**Final Report: Narrative** Nicholas Fiacco

The purpose of this project was to solve a shortest-path problem over a topographical map composed of a 16x16 grid of square units. Each square, which represented a plot of land, had a pre-determined degree of difficulty posed to the hypothetical traveller. This delay represented the time necessary to cross such a square unit, and was factored into the overall shortest-path finding algorithm. While traversing this grid, the traveller was assumed to only be able to move in the cardinal directions: north, east, south, or west. This, for each cell not along the boundary of the grid, there were four potential transitions that could be made upon the completion of the travel time. There are many known solutions to the afore-mentioned shortest path problem, including algorithms known as breadth-first search (BFS), A\*, and Dijkstra’s. However, BFS suffers from the disadvantage of being unable to handle differently weighted terrain delays, while all three are inefficient in terms of their computer resource consumption and lengthy completion times. They are also rather complex mathematical algorithms, which makes adjustments to future preferences difficult to accomplish. Instead, this project successfully utilized individual parallel processing cells to represent the square patches on the grid. By having each cell handle its own computations upon being activated, significant processing resources were saved as well as speeding up the computation time.

In order to accomplish this goal of finding the shortest path using a grid of parallel processing cells, several layers of design were utilized. The unit cell is the lowest layer of the architecture, and handles several process: programming a delay cost into the cell, determining whether the cell is active or not, storing the direction from which the activate signal came, storing whether the cell is on the shortest path, and decrementing the delay value upon activation until zero is reached. In order to program a delay cost, or the time needed to traverse a particular cell, an 8-bit register was used to store values up to 255. These values came from the top level architecture. The enable signal for this register was the result of ORing a “program” signal from the top level with the AND of an “active” signal and a “slow clock.” The active signal comes from the output of a DFF storing a bit telling whether the cell has been activated yet. Its input is the result of ORing activate signals from all four directions with a “start” signal from the top level, and is enabled by the NOT of its own output. This active signal is also used as the load signal for the 2-bit previous location register, which stored the encoded value of which direction the first active signal came from, favoring north over east, east over south, and south over west. Decrementing the delay value was achieved by wiring the output of the delay register to a subtractor, and subsequently muxing this signal with the input delay cost from the top level, using the program signal as the select bit. Upon the cell being activated and receiving a slow clock tick, the decremented value would be stored back into the register. When the delay value reached zero, which was determined by a comparator, an active signal was sent to a maximum of four adjacent cells, and a done signal was sent to the top level. Storing whether the cell is on the shortest path was accomplished by implementing a DFF with its input tied high and a load value connected to a signal from the top-level.

The top level consisted of an FSM controller, which received 8-bit data buses from a serial receiver, connected to a computer. This controller then sent enable signals to decoders that sent signals to every cell based on the input address. The top-level controller also handled the load signals for top level registers utilized in the translation of the 8-bit data blocks. In programming the start and stop cell, the FSM would also send load signals to registers with the purpose of storing the addresses of these cells from the receiver data. A current cell register would also be loaded with the receiver data at the same time as the stop cell. The output of the stop cell register was used as the select bit input to a 256x1 8-bit mux, which wired the done signal of the stop cell to the FSM. Likewise, the start cell register’s output was used as the input to a decoder, which sent a start signal to the correct start bit during a designated FSM state. Once the done signal was received by the FSM, the trace back stage would occur. The output of the current cell register, which would have already had the location of the stop cell programmed in, would be used as the select bit to a 256x1 8-bit mux connected to each cells previous location register output. This data would then be used as the select bit to 4x1 8-bit mux that routed the address of one of the current cell’s adjacent cells into the current cell register’s input. These addresses can be calculated through simple arithmetic using subtractor and adder components. During the trace back stage, the load signal for the current cell register is constantly high, and the register keeps getting loaded with the previous location until the start bit is reached, found by comparing the current cell register’s data with the start bit register’s data. Additionally, the current bit register’s data was used as the address to a decoder, which output a load signal to the corresponding unit cell’s path DFF. A VGA controller was utilized to translate the delay values of each cell into colors to be displayed on a monitor. This was accomplished by sending addresses to the cell grid one by one, which would then send the corresponding delay value along with whether it was on the path for the cell at that address. The top level would also send a bit telling whether that cell was the start or stop cell.

In order to achieve the application of this architecture, the VHDL hardware description language was used to specify the logic, and a Xilinx Nexys3 board was used to implement this design. The code was written to reflect actual component design as much as possible, with the unit cells being built from smaller components such as registers and comparators described in separate files. Likewise, the cell grid and top level were described as combinations of various low level and middle level components. Creating the cell grid was accomplished through the use of generate statements, with special cases to handle boundary situations. One of the major challenges in the implementation of this project was not the actual algorithm itself, since the parallel processing design serves to immensely simplify the process. Instead, the biggest obstacle was creating a design that did not exceed the limitations of the board. Several design choices helped this architecture to achieve this goal. First, the unit cell stored the previous location as an encoded value rather than one-hot, in order to cut down the register usage per cell from 4-bit to 2-bit. Additionally, rather than using two 8-bit registers for the delay and its decremented value, it was possible to use only one register, with combinational logic used to determine the load signal. Once this size constraint was met, the rest of the testing went rather smoothly.

One of the highlights of this project was the fact that it was a hardware implementation of a solution that is normally achieved through software. This demonstrated that expediting processing speed cannot only be achieved through modifications to an algorithm, but by thinking outside the box and following a different approach altogether. This new perspective was something that truly appealed to me. It also helped significantly that the structure of the VGA controller was supplied, since designing this component could have made an otherwise interesting project tedious and time-consuming. This cellular automata architecture is useful for any sort of situation in which adjacent cells are interacting with each other, and changing their value in response to their neighbors. An immediate example that comes to mind is the Game of Life, whose visual representation strongly resembles the cell grid made in this project. Physics simulations could also be implemented with such a design, in which moving particles reacts upon interacting with other particles.