.

Arquitectura Multi-Rate

Síntesis

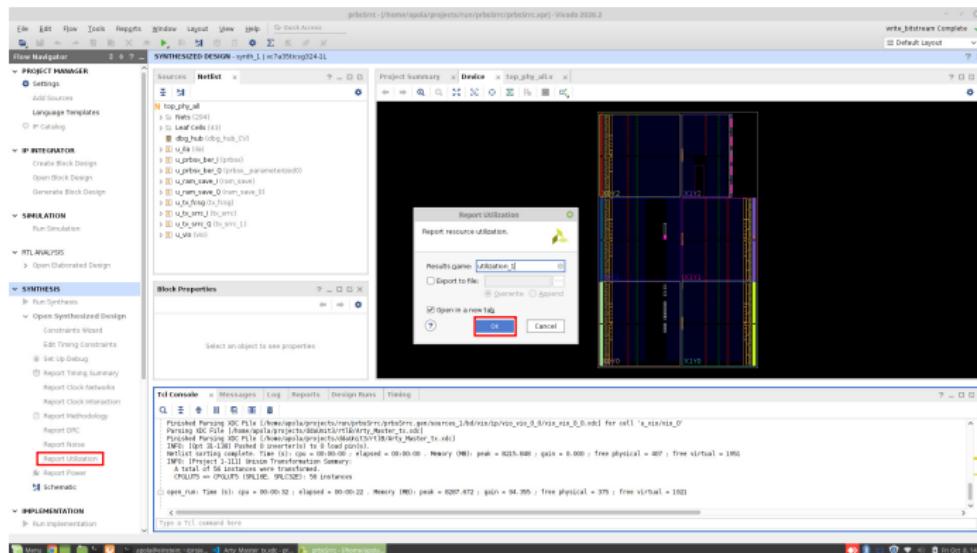


Figure: Reporte de utilización.

Arquitectura Multi-Rate

Síntesis

Name	^_1	Slice LUTs (20800)	Slice Registers (41600)	F7 Muxes (16300)	Block RAM Tile (50)	DSPs (90)	Bonded IOB (210)	BUFGCTRL (32)
top_phy_all	N	1358	2130	21	2.5	1	1	1
dbg_hub (dbg_hub_CV)	T	0	0	0	0	0	0	0
u_ilia (ila)	> I	799	1377	18	1.5	0	0	0
u_prbsx_ber_I (prbsx)	I	1	9	0	0	0	0	0
u_prbsx_ber_Q (prbsx_parameterized0)	I	4	8	0	0	0	0	0
u_ram_save_I (ram_save)	> I	16	38	0	0.5	0	0	0
u_ram_save_Q (ram_save_0)	> I	11	28	0	0.5	0	0	0
u_tx_fcsg (tx_fcsg)	I	16	12	0	0	1	0	0
u_tx_src_I (tx_src)	I	180	85	0	0	0	0	0
u_tx_src_Q (tx_src_1)	I	87	82	0	0	0	0	0
u_vio (vio)	> I	235	465	3	0	0	0	0

Figure: Reporte de utilización.

Implementación



Arquitectura Multi-Rate

Implementación - Timing

Analizando los caminos críticos

- En la última etapa es importante identificar los problemas de **timing** que puedan aparecer en el diseño.
- También es de interés conocer hasta cuanto podemos aumentar la frecuencia de trabajo según un análisis de todos los caminos críticos.
- La herramienta nos brinda un detalle de cada **path** especificando todos los componentes que utilizó entre un punto de origen y destino.
- Se pudo utilizar como referencia el documento UG906 de Xilinx.
- El reporte de timing se obtiene ejecutando la siguiente opción de la barra de herramientas: *Reports - Timing - Report Timing Summary*.

Arquitectura Multi-Rate

Implementación - Timing

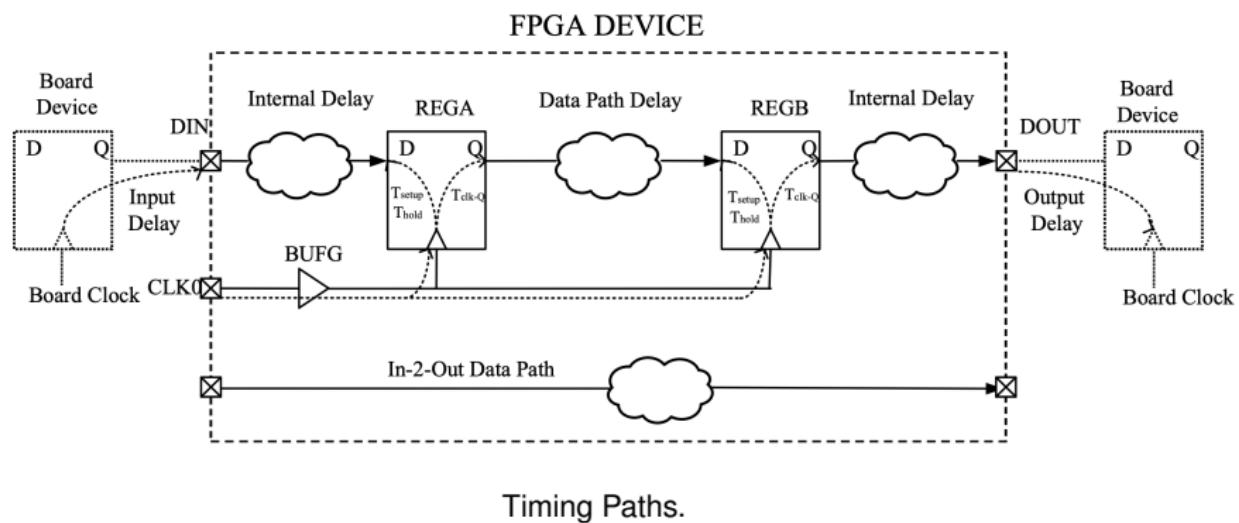
Analizando los caminos críticos

- Las secciones de análisis son

- **Source Clock Path:** Es el camino seguido por el reloj de la fuente desde su punto de origen (típicamente un puerto de entrada) hasta el pin del reloj del registro. Para una ruta de tiempo que comienza en un puerto de entrada, no hay un camino de fuente de reloj.
- **Data Path:** Es la sección del camino de tiempo en la que se propagan los datos, entre el punto de inicio del camino y el punto final del mismo. Se aplican las siguientes definiciones: (1) El punto de inicio del camino es un pin de reloj de un registro o un puerto de entrada de datos; y (2) El punto final del trayecto es un pin de entrada de datos de un registro o un puerto de salida de datos.
- **Destination Clock Path:** La ruta del reloj de destino es el camino seguido por el reloj de destino desde su punto de origen, típicamente un puerto de entrada, hasta el pin del reloj del registro. Para una ruta de tiempo que termina en un puerto de salida, no hay una ruta de reloj de destino.

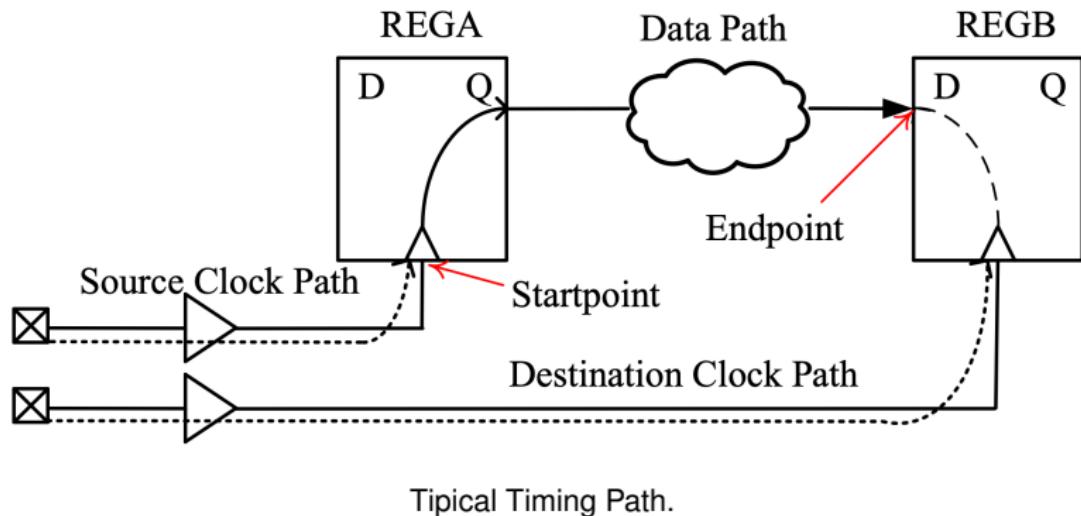
Arquitectura Multi-Rate

Implementación - Timing



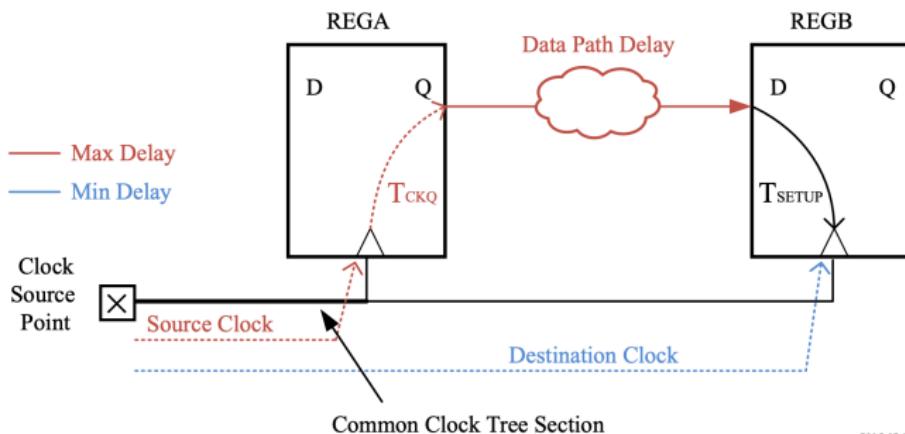
Arquitectura Multi-Rate

Implementación - Timing



Arquitectura Multi-Rate

Implementación - Timing



Common Clock Tree Section.

Arquitectura Multi-Rate

Implementación - Timing

```
Slack (MET) :          0.142ns  (arrival time - required time)
Source:           u_ram_save_Q/counter_reg[8]/C
                  (rising edge-triggered cell FDRE clocked by clock {rise@0.000ns fall@5.000ns period=20.000ns})
Destination:        u_ram_save_Q/u_bram/ram_reg_0_0/ADDRARDADDR[8]
                  (rising edge-triggered cell RAMB36E1 clocked by clock {rise@0.000ns fall@5.000ns period=20.000ns})
Path Group:         clock
Path Type:          Hold (Min at Fast Process Corner)
Requirement:        0.000ns  (clock rise@0.000ns - clock rise@0.000ns)
Data Path Delay:    0.635ns  (logic 0.164ns (25.842%) route 0.471ns (74.158%))
Logic Levels:       0
Clock Path Skew:   0.309ns (DCD - SCD - CPR)
Destination Clock Delay (DCD): 2.020ns
Source Clock Delay (SCD): 1.461ns
Clock Pessimism Removal (CPR): 0.250ns
```

Reporte de timing con un clock de 50MHz.

Arquitectura Multi-Rate

Implementación - Timing

Location	Delay type	Incr(ns)	Path(ns)	Netlist Resource(s)
E3	(clock clock rise edge)	0.000	0.000 r	
	net (fo=0)	0.000	0.000 r	clock (IN)
E3	IBUF (Prop_ibuf_I_0)	0.257	0.257 r	clock_IBUF_inst/0
	net (fo=1, routed)	0.631	0.888 r	clock_IBUF
BUFGCTRL_X0Y16	BUFG (Prop_bufg_I_0)	0.026	0.914 r	clock_IBUF_BUFG_inst/0
	net (fo=343, routed)	0.548	1.461 r	u_ram_save_Q/clock_IBUF_BUFG
SLICE_X34Y77	FDRE		r	u_ram_save_Q/counter_reg[8]/C
SLICE_X34Y77	FDRE (Prop_fdre_C_Q)	0.164	1.625 r	u_ram_save_Q/counter_reg[8]/Q
	net (fo=16, routed)	0.471	2.096 r	u_ram_save_Q/u_bram/ADDRARDADDR[8]
RAMB36_X1Y15	RAMB36E1		r	u_ram_save_Q/u_bram/ram_reg_0_0/ADDRARDADDR[8]
E3	(clock clock rise edge)	0.000	0.000 r	
	net (fo=0)	0.000	0.000 r	clock (IN)
E3	IBUF (Prop_ibuf_I_0)	0.445	0.445 r	clock_IBUF_inst/0
	net (fo=1, routed)	0.685	1.129 r	clock_IBUF
BUFGCTRL_X0Y16	BUFG (Prop_bufg_I_0)	0.029	1.158 r	clock_IBUF_BUFG_inst/0
	net (fo=343, routed)	0.862	2.020 r	u_ram_save_Q/u_bram/clock_IBUF_BUFG
RAMB36_X1Y15	RAMB36E1		r	u_ram_save_Q/u_bram/ram_reg_0_0/CLKARDCLK
	clock pessimism	-0.250	1.771	
RAMB36_X1Y15	RAMB36E1 (Hold_ramb36el_CLKARDCLK_ADDRARDADDR[8])	0.183	1.954 r	u_ram_save_Q/u_bram/ram_reg_0_0
	required time		-1.954	
	arrival time		2.096	
	slack		0.142	

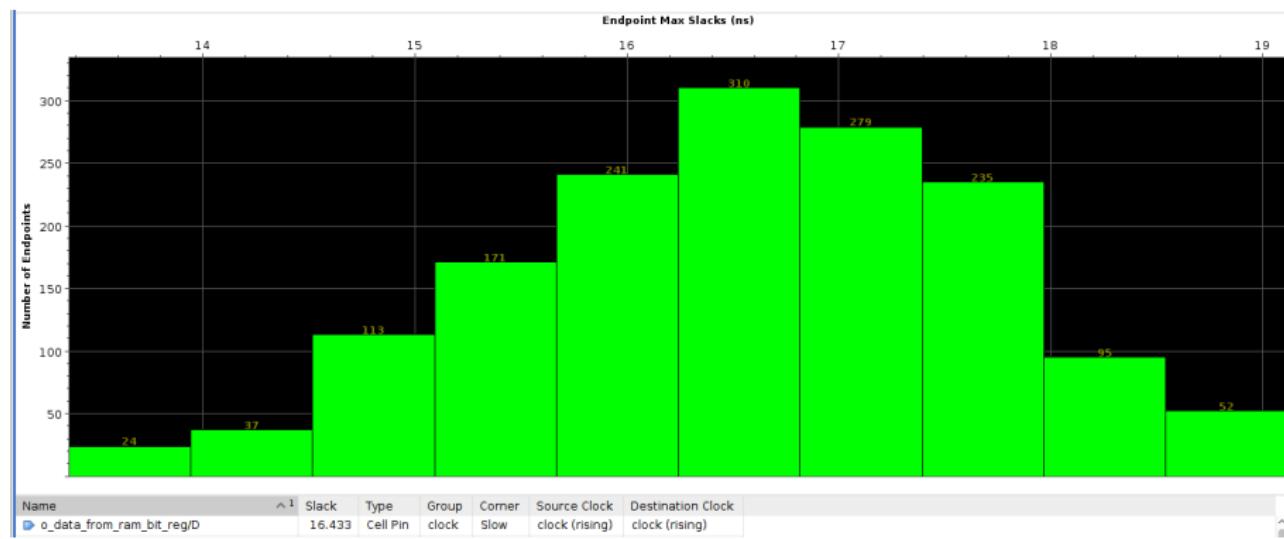
Reporte de timing con un clock de 50MHz.

Analizando los caminos críticos

- Otra forma de facilitar el análisis del timing es utilizando histogramas.
- Los histogramas de **timing** se pueden obtener una vez implementado el diseño.
- Para ejecutar el análisis el reporte de timing usando histogramas debemos ir a *Reports - Timing - Create Slack Histogram*.

Arquitectura Multi-Rate

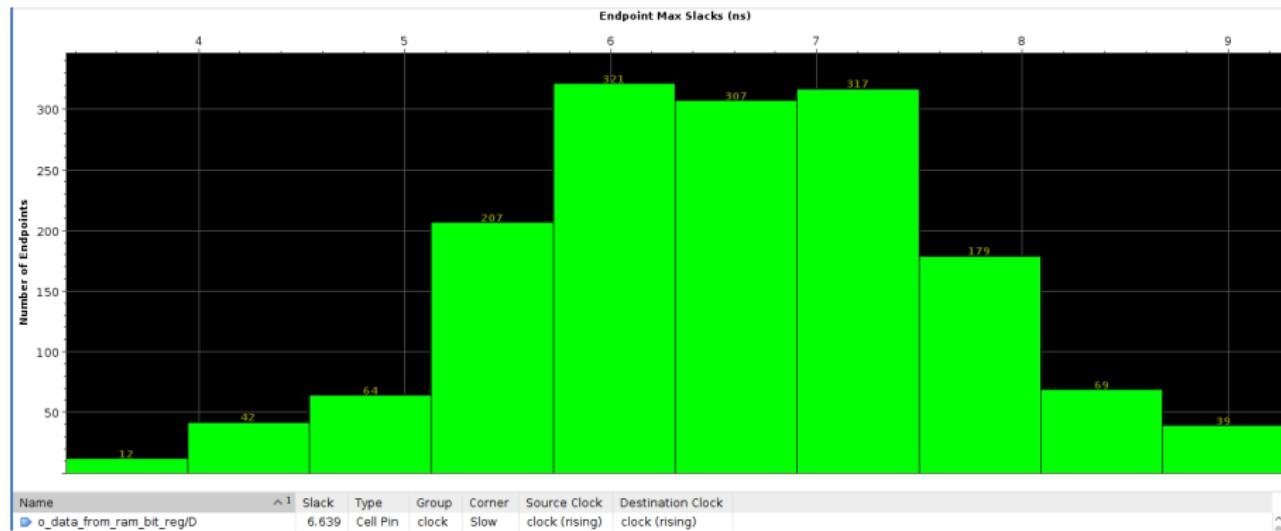
Implementación - Timing



```
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clock]
create_clock -period 20.000 -name clock -waveform {0 10} -add [get_ports clock]
```

Arquitectura Multi-Rate

Implementación - Timing

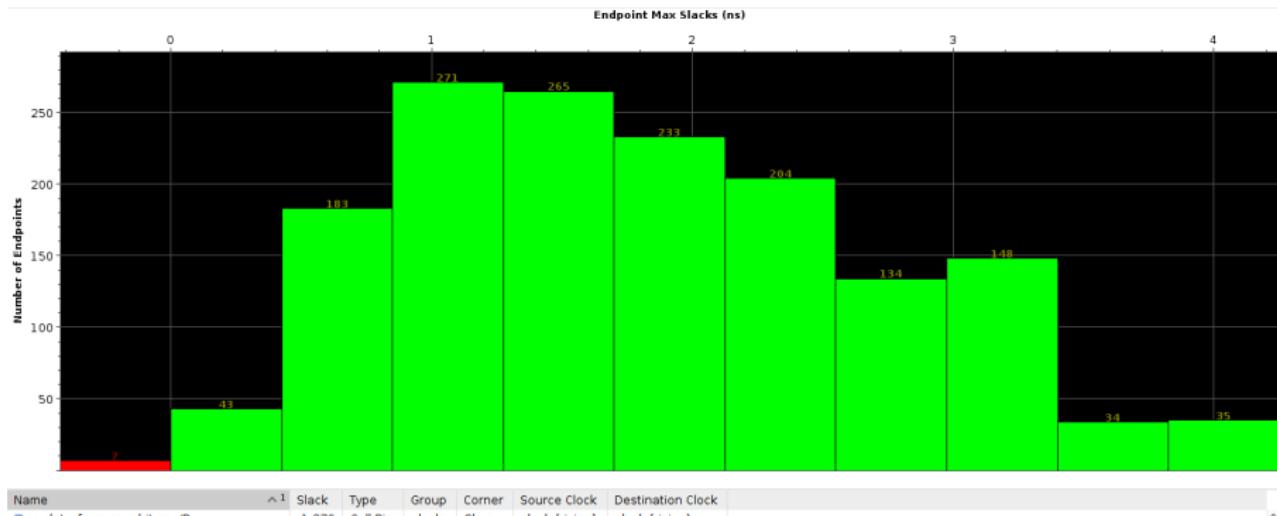


Histograma de Slacks con un clock de 100MHz.

```
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clock]
create_clock -period 10.000 -name clock -waveform {0 5} -add [get_ports clock]
```

Arquitectura Multi-Rate

Implementación - Timing



Histograma de Slacks con un clock de 200MHz.

```
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports clock]
create_clock -period 5.000 -name clock -waveform {0 2.5} -add [get_ports clock]
```

Trabajo de Laboratorio

Arquitectura Multi-Rate

Trabajo de Laboratorio

Análisis de Reportes

- 1 Realizar el diagrama de flujo de datos en base al diseño entregado sin considerar los módulos VIO e ILA.
- 2 Crear el proyecto es incluir los archivos Arty_Master_tx.xdc (constraint), tb_top_phy_all.v (testbench) y top_phy_all.v (source). Nota: el resto de los archivos se incluirán en forma automática por los includes utilizados en el código (revisar los path a los archivos).
- 3 Simular el diseño sin los módulos VIO e ILA (revisar sección de simulaciones).
- 4 Crear los módulos VIO e ILA, instanciarlos y sintetizar el sistema para una frecuencia de referencia de 100MHz (revisar sección de síntesis).
- 5 Determinar el número de celdas por jerarquía y los componentes (DSP, BRAM, LUT, etc).
- 6 Implementar en FPGA y obtener la forma de ondas desde el ILA.
- 7 Analizar los caminos críticos y obtener el histograma de timing para las frecuencias de reloj de 50MHz, 100MHz y 200MHz. Nota: Generar la implementación pero no implementarlo en la FPGA.