Prof. M. Mansour

American University of Beirut
Department of Electrical and Computer Engineering
EECE 420 – Digital Systems Design II

Project
(Vector Norm-2 List Processor)

In this project, you will implement a list processor that computes the squared-norm of a complex vector. Recall that the squared-norm of an n-dimensional complex vector $\mathbf{V} = [v_0, v_1, \cdots, v_{n-1}]$, where $v_i = x_i + iy_i$, is

$$\|\mathbf{V}\|^2 = \sum_{i=0}^{n-1}\left(x_i^2 + y_i^2\right)$$

where $n \le 128$ is the vector length. The quantities v_i are complex, while x_i, y_i are fixed-point numbers. You are told that these x_i, y_i values are stored in memory in the form of a double linked-list data structure as shown in Figure 1. Each node in the list contains **four** entries: one entry that points to the next node in the list (pointer to *first* entry in that node), one entry that points to the previous node in the list (pointer to *second* entry in that node), one entry that holds the x_i value, and one entry that holds the y_i value. The first node in the list is stored at address 0 in memory, while the last node is stored at address A.

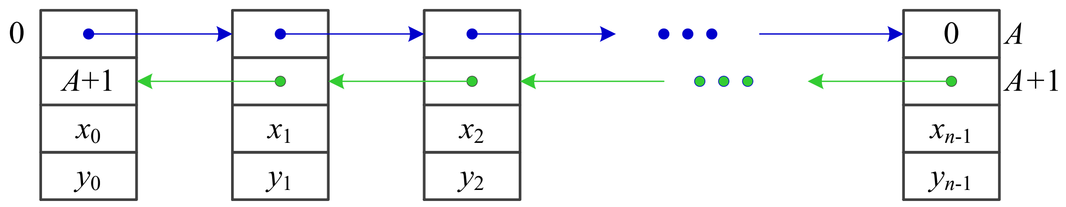

Figure 1: Double-linked list data structure

Assume memory has two read ports. Each word (entry) in memory is 24 bits wide. These 24 bits are divided into three fields, one storing the sign (s), one storing the mantissa (m), and one storing the exponent (e) as follows:

Sign (s)	Mantissa (m)	Exponent (e)
1 bit	15 bits	8 bits

The value of the corresponding number is $|x| = (-1)^s \times 0 \cdot m \times 2^e$. The mantissa always stores the normalized value of the number.

There are 512 words in memory. Hence memory can contain at most 512/4 = 128 nodes, and therefore the maximum length of vectors supported in this design is 128. The maximum value of address A is 508.

You are given a library of logic components whose propagation delay is summarized in the table below.

Component	Delay
Simple logic gates	0.4 ns
n-bit register	Clk-to-Q = 0.4 ns setup time = 0.4 ns
n-bit 2:1 mux	0.9 ns
n-bit adder	(2 log(n) + 1.5) ns
n-bit x n-bit multiplier	(0.9 n^2 + 1.5) ns
memory	12 ns (asynchronous read) 8 ns (synchronous write)
n-bit comparator	0.4 log(n) ns

You are asked to design an architecture for the vector norm-2 list processor (VNLP) using the above the library of components. The VNLP interfaces to memory as shown below.

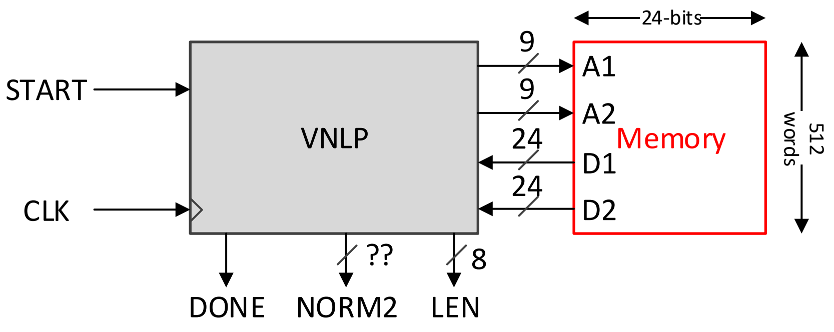

Figure 2: VNLP memory interface

The VNLP takes as input a CLK signal and a control START signal that resets all internal registers and initiates the vector norm-2 squared computations. When the processor is done, it asserts the DONE signal, generates the required norm at the output port NORM2, and generates as well as the length of the vector n at the output port LEN. You do NOT know ahead of time what the length of the vector is, nor what the address A is. But you know that the list in memory that holds the components of the vector starts at address 0, and that the list contains at least one element (i.e., $1 \le n \le 128$).

1. Determine what the largest value of the vector norm-2 is, and hence find the number of bits of the NORM2 output port. Analyze how many bits you need inside the VNLP to do arithmetic without losing too much precision. Remember, when you multiply in this fixed-point format, you need to multiply mantissas and add exponents, then possibly truncate/round the mantissa product and adjust the exponent to cut down on the number of bits; when you add, you need to first align the exponents, then add the mantissas, then truncate/round and re-adjust the exponent.
2. Write down the pseudo-code of the algorithm to be used for computing the vector norm-2 operation similar to what we did in class related to the basic list processor.
3. Draw the block diagram of an architecture that implements your pseudo-code above.
4. Draw a finite-state machine to control the datapath of the VNLP.
5. Analyze the performance of your design. Using delay values from the component library, determine the worst case propagation of the architecture and maximum clock frequency that the VNLP processor can run at. Draw a timing diagram illustrating how you came up with the maximum clock period.
6. Optimize your design for maximum speed followed by minimal gate cost. Write the pseudo-code of the final optimized architecture in RTL form, and provide the accompanying FSM, and performance analysis that shows what the new clock speed is.
7. Implement your final optimized architecture in Verilog.
8. Test your Verilog code on the list provided in Figure 3 below.

Deliverables: Provide a detailed report typed in Word or Latex that contains the following:

- Detailed analysis of the relevant bit-widths for norm-2 computations
- Pseudo-code, architecture, FSM, timing analysis of the basic and optimized architectures
- Verilog code of your optimized design according to the following structure:
    - Top level design: **VNLP.v** (instantiates **memory.v**, **datapath.v**, **control.v**)
    - **memory.v**: implements the memory block
    - **datapath.v**: implements the datapath
    - **control.v**: implements the controller
    - Testbench: **VNLP_tb.v** with memory initiated as shown in Figure 3 (all other memory entries are 0).
- Timing waveform demonstrating the execution of the original and modified programs. List all relevant signals in your waveforms to show that the program indeed executes the write operations, according to the following order:
    - CLK
    - START
    - A1
    - D1
    - A2
    - D2
    - DONE
    - NORM2

  - LEN
  - (list here only the remaining important internal control signals in your controller)

Grading:

- **Algorithm**: 20% (*wrong algorithm you get a ZERO*)
- **Architecture design**: 30% (*if basic design only without optimizations, you get at most half the grade here)*
- **Verilog**: 30% (*if Verilog code does not compile or shows erroneous output signals, you get a ZERO here*)
- **Report**: 20% (*messy or sloppy writing here you get a ZERO*)
- Cheating/copying: All parties involved will get a zero, no questions asked.

0:	49
1:	34
2:	-33.125
3:	23.75
4:	
5:	17
6:	40
7:	19.03125
8:	102.0
9:	
10:	
11:	22
12:	18
13:	25.5
14:	-93.125
15:	
16:	
17:	11
18:	6
19:	8.125
20:	90.09375
21:	
22:	33
23:	12
24:	31.28125
25:	32.34375
26:	
27:	
28:	102
29:	240
30:	47.0625
31:	-11.46875
32:	
33:	0
34:	23
35:	25.75
36:	88.125
37:	
38:	
39:	5
40:	50
41:	56.59375
42:	48.65625
43:	
44:	56
45:	88
46:	112.71875
47:	69.78125
48:	
49:	39
50:	1
51:	101.96875
52:	63.25
53:	
	⋮

Figure 3: Memory contents