Project Plan – ECE 520

/10

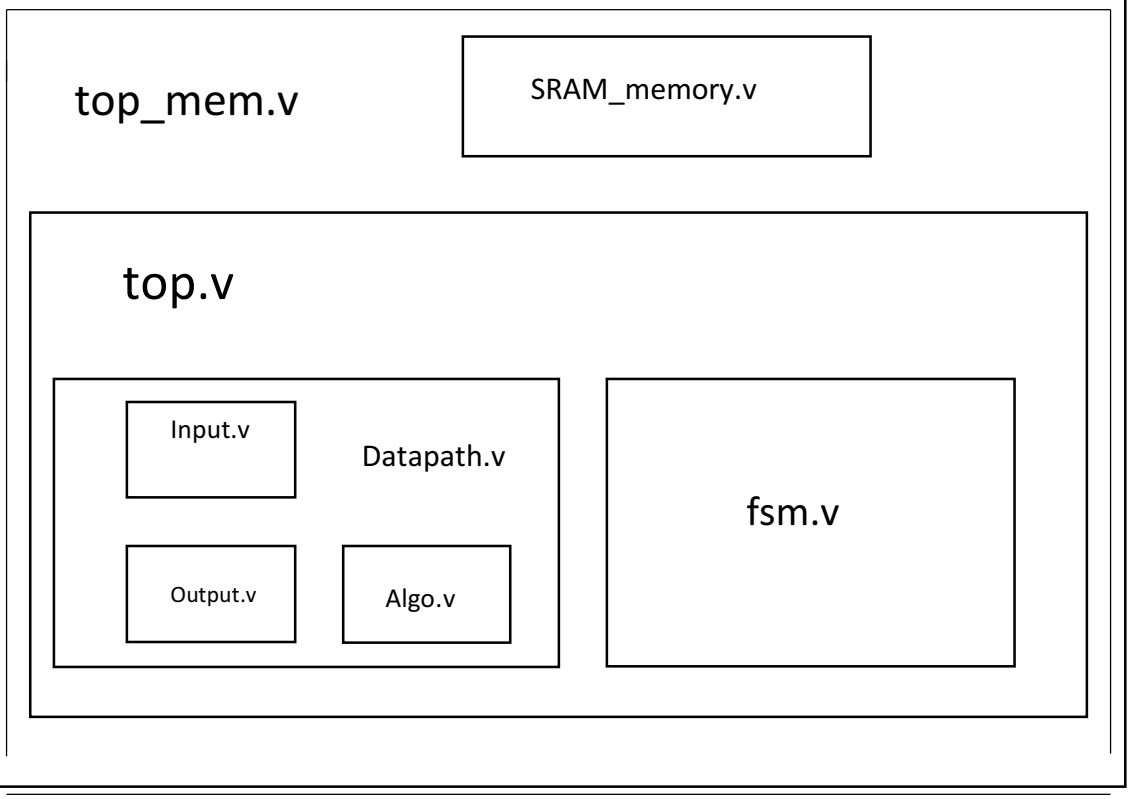
Project Responsibilities: Design and Synthesis of Hardware Accelerator for Bellman-Ford Algorithm
StudentID: 200108628
Name: Parth Bhogate

Summary Risk Plan: 1. In case of multiple shortest paths existing in the graph, logic needs to be designed to choose the suitable shortest path from the the available paths. The count field in the Work SRAM is used to store the data needed to determine the best path.
2. If the number of daughter nodes is greater than 7, the data for a node spans multiple lines in the Graph SRAM. In this case, the circuitry for prefetch needs to be carefully designed.
3. In synthesis, the script files need to be properly designed to account for delays in interfacing with memory modules.

Schedule:
18th Oct: Complete hardware design
25th Oct: Complete Verilog Code for design
2th November: Start synthesis runs
9th November: Compile results and report

Brief Description of Mode of operation, including selected algorithms:
Bellman-Ford Algorithm. Operation will be controlled by a Finite State Machine Controller.

High level sketch: Describes modules that will be implemented in Verilog code.



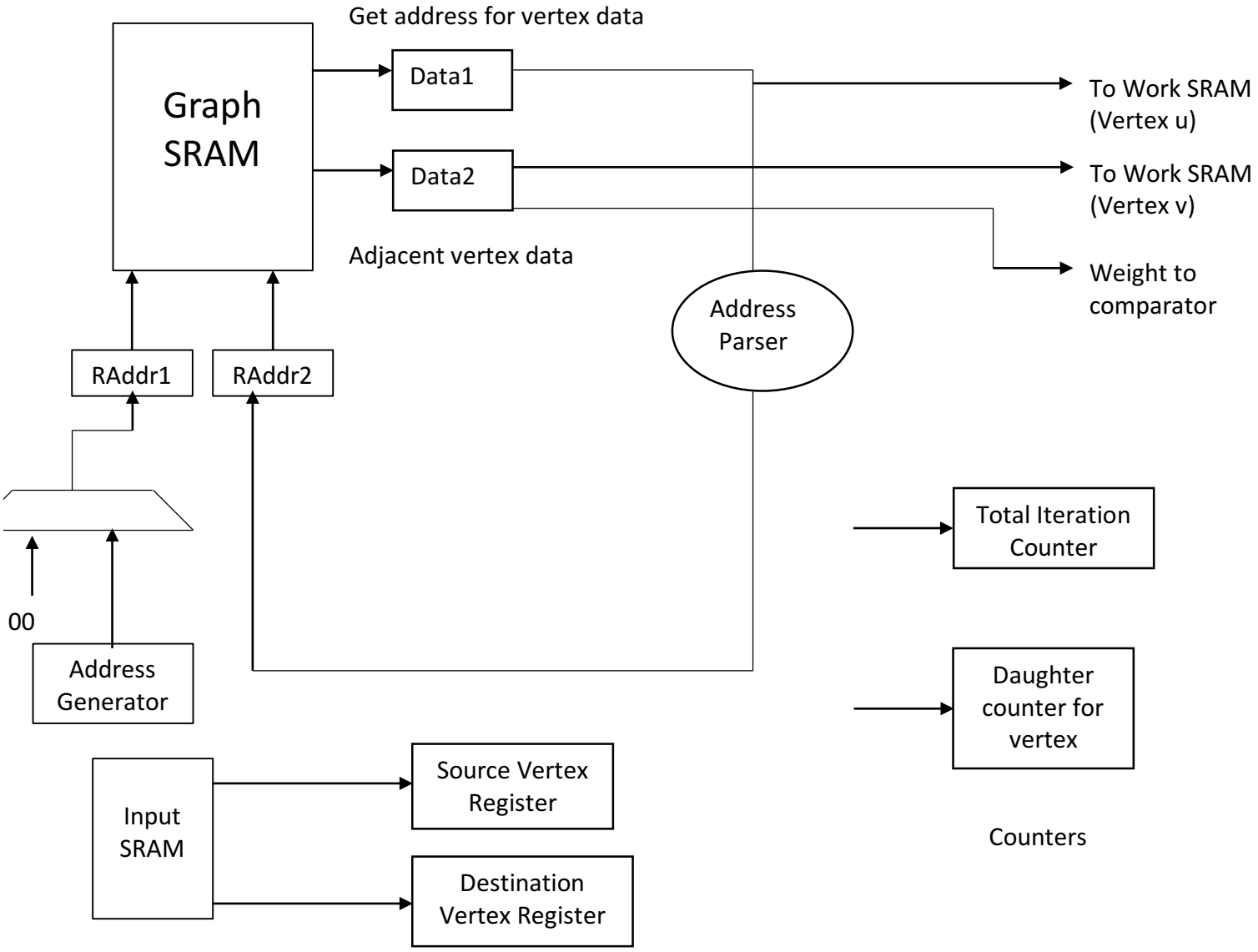
Design and Synthesis of a Hardware Accelerator for Bellman-Ford Algorithm

---

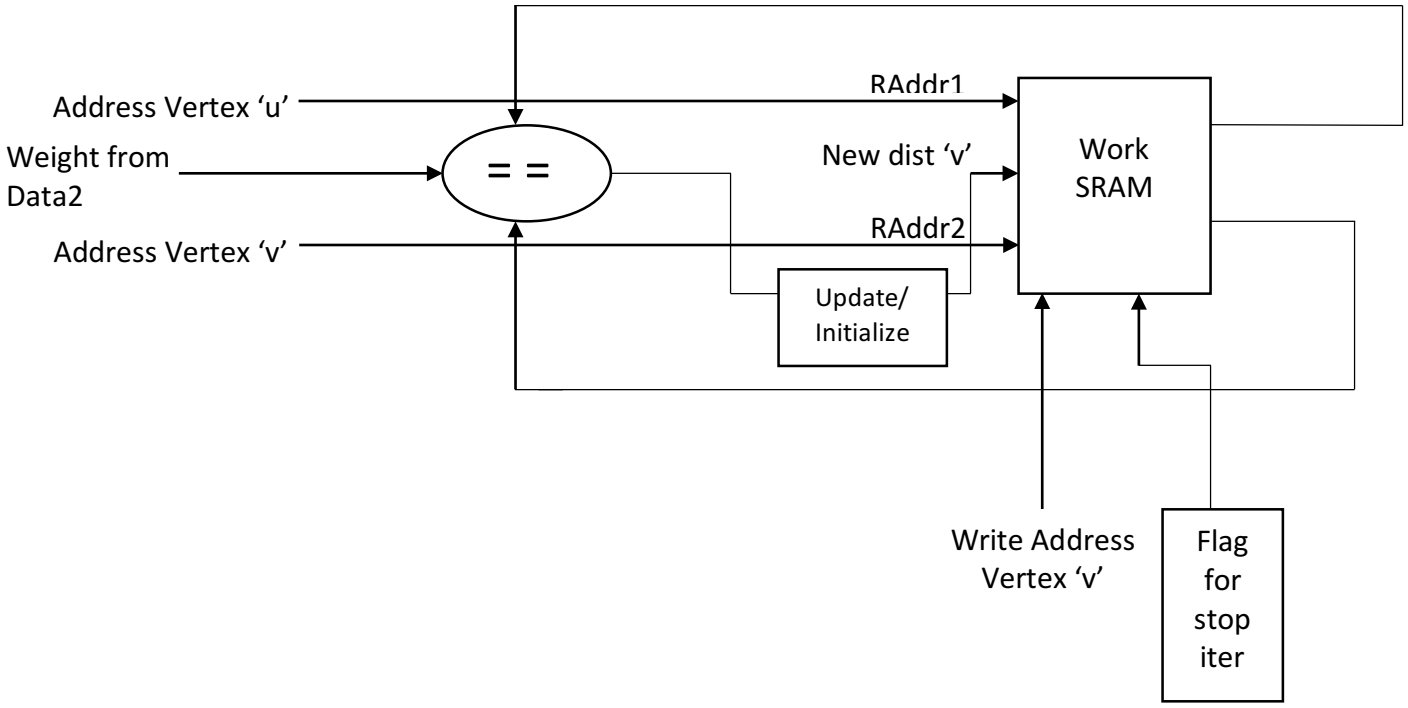
Block Diagram

Different parts of the design are described:

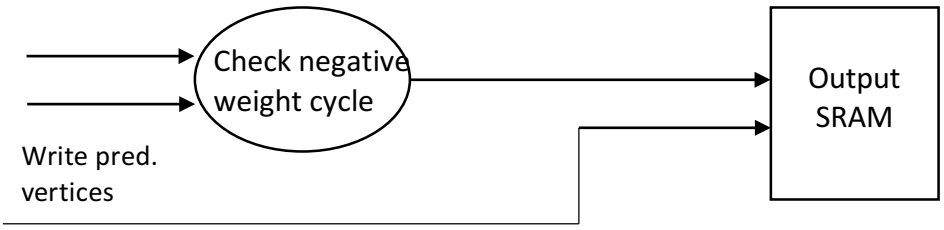
Reading from Graph SRAM



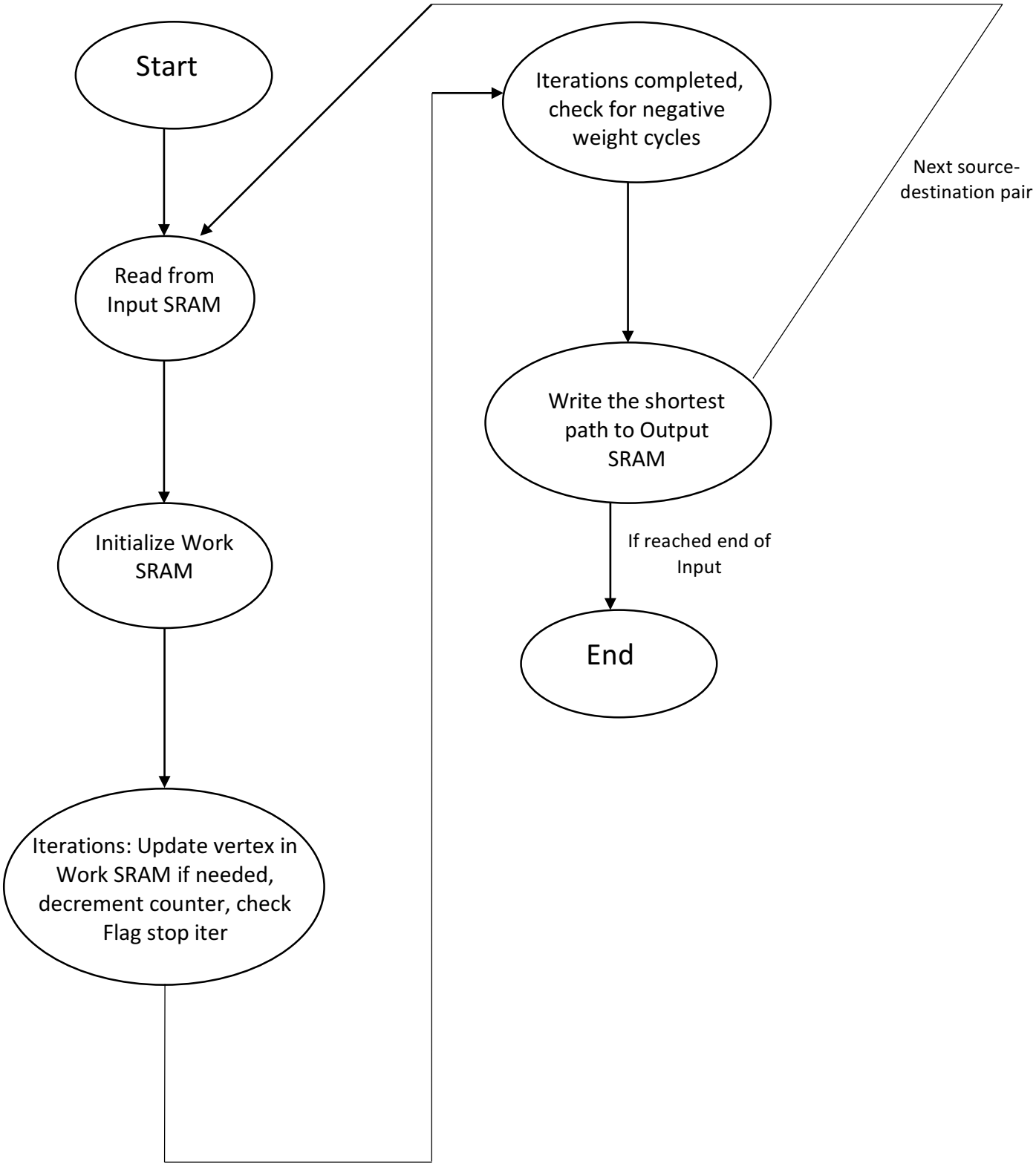
Work SRAM Functionality:



Read the shortest path and write to Output SRAM:



Controller State Sequence:



Format of data stored in Work SRAM:

Address (not stored in SRAM)	Distance of Vertex (16 bits)	Predecessor Vertex (8 bit)	Count for number of previous vertices
0000h			
0001h			
0002h			

This is the format of data storage in the Work SRAM. The data for each node will be addressed according to the vertex number. For example, the values for vertex 3 will be stored at address 03h in the SRAM. The count field in Work SRAM stores the number of previous vertices for a particular vertex. This is to be used to determine the shortest path which has the minimum number of vertices from source to destination.

Description of Logic Blocks and Functionality:

The various steps through which the design goes are as follows:
1. In the initialization step, the source vertex number is read from Input SRAM and stored in the Source Vertex Register. Also, destination vertex number is stored in Destination Vertex Register. Then the Graph SRAM is read and the Work SRAM is initialized with 0 distance for Source Vertex and infinity distance for all other vertices. Also, the predecessor nodes are initialized to undefined.
2. After initialization, the actual algorithm iterations are started. In each iteration step, one data line is read from Graph SRAM and the address is parsed to get the daughter vertex number and distance. From the Work SRAM, the current distances of 'u' and 'v' are read, weight 'w' is added and compared in the Comparator block. If the new distance is lesser than the current distance of 'v', then that particular entry in Work SRAM is updated. The iteration is executed for 'Number of Vertices – 1' number of times.
A prefetch logic is implemented to prevent the wastage of one cycle to read the data for a particular vertex from Graph SRAM. While the last daughter vertex of one iteration is being computed, the next vertex and its data address in Graph SRAM is fetched so that 1 cycle is not wasted in reading the vertex data. This prefetch mechanism needs to be handled when the vertex data is on multiple lines.
3. The Flag stop iter is used to stop the iterations if the graph vertex distances do not update from the previous iteration. This saves extra iterations during which the distances will not be updated. The Flag is loaded with a value of 1 at the start of each iteration, and it is changed to 0 if any vertex distance is updated in the Work SRAM. At the end of each iteration, the Flag is checked to see whether it is 0 or 1. If it is 1, the iterations are stopped.
4. After the end of 'Number of Vertices – 1' iterations or if the Flag is 1, the graph distances are used to check for the presence of negative-weight cycle in the graph. After check, the Work SRAM is read from destination to source using the predecessor vertex field in SRAM. The vertices read out from Work SRAM are written to the Output SRAM.

Risk Assessment:

1. In case of multiple shortest paths existing in the graph, logic needs to be designed to choose the suitable shortest path from the the available paths. The count field in the Work SRAM is used to store the data needed to determine the best path.
2. In synthesis, the script files need to be properly designed to account for delays in interfacing with memory modules.
3. Multiple lines of data for a vertex: If the number of daughter nodes is greater than 7, the data for a node spans multiple lines in the Graph SRAM. In this case, the circuitry for prefetch needs to be carefully designed.
4. In the design, datapath and control path needs to be clearly separated. The clock cycle usage needs to be analyzed to ensure that data is latched properly at the end of each cycle. Correct status and control lines need to generated.

Milestone Chart:

The anticipated dates for completing stages of the project are:

18th October: Complete the hardware design and hand simulate to determine clock cycle behavior.

20th October: Begin Verilog Code. Simulate and verify individual modules while designing.

25th October: Complete the Verilog Code for the design. Start verifications of all modules connected together.

2th November: Complete verification of the entire design, start synthesis runs.

5th November: Iterate over the Synthesis runs to generate best possible area and clock cycle time. Go back and make suitable changes in RTL to generate a better design.

9th November: Complete synthesis of the project. Start compiling area and performance reports.

11th November: Project final submission.