Applications of FPGAs in Music and Computer Architecture Education

Matthew Harrison

April 4, 2019

1 Overview of FPGAs

Field Programmable Gate Arrays (FPGAs) are a semiconducting device composed of arrays of logic cells. Those cells can be reconfigured to implement countless logic operations for specific tasks. This reprogrammable hardware is the basis of FPGA technology, but it is quite often taken much further than just these logic cells. Companies such as Digilent and Papilio make FPGA development boards, which include peripherals that can be easily incorporated into a design, such as HDMI, sevens segment displays, audio jacks, and USB.(Xilinx, Monk) Most development boards also have sets of I/O pins so the FPGA can send and receive signals. These I/O ports can be used to implement Peripheral Modules (PMODs), which are external devices made to expand the capabilities of FPGA applications. PMODs can be sensors, connectors, additional memory, or a variety of components. (Romano)

1.1 Applications

The versatility of FPGA applications is nearly endless. Reprogrammable logic allows programmers to design custom systems for a specific task, but one of the most common uses is prototyping. FPGAs are fast, however, not usually the fastest solution. Application Specific Integrated Circuits (ASICs) are exactly what the title implies: printed ICs specialized for a specific application. These semiconducting chips are found in most modern electronics and were invented long before the FPGA. Because ASICs are static printed devices, the internal connections and gates are optimally placed, and they are therefore among the fastest known implementations of logical operations. Many silicon chip production companies like Intel or Texas Instruments begin product development on FPGA hardware.

Designs can be reconfigured as many times as needed until the configuration is optimized, at which point it can be printed as an ASIC and used in a device.

2 Computer Architecture

3 Integrated Developmment Environments

3.1 Vivado Design Suite

Vivado provides all the necessary tools to design and program FPGA hardware. Designs can be made either by HDL programming in VHDL or Verilog, or using Vivado's IP Integrator. The Vivado IPI uses block diagrams to create graphic representations of a design which can then be validated and translated into an HDL wrapper. The IP Integrator wizard speeds up this flow by giving the option to run a connection automation, making all necessary connections such as connections to a clock source. Vivado also allows for High Level Synthesis, meaning the implementation of high level code, such as C, on FPGA IP. The Vivado IPI has blocks to resemble software components and even microprocessors. (Xilinx)

3.2 System Generator

The Xilinx System Generator for Matlab uses the Simulink environment to create and implement FPGA designs. Simulink uses block designs to model systems. System Generator gives access to the default Simulink blockset, and a Xilinx blockset that allows a Simulink design to communicate with an FPGA, or design IP or bitstreams for a specific FPGA device. (Xilinx)

3.2.1 Hardware Co-Simulation

System Generator allows for a Simulation to run both on the host PC and the attached FPGA. This can be very useful for digital signal processing designs, particularly when validating output from the board. (Xilinx)

3.3 Software Development Kit

The Xilinx Software Development Kit (SDK) allows programmers to embed software designs on an FPGA. This can be done through microprocessor design such as the MicroBlaze IP, or using the dual core ARM Cortex processor built into the Zynq 7 series FPGAs. The Vivado IP integrator can interface an FPGAs reprogrammable hardware with the microprocessor, which can then be opened with

SDK to implement a software design on the hardware. SDK allows for the creation and debugging of software, and can be used to run a design on an FPGA. If the design supports communication with the host PC, the SDK terminal can be used.(Xilinx)

4 Projects

4.1 Sequencer

Following a tutorial by Lincoln Tran on instructables.com, I created an 8 step sequencer, able to play a unique pitch on 8 different beats to make a looping pattern. The method of creating sine waves to output in this design was to toggle an output on an off according to a frequency derived from the clock signal. Unlike many other music applications which use software to store tables of frequencies that may be outputted, this design creates its output purely from hardware. To implement this design on a Basys3 board, rather than the Nexus2 the creator used, I had to create a ucf constraint file in Vivado, being sure to use the names for the pins as declared in the top module of the design.