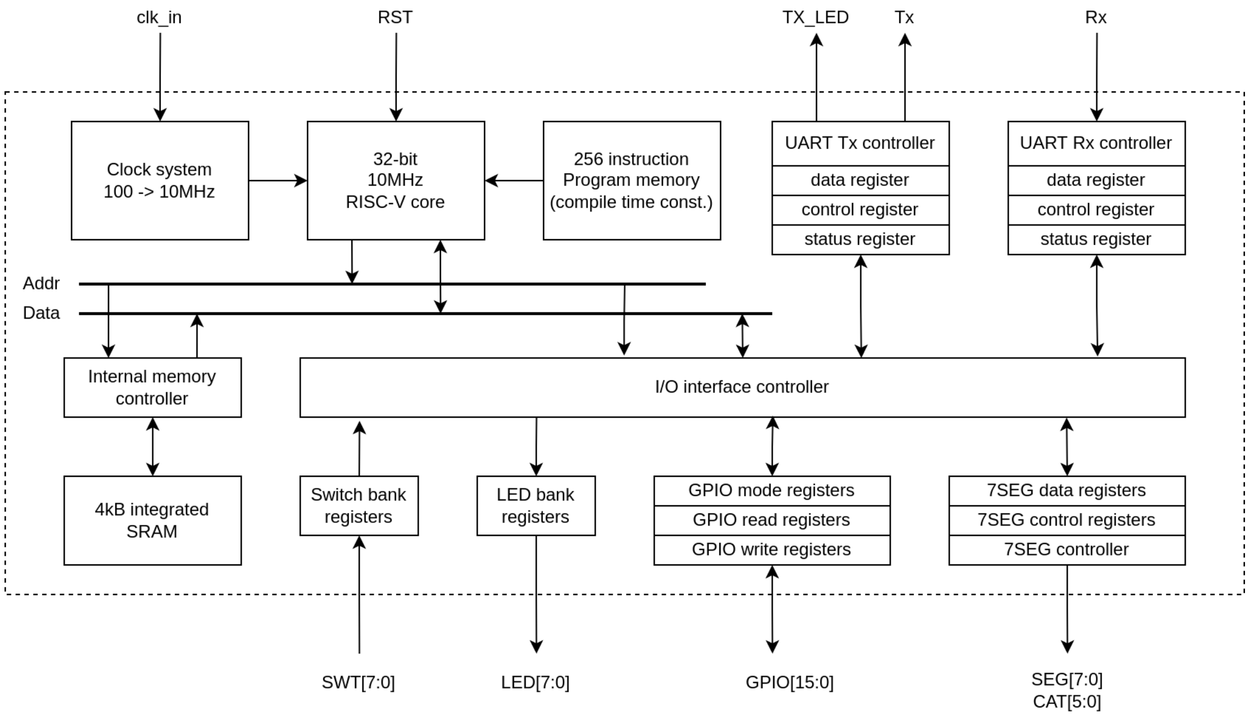
Pēteris Račinskis
pr20015

MDCPU microcontroller datasheet

Overview



MDCPU is a soft-core microcontroller designed for use with the Digilent Anvyl FPGA development board implementing a reduced version of the RISC-V 32-bit base integer instruction set. At this time it is not reconfigurable. Therefore the design is constrained for use with the Xilinx Spartan6 xc6slx45-3csg484 with hardware integration (clock source, pinout) dictated by the specific implementation found on the aforementioned development board.

Features:

- **10MHz RISC-V CPU with 32 general-purpose registers;**
- **4096 bytes of on-chip static RAM;**
- **256 words of program memory (determined at HDL compile time);**
- **Memory-mapped I/O;**
- **Full-duplex Universal Asynchronous Receiver-Transmitter (UART) interface (baud rate configured at HDL compile time, default – 9600);**
- **7-segment display controller with support for up to 6 digits at a 600Hz refresh rate;**
- **16 configurable general purpose input-output pins;**
- **8 input-only switch pins;**
- **8 output-only LED pins.**

Pinout description

The pinout table is specified by physical connections found on the Anvyl board. Pin identifiers in accordance with the xc6slx45-3csg484 package specification.

Symbolic name	**Pin identifier**	**Description**
clk_in	d11	100 MHz clock source
SWT[0]..SWT[7]	v5, u4, v3, p4, r4, p6, p5, p8	Input only switch pins
LED[0]..LED[7]	w3, y4, y1, y3, ab4, w1, ab3, aa4	Output only LED pins
TX_LED	t7	Transmitter status indicator
RST	e6	Global reset
SEG[0]..SEG[7]	aa21, aa22, y22, n15, ab19, p20, y21, p15	7-segment display individual anodes
CAT[0]..CAT[5]	m17, p16, p19, n16, ab21, aa20	7-segment display common cathodes
Rx	t19	UART Rx (connected to USB-UART converter)
Tx	t20	UART Tx (connected to USB-UART converter)

Instruction set

The CPU core implements a reduced version of the RV32I base integer instruction set. The executable program code is read into memory during HDL compilation and therefore constant when loaded on hardware. Program execution always starts with the program counter at 0. Below is the table of implemented instructions as subsets of RV32I:

Instruction class	**Implementation status**
Integer computation	implemented
Control transfer	implemented
Load/store	implemented
Memory ordering	not implemented
Environment call and breakpoints	not implemented
HINT	not implemented

Memory map

All I/O operations are performed through the use of memory access instructions. Addresses can be found in the table below, while the detailed operation of every module can be found further down. I/O operations are all mapped in the upper half of the address space. **Addresses in the I/O mapped range are not treated as contiguous** - reading/writing a byte, half-word or word to the same address will produce the same byte-valued result, and not return or alter the contents of the neighboring addresses.

Address range	Function
0x00000000 – 0x0000ffff	General purpose RAM
0x00001000 – 0x7fffffff	Not mapped
0x80000000	Switch register (read-only)
0x80000001	LED register (rw)
0x80000002 – 0x80000003	GPIO mode registers (rw)
0x80000004 – 0x80000005	GPIO read registers (read-only)
0x80000006 – 0x80000007	GPIO write registers (rw)
0x80000008 – 0x80000013	7-SEG data registers (rw)
0x80000014	7-SEG control register (rw)
0x80000015	Rx controller status register (read-only)
0x80000016	Rx controller control register (rw)
0x80000017	Rx buffer data-out (read-only)
0x80000018	Tx controller status register (read-only)
0x80000019	Tx controller control register (rw)
0x80000020	Tx transmit data-in (rw)

I/O module usage instructions

Input, output, GPIO pins (SWT, LED, GPIO)

Input registers can be read in exactly the same way as regular memory addresses, with the aforementioned caveat – word/half-word memory access instructions are treated exactly the same as byte instructions by the underlying hardware, therefore interacting with multiple registers in a single access operation is not possible.

Pure output registers (LED) behave exactly like regular memory locations during writes as well. Pure input registers (SWT) do not accept write data. The GPIO pin bank has three register sets:

- read data: read-only registers, contain the synchronized pin value;
- mode: read-write registers that control the direction of each pin;
- write data: read-write registers the value of which gets written to the pin (bitwise) only if the corresponding mode register bit is set to 1. If this value is set to 0 instead, the pin acts as an input and the write-data value is ignored.

7-segment display

MDCPU has a hardware 7-segment display controller that handles the scanning of the display segments. It exposes a data register array and a control register.

- data register array: 6 bytes (corresponding to one digit display each). Each bit corresponds to an anode in the common-cathode display configuration, with each byte mapped as [7:0] – [dot,G,F,E,D,C,B,A];
- control register: setting this to 0 disables the display, but the values in the data register array are conserved.

Tx Controller

The UART Tx controller is capable of transmitting one byte of data at a time. It is configured to use a baud rate of 9600 and two stop bits at the HDL level. There is no transmit data buffer, so software must handle flow control. Interaction with the Tx controller is mediated by three registers:

- Tx data-in: byte to be transmitted. Gets read once during the start of a transmit cycle;
- Tx status: value of -1 (255) indicates that the controller is busy with an ongoing transmission;

- Tx control: value of -1 (255) triggers a transmit cycle. Does not reset automatically.

Rx controller

The UART Rx controller has a 256-byte receive buffer that stops admitting writes in case of overflow, always allows reads from the current read pointer address but only increments the read pointer on command. The receiver line is also configured to use a baud rate of 9600 and requires at least 1 stop bit. The controller has three registers:

- Rx data-out: value in the receive buffer currently pointed to by the FIFO read pointer. Can always be accessed;
- Rx status: indicates whether there is any new data in the buffer (values of 0 for no new data and -1 (255) for new data);
- Rx control: setting this to -1 (255) increments the read counter unless the buffer is already empty.