

- This document will guide you through compiling and testing the provided mcx16 demonstration code on the Avnet Spartan-6 LX9 MicroBoard. The code was tested using Xilinx ISE 13.2.
- The MicroBoard is inexpensive (\$89 when I bought one), but even if you don't have one you can still use the provided sample code to experiment with synthesizing the mcx16 microcontroller.

mcx16 is a 16-bit microcontroller written in VHDL. It was inspired by PicoBlaze, an 8-bit microcontroller designed by Ken Chapman at Xilinx. mcx16 uses an instruction set and architecture similar to PicoBlaze, so if you've already learned one, using the other should be straightforward. The mcx16 code is tested on Xilinx FPGAs.

Some highlights of mcx16:

- 16-bit addresses, data, port addresses and ports
- Can use 1 (non-pipelined) or 2 (pipelined) clocks per instruction
- 74MHz (74 MIPS) on a Spartan 3 XC3S200AN-4 non-pipelined, and 119 MHz when pipelined (59 MIPS), using 110/114 slices
(area optimized, 16 registers, 32 deep stack, 512 word program, distributed stack)
- 107 MHz (107 MIPS) on a Spartan 6 XC6SLX9-2 non-pipelined, and 166 MHz when pipelined (83 MIPS), using 46/41 slices
- Pipelining, register count (1 – 256), stack depth (1 – 65536), and program/stack type (BRAM/distributed) are all user-set generics; program size up to 65536 words
- Cross-platform assembler
- Optional JTAG loader using Kris Chaplin's technique and cross-platform Xilinx tools

- mcx16s6mb.xise: Project file
- mcx16s6mb.vhd: Top level VHDL
- mcx16s6mb.ucf: Pin and clock constraints
- mcx16.vhd: Microcontroller VHDL
- mcx16s6mb_rom.txt: Program ROM source
- mcx16s6mb_rom.vhd: Program ROM VHDL
- mcx16loader.vhd: JTAG loader VHDL
- mcx16uart.vhd: UART VHDL
- mcx16util.vhd: Shared utility VHDL

- Extract the mcx16s6mb.zip file into a directory
- Open the mcx16s6mb.xise project file with ISE
- Let it create a “work” directory if prompted
- Double-click “Generate Programming File”
- This should complete without errors
- Click the “Design Summary” button to inspect the design’s resource usage

COMPILE PROJECT

The screenshot displays the Xilinx ISE Project Navigator interface. The 'Design Summary' window is open, showing a table of resource usage statistics. A red arrow points to the 'Generate Programming File' button in the left-hand pane, and another red arrow points to the 'Run' button in the top toolbar.

Resource	Used	Available	Usage %
Number of occupied Slices	83	1,430	5%
Number of LUT Flip Flop pairs used	272		
Number with an unused Flip Flop	148	272	54%
Number with an unused LUT	50	272	18%
Number of fully used LUT-FF pairs	74	272	27%
Number of unique control sets	26		
Number of slice register sites lost to control set restrictions	129	11,440	1%
Number of bonded IOBs	4	200	2%
Number of LOCed IOBs	4	4	100%
IOB Flip Flops	2		
Number of RAMB16BWERS	1	32	3%
Number of RAMB8BWERS	0	64	0%
Number of BUFIO2/BUFIO2_CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_CLKs	0	32	0%
Number of BUFG/BUFGMUXs	2	16	12%
Number used as BUFGs	2		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	4	0%
Number of ILOGIC2/ISERDES2s	1	200	1%
Number used as ILOGIC2s	1		
Number used as ISERDES2s	0		

The console window shows the following output:

```

Command Line: bitgen -intstyle ise -f mcx16s6mb.ut mcx16s6mb.ncd
WARNING:PhysDesignRules:367 - The signal <uart/tx_fifo/Mram_fifo1_RAMD_D1_O> is
incomplete. The signal does not drive any load pins in the design.
WARNING:PhysDesignRules:367 - The signal <uart/rx_fifo/Mram_fifo1_RAMD_D1_O> is
incomplete. The signal does not drive any load pins in the design.

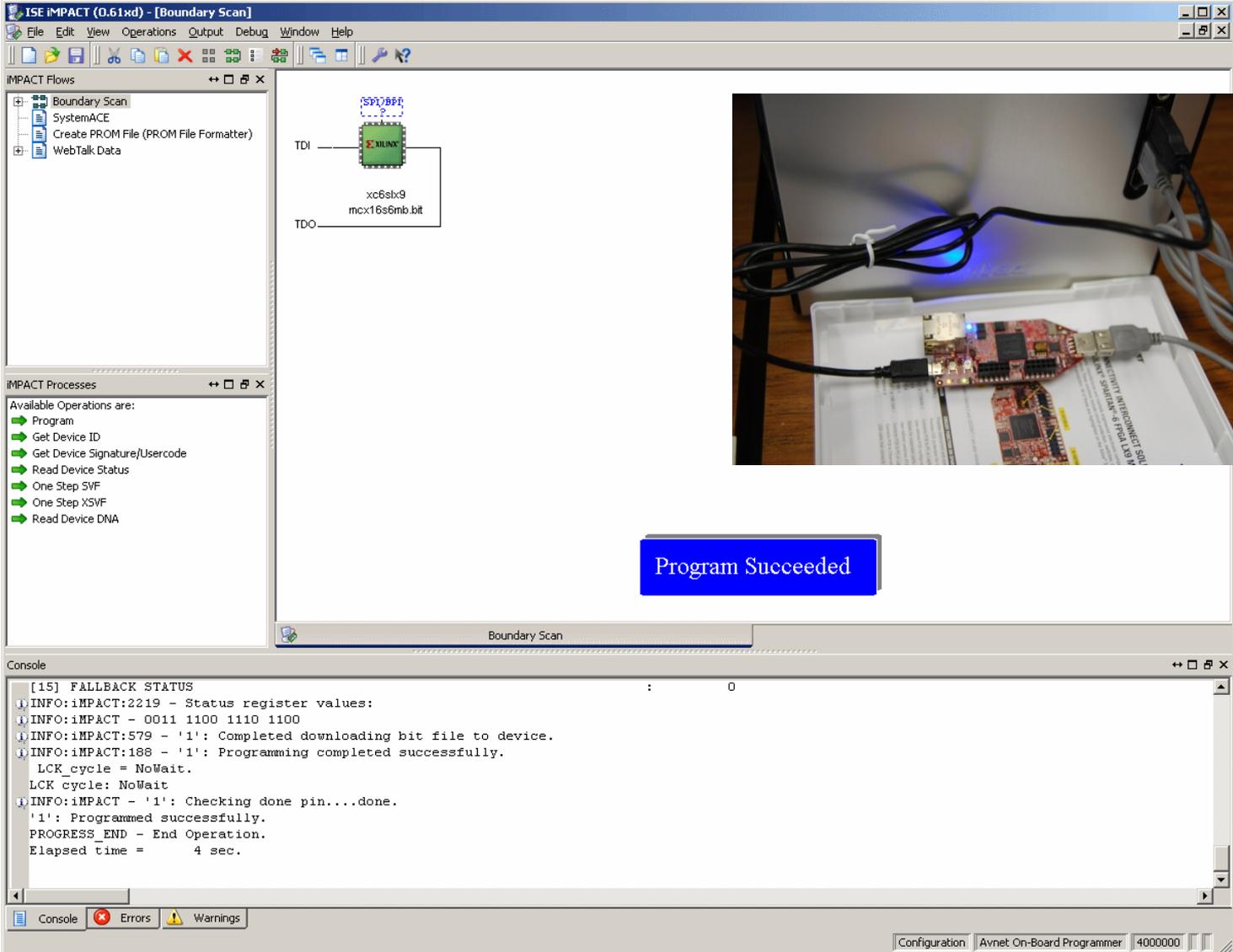
INFO:WebTalk:4 - D:/mcx16/mcx16s6mb/work/usage_statistics_webtalk.html WebTalk
report has been successfully sent to Xilinx. For additional details about this
file, please refer to the WebTalk log file at
D:/mcx16/mcx16s6mb/work/webtalk.log

WebTalk is complete.

Process "Generate Programming File" completed successfully
Launching Design Summary/Report Viewer...
    
```

- It is assumed that you've already installed the Digilent JTAG drivers for the MicroBoard, along with the Silicon Labs CP2102 serial port drivers (these steps are described in detail in the MicroBoard documentation)
- With the MicroBoard connected to your computer (both through the microUSB connector for serial and through the full USB connector for JTAG), launch iMPACT, create a new project, and assign the `work/mcx16s6mb.bit` file to the FPGA

ASSIGN BIT FILE



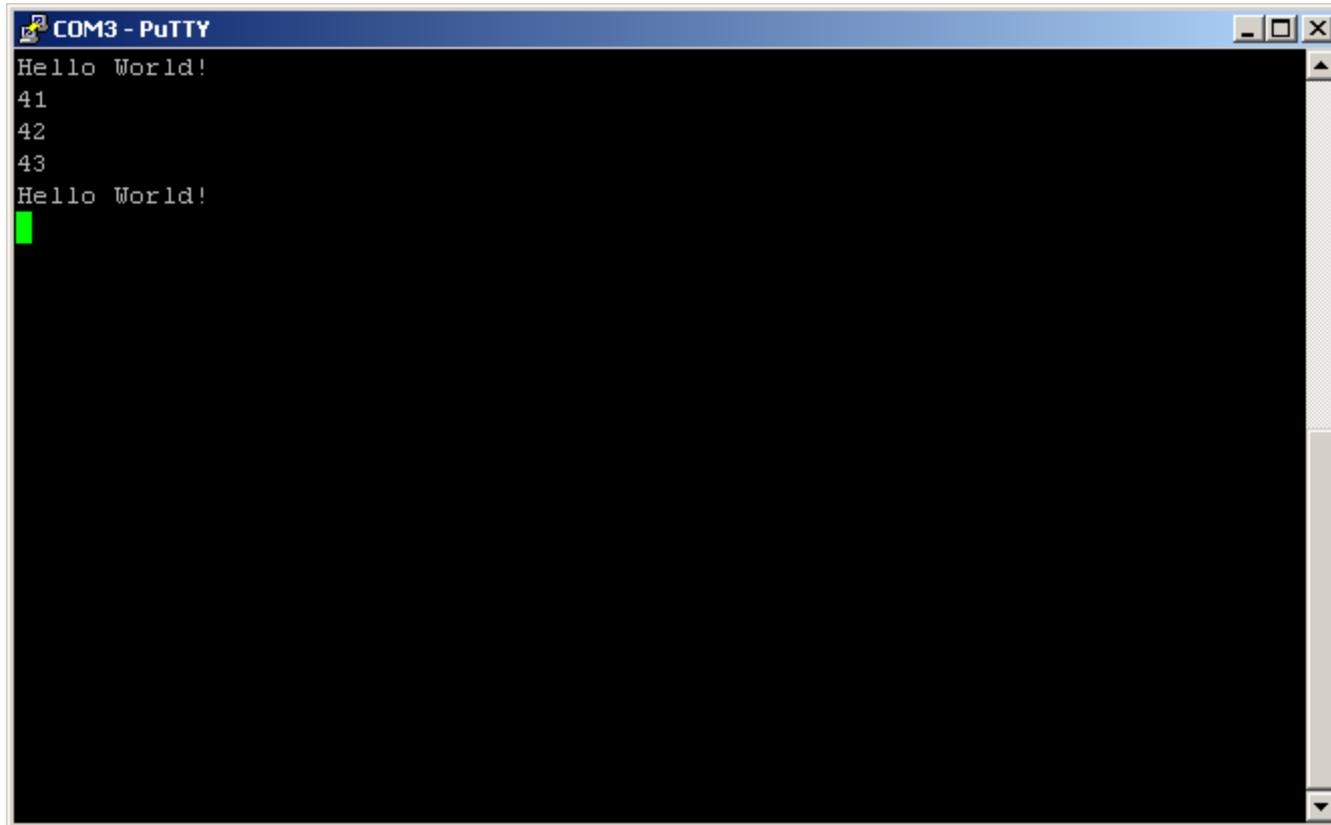
The screenshot displays the ISE iMPACT (0.61.xd) - [Boundary Scan] interface. The main workspace shows a diagram of a device with TDI and TDO connections, labeled 'xc6sxb9' and 'mcx16s6mb.bit'. The left sidebar shows 'IMPACT Flows' and 'IMPACT Processes'. The console window at the bottom displays the following log output:

```
[15] FALLBACK STATUS
INFO:iMPACT:2219 - Status register values:
INFO:iMPACT - 0011 1100 1110 1100
INFO:iMPACT:579 - '1': Completed downloading bit file to device.
INFO:iMPACT:188 - '1': Programming completed successfully.
LCK_cycle = NoWait.
LCK Cycle: NoWait
INFO:iMPACT - '1': Checking done pin....done.
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time = 4 sec.
```

A blue box with the text "Program Succeeded" is overlaid on the console area. An inset image shows a physical circuit board connected to a USB cable, with a blue light visible on the board.

- Open a serial terminal to the MicroBoard (using HyperTerm, PuTTY, etc, 115200 N81)
- Program the MicroBoard with the bit file
- You should see “Hello World!” after programming, and every time you press the reset switch on the MicroBoard (SW5)
- When you type into the terminal, it should respond with the two digit hex codes of the characters you typed (programmed, typed “ABC” and then pressed SW5 in the following screen)

TERMINAL OUTPUT



```
COM3 - PuTTY
Hello World!
41
42
43
Hello World!
█
```

- Download the mcx16 files (which includes the assembler) into a directory, and copy the “mcx16s6mb_rom.txt” file into the same directory
- Launch the “ISE Design Suite Command Prompt” and navigate to the directory with the mcx16 files
- Edit the “mcx16s6mb_rom.txt” file (change the ‘H’ to ‘J’ in “Hello World”, for example), and then run:

```
mcx16asm mcx16s6mb_rom.txt -l
```
- This will compile and load the new code via JTAG into the mcx16rom, and restart the processor

```
C:\ ISE Design Suite Command Prompt
D:\mcx16>mcx16asm mcx16s6mb_rom.txt -1
=== mcx16 Assembler R1 ===
Assembly successful.
Highest address used: 43 (0x2B).
ROM size: 512 (0x200).
Starting impact...
Program successfully uploaded.
D:\mcx16>_
```

```
COM3 - PuTTY
Hello World!
41
42
43
Hello World!
Hello World!
Jello World!
Jello World!
```

CONCLUSION

- This concludes the quick mcx16 demo. Feel free to experiment - compare resource usage for different numbers of registers, stack depths, single/dual cycle, or check maximum clock frequencies (> 111 MHz for the example single cycle project).
- Thanks for giving mcx16 a try, I hope it proves useful! The latest mcx16 info will be posted at <http://www.bitpond.com/mcx16>

- Greg Bredthauer