University of Toronto
Department of Electrical and Computer Engineering
ECE496 Design Project

Virtual FPGA fabrics
Implementation of a Virtual FPGA Architecture

Project Proposal
Final Version

October 18, 2011

Neil Isaac n.isaac@utoronto.ca

Keyi Shi keyi.shi@utoronto.ca

Project ID:	2011017
Supervisor:	Jason Anderson
Administrator:	Ross Gillett
Section:	#7

ECE496Y Project Proposal Document Evaluation Form

Student				
Project Title: Virtual FPGA fabrics: Implementation of a virtual FPGA architecture				
Project ID: 2011017	**Supervisor:** Jason Anderson			
Section #: 7	**Administrator:** Ross Gillett			
Estimate # contact hours per month with supervisor: 4			**Suggest optimal # hours per month:** 4	

Administrator's Project Proposal Evaluation *(Provide a rating for each section)*	Excellent	Good	Adequate	Marginal	Poor	***Comments***
Project Description						
Background and Motivation (design problem, past work, references)						
Project Goal and Requirements (Verifiable? Link to original problem?)						
Validation and Acceptance Tests (Link to goal and requirements?)						
Technical Design						
Possible Solutions and Design Alternatives (key trade-offs)						
System-level overview (system block diagram)						
Module-level descriptions (inputs/outputs, functional description)						
Assessment of Proposed Design (strengths, limitations, trade-offs)						
Work Plan						
Work breakdown structure (tasks verifiable? Complete, clear and fair division of work)						
Gantt chart (task described and numbered, logical scheduling)						
Financial plan (Justification for funding? Contingency plan?)						
Feasibility Assessment (skill & resources, risk mitigation plan)						
Overall						
Presentation (grammar, spelling, writing style, figures)						
Content (Exec. summary, organization, logical coherence, substance)						
Other problems (Late, student-supervisor agreement unsigned, supervisor contact time too low)						
❐ **Ethics review** (if gathering personal information or testing on human or animal subjects)						

Administrator's grade only (/10) (*Supervisor's grade on <u>next</u> page)	**Document section average (/10)**	**Administrator's signature:**

Administrator's comments (also see comments in report)

ECE496Y Design Review Meeting Evaluation Form

Student			
Project Title:	Virtual FPGA fabrics: Implementation of a virtual FPGA architecture		
Project ID: 2011017	**Supervisor:** Jason Anderson		
Section #: 7	**Administrator:** Ross Gillett		
Estimate # contact hours per month with supervisor: 4		**Suggest optimal # hours per month:** 4	

Administrator's Evaluation **Group Evaluation**	Excellent	Good	Adequate	Marginal	Poor/unclear	**Comments to Group**
Project Description (background, goals, requirements, acceptance tests)						
Technical Design (alternatives, trade-offs, system overview, testing)						
Work Plan (scheduling, risks, resources)						
December Review Target Received ☐ **Additional deliverables for December Review:**						

Individual Evaluations (roles and contributions)						**Comments to Individuals**
Student 1: **Mark: /10**						
Student 2: **Mark: /10**						
Student 3: **Mark: /10**						
Student 4: **Mark: /10**						
Section average (/10):	❒ **Ethics review recommended**					**Administrator's Signature:**

Note to Supervisors:

Your evaluation was done online. There is no need to return anything else to the administrator.
Please return this report to your students.

Executive Summary

Academic studies of Field Programmable Gate Array (FPGA) chip architecture rely on simulations, as commercial FPGA chips contain proprietary designs that make their underlying architecture inaccessible to academic researchers. The goal of this project is to provide a physical platform for researchers to carry out FPGA architecture studies. The finished design should be financially accessible and capable of running common benchmark circuits.

The proposed design will use an overlay circuit to implement a virtual FPGA on an existing, commercially available FPGA chip. Using a commercial FPGA as the physical medium for this project makes the design cheaper and more accessible to researchers, as they may have an appropriate FPGA chip already.

The overlay design may use architectural features that may specific to a particular FPGA family. Although this will limit the models FPGA chips the overlay circuit can be implemented on, it should reduce the design's area overhead and improve its timing characteristics.

We have selected the Xilinx Virtex 5 FPGA as our development platform. Any architectural features we use on the Virtex 5 will be forward-compatible with all current-generation Xilinx FPGA products, allowing the researcher to use a variety of FPGAs.

The overlay FPGA is intended to be used in conjunction with VPR, an open-source placement and routing tool that is used in FPGA architecture research. As such, the validation of the design will involve testing a set of benchmark circuits by placing and routing them with VPR, then transferring them to the FPGA overlay. The circuits can then be tested for correct behavior, confirming that the overlay design can be correctly programmed using VPR output, and that the inputs and outputs to the design are functioning properly.

The current budget for the proposed design is $14.00 and will be covered by the students. The required FPGA development boards and software licenses have been provided by the supervisor.

Contents

1 Project Description

1.1 Background and Motivation

Academic researchers who study Field Programmable Gate Array (FPGA) design commonly use variations of an FPGA design architecture, described by Kuon et al[1], which we will refer to as the *Academic FPGA Model*. While FPGA chips are available from a variety of commercial vendors, their inner design is proprietary, making their architectures difficult to study. Furthermore, there is no existing physical implementation[1] of an Academic FPGA Model.

One of the tools used in FPGA architecture research is VPR[2]. VPR is an open-source placement and routing tool that accepts a wide variety of architecture parameters. It can handle many variations of the Academic FPGA Model. It is not currently possible to realize circuits produced by VPR on a commercial FPGA.[2]

As such, Computer Aided Design (CAD) researchers who work on placement and routing algorithms for FPGA designs are presently limited to using simulations to evaluate or verify their work. They may be interested in testing circuits on a physical medium because it would be much faster than simulating the circuits.

1.2 Project Goal

The goal of this project is to produce a circuit design based on the Academic FPGA Model. Researchers will be able to use the circuit to study FPGA architecture and CAD algorithms with circuits produced by VPR.

[1]Alex Brant is also developing a comparable FPGA overlay platform with Prof. Guy Lemieux at University of British Columbia.

[2]A technology-mapped input netlist for VPR can be converted to an Altera Quartus VQM netlist file using *nettovqm*[3], but the placement and routing can not be converted.

1.3 Project Requirements

1.3.1 Functional Requirements

Researchers must be able to:

- implement the overlay FPGA circuit on commercially available FPGA chips,
- tune the number, arrangement, and logic cell connectivity of the overlay FPGA,
- program the overlay FPGA using a circuit produced by VPR,
- modify the inputs and outputs of the overlay FPGA.

1.3.2 Constraints

- The overlay circuit must support at least 3000 logic cells[3] in order to accommodate the *"Golden 20"* MCNC benchmark circuits[4] commonly used in FPGA research.

1.3.3 Objectives

- Be compatible with commercial FPGAs that are available to researchers.
- Take advantage of the underlying FPGA architectural features in the overlay FPGA design to reduce area and latency.

1.4 Validation and Acceptance Tests

1.4.1 Functional validation

To ensure that the overlay FPGA circuit design is functional, we will:

1. select and use a benchmark circuit commonly used to test VPR,

---

[3] 3000 logic cells was chosen as the minimum target because the largest of the "Golden 20" circuits, *"s38417"* requires 2567 6-input logic cells[4].

[4] The "Golden 20" MCNC circuits are available in BLIF format at http://www.ece.ubc.ca/~julienl/benchmarks.htm.

2. configure VPR to match our architecture and dimensions,
3. place and route the benchmark circuit with VPR,
4. convert the VPR output into a bitstream for the overlay FPGA,
5. load the bitstream onto the overlay FPGA, then
6. test the functionality of the benchmark circuit running on the overlay FPGA.

This test procedure ensures that:

- circuits can be implemented using VPR output,
- circuits can be transferred correctly to the overlay FPGA, and
- inputs can be set and outputs can be read.

The exact verification process for inputs and outputs will depend on the benchmark circuit's intended function. We will need to develop an appropriate testing mechanism for the benchmark circuit.

1.4.2 Size and overhead validation

To ensure that the overhead is low enough that the overlay FPGA can fit useful circuits, we will test it using the *"Golden 20"* MCNC benchmark circuits. For each circuit, we will:

1. run synthesis and technology mapping using ABC,
2. run placement and routing using VPR configured, and
3. confirm that VPR can place and route the benchmark circuit using the number and arrangement of logic blocks that we can fit.

We discuss risk mitigation for high overhead in Section 3.4.2.

1.4.3 Validation of improvements from architectural optimizations

To evaluate the benefits of utilizing architectural FPGA features of the host FPGA board, an alternate design can be created that implements the same functionality using only standard

Verilog. The size and timing of the two designs can then be compared to measure any efficiency gained by the design that uses special FPGA features.

For example, in select Xilinx boards, a lookup table can be used as a 32-bit shift register; the same function could be implemented in plain Verilog using multiplexers and flip-flops, but is expected to be slower and consume more area. The two equivalent circuits can be compiled separately in order to compare their resource use and limiting timing path.

Because the utilization of architectural features limit the design to specific board families, we will demonstrate that they enhance the circuit efficiency for the project in order to justify their use.

2 Technical Design

2.1 Design Alternatives

2.1.1 Implementation medium

The implementation medium for our circuit is a major decision impacting how accessible our circuit will be to researchers. The main criteria are cost, size, and ease of use. The lower the cost of the finished design to the researcher, the better. We must also ensure that the design is large enough to handle circuits the researchers wish to test. Finally, we want to make interfacing with the design's inputs and outputs as hassle-free as possible. The alternatives are as follows:

1. Custom integrated circuit
    - Faster, smaller and more power efficient.
    - High design and manufacturing costs.
    - Lengthy design and manufacturing time-line.
    - Once built, the parameters can't be modified without manufacturing a new chip.
    - Inputs and outputs will require extra circuitry to interface with the circuit.
2. Overlay FPGA implemented on commercial FPGA

- Researchers may already own a compatible FPGA so they won't need to purchase new hardware.
- Using an FPGA allows the researcher to implement a virtual circuit to interface with the overlay FPGA.
- Need to pick a FPGA platform to target:
  (a) Basic FPGA without using architecture-specific features
    - Circuit will work on most FPGAs from most vendors, so it is the most widely accessible.
    - Can't use architecture-specific features to save area and gain performance.
  (b) Xilinx Virtex 5 or newer
    - Lookup tables can be programmed directly as 32-bit shift registers.
    - Large FPGAs with 330,000 logic cells for Virtex 5[5] will fit a larger overlay circuit. Virtex 6 and 7 feature up to 760,000 and 2,000,000 logic cells respectively[6].
    - Higher cost for researchers.
  (c) Xilinx Spartan 6
    - Lookup tables can be programmed directly as 32-bit shift registers.
    - Smaller FPGA with 150,000 logic cells[6], allowing smaller overlay circuit.
    - Lower cost than Virtex 5.
  (d) Altera Stratix IV or newer
    - Higher cost than Xilinx Spartan FPGAs.
    - Large FPGAs with up to 820,000 logic cells for Stratix IV[7] and up to 952,000 for Stratix V[8].
    - Features a similar 32-bit shift register, but it isn't directly compatible with the Xilinx boards.

Developing a custom integrated circuit is far too costly and time consuming for the scope of this project. It was explored as an alternative to illustrate by contrast the necessity of targeting an existing FPGA. We have tentatively selected the Virtex 5 FPGA because our supervisor has numerous development boards and software licenses readily available. We also intend to use the 32-bit shift register functionality that is available in logic blocks in Virtex 5 and newer FPGAs. This feature will allow us to reduce the overhead of the overlay FPGA circuit by directly using the native FPGA's features. This selection limits the use of our circuit to modern Xilinx FPGAs including Spartan 6, Artix 7, Kintex 7, and Virtex 5, 6 and 7.

2.1.2 Configuration mechanism

Various parameters of our circuit, including the number, arrangement, and connectivity of the logic cells will be tunable. There are two alternatives for the implementation of the configuration mechanism:

1. Parameterized Verilog
   - Requires the user to modify values within the Verilog source.
   - Involves more complex Verilog code to accommodate flexible parameters.
2. Software front end to generate Verilog code
   - The generation software would be easier to use than modifying Verilog code.
   - Front end code will be easier to write than parameterized Verilog.
   - User may need to install a compiler or interpreter to run the software.

We have tentatively decided to use parameterized Verilog because we deemed that the complexity of the configuration in our present design concept does not warrant a front end code generator. If added features or a reevaluation of the design add configuration complexity, we may reconsider this, as this decision could be changed without revising a great deal of work.

2.2 Assessment of Proposed Design

The decision to take advantage of the custom 32-bit shift registers in implementing our design entails the following trade-offs versus using only basic Verilog logic:

- The design is more efficient area and timing-wise.
- The data describing the circuit (known as the *bitstream*) which is needed to program the design will be larger.
- The maximum size of the design will be limited by the amount of 32-bit shift registers available on the FPGA board, as opposed to the amount of flops.
- The design will be incompatible with boards that do not have custom 32-bit shift registers

We decided that the performance efficiency outweighed the negative aspects of the larger bitstream and limitation of implementation platforms. If the design is successful, adaptations can be made in the future to support the implementation of the design on more FPGA boards.

2.3 System-level overview

Figure 1 shows the high-level interactions of the components used in our project. The rectangles show software components, the circles show intermediate products or interfaces, and the parallelograms show primary inputs and outputs. The components we are producing directly are contained within the dotted lines.

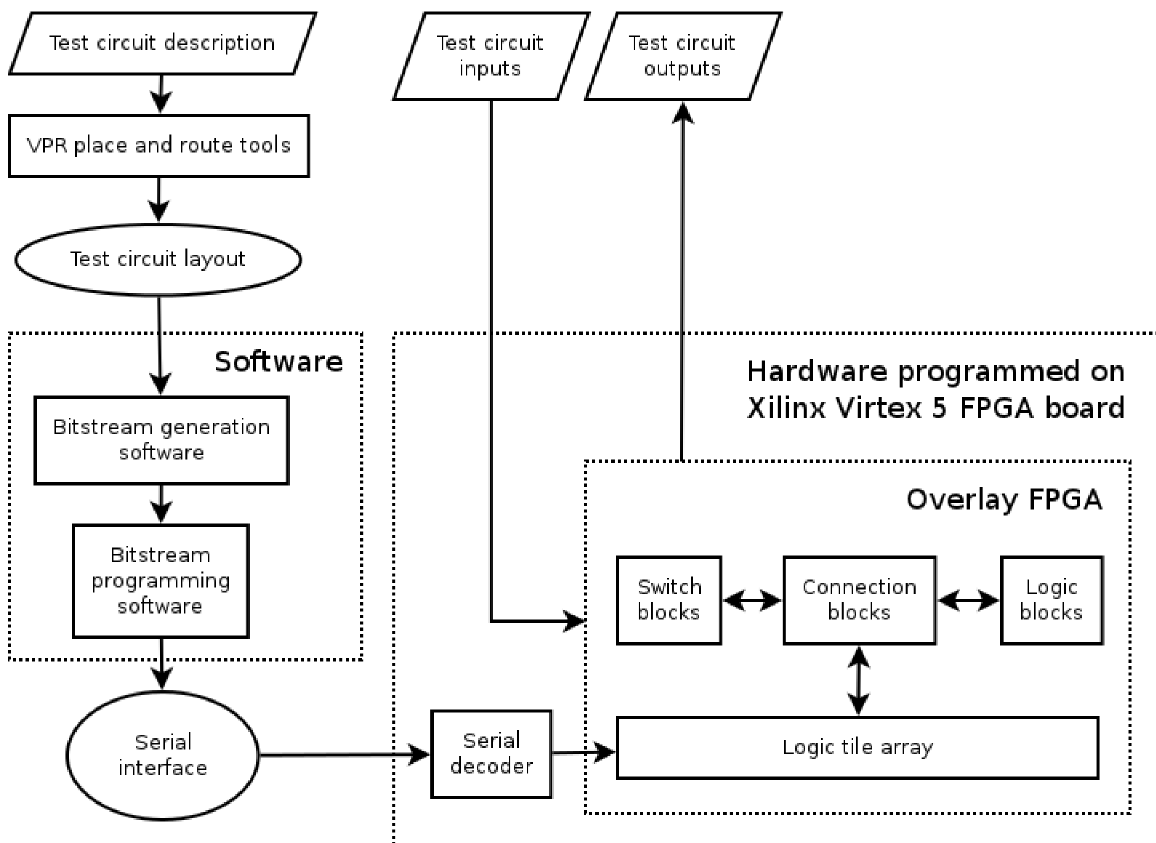

Figure 1: Module interaction

The finished project will consist of three main parts: The overlay FPGA, bitstream generation software, and interfacing of inputs/outputs of the overlay FPGA.

The *Overlay FPGA* is a Verilog HDL circuit implementation of the academic FPGA model which will be constructed as an overlay on a Xilinx FPGA board. The arrangement, size and connectivity of the overlay circuit will be controllable via parameters in the source Verilog. The overlay FPGA will consist of organized tiles of *logic block*, *connection block*, and *switch block* modules. Together, these modules will allow the overlay FPGA to implement different logic circuits.

Once built, the overlay FPGA can be configured to implement user-specified test circuits that have been placed and routed by VPR. This will be achieved by creating *bitstream generation and programming software* that will translate the VPR output circuit into a bitstream that the overlay FPGA will understand. The bitstream will then be injected into the overlay FPGA via a *serial interface*. The FPGA will receive and decode the bitstream into the appropriate test circuits on the overlay.

Finally, the circuits on the overlay FPGA can be tested for functionality by connecting devices to the *Test circuit inputs* and *Test circuit outputs* of the Overlay FPGA (e.g. Switches and LEDs).

2.4 Module-level descriptions

The *Overlay FPGA* will be composed of *Logic tiles*. The logic tiles make it easier to build a large overlay, and help keep the internal logic modules organized. Each logic tile will consist of one *logic block module*, two *connection block modules*, and one *switch block module*. Figure 2 shows the internal composition of a single logic tile.

The *Logic block module* consists of programmable look-up tables that perform all of the logical funcionality required by the circuit. A logic block module may be composed of multiple look-up tables, the number of which can be determined by verilog parameters.

The *Connection block* and *Switch block* modules regulate the routing of signals in the overlay. *Connection blocks* connect logic block signals to buses that run throughout the overlay. *Switch blocks* control the routing between buses when they cross each other.

The *Bitstream generation software* will be a program that translates VPR output circuits into a bitstream capable of programming the overlay FPGA directly. The program will consist of functions that parse the output from VPR and generate the appropriate bitstream from the parsed information.

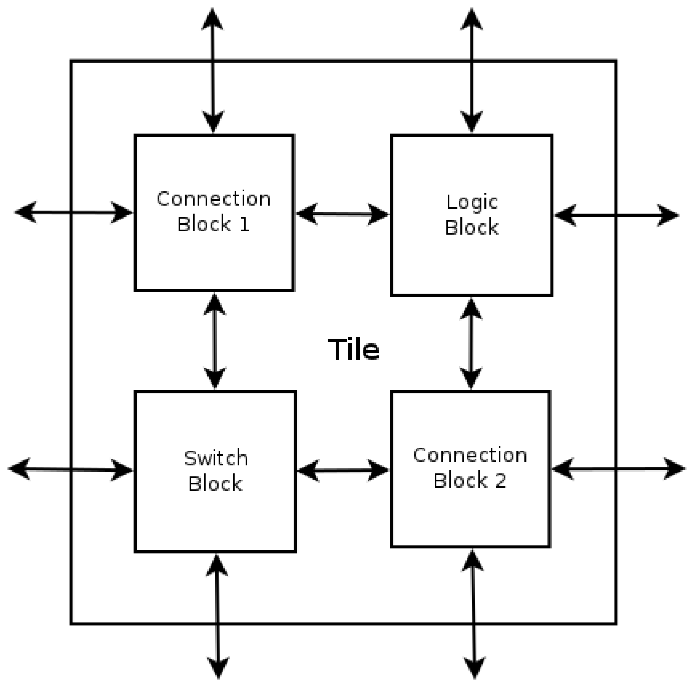

Figure 2: Logic connections within a tile

The *Bitstream programming software* will take the bitstream formed by the bitstream generator and format it for proper transmission over a serial interface.

The *Serial decoder* will be a circuit attached to the overlay FPGA that receives the bitstream sent through the serial interface. It will decode and extract the bitstream from the serial format, then inject it into the overlay circuit to configure the overlay.

3 Work plan

3.1 Work breakdown structure

The work breakdown structure is show in Table 1.

For each day allocated to a task (denoted "1d,") we allocated one hour of labour. This is based on seven hours of work per week devoted to the project for each group member.

Note that this work breakdown structure projects completion mid-February rather than mid-March to allow extra time for unforeseen engineering challenges.

WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to	% Complete
1	**Virtual FPGA circuit**	**Jul 21**	**Feb 15**	**358d**	**210d**		**0**		0
1.1	Basic logic element	Jul 21	Aug 5	16d	16d	194d	0	Keyi	100
1.2	UART interface	Jul 21	Oct 1	73d	73d	137d	0	Neil	95
1.3	Logic block	Jul 21	Aug 29	80d	40d		0	Keyi, Neil	100
1.4	Shift multiplexing	Aug 30	Sep 19	21d	21d		0	Keyi	100
1.5	Connection block	Sep 20	Oct 7	18d	18d		0	Neil	100
1.6	Switch block	Sep 20	Oct 7	18d	18d		0	Keyi	100
1.7	Logic tile	Oct 8	Oct 15	8d	8d		0	Keyi	100
1.8	Logic tile grid	Oct 16	Oct 31	16d	16d	107d	0	Neil	0
1.9	Logic tile boundary	Oct 16	Nov 30	46d	46d		0	Keyi	0
1.10	Basic tile optimization	Dec 16	Feb 15	62d	62d		0	Keyi	0
2	Software support	Dec 1	Dec 15	15d	15d		0	Neil	0
3	Test vector injection	Dec 1	Dec 31	31d	31d	46d	0	Keyi	0
4	Gathering results	Dec 16	Feb 15	62d	62d		0	Neil	0

Table 1: Work breakdown structure

Task Descriptions:

1. Virtual FPGA circuit: the hardware component of the project
   - 1.1 Basic logic element: write Verilog sub-circuit of basic LUT element.
   - 1.2 UART interface: write Verilog sub-circuit and a python program for bitstream programming via a serial cable.
   - 1.3 Logic block: write Verilog sub-circuit of bundled logic cells with full input crossbars.
   - 1.4 Shift multiplexing: write Verilog sub-circuits for efficient 2-layer and 3-layer multiplexers constructed from 32-bit shift registers.
   - 1.5 Connection block: write Verilog for the connection block module.
   - 1.6 Switch block: write Verilog for the switch block module.
   - 1.7 Logic tile: write Verilog module of assembling the logic block, connection blocks, and switch block in a tile ready for tessellation.
   - 1.8 Logic tile grid: assemble logic tiles into a grid to build the virtual FPGA.
   - 1.9 Logic tile boundary: handle input and output to and from the virtual FPGA.
   - 1.10 Basic tile optimization: tune the logic cell and routing for performance and area.
2. Software support: write software to convert a VPR circuit to a bitstream for our circuit.
3. Test vector injection: create a mechanism for test inputs to be transferred alongside and executed on the circuit, and for the results to be retrieved.
4. Gathering results: Study and document the area overhead and performance of the circuits using different configurations.

3.2 Gantt chart

Figure 3 shows the proposed project plan corresponding to the work breakdown structure outlined in Section 3.1.

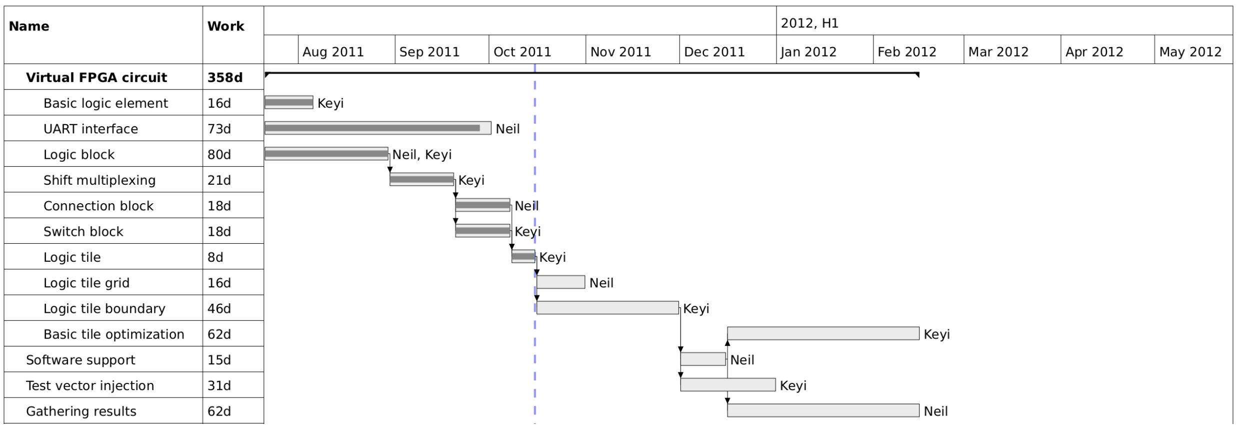

Figure 3: Gantt chart showing project plan

3.3 Financial plan

The expenses of our project are documented in Table 2. No additional financing is requested.

Item	Source	Unit cost	Quantity	Total
FPGA development board	Supervisor	$750.00	2	$0.00
ISE software licenses	Supervisor		2	$0.00
Personal computer	Students		2	$0.00
Serial cable	Students	$5.00	2	$10.00
USB serial adapter	Students	$2.00	2	$4.00
Total				$14.00

Table 2: Project expenses

3.4 Feasibility Assessment

3.4.1 Skills and Resources

- Required Software

  - ABC synthesis system to produce input for VPR - available online[5]
  - VPR placement and routing to produce our input data - available online[6]
  - FPGA vendor tools to implement our circuit: Xilinx ISE - licenses through supervisor

- Required Hardware
  - FPGA development board - currently using Xilinx Virtex 5 from supervisor; Spartan 6 boards are also available from supervisor.
  - Host computer to program the physical and overlay FPGAs - personal laptops
  - Serial cable and USB adapter to interface with overlay FPGA - available at electronics stores

- Required Skills and Knowledge
  - Verilog circuit design skills - gained from previous coursework and projects
  - Knowledge of FPGA architecture - consulting with supervisor, studying recommended papers, software tools, and online resources
  - VPR interfacing - consulting with graduate students who are working on VPR

3.4.2 Risk Assessment

Overlay FPGA implementation overhead

- Implementing the overlay's logic cells and interconnect on a physical FPGA will consume more area than the area consumed by an equivalent amount of logic cells and interconnect on the physical FPGA.
- If the overhead is too high, then we may not be able to fit enough logic cells for our benchmark circuits.
- For example, if a benchmark circuit consumes 5% of a normal FPGA and the overlay has a 20x overhead, then the benchmark circuit may not fit on the overlay FPGA.
- Risk Mitigation strategies:
  - Use a larger, more costly FPGA.

[5]ABC can be downloaded from: http://www.eecs.berkeley.edu/~alanmi/abc/

[6]VPR 5.0.2 can be downloaded from: http://www.eecg.utoronto.ca/vpr/

  - Look for more architectural features in the physical FPGA that can be exploited to reduce the overhead of the overlay FPGA circuit.
  - Use a smaller benchmark circuit for the proof of concept.

Unbalanced timing

- Logic cells in the overlay FPGA would ideally be arranged in a perfect grid, but we expect that the placement algorithm in ISE will not produce this arrangement on the physical FPGA.
- This means that the timing delay of two different cells to their respective "right" neighbours may differ.
- If this imbalance is too big, then the performance of the benchmark circuits may not be satisfactory.
- Risk Mitigation: Constrain a "tile" containing only one logic block to a rectangular shape, then tessellate it to form a grid, producing a more balanced design.

References

[1] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2007.

[2] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *International Workshop on Field Programmable Logic and Applications*, 1997.

[3] V. Betz. (2011, Sep.) "A Utility to Convert a VPR netlist to Quartus format". [Accessed: September 17, 2011]. [Online]. Available: http://www.eecg.toronto.edu/~vaughn/vpr/download.html

[4] J. H. Anderson, Q. Wang, and C. Ravishankar, "Raising FPGA logic density through synthesis-inspired architecture," 2010.

[5] Xilinx Inc. (2011, Sep.) "Virtex-5 FPGA Family". [Accessed: September 17, 2011]. [Online]. Available: http://www.xilinx.com/products/virtex5/

[6] ——. (2011, Sep.) "Xilinx FPGAs Offer High-Performance, Low-Power, Low-Cost Silicon Devices". [Accessed: September 17, 2011]. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/index.htm

[7] Altera Corporation. (2011, Sep.) "Stratix IV FPGA: High Density, High Performance AND Low Power". [Accessed: September 17, 2011]. [Online]. Available: http://www.altera.com/products/devices/stratix-fpgas/stratix-iv/stxiv-index.jsp

[8] ——. (2011, Sep.) "Stratix V FPGAs: Built for Bandwidth". [Accessed: September 17, 2011]. [Online]. Available: http://www.altera.com/products/devices/stratix-fpgas/stratix-v/stxv-index.jsp

Appendices

A Student-Supervisor Agreement

Our signatures below indicate that we have read and understood the following agreement, and that all parties will do their best to live up to the word as well as the spirit of it.

We agree to meet at least once every two weeks for at least half an hour to discuss progress, plans, and problems that have arisen. Before each meeting, the group will prepare a brief progress report that will form the basis for the discussions at the meeting.

If a meeting has to be canceled by the supervisor, he should advise the group as early as possible. If a student cannot attend a meeting, she/he should advise members of the group as well as the supervisor as early as possible.

Both the supervisor and the students will:

- Inform themselves of the course expectations and grading procedure.

The supervisor will:

- Provide regular guidance, mentoring, and support for his design project group,
- Take an active role in evaluating the work and performance of the students' by completing the supervisor's portion of the grading forms for each course deliverable expediently.
- Return a photocopy of the completed grading evaluation forms to the appropriate section administrator in a timely fashion.
- Be aware of the aims and processes of the course as outlined in the Supervisor's Almanac.

Jason Anderson Date

Neil Isaac Date

Keyi Shi Date

B Attribution Table

Section	Neil	Keyi
Executive Summary	ET	RD
Background and Motivation	RD	ET
Project Goal	RD	ET
Project Requirements	RD	ET
Validation and Acceptance Tests	MR, ET	RD
Design Alternatives	RS, MR	RD
Assessment of Proposed Design	ET	RD
System-level Overview		
Module-level Overview		
Work Breakdown Structure	RD	ET
Gantt Chart	RD	RS, ET
Financial Plan	RD	ET
Feasibility Assessment	ET	RD
All	FP, CM	FP

Abbreviation Codes

		"All" row entries	
RS	research and information		
RD	wrote first draft	FP	final read through
MR	major revision	CM	compiling elements
ET	editing spelling grammar and expression	OR	other (describe OR1, OR2, etc)

Signatures

By signing below, you verify that you have read the attribution table and agree that it accurately reflects your contribution to this document.

Neil Isaac Date Keyi Shi Date