http://members.optushome.com.au/jekent/

Created by: John Kent
If you find this project useful, please email me at
jekent@optushome.com.au

www.RetroMicro.com

Tested by: Douglas Hodson
If you find this project useful, please email me at
doug@RetroMicro.com

Project Name: micro8-xsa

Creation Date: May 2003
Development Board: XESS XSA-100 Plus XStend Version 2
Development Software: Xilinx ISE Version 5.1.03i

Description

Micro8 is a very simple VHDL microprocessor that was designed by John Kent. Doug at RetroMicro simply packaged it a little differently and tested it on the XESS XSA-100 plus Xstend board setup. During this process the project was upgraded to to be fully Wishbone compliant.

Micro8 started of as a minimal set 4 instruction computer By Tim Boscke designed to fit in a 32 Macrocell CPLD.

http://www.tu-harburg.de/~setb0209/cpu/

Tim's computer had the following instructions ADD, NOR , STA and JCC. It had a single carry bit which was reset by the JCC (Jump on Carry) instruction. Most microprocessor instructions can be built up using these basic instructions. I have extended the instruction set by:

- Adding an 8 bit index register.

Bit 7 selects between accumulator or index register instructions

B7
0 - A - Accumulator
1 - X - Index Register

- The four instructions remain much the same

B6, B5
00 - ADD - add memory to register
01 - NOR - nor register with memory
10 - STO - Store register to memory
11 - Bcc - Branch on condition code

- I Added 4 addressing modes:

Tim's computer only had absolute addressing

B4, B3
00 - I - Immediate
01 - A - Absolute
10 - X - Indexed
11 - P - PC Relative

- Added 8 conditional branch instructions (PC relative)

In Tim's computer, there was only one carry bit which was reset by the jump. This was so that subsequent jump instructions where always takem. The jump instruction also used absolute addressing. On my version, only ADD, NOR and STO affect the condtion codes, not the branches. The branch instructions on my computer have a 11 bit displacements so can jump anywhere in the address range. It would be nice to have an absolute JMP instruction, to jump to fixed locations but there is not enough room in the opcode map.

B7, B6, B5, B4, B3
01100 - BRA - Branch Always
01101 - BEQ - Branch if Zero flag set
01110 - BCS - Branch carry flag set
01111 - BMI - Branch if Negative flag set (Negative)
11100 - BRN - Branch never
11101 - BNE - Branch if not Zero
11110 - BCC - Branch carry clear
11111 - BPL - Branch if Negative flag Clear (Positive)

- Instructions are now all 2 bytes long.

The high address is formed by Bit B2, B1 and B0 of the opcode and the low order address or immediate value by B7 to B0 of the second byte. This means you can address 2K bytes of memory, where as Tim's computer could only address 6 bits or 64 bytes of memory.

Instruction Macros

To Clear a register,
NORAI #FF

To Load a register from memory
NORAI #FF
ADDAA address

To perform ones compliment
NORAI #00

To test a bit, say bit 0
NORAI #FE ; set all other bits and invert
BEQ branch_address ; test for opposite state

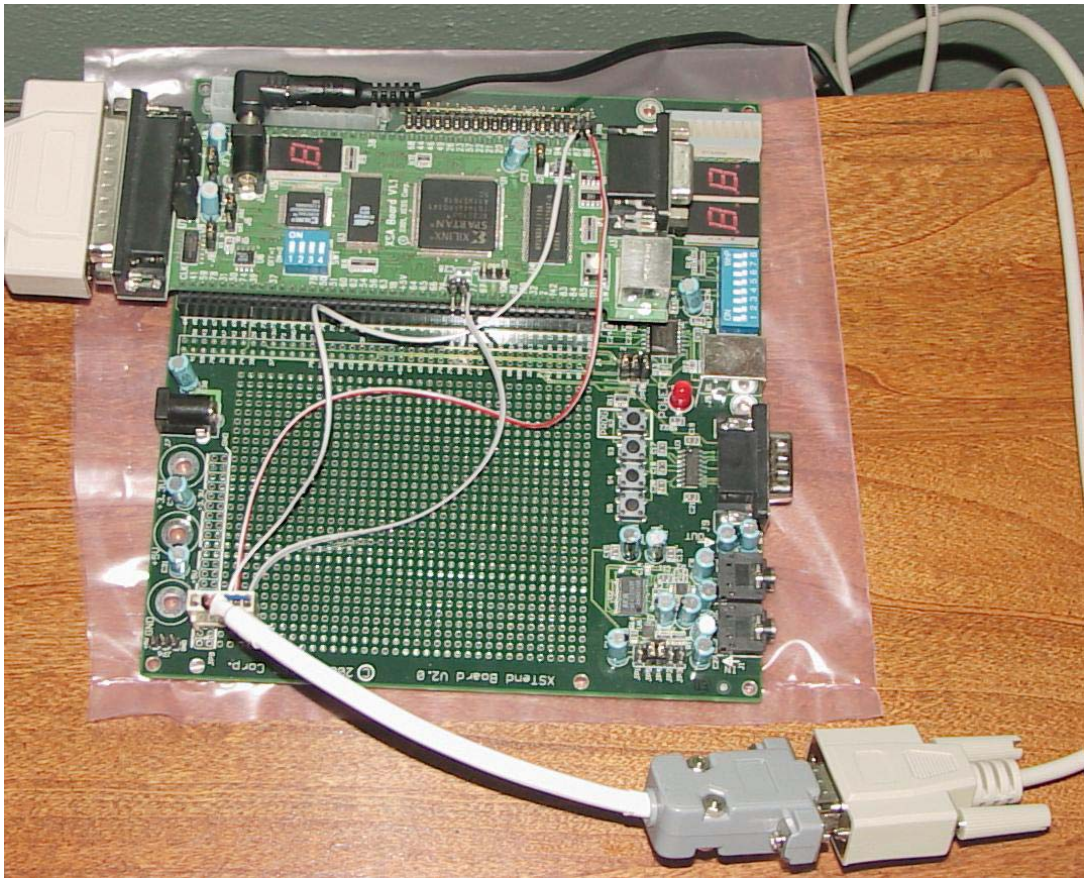
I have not tested all the instructions but the code prints out "Hello World" on the miniuart and waits for a character to be entered.

The CPU is interfaced to a boot ROM entity, real SRAM and a RS232 port. To use this project with an XSA setup you will probably need to extend your Xstend board by adding another RS232 interface that will not conflict with SRAM signals.

This can be accomplished by purchasing a RLC-1 RS232 to 3-5 volt level converter from DigitalNemesis.

http://www.digitalnemesis.com/catalogue/RLC1/RLC1.htm

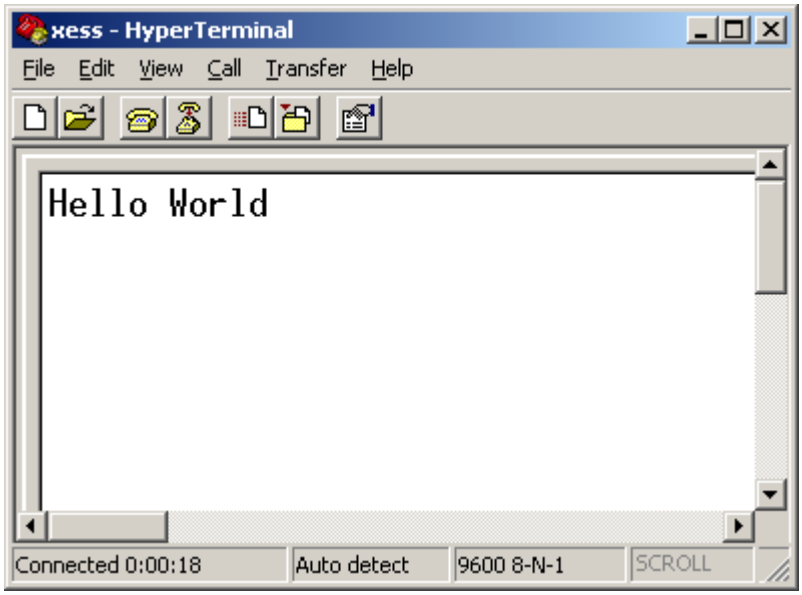
Power for the converter can be pulled from the Xstend. Pins 86 and 87 are the pins I used for rs232 transmit and receive signals. RTS and CTS can be ignored. Use a straight (non-null) cable to connect up to a PC.



Use a simple terminal emulator such as HyperTerminal (provided with Windows) and set terminal to the appropriate com port with settings of 9600 baud, 8 data bits, no parity, 1 stop bit and no flow control.

The design is set to run at 10MHz. Make sure the external oscillator is programmed for this frequency. This is the only requirement due to the fact that the baud rate generator needs to run at the correct frequency. A generic is provided to change this if a different external freq is provided.

Output should look as follows:



Project Directory Structure

This project is organized into the following directories:

./ - contains all the files need to synthesize the design.

./config – effectively a backup of all main files in the root directory

./docs – this PDF file. Source files used to build this PDF are located in ./docs/src.

./src – source directory for all HDL files

./temp – temporary directory used during the build process.

Synthesis

The project is built and maintained using two windows based batch files. The first is "make.bat". It simply issues are the commands required to "compile" all the source files in the ./src directory and eventually generate the chipIO.bit file to downloaded to the fpga.

The second batch file is called "clean.bat". Its sole purpose is to delete most of the unwanted files generated during the build process.