



**FACULTAD
DE INGENIERIA**
Universidad de Buenos Aires

[66.17/86.41] SISTEMAS DIGITALES

TRABAJO PRÁCTICO 2

2^{DO} CUATRIMESTRE 2019

Recepción de caracteres por UART y visualización VGA

AUTOR

Verstraeten, Federico. - #92892
<federico.verstraeten@gmail.com>

CÁTEDRA

Ing. Álvarez, Nicolás .

CURSO

Ing. Martos, Pedro.
Ing. Diograzia, Federico.
Ing. Alpago, Octavio.

FECHA DE ENTREGA

10 de diciembre de 2019

FECHA DE APROBACIÓN

CALIFICACIÓN

FIRMA DE APROBACIÓN

Índice

1. Objetivos	2
2. Diseño e implementación del programa	2
2.1. text_screen_top	2
2.2. text_screen_gen	2
2.3. uart	2
2.4. vga_sync	3
2.5. clock_manager_wrapper	3
3. Diagramas en bloque	4
3.1. Conexiones del hardware	4
3.2. Diseño implementado	4
4. Esquemático sintetizado simplificado	5
5. Recursos de hardware utilizados	6
6. Casos de prueba	7
7. Herramientas de hardware y software utilizadas	8
Referencias	9

1. Objetivos

El presente trabajo tiene los siguientes objetivos:

- Diseñar e implementar un procesador de caracteres ASCII que se ejecute sobre una FPGA.
- Repasar los conceptos de transmisión asincrónica de datos del protocolo UART.
- Utilizar el lenguaje *VHDL*, en cada una de las implementaciones necesarias para el desarrollo de los programas.
- Utilizar el programa *Vivado* para la síntesis de los circuitos digitales descriptos en el código desarrollado.
- Cargar el programa desarrollado en el kit FPGA provisto por la cátedra *Arty Z7-10*.
- Verificar el funcionamiento enviando caracteres por teclado e imprimiendo los mismos en un monitor con entrada VGA.

2. Diseño e implementación del programa

Se diseñaron los siguientes programas en VHDL que permiten imprimir caracteres ASCII por pantalla, ingresando los mismos desde el teclado de la computadora, transmitiéndolo con un adaptador USB-UART a la FPGA y luego a un monitor con entrada VGA.

2.1. `text_screen_top`

Programa principal o código top-level. Incluye cada uno de los diseños, combinando el circuito generador de texto (`text_screen_gen`), el circuito de sincronización (`vga_sync`), el receptor UART (`uart`) y el controlador de pulsos de reloj (`clock_manager_wrapper`).

2.2. `text_screen_gen`

Circuito generador de texto a partir de un mapa de caracteres o *Tile memory* (`font_rom`). Se define la lógica de movimiento del cursor en el proceso de escritura y los colores de la pantalla. Por defecto el fondo (background) es de color negro, los caracteres y el cursor de color verde.

2.3. `uart`

Implementación del circuito receptor de la UART. Recibe en serie los datos por la entrada `rx` y transmite en paralelo el código del caracteres ASCII correspondiente por la salida `r_data` al circuito `text_screen_gen`. Además activa el flag `rx_ready` por cada carácter enviado.

Valores por defecto: Baud rate = 19200, Clock rate = 50 MHz.

2.4. vga_sync

El circuito vga_sync genera las señales de sincronización hsync y vsync que están conectadas al puerto VGA para controlar los posiciones horizontal y vertical de los pixeles en la pantalla. Las dos señales se decodifican de los contadores internos, cuyas salidas son las señales pixel_x y pixel_y que indican la posiciones relativas de los pixeles en pantalla. El circuito vga_sync también genera la señal de video_on para habilitar o deshabilitar la transmisión de video.

Por defecto está definida una resolución VGA de 640x480 píxeles.

2.5. clock_manager_wrapper

Programa *wrapper* del divisor de frecuencia de reloj clk_wiz_0 mediante un IP-Block creado con el IP-Wizard de Vivado [8].

Frecuencia de entrada: 125 MHz (interna FPGA).

Frecuencia de salida: 5 MHz.

3. Diagramas en bloque

3.1. Conexiones del hardware

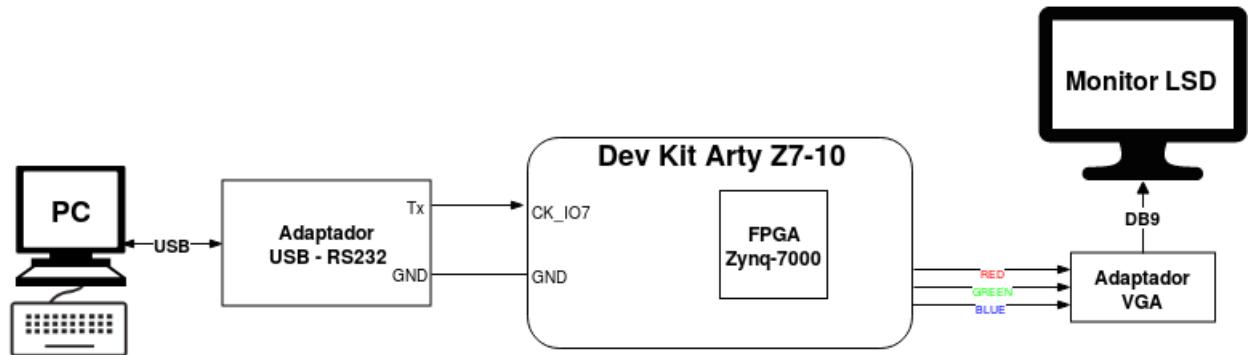


Figura 3.1: Diagrama de conexiones del hardware utilizado

3.2. Diseño implementado

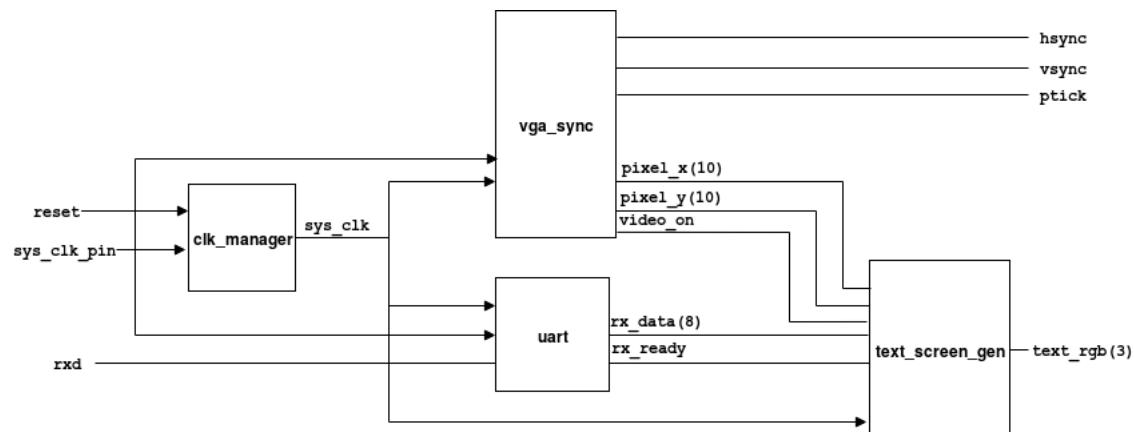
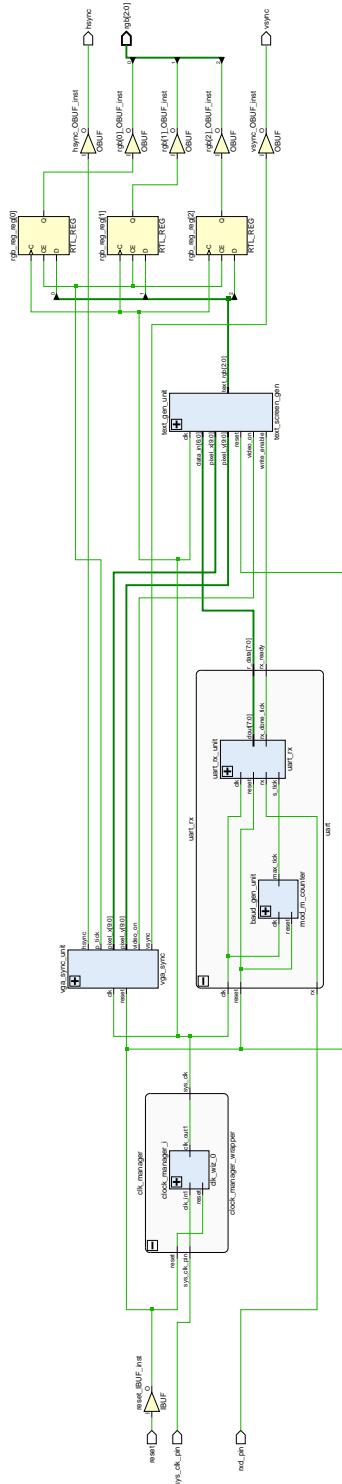


Figura 3.2: Diagrama en bloques del diseño del circuito

4. Esquemático sintetizado simplificado



5. Recursos de hardware utilizados

El cálculo de recursos utilizados se realiza mediante la síntesis de componentes con la suite de *Vivado*. Se seleccionó un *Baud Rate* de 19200 y un *Clock Rate* de 5 MHz generado mediante un IP block.

Lista de dispositivos utilizados			
Utilización Lógica	Usados	Disponible	% Utilización
Slice	30	4400	0.68
LUT Flip-Flop Pairs	89	17600	0.51
Register Flip-Flop	91	35200	0.26
LUT logic	64	17600	0.36
Bonded I/O	8	100	8.00
Block RAM Tile	1.50	60	2.50
RAMB36 FIFO	1	60	1.67
RAMB18	1	120	0.83
BUFG	2	32	6.25
MMCM	1	2	50.00

Tabla 1: Tabla resumen post implementación.

Tabla 2: Diagrama de conexiones del hardware utilizado

Lista de dispositivos utilizados			
Utilización Lógica	Usados	Disponible	% Utilización
Register FF	91	35200	0.26
LUT	64	17600	0.36
I/O	7	100	7.00
BRAM	1.50	60	2.50

Tabla 3: Tabla resumen post síntesis.

6. Casos de prueba

Para realizar los casos de prueba se sintetizaron los códigos VHDL desarrollados y se cargaron en la FPGA de la placa *Arty Z7-10*[4] por medio del programa *Vivado* [8]. Según las conexiones que se muestran en la figura 3.1 se ejecutaron pruebas de escritura, transmitiendo la información por la interfaz UART a través de un adaptador *USB-RS232*.

Por medio del programa *miniterm* [14] se enviaron caracteres ASCII desde la PC hacia la FPGA, utilizando un *Baud Rate* de 19200 como se muestra a continuación.

```
1 $ sudo miniterm.py /dev/ttyUSB0 19200
```

En la imagen 6.1 se muestra por pantalla una prueba donde introdujo desde la consola un fragmento del texto *Lorem Ipsum* [13].

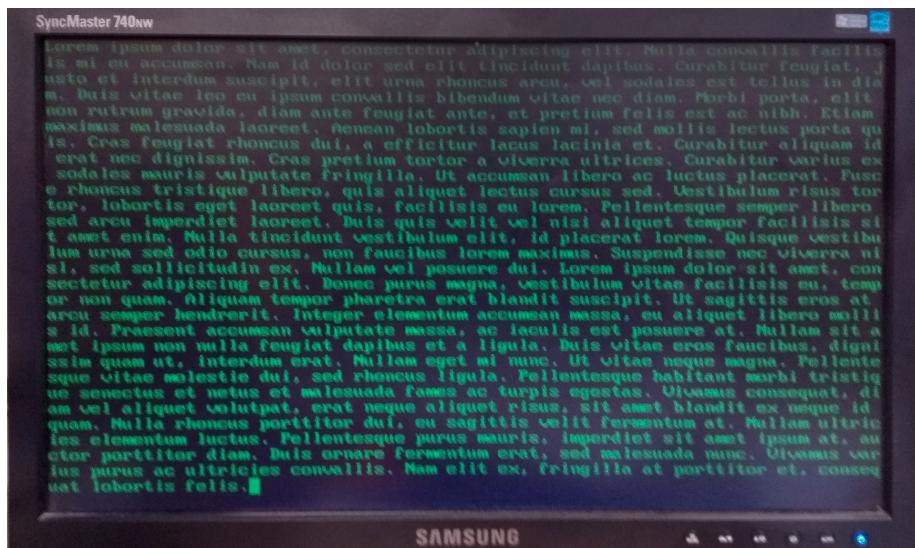


Figura 6.1: Muestra de texto por pantalla

7. Herramientas de hardware y software utilizadas

La computadora utilizada para realizar el desarrollo y las pruebas tiene las siguientes especificaciones:

- Procesador: Intel(R) Core(TM) i3-4005U CPU 64 bits @ 1.70GH.
- Memoria: 8GB RAM DDR4.
- Almacenamiento: Disco magnético de 500GB de 7200RPM.

Los programas se desarrollaron en el sistema operativo Linux Ubuntu, cuyos datos de distribución son

```
Distributor ID: Ubuntu  
Description: Ubuntu 14.04.2 LTS  
Release: 14.04  
Codename: Trusty Tahr
```

Elementos de *Hardware* utilizados:

- Kit FPGA: Arty Z7 development platform designed around the Zynq-7000(AP SoC) from Xilinx. Zynq-7000 architecture dual-core, 650 MHz ARM Cortex-A9 processor with Xilinx 7-series FPGA logic [4].
- Adaptador VGA: Standard VGA port for connecting commonly found displays, 12-bit RGB444 color depth, High-speed buffers support pixel clocks up to 150 MHz [5].
- Adaptador USB-RS232: CP2102 USB to TTL Serial Module Adapter (USB-TTL-RS232).

Herramientas de *Software* utilizadas:

- Compilador VHDL: GHDL 0.33 (20150921) [Dunoon edition]. Compiled with GNAT Version: 4.8, llvm code generator, Written by Tristan Gingold [6].
- Analizador de ondas: GTKWave Analyzer v3.3.58 (w)1999-2014 BSI [7].
- Síntesis de hardware: Vivado v2015.1 (64-bit) - SW Build 1215546 - IP Build 1209967 - Copyright 1986-2015 Xilinx, Inc [8].
- Control del proceso de compilación: GNU Make 4.1 [9].
- Compilador del presente informe: Latex pdfTeX 3.14159265-2.6-1.40.16 (TeX Live 2015/Debian) [10].
- Edición de código fuente: VIM - Vi IMproved 7.4 (2013 Aug 10, compiled Nov 24 2016 16:44:48) y Sublime 3.2 [11] [12].

Referencias

- [1] Goldberg, David - *Computer Arithmetic - Appendix H* - Xerox Palo Alto Research Center - Elsevier Science USA - 2003.
- [2] Hennessy, J.L. & Patterson D.A. - *Computer Architecture a Quantitative Approach* - Morgan Kaufmann (ISBN 0123704901) - 2006.
- [3] Zaianalabedin, Navibi - *Analysis and Modeling of Digital Systems* - McGraw-Hill Professional - 1997.
- [4] Arty Z7 - <https://reference.digilentinc.com/reference/programmable-logic/arty-z7/reference-manual>
- [5] Pmod VGA - <https://reference.digilentinc.com/reference/pmod/pmodvga/reference-manual>
- [6] *GNU GHDL* - <http://ghdl.free.fr/>
- [7] *GTKWave Analyzer* - <http://gtkwave.sourceforge.net/>
- [8] *Vivado* - <https://www.xilinx.com/products/design-tools/vivado.html>
- [9] *GNU Make* - <https://www.gnu.org/software/make/>
- [10] *LATEX*- www.latex-project.org/
- [11] *VIM* - vim.sourceforge.io/
- [12] *Sublime Text* - www.sublimetext.com/
- [13] *Lorem Ipsum* - en.wikipedia.org/wiki/Lorem_ipsum
- [14] *Miniterm* - pyserial.readthedocs.io/en/latest/