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

Virtual FPGA fabrics
Implementation of a Virtual FPGA Architecture

Individual Progress Report

January 17, 2012

Keyi Shi keyi.shi@utoronto.ca

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

ECE496Y Individual Progress Report Evaluation Form

Student				
	Project ID: 2011017	**Project Title:** Virtual FPGA Fabrics		
	Student Name: Keyi Shi		**Supervisor:** Jason Anderson	
	Section # 7	**Administrator:** Ross Gillett		
	Contact hours per month with supervisor :	4	**Optimal # contact hours per month:**	4

Administrator's Evaluation (Provide a rating for each section)	Excellent	Good	Adequate	Marginal	Poor/Missing	**Comments** *(also see comments in report)*
Progress: execution of work plan, significance of reported work Problems(circle): *claims not justified, status of tasks unclear*						
Method: Sound engineering practices reflected in actions, decisions, and testing. Changes reflect good project management. *Problems(circle): actions, decisions, testing*						
Reported Results: Quality and completeness of documentation *Problems(circle): inadequate documentation or analysis, inadequate test results, completion of task not verifiable*						
Overall Quality of Document						
Presentation (circle problem areas) *Grammar, spelling, clarity, writing style, use of figures & tables*						
Content (circle problem areas) *Organization, substance, references, Gantt chart incomplete*						
Other problems (circle): *late submission, other (specify)*						

Administrator's grade (/10):	**Section average (/10):**	**Administrator's signature:**

Note to supervisors: Your evaluation was done online. There is no need to return anything to the administrator.

Executive Summary

The project overall has progressed smoothly to the overlay building stage, and we are now attempting communicate between the overlay and a VPR-generated bitstream.

I have worked exclusively on the hardware side of this project, building and testing the basic building blocks of the overlay, as well as generating the overlay itself. My current goals are to work with Neil to get the overlay and VPR bitstream generator working together, as this is the next crucial step in the project.

We have extended our schedule and deadlines from our initial proposal. We intentionally budgeted a large amount of spare time at the end of the term to be used on possible complications and bugs in the project, as the nature of this projects requires extensive debugging in order to get all of the components to function together.

Our overlay design turned out to be quite resource consuming, and as it stands we cannot fit as large a virtual FPGA onto a commercial board as we would have liked. This was a risk identified in the proposal, but we believe we can still produce an overlay large enough to house meaningful circuits for demonstration and proof-of-concept purposes. If there is extra time left at the end of the project, we may attempt to improve the efficiency of our overlay design.

1 Individual Progress and Contributions

The following table is an overview of my contributions to the project. An updated Gantt chart may be found in Appendix A.

Task title	**Category**	**Status**	**Original date**	**New date**
1. Basic virtual overlay modules	hardware	completed	Aug-Oct 15	Aug-Nov 2
2. Complete overlay design	hardware	debugging	Oct 15-Jan 15	Nov2-Jan 20
3. Stress-testing and improvement	testing	not started	Dec 15-Apr 1	Jan 15-Apr 1

2 Information on individual Milestones

1. Basic virtual overlay modules
Category: hardware
Original date: August - October 15
New date: August - November 2

Responsibility: Keyi, Neil
I wrote the basic logic element, multiplexer, and switch block.

Status: completed: individual modules are tested and functional according to planned requirements.

Actions

- I wrote the logic element, multiplexer, and switch block modules.
- I wrote multi-layer shift-multiplexers for different numbers of inputs.
- All modules needed to be programmable via a scan chain.

Decisions

- We used the native 32-bit shift registers available on the Virtex 5 board as the building block for our basic modules. The shift registers allowed for the use of a scan chain set up to program all components of the overlay.
- The different modules required multiplexers of different sizes. In order to simplify coding and save on area, I created a master multiplexer module that adjusted its size to suit the number of inputs required.
- To keep our project simple and the overhead area down, we decided against giving our switchblock full connectivity.
- Each module was coded to allow the user to define the number of inputs and outputs from logic blocks, as well as the width of the bus interconnecting the modules.

Testing, verification and results

- Each module was tested using manually created bitstream files to program different functionalities to ensure correct performance. Functionality was observed by using on-board switches and LEDs

2. Complete overlay design

Category: hardware

Original date: October 15 - January 15

New date: November 2 - January 20

Responsibility: Keyi

Status: debugging

The design is complete, but not yet functional with the VPR-generated bitstream. Further testing is required to determine the source of the problems.

Actions

- I wrote the overlay grid boundary and interconnects between tiles.
- I connected the appropriate signals to the inputs and outputs of the overlay.

Decisions

- In order to keep track of interconnects and ensure they were placed correctly, we utilized a 2D coordinate system for the overlay grid.
- We decided to re-use the existing connection block and switch block modules for the boundaries of the overlay in order to save time and keep the code simple.
- Because of the structure of our connection blocks, the number of outputs per connection block is restricted indirectly by the number of basic logic elements per logic block. We decided this was an acceptable restriction.

Testing, verification and results

- Testing is still in progress, and will be more complex than previous hardware testing procedures, as the overlay is much more complex and multi-layered.
- The resources taken by single tile generated with different parameters has been tested and documented, the results of which are available in Appendix C. The data shows us just how resource-intensive a single tile is currently.

3. Stress-testing and improvement *Category:* testing, hardware *Original date:* December 15 - April *New date:* January 15 - April
Responsibility: Keyi, Neil I will focus on the verification and streamlining of the overlay itself.
Status: not started We will begin extensive verification and improvement once the overlay and bitstream generator work together.
Actions n/a
Decisions n/a
Testing, verification and results • Make sure bitstream programs expected circuits onto overlay. • Make sure signal routing is always correct. • Make sure size of overlay is sufficient. • Look into the use of Clos networks to reduce the size of multiplexers.

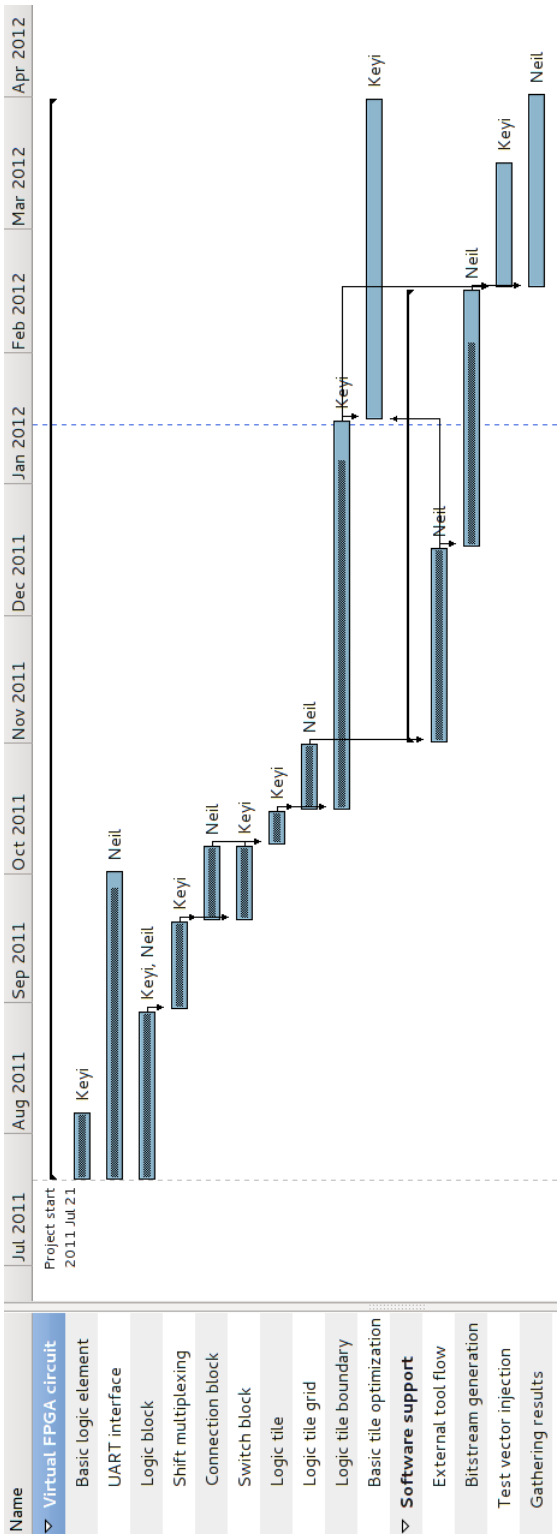
3 Progress Assessment

The majority of the hardware for the project has been coded, and now require extensive testing and debugging to get it working with the bitstream generator. Once that is done, there are possible improvements to be made to the design to improve the overlay area efficiency on the board.

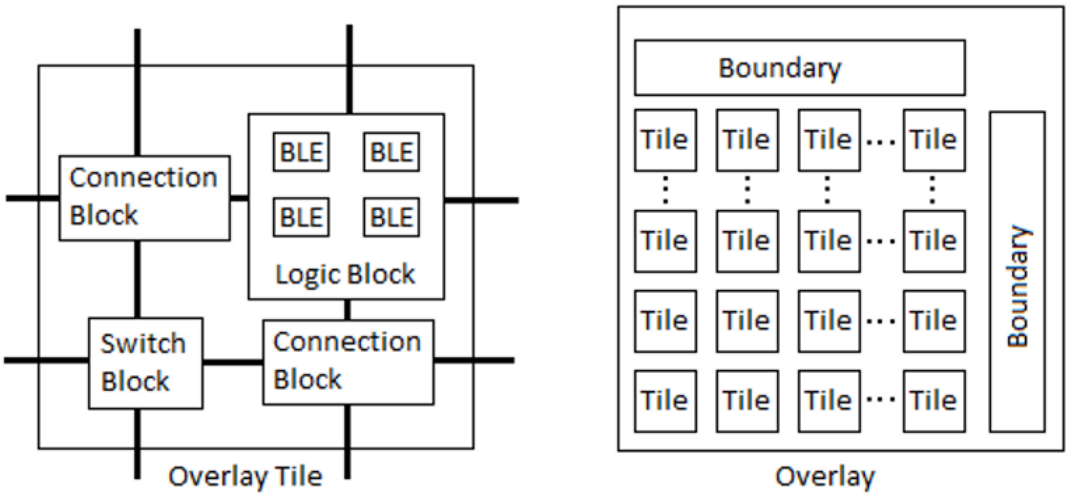
Overall, the project is progressing well. We have extended our schedule from our initial plans in the proposal, but it was an extremely optimistic schedule and we did budget extra time in the end to allow extensions. As such, we do believe that the project is still well on its way to completion before the design fair.

Appendices

A Gantt Chart



B Overlay Schematic



The above schematic illustrates the organization of both the structure of a single tile in the overlay, as well as the overlay itself. Each tile is composed of 3 different types of modules: logic block, connection block, and switch block. The tiles are then tessellated together into the overlay, with extra boundaries added on two sides to make it symmetrical.

At the moment, the logic block, connection block, and switch blocks are all complete. The overlay itself still requires testing and debugging to ensure functionality.

C Resource Usage Per Tile

Bus Width	Logic Block Input	Logic Block Output	SRL/Tile	Tiles/Board
2	16	4	160	216
3	16	4	200	172
2	24	4	264	130
3	24	4	320	108
4	24	4	328	105
5	24	4	360	96
5	12	4	216	160
5	16	4	232	148
5	20	4	272	127
5	16	8	408	84
5	20	8	568	60
5	24	8	632	54
3	24	8	592	58
4	24	8	600	57

The table above shows the number of shift registers used in a single tile in the overlay for different parameters chosen. The numbers were acquired from Xilinx Xpower Analyzer after a single tile with the specified parameter was implemented.

The numbers have a large range depending on the parameter settings, with the most significant parameter being the number of outputs (basic logic elements) per logic block. The board we're using contains approximately 35,000 shift registers of the appropriate type. This means that we can fit only around 200 tiles on a single board with the lowest parameter settings. This number is significantly less than our initial goal, but still enough to make the project a successful proof-of-concept. If we have enough time after the overlay and the bitstream are debugged and tested, we may look to reducing the number of SRL/tile used to boost the maximum possible size of the overlay.