Sequencer

User Guide

January 29, 2020

Contents

List of Tables

©2017 IObundle Lda All rights reserved www.iobundle.com **Confidential**

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 in order to produce a sound. The sequencer 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 Basys 2 FPGA by Digilent, for that, custom-made peripherals are developed in order to use the board's features (e.g.LEDs, Switches, etc...).

2 Sequencer operation

In normal operation, the sequencer will loop through all the steps in a cyclic fashion, the current step can be seen by the indicated LED. In order to create a rythm, the user can activate the desired steps by activating the correspondant switch. The activated steps are indicated by their correspondant LEDs being on.

The sound output of the signal is a DC biased square wave outputted by an IO pin of the Basys 2 Board, this should be connected to and external powered speaker in order to the sound to be apmplified and filtered.

Figure 1 shows the Basys 2 peripherals that the user can use in order to interact with the sequencer.

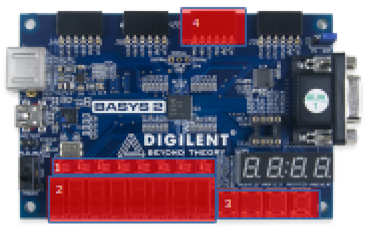

Figure 1: Basys 2 board peripherals

3 Implementation

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 is divided into two routines: one for the main sequencer loop, implemented as a standalone peripheral, the "Sequencer loop controller", and other for the reading and debouncing of the pushbuttons and for keeping track of the frequency and loop values.

All the peripherals and modules are interconnected as described in the following picture:

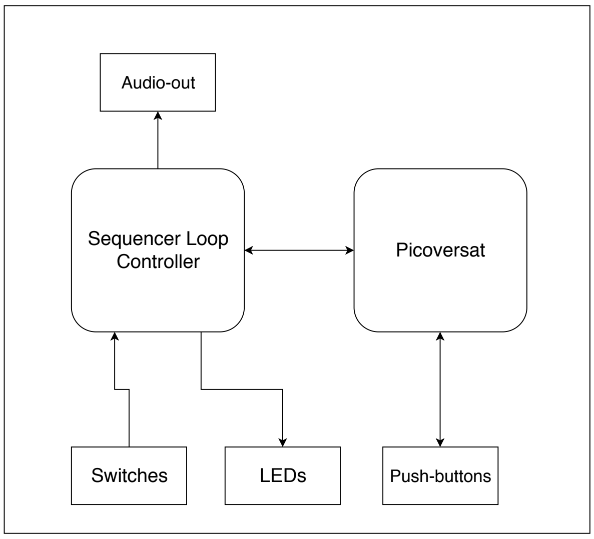

Figure 2: Sequencer modules

3.1 Sequencer Loop Controller

The sequencer loop controller 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 and selected note.

The sequencer loop controller operation is described by the fluxogram on Picture 7.

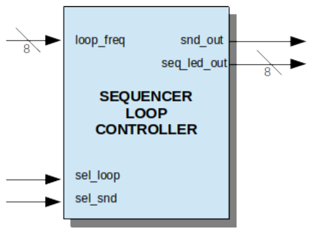

Figure 3: Sequencer Loop Controller module

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

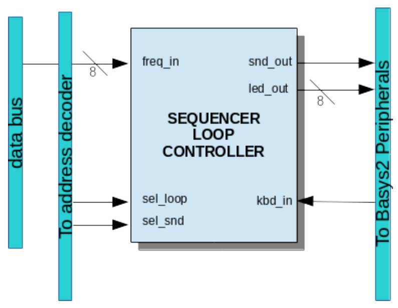

Figure 4: Sequencer Loop Controller bus connections

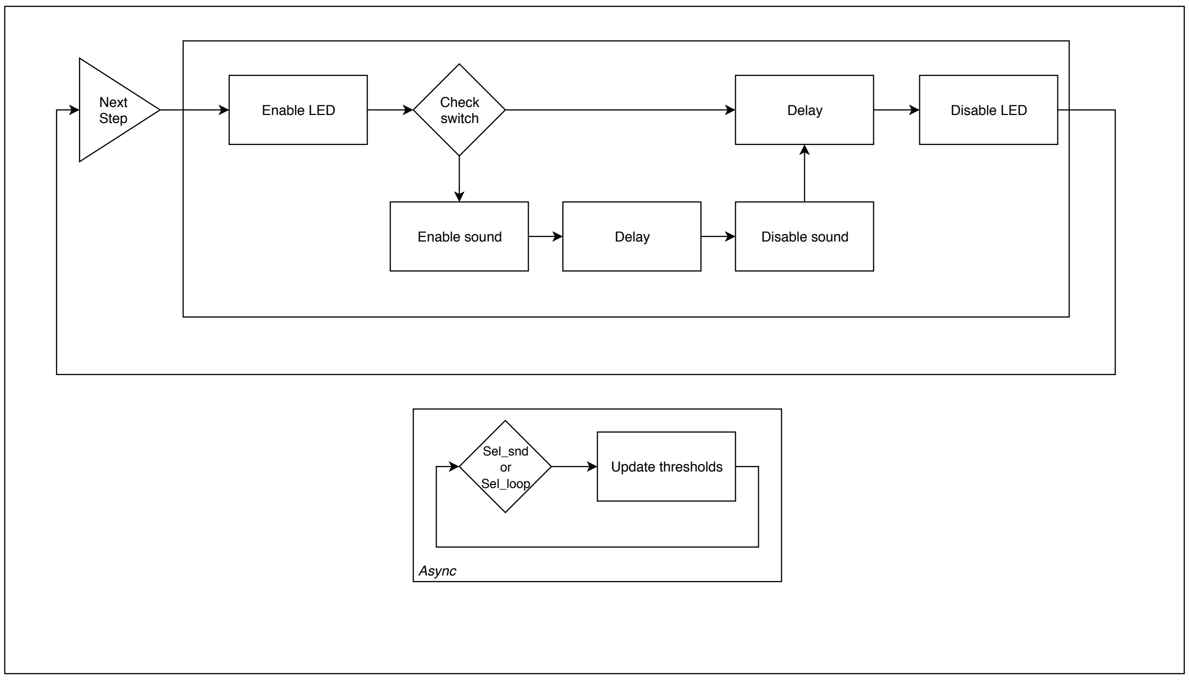

Figure 5: Sequencer Loop Controler fluxogram

3.2 Picoversat

For reading the pushbuttons, debouncing and managing loop and sound frequencies, the picoversat SoC was used. The Sequencer Loop Controler is connected to the picoversat data bus and the loop and sound frequency registers are mapped in memory. For this, the picoversat address decoder was altered so that the sequencer loop controller's registers can be accessed. This was also done for the push-buttons.

Some auxiliary modules were also added to the picoversat:

3.2.1 General Purpose Register File

This peripheral contains a 16x32bit register file that can be used by user programs. Refer to the picoversat manual for more information about this peripheral.

3.2.2 Debug Printer

This peripheral can be used by user programs to print characters, mainly for debug purposes. Refer to the picoversat manual for more information about this peripheral.

All of the aforementioned peripherals connect to the picoversat as described below:

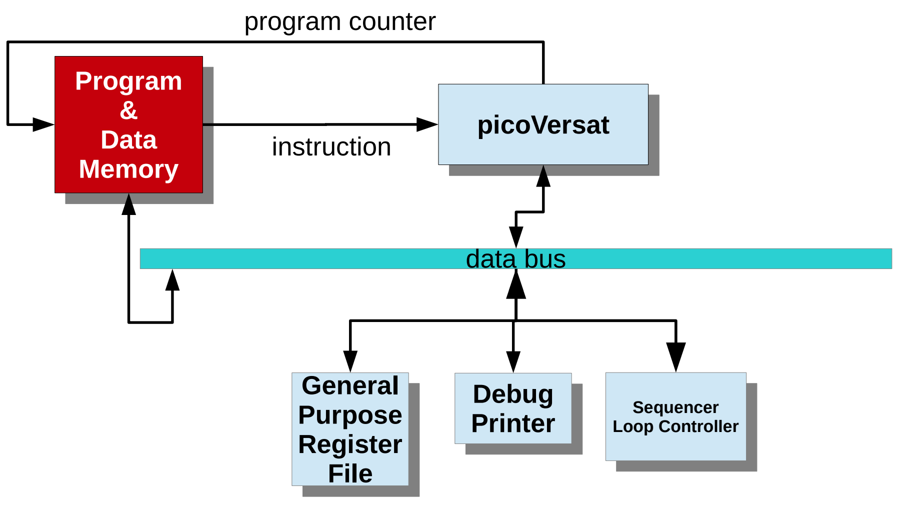

Figure 6: PicoVersat SoC with the peripherals

The final memory map is represented on following table:

Table 3: Memmory map

Mnemonic	Address	Read/Write	Read/Write Latency	Description
SND_BASE	0x25E	Write only	0	Sound frequency address for sequencer loop controller
LOOP_BASE	0x25C	Write only	0	Loop frequency address for sequencer loop controller
PUSH_BASE	0x25A	Read only	0	Push-button peripheral
CPRT_BASE	0x258	Read only	N/A	Debug printer peripheral
REGF_BASE	0x200	Read+Write	0	Register file peripheral
PROG_BASE	0x0	Read+Write	1	User program and data

3.2.3 Software

The picoversat was firstly developed in C and cross-compiled into picoversat assembly using the lcc compiler. Unfortunately due to some setbacks with the lcc code file size and picoversat's memory size, the code was implemented again from scratch in assembly. This resulted in a smaller and better optimized program. The general program logic is represented below:

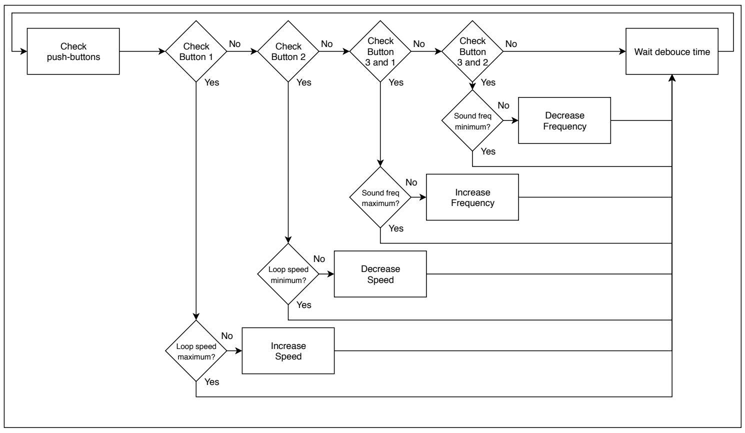

Figure 7: Picoversat code fluxogram.

4 Conclusions

The developed system works as expected, everything defined in the beginning of the project was achieved. There were a few challenges while developing the system, nonetheless they were overcomed. The biggest setback was the fact that, initially the system's code was develop in C and compiled with lcc for the picoversat. However, in the end the compiled code was bigger then the available memory. After several attempts to widen the memory available without success, the code was re-written using Assembly. In the end it was possible to use the sequencer to produce a rhythmic pattern with varying speeds and frequencies. The final implementation was also converted into a standalone rom file that was flashed into the boards persistent memory in order for the bitstream to be loaded into the FPGA on boot.

Link to a short demo