*** N.I.G.E. MACHINE ***

Application Notes

Contents

1. Installation and set-up

1.1 Introduction

The N.I.G.E. Machine is a user-expandable micro-computer system that runs on an FPGA development board. It has been designed specifically for the rapid prototyping of experimental scientific hardware or other devices. The key components of the system include a stack-based softcore CPU optimized for embedded control, a FORTH software environment, and a flexible digital logic layer that interfaces the micro-computer components with the external environment.

The N.I.G.E Machine is presently available for both the Digilent Nexys 2 (1200K gate) and Nexys 4 FPGA boards.

Further information on the N.I.G.E. Machine design is available in papers presented at EuroFORTH:

- http://www.complang.tuwien.ac.at/anton/euroforth/ef12/papers/
- http://www.complang.tuwien.ac.at/anton/euroforth/ef13/papers/
- http://www.complang.tuwien.ac.at/anton/euroforth/ef14/papers/

Two short video introductions are also available:

- http://www.youtube.com/watch?v=0v-HuVLRoUc
- http://www.youtube.com/watch?v=0Kj5EMdnkMk

1.2 Set-up preliminaries

- Install Xilinx ISE version 14.6
  - The N.I.G.E. Machine is being developed on ISE 14.6, so using this version will avoid compatibility issues
- Install a suitable GIT repository manager, e.g.:
  - http://www.syntevo.com/smartgithg
- Clone the open-source N.I.G.E. Machine repository from GitHub to your local machine
  - Remote repository location:
    - https://github.com/Anding/N.I.G.E.-Machine
  - To preserve absolute file references used by ISE, the local directory for the repository must be exactly as follows:
    - E:\N.I.G.E.-Machine
- Within the local repository, switch to the appropriate branch for your Nexys board:
  - Nexys 2 (1200K gate) v2.0
  - Nexys 4 v4.0 (default branch)

1.3 Quick start

- Attach a VGA monitor, keyboard (PS/2 - Nexys 2 or USB - Nexys 4), and 5V power supply
- Set the FPGA board jumper wires according to the Nexys reference manuals

- Configure the FPGA board with the pre-compiled N.I.G.E. Machine bit file
  - Nexys 2: E:\N.I.G.E.-Machine\board_Nexys 2_1200_v2.0.bit
  - Nexys 4: E:\N.I.G.E.-Machine\board_Nexys 4_v4.0.bit
- The N.I.G.E. Machine is now running as a FORTH microcomputer
  - See chapter 2 for further guidance

1.4 Full start

- Unzip the file Xilinx_ISE.zip to the local repository folder
  - E:\N.I.G.E.-Machine
- The Xilinx project files should now be found in
  - E:\N.I.G.E.-Machine\Xilinx_ISE
  - Watch out for inadvertent folder duplication ("\Xilinx_ISE\Xilinx_ISE") caused by the unzip process or absolute project file references used by ISE will be invalid
- Double click on the ISE project file
  - E:\N.I.G.E.-Machine\Xilinx_ISE\NIGE_Machine.xise
- The N.I.G.E. Machine design files are now open in Xilinx ISE
- Synthesize the design files
- Configure the Nexys board with the newly compiled N.I.G.E. Machine bit file
  - E:\N.I.G.E.-Machine\Xilinx_ISE\[xxx].bit

1.5 Optional SD card interface

- Nexys 2: utilize a full-size SD card and a Digilent SD card PMOD device plugged into port A
  - http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,513&Prod=PMOD-SD
- Nexys 4: utilize a micro-SD card and the micro-SD slot on the Nexys board
- Format the SD card as FAT32
  - Both normal (<2GB) and high capacity (>2GB) SD cards are acceptable
  - The card must be formatted as FAT32 irrespective of capacity
- Copy all of the files from the following folder onto the SD card
  - E:\N.I.G.E.-Machine\Software
- Insert the SD card into the slot on the PMOD (Nexys 2) or the directly into board (Nexys 4)
- See chapter 2 for further guidance on using the SD card as a file system

2. Using the N.I.G.E. machine as a FORTH microcomputer

2.1 ANSI FORTH

At power-on the N.I.G.E. Machine is a self-contained microcomputer with FORTH system software. As far as possible the system software has been designed to be compliant with ANSI FORTH. Some minor ANSI deviations were allowed given the constraints of an embedded system. See appendix 6 for a list of the ANSI FORTH words that are available on the N.I.G.E. Machine and appendix 7 for a list of N.I.G.E. Machine specific control words.

2.2. File System

SD card usage and file system navigation

The N.I.G.E. Machine uses SD cards for file system external storage. See Chapter 1 for SD card set-up details. At power-on the only available file access word is INCLUDE. Prepare and insert an SD card, then issue the following commands to make available additional ANSI FORTH words from the file access and other word sets.

```
MOUNT               \ Mount an SD card and initialize FAT32 data
INCLUDE SYSTEM.F    \ SYSTEM.F extends the FORTH system software
```

The N.I.G,E. Machine reads and writes filenames in 8+3 format only (e.g. “FILENAME.EXT”), but directory structures are supported. The file path directory separator character is either forward slash “/” or back slash “\”, and these may be used interchangeably. A leading slash or a double slash (“//” or “\\”) within a file path are interpreted as go-up-one-directory-level. No special character is used to specify the root folder.

Examples:

```
S” PROJECT/TEST.F”  \ specify the file TEST.F in the PROJECT directory
                    beneath \ the current directory

S” \B\DATA.TXT”      \ go up one level from the current directory and down
                     \ again into folder B to specify the file DATA.TXT
```

New-line character

The system software recognizes LF as the line terminator disregards CR characters. The WRITE-LINE word terminates lines with CRLF. The N.I.G.E. Machine can therefore read and process both Windows and LINUX format files, but files are written in Windows format.

2.3 Memory address regions

The N.I.G.E. Machine address space is spanned in bytes. It comprises three separate memory regions:

1. System memory is available for storage of the FORTH dictionary and other data. This is referred to as SRAM in the N.I.G.E. Machine documentation. System memory is comprised of FPGA BLOCK RAM and is the only memory from which the CPU can fetch and execute code.

2. Memory mapped hardware registers that control the operation of the N.I.G.E. Machine or connect to external pins on the FPGA. These registers are implemented in FPGA fabric logic and route directly into the N.I.G.E. Machine's own functional modules.

3. The external pseudo-static dynamic RAM chip included on the Nexys boards. This memory is referred to as PSDRAM. The N.I.G.E. Machine CPU can fetch and store data from PSDRAM, but cannot execute code there. PSDRAM is also used to hold the VGA screen display buffer. Access to PSDRAM is arbitrated by a direct memory access controller module within the N.I.G.E. Machine.

In terms of the choice of RAM for data storage, SRAM has the advantage that it is fast (1 clock cycle access) and deterministic. However total capacity is limited to tens of kilobytes. PSDRAM is more plentiful (16 megabytes), but access is slower and subject to arbitration with the VGA display.

The CPU has separate instructions for byte, word and long-word memory access and the corresponding FORTH words C!, C@, W!, W@, !, @ are available in system software.

Access to SRAM does not need to be aligned. Both words and long-words can successfully be read/written to odd address boundaries without penalty. PSDRAM access on the other hand must be aligned: words can only be accessed on even address boundaries and long-words accessed on address boundaries divisible by four.

The N.I.G.E. Machine is big-endian format. Memory access for words and long-words is such that the highest value byte is placed in the lowest memory address.

2.4 RS232 port

In the default configuration the N.I.G.E. Machine includes a single RS232 port.

In the N.I.G.E. Machine v2.0 the RS232 port is connected directly to the Nexys 2 board's RS232 D-sub connector. However the Nexys 4 board does not include a D-sub connector. For the N.I.G.E. Machine v4.0 a PMOD RS232 expansion should be connected to PMOD socket C, lower pin row. Alternatively it is possible to re-route the RS232 port to the Nexys 4 board's RS232/USB interface. Comment/uncomment the relevant NET definitions in the FPGA constraints file to make this change:

```
E:\N.I.G.E.-Machine\Software\Board_Nexys 4.ucf
```

RS232 communication over an RS232/USB adapter may be less reliable due to buffer/latency issues in the adapter or PC driver. Appendix 7 documents N.I.G.E. Machine specific RS232 input/output words.

The UART adapter is hardwired to 8 bits, 1 stop bit, no parity, no handshaking. The baud rate is user configurable. Default settings are as follows:

v2.0, Nexys 2: 9,600 baud

v4.0, Nexys 4: 57,600 baud

3. Customizing the system software

The system software is written in assembly language. A two-pass cross-assembler is available to prepare the necessary binary files from the assembly language source.

The CPU instruction set is documented in Appendix 2. Because the N.I.G.E. Machine CPU is designed as a FORTH processor, the instruction set is essentially a sub-set of the most primitive FORTH words. The cross assembler provides additional macros corresponding to higher-level FORTH word, mostly for flow control. Appendix 9 documents the cross assembler macros and directives.

3.1 Running the cross-assembler

The cross assembler should be run in an ANSI FORTH environment such as a PC running VFX FORTH. The command to run the cross assembler is ASMX. The N.I.G.E. Machine system software source file is E:\N.I.G.E.-Machine\System\FORTH.ASM.

```
INCLUDE ASMX.F   \ include the cross assembler

S" E:\N.I.G.E.-Machine\System\FORTH.ASM" ASMX
                 \ cross-assemble the system software
```

The three output files are as follows, and **they will always be placed in the folder E:\N.I.G.E.-Machine\System** to maintain integrity of the absolute file references used by Xilinx ISE.

- SRAM.BIN
  - Binary file suitable for transfer to the N.I.G.E. Machine by boot-loader
- SRAM.TXT
  - Text file that is used during simulation with the Xilinx ISE simulator
- SRAM.COM
  - Text file that is used when Xilinx ISE generates the SRAM memory module. Note that the SRAM module must be explicitly regenerated in ISE each time the system software source is reassembled

All of the output files are placed in this folder

```
E:\N.I.G.E.-Machine\System
```

so it is important that the exact directory structure of the source file depository is maintained.

3.2 Updating the system software

The most straightforward way to test the system software is to transfer the completed binary machine code file to the FPGA board on a temporary basis using the boot loader. However the file transferred by boot loader will be lost when the board is powered off or the FPGA is reprogrammed. To incorporate the revised system software on a permanent basis the FPGA configuration .bit file needs to be regenerated with the revised machine code file.

In either case the first step is to re-assemble the updated system software and confirm that a new binary file has been created:

```
E:\N.I.G.E.-Machine\System\SRAM.bin
```

Note that this **exact filepath and filename** must be used to ensure compatibility with the absolute file references used by Xilinx ISE.

Using the boat loader to update the system software until power off

Nexys 2, v2.0

The Digilent Adept application should be used to transfer the file SRAM.bin to via the Epp interface into register 0xFF. The system will automatically reset when the transfer is started and reboot with the new system software.

Nexys 4, v4.0

The Digilent Adept application is not compatible with the Nexys 4 board and so the boot loader utilizes transfer via the RS232 port.

- Establish an RS232 connection between the PC and the Nexys 4 board. (See section 2.5)
- Press the CPU reset button on the Nexys 4 board. The board will reset for 4 seconds
- Immediately transfer the file SRAM.bin to the Nexys 4 board via the RS232 interface at the UART default settings of 57,600 baud, 8 bits, 1 stop bit, no parity, no handshaking. A blue LED on the Nexys 4 board will light during transfer. Ensure that the binary file is transferred without modification (e.g. no CR/LF substitutions)
- At the end of the 4 second reset reboot will occur with the new system software. This will be retained in memory until either the board is switched off or the FPGA is reprogrammed.

Updating the FPGA configuration file for a permanent system software update

The procedure for permanently updating the FPGA configuration bit file is as follows:

- Open the SRAM core in Xilinx ISE

  ```
  inst_SYS_RAM
  ```

- Generate the core and the regenerate the programming file in ISE. **Note that the filepath to the Xilinx memory core initialization module must be exactly as follows:**

  ```
  E:\N.I.G.E.-Machine\System\SRAM.coe
  ```

- Transfer the FPGA configuration .bit file to the Nexys board in the usual manner

4. Customizing the system hardware

The N.I.G.E. Machine is designed at the outset to be extended for specific applications through customized hardware that control or interact with external apparatus. The overall scheme for doing this is as follows:

- Customized hardware is developed as VHDL (or Verilog) modules within the N.I.G.E. Machine design project
- The custom modules interface with the outside world either through the electronic components available on the Nexys board (e.g. LED's, microphone, etc.) or through circuits attached to the PMOD expansion ports. Digilent supply a range of ready-made PMOD circuits that may be suitable for a variety of purposes
    - http://www.digilentinc.com/Products/Catalog.cfm?NavPath=2,401&Cat=9
- The custom modules interface with the N.I.G.E. Machine through either or both of two channels:
    - Additional user-defined hardware registers (that extend the list documented in appendices 3 and 4). These hardware registers may be readable, writable or both. The CPU access them through normal memory fetch and store instructions, while the bit level signals are routed directly into the design of the custom hardware module
    - Additional user-defined system interrupts (that extend the list documented in appendix 11). After creating additional system interrupts appropriate entries will need to be created for them in the interrupt vector table and appropriate interrupt handlers will need to be developed. This will involve customizing the system software in assembly language (see chapter 3)
- Extending the hardware registers and interrupts involves making changes to the following modules in the project design files. In general this should be as straightforward as duplicating the existing logic for each new item using the design file comments as a guide:
    - Inst_HW_Registers
    - Inst_Interrupt
- The FORTH system software is used to debug, test, and develop the necessary control applications for the customized hardware. Because FORTH can be used as an interpreted language, and because the N.I.G.E. Machine includes native keyboard and video display interfaces, the benefits of both rapid prototyping and fast execution speed are available to enhance software development compared with a traditional embedded development using a remote editor and compiler

In addition the N.I.G.E. Machine design is released under a dual license with open source rights for non-commercial use (please see the cover page of this manual for further details.) The design of the system itself can be remade.

This manual cannot fully describe the process of custom hardware development in VHDL or Verilog, but some further suggestions in respect of the N.I.G.E. Machine are made as follows:

- Testing design changes in the electronic simulator (Xilinx ISM) is essential before testing in hardware. For this purpose a faster-to-simulate SRAM module

(RAM_for_Testbench) is provided. Comment out the instance of SYS_RAM in Board_Nexys4 and comment in RAM_for_Testbench. The principal advantage is that this module can read a revised SRAM.bin file directly without being regenerated, thus saving time

- Regression testing of CPU functionality is also essential after modifications are made. Several regression test programs are provided in the folder E:\N.I.G.E.-Machine\System to test the CPU instructions set, memory access and other functions. They can be run in both the electronic simulator and in hardware. Each test in the sequence is announced via the seven-segment display. If a test fails then its sequence number will remain on display. If the complete sequence of tests completes successfully then the value 0xFF is displayed
- It is critical to ensure that the new design meets timing. ISE SmartXplorer is very helpful for obtaining the best place and route for a given design and optimizing timing