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).

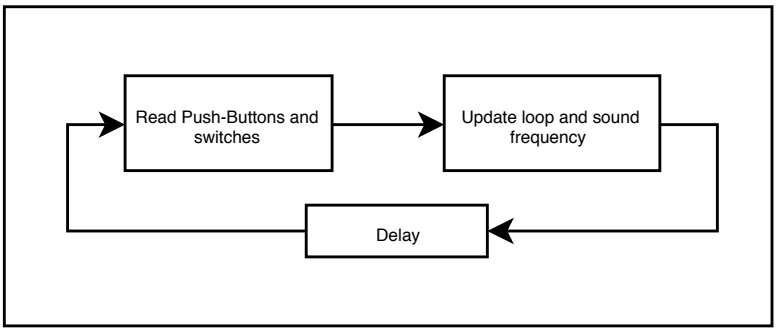

Figure 1: Picoversat code main loop.

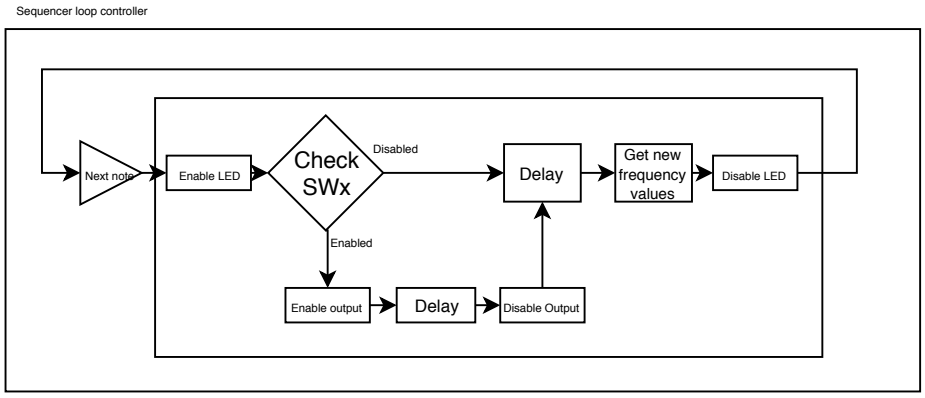

Figure 2: Sequencer Loop controler peripheral main loop.

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 **??**

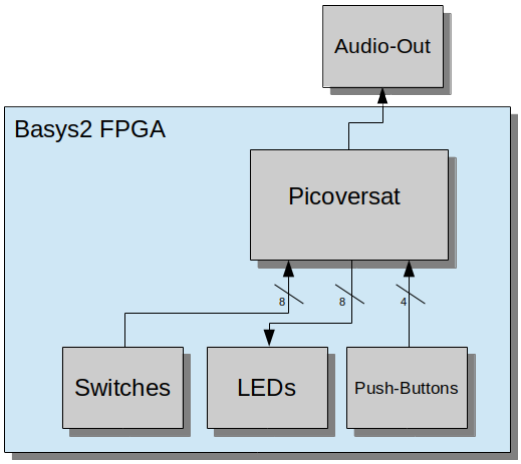

Figure 3: Block Diagram

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

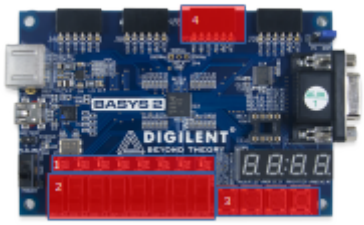

Figure 4: Basys 2 board peripherals

1. 8xLED
2. 8xSlide-Switches
3. 4xPush-Button
4. 1xAudio Output

4 Peripherals

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

4.1 General Purpose Register File

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

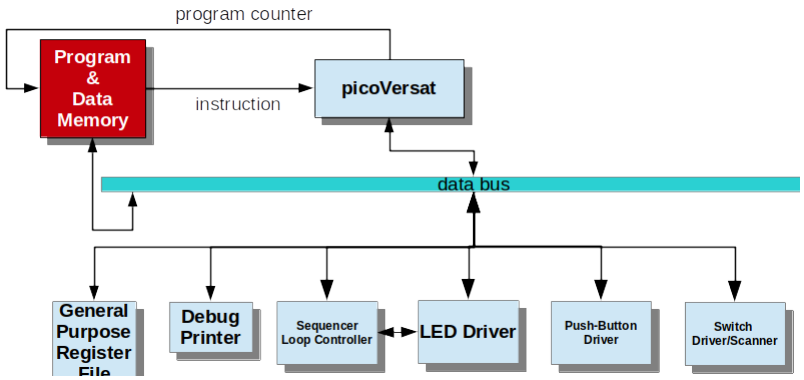

Figure 5: PicoVersat SoC with two peripherals

4.2 Debug Printer

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

4.3 Sequencer Loop Controller

This peripheral is used to generate the sequencer loop. The loop and note frequency can be set by using the *freq* input and by selecting the according selector signal. The Sequencer loop controller will output a square wave corresponding to the loop ouput. The led outputs are directly connected to the LED driver peripheral and send information about the current note.

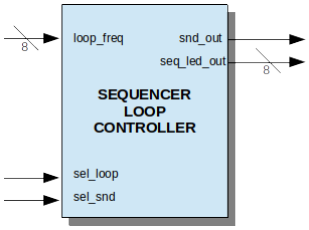

Figure 6: PicoVersat SoC with two peripherals

Table 1: Sequencer Loop Controller Inputs

Name	#bits	Description
freq	8	Loop Period / Note Frequency
sel_loop	1	Loop period select signal (address decoder)
sel_snd	1	Note frequency select signal (address decoder)

Table 2: Sequencer Loop Controller Outputs

Name	#bits	Description
snd_out	1	Audio Output
seq_led_out	8	Current note led output

4.4 LED Driver

The LED driver will display the current note.

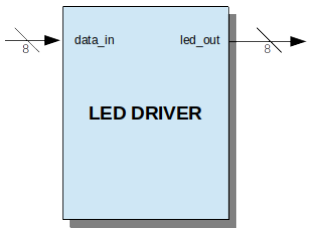

Figure 7: PicoVersat SoC with two peripherals

Table 3: Led driver inputs

Name	#bits	Description
data_in	8	Led display information

Table 4: Led driver outputs

Name	#bits	Description
led_out	8	Led display output

4.5 Switch Driver

Information about the driver

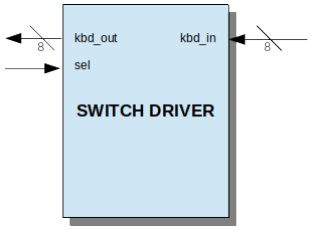

Figure 8: PicoVersat SoC with two peripherals

Table 5: Switch driver inputs

Name	#bits	Description
sw_in	8	Switch state inputs
sel	1	driver selector (address decoder)

Table 6: Switch driver outputs

Name	#bits	Description
kbd_out	8	Switch state outputs

4.6 Push-Button Driver

Information about the driver

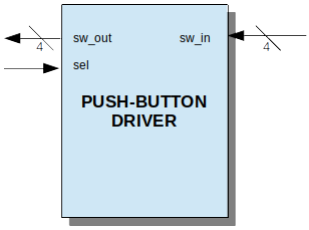

Figure 9: PicoVersat SoC with two peripherals

Table 7: Push-button driver inputs

Name	#bits	Description
sw_in	4	Push-button state inputs
sel	1	Driver selector (address decoder)

Table 8: Push-button driver outputs

Name	#bits	Description
sw_out	4	Switch state outputs

5 Interface Signals

The interface signals of the Versat controller core are described in Table 9..

6 Memory Map

The memory map of the system, as seen by picoVersat programs, is given in Table 10.

7 Implementation Results

8 Conclusions

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 9: Interface signals.

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 10: Memory map base addresses