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 Block Diagram

The picoVersat block diagram is shown in Fig. 1. PicoVersat contains 4 main registers: the accumulator (register A), the data pointer (register B), the flags register (register C) and the program counter (register PC).

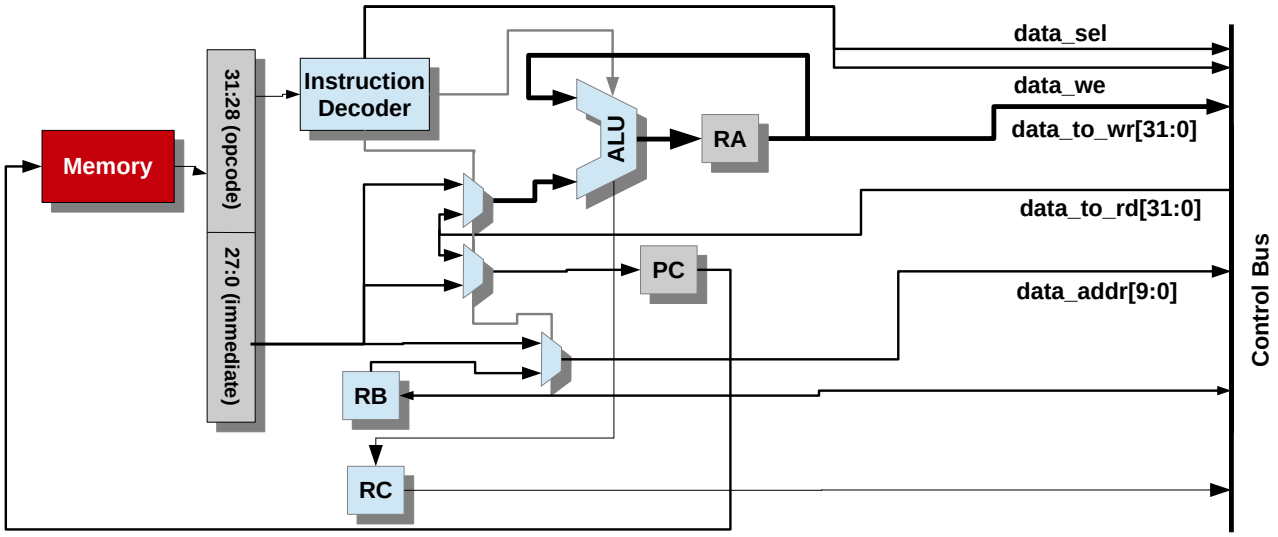

Figure 1: Block Diagram

2.1 Accumulator register

Register A, the accumulator, is the main register in this architecture. It can be loaded with an immediate value from the instruction itself (immediate value) or with a value read from the data interface. It is the destination of operations using as operands register A itself and an immediate or addressed value. Its value is always driven out to the data interface.

2.2 Pointer register

Register B, the memory pointer, is used to store the address in indirect loads and stores to/from the accumulator, respectively, and to store the target address in branch instructions. Register B itself is in the memory map so it can be read or written as if accessing the data interface.

2.3 Flags register

Register C, the flags register, is used to store three operation flags: the negative, overflow and carry flags. Register C itself is in the memory map and it is read-only. The flags are set by the controller ALU and can be read by programs for decision taking. The structure of register C is shown in Table 1.

Bits	Name	Description
31-3	NA	Reserved for future use
2	Negative	Asserted if last ALU operation generated a negative result
1	Overflow	Asserted if last ALU operation generated an arithmetic overlow
0	Carry	Asserted if last ALU operation generated a carry

Table 1: Register C: flags

2.4 PC register

The Program Counter (PC) register contains the address of the next instruction to be fetched from the Memory. The PC normally increments to fetch the next instruction, except for program branch instructions, in which case the PC register is loaded with the instruction immediate or with the value in register B, depending on the branch instruction type, direct or indirect, respectively.

3 Interface Signals

The interface signals of the Versat controller core are described in Table 2.

3.1 Instruction Bus Timing Diagram

The timing diagram for an instruction read transaction is shown in Figure 2.

3.2 Data Bus Timing Diagram

The timing diagrams for data reads and writes are shown in Figure 3 and Figure 4, respectively. These operations may be consecutive or not, as illustrated.

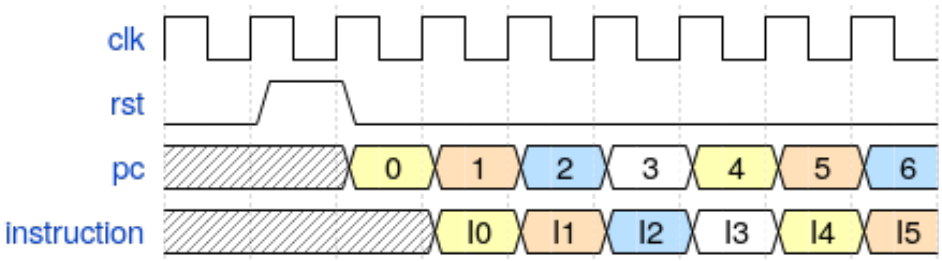

Figure 2: Instruction (pipelined) reads.

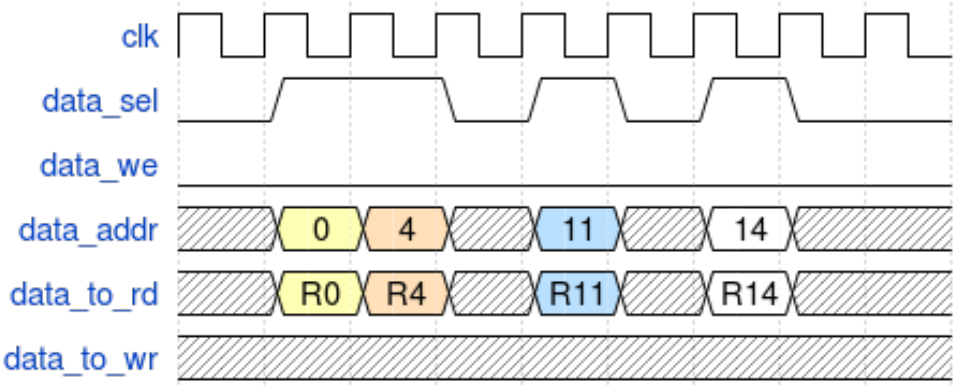

Figure 3: Data Bus reads.

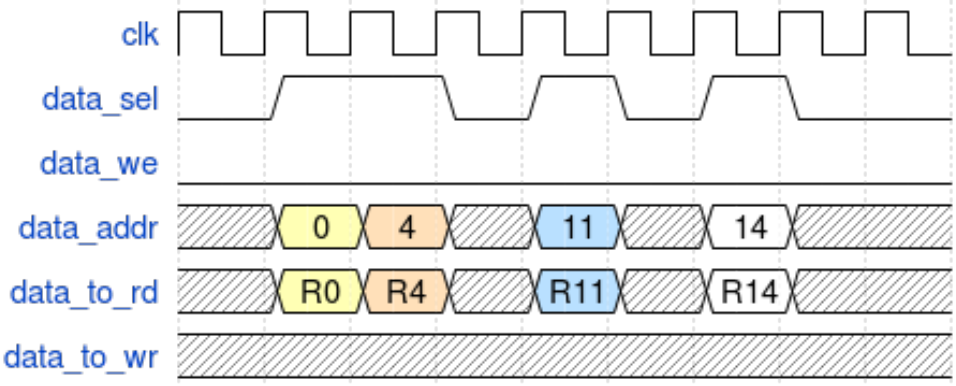

Figure 4: Data Bus writes.

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 2: Interface signals.

4 Peripherals

attached to the data bus is shown in Figure 5.

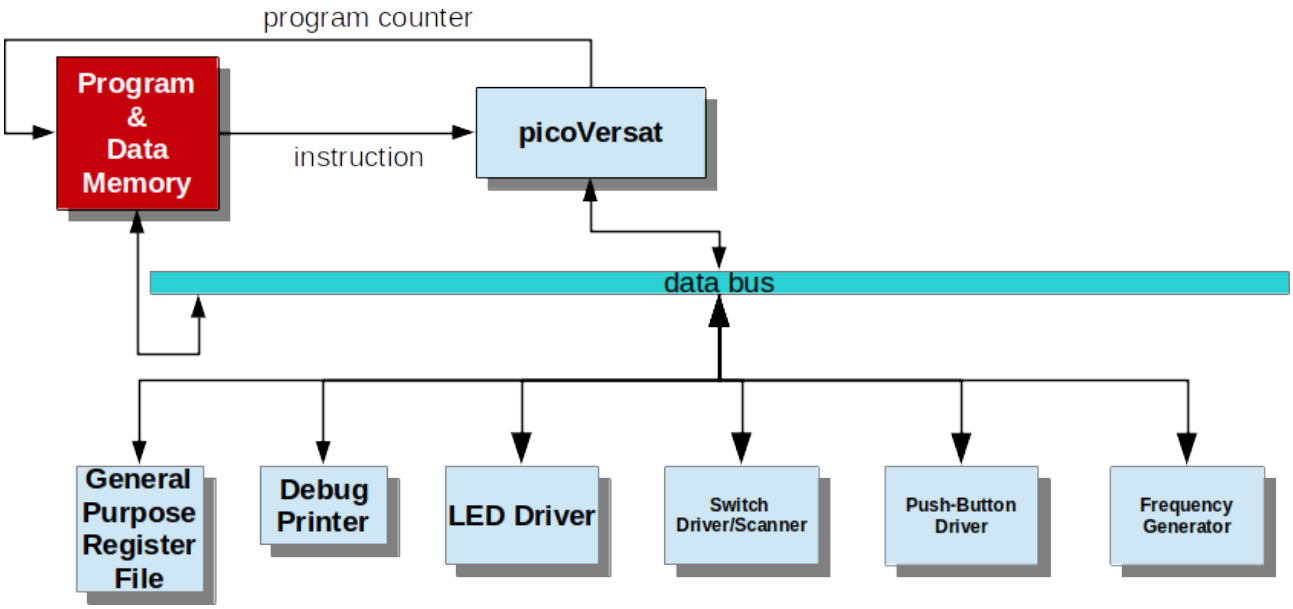

Figure 5: PicoVersat SoC with two peripherals

Refer to the memory map in section 5 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.

4.2 Debug Printer

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

4.3 LED Driver

Led driver.

4.4 Switch Driver/Scanner

Switch Driver.

4.5 Push-Button Driver

Push-Button Driver.

4.6 Frequency Generator

Frequency Generator.

5 Memory Map

The memory map of the system, as seen by picoVersat programs, is given in Table 3.

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 3: Memory map base addresses

6 Implementation Results

7 Conclusions