# Alternate Final Exam Report Template (ver 1.0)

Live URL: https://docs.google.com/document/d/1J3obz0IvGOjJtRWLi5E4kmmnpVHjZhic5pUVNs_pi-A

## Student Name: Le'on Norfleet

### Collaboration Information

*Please provide below anyone other than course staff you sought help from or collaborated with on any aspect of the exam, and the nature of help/collaboration.*

N/A

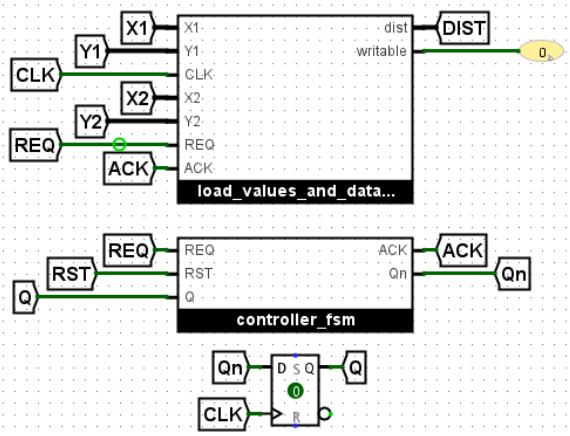
### Reference Material Used

*Please list below any information or material that you used beyond that made available to you during the course, e.g., articles from the web, software tools, design files, etc.*

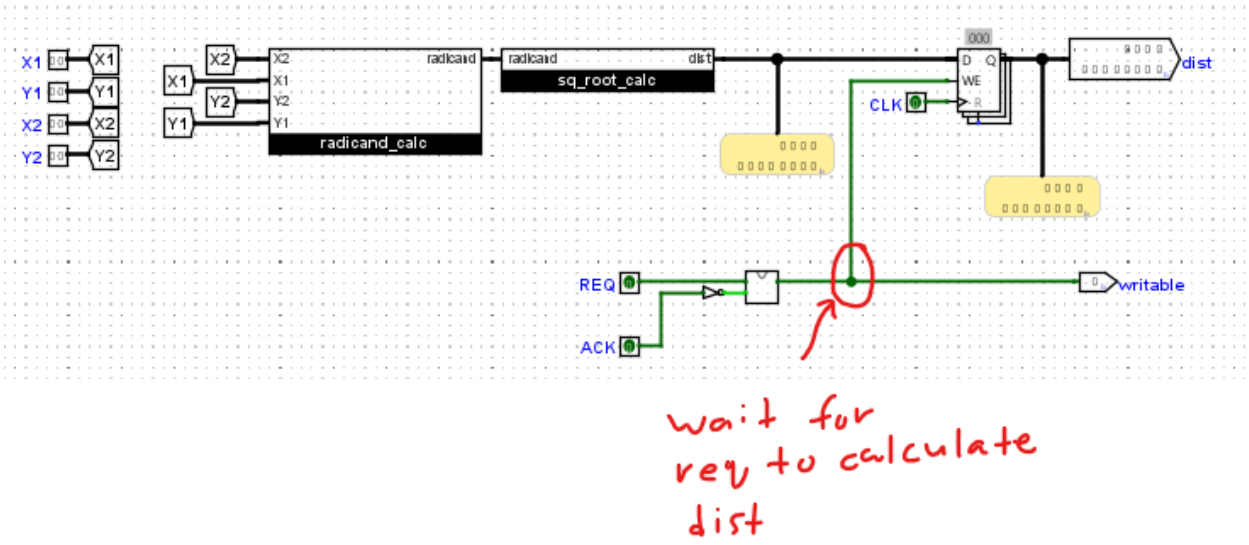
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.174.6832&rep=rep1&type=pdf

## Task 1: Euclidean Distance Computer with 4-Phase Handshake Interface

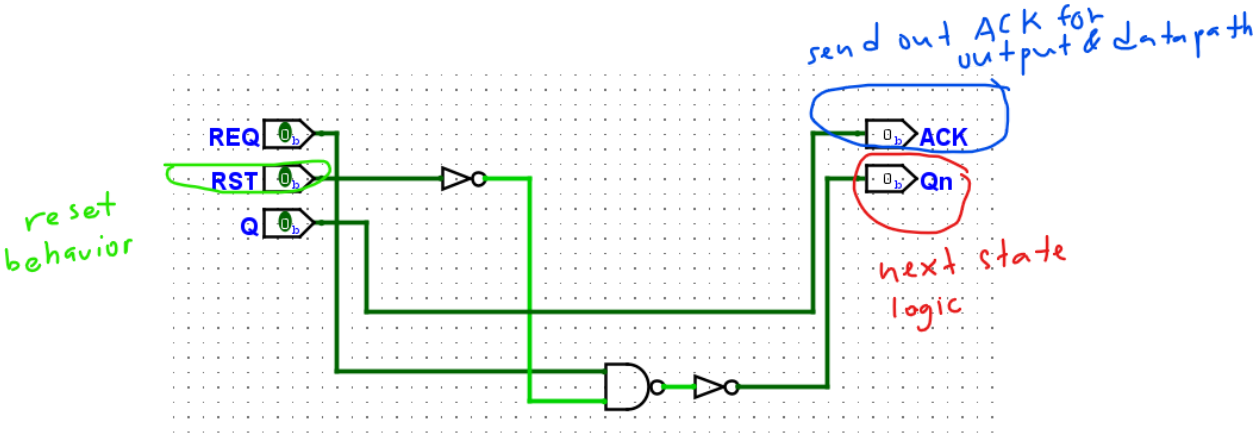
### Architecture showing Datapath, Controller, and Main Signals



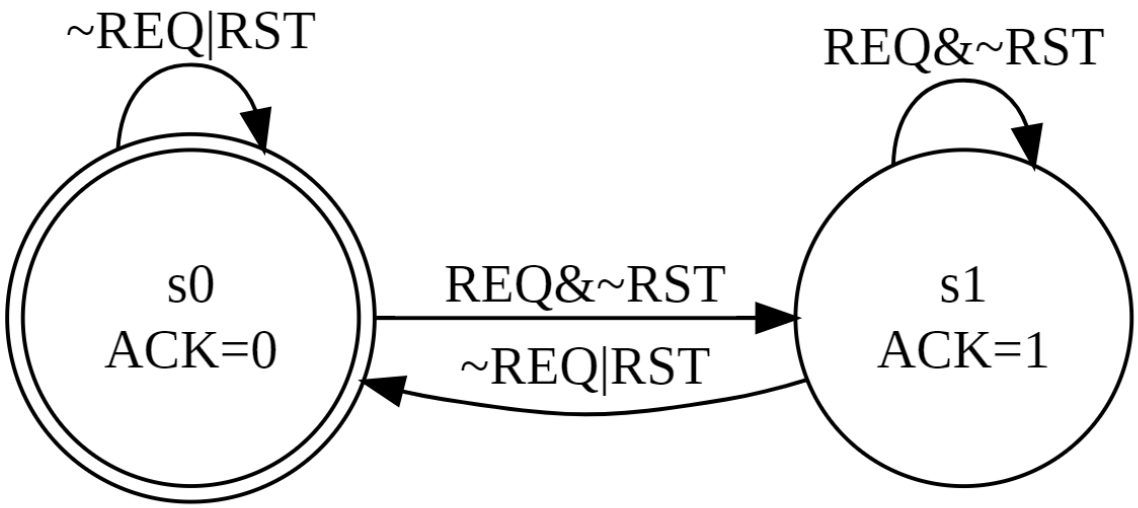
Datapath: **not calculate, update value of dist




Controller:




## State Transition Diagram (for every FSM in Controller)




## State Transition Tables (for every FSM)

**Q is the current state, Qn is the next state

*#Automatic compact state assignment: {'s0': 0, 's1': 1} using 1 bits*

*# State Assignment: {'s0': 0, 's1': 1}*

*REQ RST Q | ACK Qn*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

| | | | | |
|---|---|---|---|---|
| *0* | *1* | *0* | *0* | *0* |
| *0* | *0* | *0* | *0* | *0* |
| *1* | *1* | *0* | *0* | *0* |
| *1* | *0* | *0* | *0* | *1* |
| *0* | *1* | *1* | *1* | *0* |
| *0* | *0* | *1* | *1* | *0* |
| *1* | *1* | *1* | *1* | *0* |
| *1* | *0* | *1* | *1* | *1* |

## State Encoding

*For every FSM describe how you encoded the states into bit vectors and why you picked this encoding.*

I chose binary encoding because there are only states involved with the handshake behavior which is 2 states. This was possible because it takes 0 delay to calculate the square root.

## Circuit Area
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Total Circuit Area = *9588*

## Distance Computation Delay (in clock cycles)

The “delay” of your circuit is the # of clock cycles between both points being available to the system (i.e., the clock edge at which your system sees the latter of $GO1 = 1$ and $GO2 = 1$) and when the result is seen by the external work (i.e., the clock edge at which is sees $DONE = 1$). Note that we will validate what you write below using the Autogtader

Fill the table below for the 10 test cases with the $DIST$ that your system computer and the delay it had.

| $X1$ | $Y1$ | $X2$ | $Y2$ | $DIST$ | DELAY (# of cycles) |
|---|---|---|---|---|---|
| -128 | -128 | -128 | -128 | 0x000 | 1 |
| -128 | -128 | 127 | 127 | 0xb44 | 1 |
| -128 | -128 | 0 | 0 | 0x5a8 | 1 |
| -128 | -128 | 127 | -128 | 0x7f8 | 1 |
| -96 | -96 | -128 | 127 | 0x70a | 1 |
| 0 | 0 | 127 | 127 | 0x59c | 1 |
| -63 | -63 | 63 | 63 | 0x591 | 1 |
| -10 | 10 | 10 | -10 | 0x0e2 | 1 |
| 96 | -96 | 63 | 63 | 0x513 | 1 |
| 33 | -33 | -50 | 50 | 0x3ab | 1 |

Average Delay for Distance Computation = 1 cycle
Median Delay for Distance Computation = 1 cycle
Best Case Delay for Distance Computation = 1 cycle
Worst Case Delay for Distance Computation = 1 cycle

## Area-Delay Product

$Area \times Median\ Delay =$ 9588

## Design Approach

*Give a brief description of your controller + datapath design approach, including how you split between the two, what is the information exchange between them, how are the two coordinated, and how did you try to optimize for area and delay.*

What I noticed from the spec is that the hand shake can be done in 2 cycles before factoring in the delay of calculating the euclidian distance. With that information I used REQ and ACK to control when the dist output would be updated. The value of ACK was determined by the controller FSM which used the REQ input to switch states. Since I was able to implement distance calculation with no delay, I made the system stall the updated value of dist for 1 cycle to make sure that dist could not be overwritten during periods where the values of the coordinates were ambiguous. Tried to optimize for area and delay by using as few arithmetic modules as possible.

## Testing Approach

*Give a brief description of how you tested your design, including a description or screenshots of any testing circuit you created.*

I used a list of different sets of coordinates and what the correct DIST output should be in binary/hexadecimal and then used a timing diagram while observing this.

## Additional Remarks

*Provide any additional information to help us understand your design better.*

*It takes 2 cycles for the entire system (computation + handshake) to finish. This was made using sketchviz + gv2fsm and there was no hand patching of the state machine/controller+datapath. I interpreted the spec as:*

*1.Outside world inputs values of x1, y1, x2, y2 at the same time (these value inputs are ambiguous while REQ != 1)*
*2.Outside world then sets REQ to 1*
*3.Distance is calculated (outside world will not tamper with coordinate input values while REQ = 1 and ACK = 0) and when finished system sets ACK to 1*
*4.When outside world is finished viewing DIST during the period where ACK is 1 (DIST should not change while REQ = 1 and ACK = 1), it will set REQ to 0 (coordinate value inputs can be ambiguous during this period)*
*5.The system will set ACK to 0 in response to this (coordinate value inputs can be ambiguous during this period)*