picoVersat

User Guide

October 17, 2018

Contents

List of Tables

List of Figures

1 Introduction

PicoVersat is a minimal hardware controller. It is designed for simple control tasks where normally finite state machines are employed. Being a programmable solution, the risk of hardware design errors is mitigated and the design of complex control structures is simplified.

PicoVersat features a minimal instruction set and is not designed for high performance computing. However, it can effectively implement simple algorithms and perform simple calculations. PicoVersat has a program memory interface and a memory mapped data interface, only.

2 Block Diagram

The picoVersat block diagram is shown in Fig. 1.

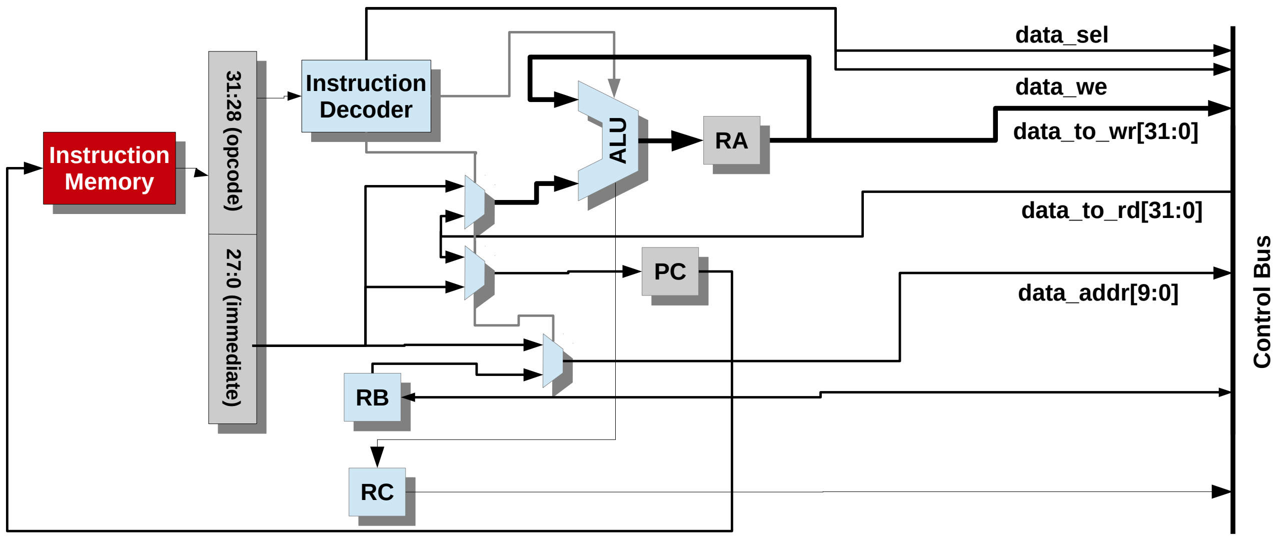

Figure 1: Block Diagram

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).

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 driven out to the data interface.

2.2 Pointer register

Register B, the memory pointer, is used to implement indirect loads and stores to/from the accumulator, respectively. Its contents is the load/store address for the data interface. 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	Negative	Asserted if last ALU operation generated a negative result
30	Overflow	Asserted if last ALU operation generated an arithmetic overlow
29-1	NA	Reserved for future use
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 Program 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.

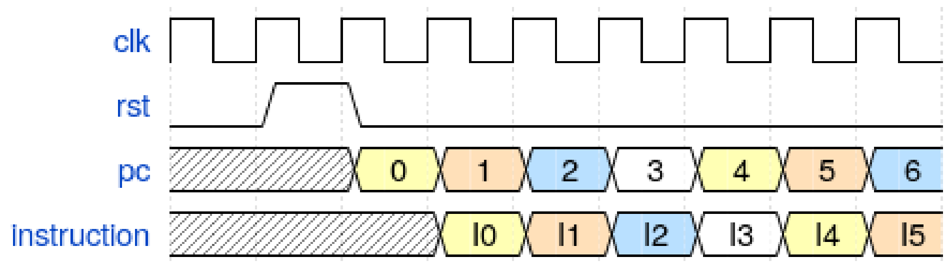

Figure 2: Instruction (pipelined) reads.

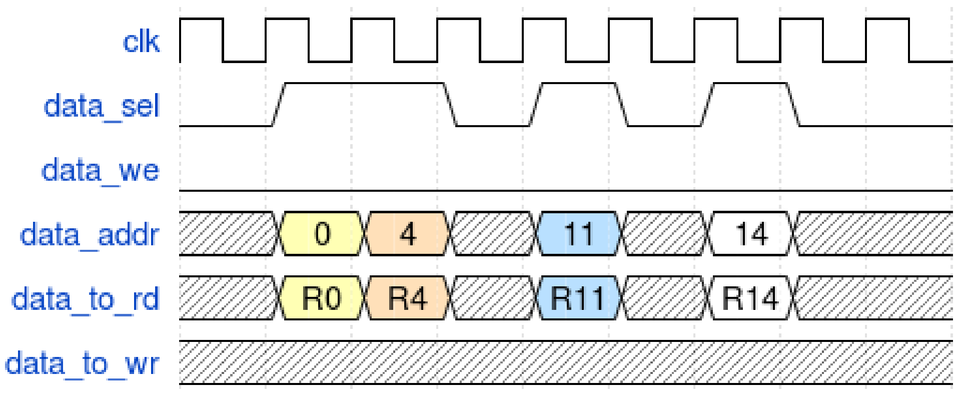

Figure 3: Data Bus reads.

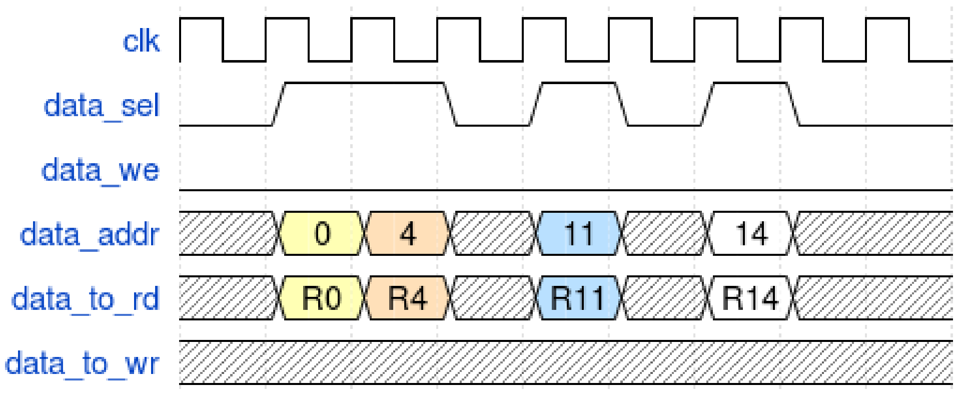

Figure 4: Data Bus writes.

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

PicoVersat is supplied with two useful peripherals, a general purpose register file and a debug print facility.

A simple System on Chip (SoC) including picoVersat, a program and data memory, and the two peripherals attached to the data bus is shown in Figure 5.

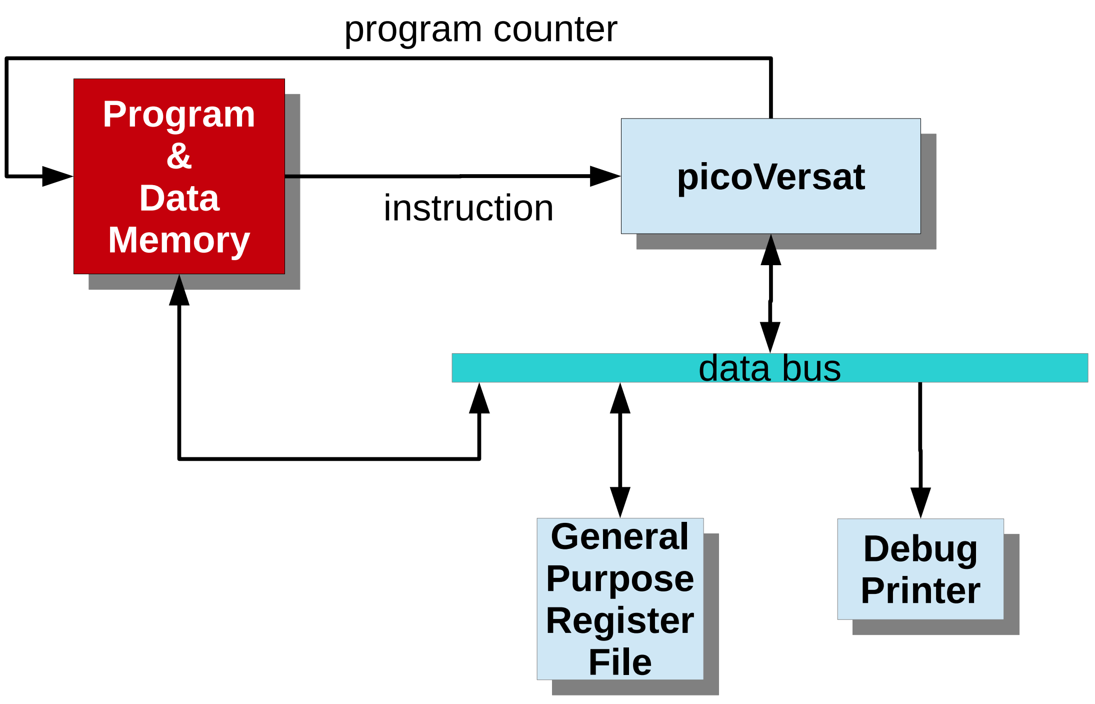

Figure 5: PicoVersat SoC with two peripherals

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.

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 Instructions

6.1 Instruction Format

PicoVersat features a very minimal set of instructions. There is only one instruction type as illustrated in Table 4.

Bits	Description
31-28	Operation code (opcode)
27-0	Immediate constant

Table 4: Instruction format.

6.2 Instruction Set Architecture

The instruction set is given in Table 5. The notation M[Imm] represents the contents of position (address) Imm in the memory map.

6.3 Aliased Instructions

There are two instruction aliases:

`nop`: same as `addi 0`. It means No OPeration or do nothing.

Mnemonic	Opcode	Description
Arithmetic / Logic		
addi	0x0	RA = RA + Imm; PC=PC+1
add	0x1	RA = RA + M[Imm]; PC=PC+1
sub	0x2	RA = RA - M[Imm]; PC=PC+1
shft	0x3	RA = (Imm $<$ 0)? RA $<<$ 1: RA $>>$ 1; PC=PC+1
and	0x4	RA = RA & M[Imm]; PC=PC+1
xor	0x5	RA = RA $\oplus$ M[Imm]; PC=PC+1
Load / Store		
ldi	0x6	RA = Imm; PC=PC+1
ldih	0x7	RA[31:16] = Imm; PC=PC+1
rdw	0x8	RA = M[Imm]; PC = PC+1
wrw	0x9	M[Imm] = RA; PC = PC+1
rdwb	0xA	RA = M[RB+Imm]; PC = PC+1
wrwb	0xB	M[RB+Imm] = RA; PC = PC+1
Branch		
beqi	0xC	RA == 0? PC = Imm: PC += 1; RA = RA-1
beq	0xD	RA == 0? PC = M[Imm]: PC += 1; RA = RA-1
bneqi	0xE	RA != 0? PC = Imm: PC += 1; RA = RA-1
bneq	0xF	RA != 0? PC = M[Imm]: PC += 1; RA = RA-1

Table 5: Instruction Set.

`wrc A,B`: same as `wrw (A+B)`, where A and B are numeric and integer constants given with the `wrc` alias.

6.4 Delayed Branches

The instruction following a branch instruction is always executed to the instruction pipeline latency (delayed branch or slot). Make sure instructions placed after branch instructions are useful and not harmful.

Insert `nop` instructions after a branch if there is no suitable candidate for the delay slot

7 Assembler

PicoVersat can be programmed in its assembly language. The assembler is a Python program called `va`. It can be used to create the boot ROM program or the user program.

The assembler reads a dictionary (json format) named xdict.txt, which associates user friendly names with numeric constants.

The dictionary xdict.txt is produced by running the xdict.v Verilog file in a simulator. The xdict.v file contains a testbench-like module and includes a few header files where the constants of interest are defined.