Cómo añadir la biblioteca de componentes oculta de Vivado

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/05/26/como-anadir-la-biblioteca-de-componentes-oculta-de-vivado/

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

GitHub: https://github.com/DRubioG

Fecha última modificación: 22/02/2025

Vivado tiene una biblioteca oculta que no enseña fácilmente pero existe. La biblioteca se llama «xpm»

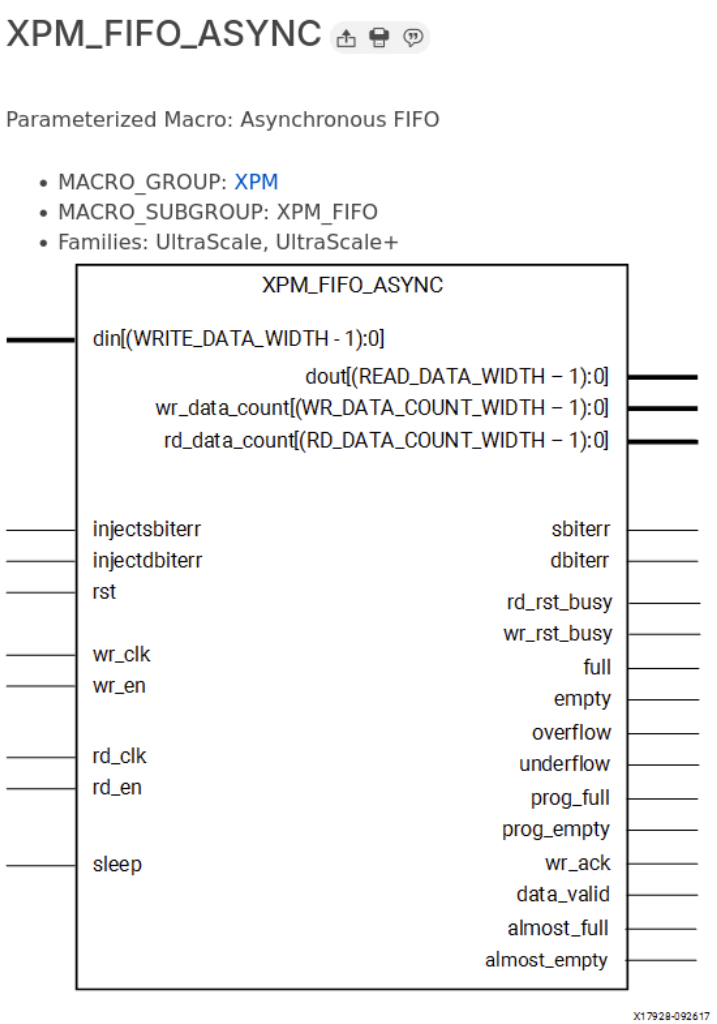
Esta biblioteca contiene componentes que permiten hacer CDC(**clock domain crossing**), FIFOs síncronos y/o asíncronos, además de añadir memorias internas, etc.

Para acceder a estas librerías solo tienes que saber cómo se llama el componente. Los nombres están en la página web:

- Serie 7000: https://docs.amd.com/r/en-US/ug953-vivado-7series-libraries/Xilinx-Parameterized-Macros
- UltraScale: https://docs.amd.com/r/en-US/ug974-vivado-ultrascale-libraries/Design-Elements

Y con el componente solo hay que declararlo.

Ejemplo: imaginemos que necesito un FIFO asíncrono, pues solo necesito saber cuál es el nombre del componente que le da Vivado y declararlo.



Para implementarlo solo se necesita declarar la librería

```
Library xpm;
use xpm.vcomponents.all;
```

Y después hacer la implementación, para la cuál en la página web describe cómo hacerlo.

```
-- xpm_fifo_async: Asynchronous FIFO
-- Xilinx Parameterized Macro, version 2023.2

xpm_fifo_async_inst : xpm_fifo_async
generic map (
   CASCADE_HEIGHT => 0,         -- DECIMAL
   CDC_SYNC_STAGES => 2,        -- DECIMAL
   DOUT_RESET_VALUE => "0",     -- String
   ECC_MODE => "no_ecc",        -- String
   FIFO_MEMORY_TYPE => "auto", -- String
   FIFO_READ_LATENCY => 1,      -- DECIMAL
   FIFO_WRITE_DEPTH => 2048,    -- DECIMAL
   FULL_RESET_VALUE => 0,       -- DECIMAL
   PROG_EMPTY_THRESH => 10,     -- DECIMAL
   PROG_FULL_THRESH => 10,      -- DECIMAL
   RD_DATA_COUNT_WIDTH => 1,    -- DECIMAL
   READ_DATA_WIDTH => 32,       -- DECIMAL
   READ_MODE => "std",          -- String
   RELATED_CLOCKS => 0,         -- DECIMAL
   SIM_ASSERT_CHK => 0,         -- DECIMAL; 0=disable simulation messages,
1=enable simulation messages
   USE_ADV_FEATURES => "0707", -- String
   WAKEUP_TIME => 0,            -- DECIMAL
   WRITE_DATA_WIDTH => 32,      -- DECIMAL
   WR_DATA_COUNT_WIDTH => 1     -- DECIMAL
)
port map (
   almost_empty => almost_empty,   -- 1-bit output: Almost Empty : When
asserted, this signal indicates that
                                   -- only one more read can be performed before
the FIFO goes to empty.

   almost_full => almost_full,     -- 1-bit output: Almost Full: When asserted,
this signal indicates that
                                   -- only one more write can be performed
before the FIFO is full.

   data_valid => data_valid,       -- 1-bit output: Read Data Valid: When
asserted, this signal indicates
                                   -- that valid data is available on the output
bus (dout).

   dbiterr => dbiterr,             -- 1-bit output: Double Bit Error: Indicates
that the ECC decoder
                                   -- detected a double-bit error and data in
the FIFO core is corrupted.

   dout => dout,                   -- READ_DATA_WIDTH-bit output: Read Data: The
output data bus is driven
                                   -- when reading the FIFO.
```

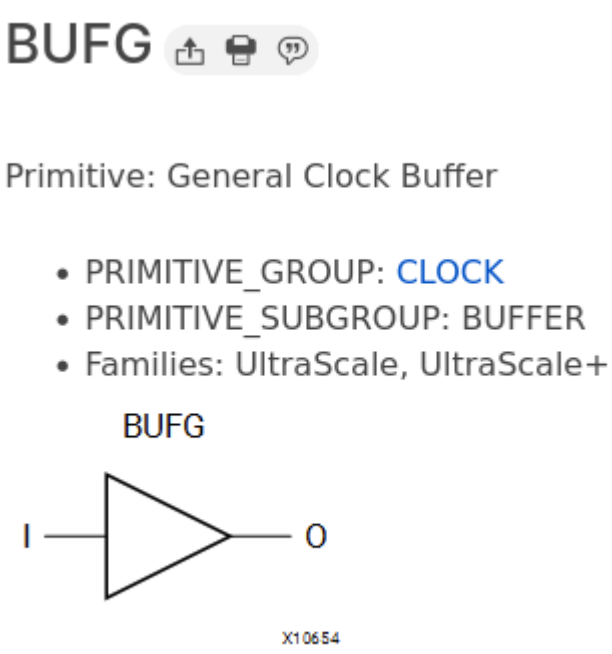
```
   empty => empty,                 -- 1-bit output: Empty Flag: When asserted,
this signal indicates that
                                   -- the FIFO is empty. Read requests are
ignored when the FIFO is empty,
                                   -- initiating a read while empty is not
destructive to the FIFO.

   full => full,                   -- 1-bit output: Full Flag: When asserted,
this signal indicates that the
                                   -- FIFO is full. Write requests are ignored
when the FIFO is full,
                                   -- initiating a write when the FIFO is full
is not destructive to the
                                   -- contents of the FIFO.

   overflow => overflow,           -- 1-bit output: Overflow: This signal
indicates that a write request
                                   -- (wren) during the prior clock cycle was
rejected, because the FIFO is
                                   -- full. Overflowing the FIFO is not
destructive to the contents of the
                                   -- FIFO.

   prog_empty => prog_empty,       -- 1-bit output: Programmable Empty: This
signal is asserted when the
                                   -- number of words in the FIFO is less than
or equal to the programmable
                                   -- empty threshold value. It is de-asserted
when the number of words in
                                   -- the FIFO exceeds the programmable empty
threshold value.

   prog_full => prog_full,         -- 1-bit output: Programmable Full: This
signal is asserted when the
                                   -- number of words in the FIFO is greater
than or equal to the
                                   -- programmable full threshold value. It is
de-asserted when the number
                                   -- of words in the FIFO is less than the
programmable full threshold
                                   -- value.

   rd_data_count => rd_data_count, -- RD_DATA_COUNT_WIDTH-bit output: Read Data
Count: This bus indicates
                                   -- the number of words read from the FIFO.

   rd_rst_busy => rd_rst_busy,     -- 1-bit output: Read Reset Busy: Active-High
indicator that the FIFO
                                   -- read domain is currently in a reset state.

   sbiterr => sbiterr,             -- 1-bit output: Single Bit Error: Indicates
that the ECC decoder
                                   -- detected and fixed a single-bit error.

   underflow => underflow,         -- 1-bit output: Underflow: Indicates that
the read request (rd_en)
                                   -- during the previous clock cycle was
rejected because the FIFO is
```

```
                                -- empty. Under flowing the FIFO is not
destructive to the FIFO.

   wr_ack => wr_ack,            -- 1-bit output: Write Acknowledge: This
signal indicates that a write
                                -- request (wr_en) during the prior clock
cycle is succeeded.

   wr_data_count => wr_data_count, -- WR_DATA_COUNT_WIDTH-bit output: Write Data
Count: This bus indicates
                                -- the number of words written into the FIFO.

   wr_rst_busy => wr_rst_busy,  -- 1-bit output: Write Reset Busy: Active-
High indicator that the FIFO
                                -- write domain is currently in a reset
state.

   din => din,                  -- WRITE_DATA_WIDTH-bit input: Write Data:
The input data bus used when
                                -- writing the FIFO.

   injectdbiterr => injectdbiterr, -- 1-bit input: Double Bit Error Injection:
Injects a double bit error if
                                -- the ECC feature is used on block RAMs or
UltraRAM macros.

   injectsbiterr => injectsbiterr, -- 1-bit input: Single Bit Error Injection:
Injects a single bit error if
                                -- the ECC feature is used on block RAMs or
UltraRAM macros.

   rd_clk => rd_clk,            -- 1-bit input: Read clock: Used for read
operation. rd_clk must be a
                                -- free running clock.

   rd_en => rd_en,              -- 1-bit input: Read Enable: If the FIFO is
not empty, asserting this
                                -- signal causes data (on dout) to be read
from the FIFO. Must be held
                                -- active-low when rd_rst_busy is active
high.

   rst => rst,                  -- 1-bit input: Reset: Must be synchronous to
wr_clk. The clock(s) can be
                                -- unstable at the time of applying reset,
but reset must be released
                                -- only after the clock(s) is/are stable.

   sleep => sleep,              -- 1-bit input: Dynamic power saving: If
sleep is High, the memory/fifo
                                -- block is in power saving mode.

   wr_clk => wr_clk,            -- 1-bit input: Write clock: Used for write
operation. wr_clk must be a
                                -- free running clock.

   wr_en => wr_en               -- 1-bit input: Write Enable: If the FIFO is
not full, asserting this
```

```
                                    -- signal causes data (on din) to be written
to the FIFO. Must be held
                                    -- active-low when rst or wr_rst_busy is
active high.
);
-- End of xpm_fifo_async_inst instantiation
```

Nota final

En esta misma página también viene descrito como utilizar la librería UNISIM, que permite utilizar lógica de la FPGA de forma directa (LUTs, BUFG, DSP48E2, RAM, etc). Como por ejemplo como meter un buffer para reloj en una FPGA.



Con un ejemplo de instanciación

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- BUFG: General Clock Buffer
--       UltraScale
-- Xilinx HDL Language Template, version 2023.2

BUFG_inst : BUFG
port map (
   O => O, -- 1-bit output: Clock output.
   I => I  -- 1-bit input: Clock input.
);

-- End of BUFG_inst instantiation
```