Sequencer

User Guide

October 20, 2019

Contents

List of Tables

List of Figures

1 Introduction

A sequencer is a device that can produce rythmic loops programmed by the user. The loop is devided into 8 equally spaced steps and each step can be activated by the user. The sequancer will go trough the loop and will play a sound on the activated steps. The loop period and the sound frequency can be set by the user.

The sequencer hardware is implemented in Verilog and uses the Picoversat SoC as the basic processing unit. Refer to the Picoversat manual for more information. This sequencer implementation is meant to be used on the Basys2 FPGA, and in order to make that possible, custom-made peripherals are used in order to use the board's features (e.g.LEDs, Switched, etc...). For more information is present on the peripherals chpater (insert ref).

2 Sequencer operation

Since the sequencer depends on multiple time-dependant routines and there is no trivial way to deal with this on the picoversat controller (because of the lack of interrupts), the main logic needed to be divided into two routines: one for the main sequencer loop, implemented as a standalone peripheral, and other for the reading and debouncing of the switches and pushbuttons and for data handling (frequencies).

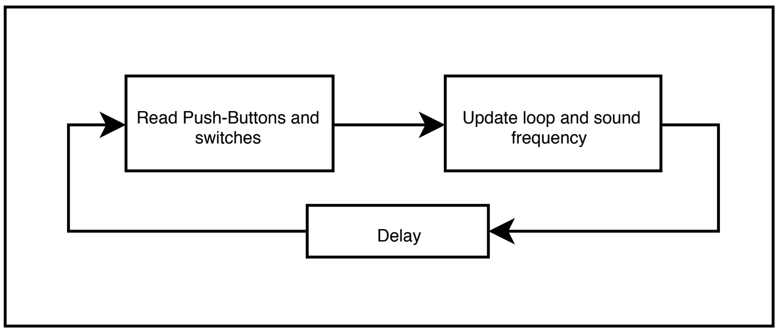
Read Push-Buttons and switches
Update loop and sound frequency
Delay

Figure 1: Picoversat code main loop.

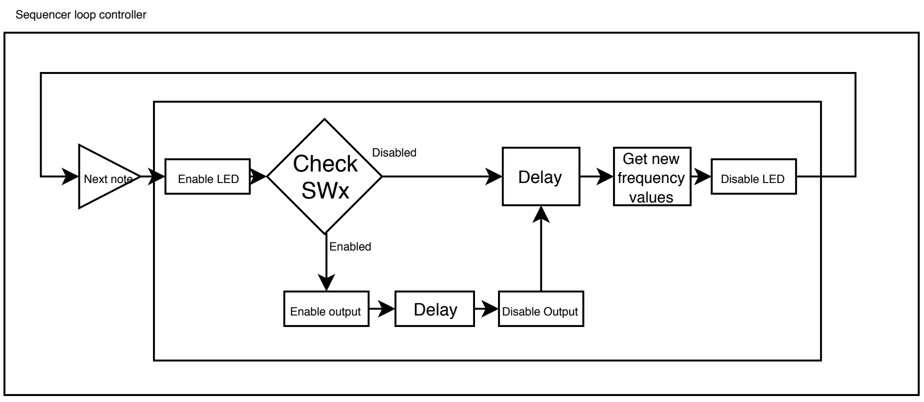

Figure 2: Sequencer Loop controler peripheral main loop.

3 Block Diagram

The hardware platform used for the developed sequencer is the Basys 2 FPGA board by digilent. The sequencer will interface with some of the board peripherals as shown in the Figure **??**

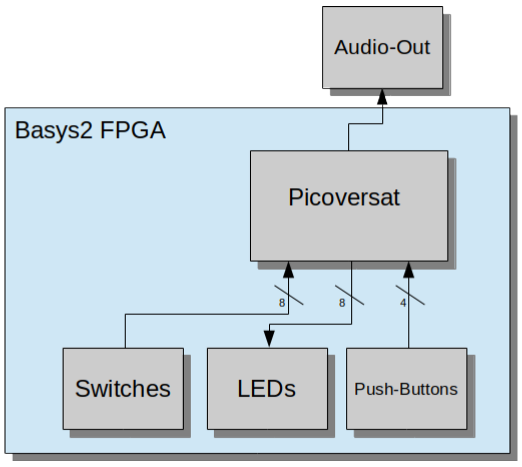

Figure 3: Block Diagram

Figure x shows the Basys 2 peripherals that the user can use in order to interact with the sequencer

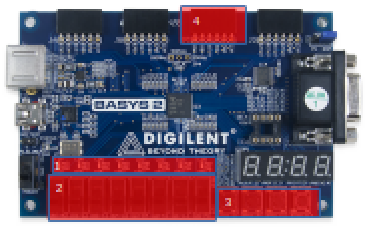

Figure 4: Basys 2 board peripherals

attached to the data bus is shown in Figure 5.

4 Interface Signals

The interface signals of the Versat controller core are described in Table 1.

4.1 Instruction Bus Timing Diagram

The timing diagram for an instruction read transaction is shown in Figure 6.

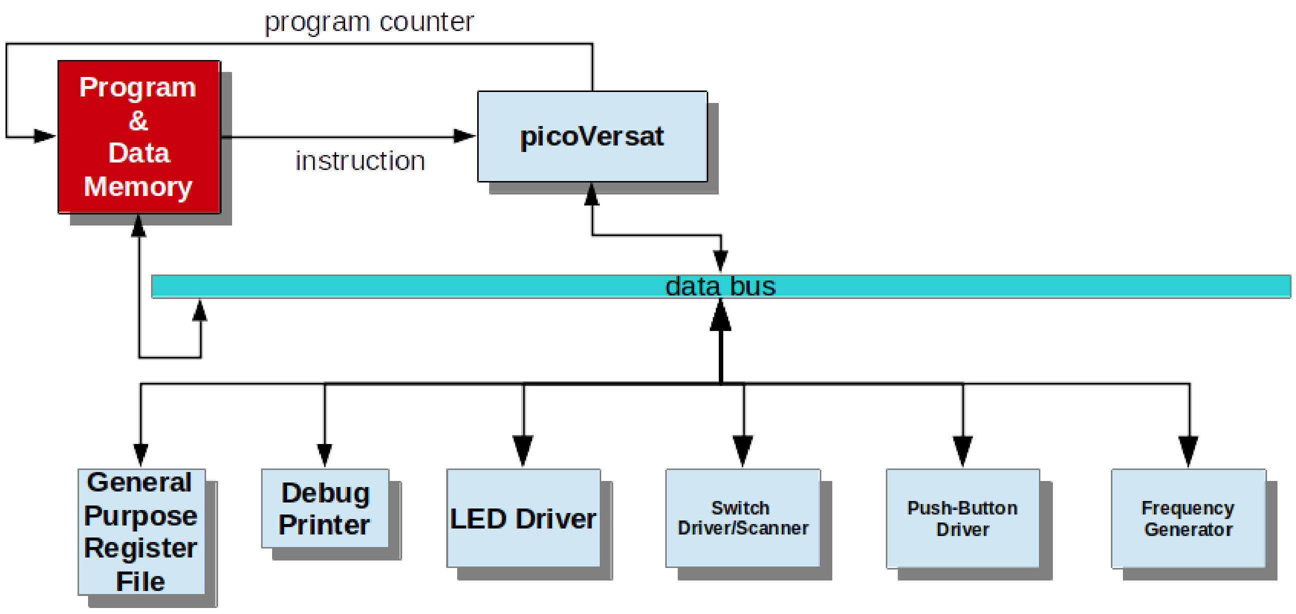

Figure 5: PicoVersat SoC with two peripherals

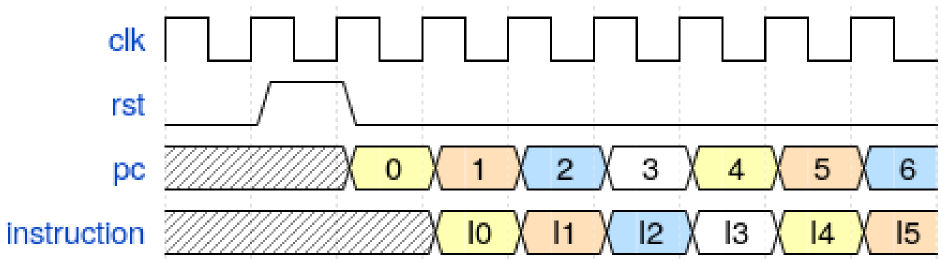

Figure 6: Instruction (pipelined) reads.

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
Instruction Bus Interface		
instruction[31:0]	IN	Instruction to execute.
pc[9:0]	OUT	Program Counter (instruction address).
Data Bus Interface		
data_sel	OUT	Read or write request.
data_we	OUT	Write enable.
data_addr[9:0]	OUT	Data address.
data_to_rd[31:0]	IN	Data to be read.
data_to_wr[31:0]	OUT	Data to be written.

Table 1: Interface signals.

4.2 Data Bus Timing Diagram

The timing diagrams for data reads and writes are shown in Figure 7 and Figure 8, respectively. These operations may be consecutive or not, as illustrated.

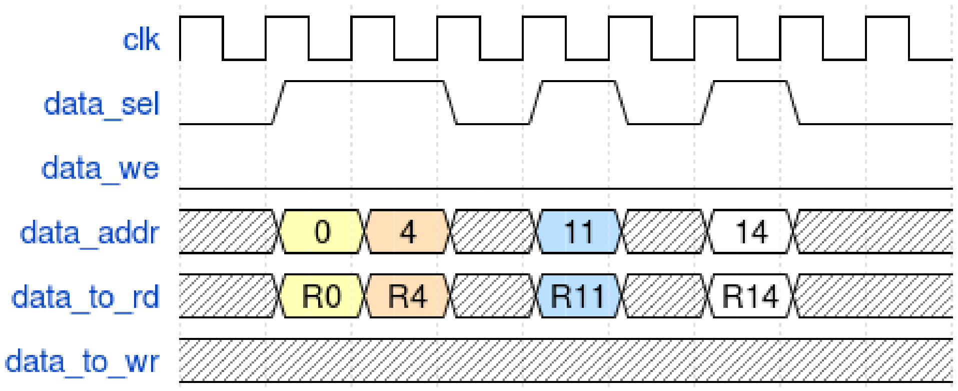

Figure 7: Data Bus reads.

5 Peripherals

Refer to the memory map in section 6 to check the base addresses of the peripherals.

5.1 General Purpose Register File

This peripheral contains a 16x32bit register file that can be used by user programs.

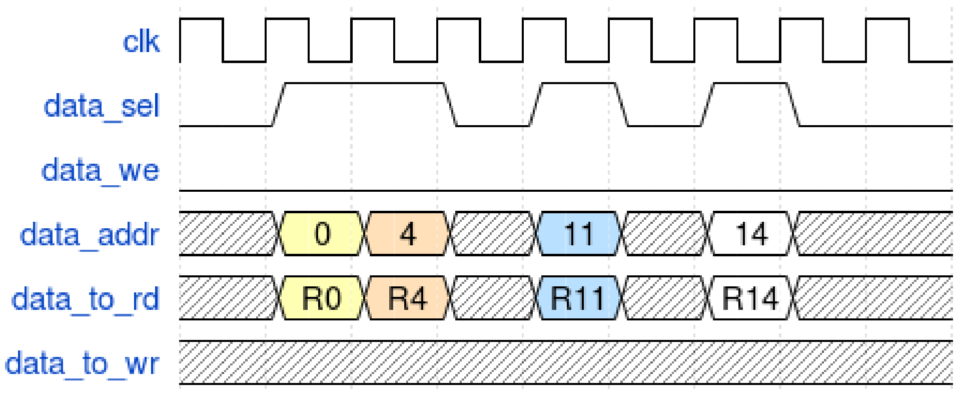

Figure 8: Data Bus writes.

5.2 Debug Printer

This peripheral can be used by user programs to print characters, mainly for debug purposes.

5.3 LED Driver

Led driver.

5.4 Switch Driver/Scanner

Switch Driver.

5.5 Push-Button Driver

Push-Button Driver.

5.6 Frequency Generator

Frequency Generator.

6 Memory Map

The memory map of the system, as seen by picoVersat programs, is given in Table 2.

Mnemonic	**Address**	**Read/Write**	**Read Latency**	**Description**
REGF_BASE	0	Read+Write	0	Register file peripheral
CPRT_BASE	1	Write only	NA	Debug printer periheral
PROG_BASE	3	Read+Write	1	User programs and data

Table 2: Memory map base addresses

7 Implementation Results

8 Conclusions