Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

The List of Inputs:

- **'InputP'** is a 32-bit integer that is an input to the following modules: add_sub, Multiplication, Division, Modulo, and Exponent. It will also work on the operations AND, NAND, NOR, NOT, OR, XNOR, XOR.
- **'InputQ'** is a 32-bit integer that is an input to the following modules: add_sub, Multiplication, Division, Modulo, and Exponent. It will also work on the operations AND, NAND, NOR, NOT, OR, XNOR, XOR. For this part of the project,
- **'opcode'** is a 4-bit operational code that is fed into a 4-to-16 Decoder that will translate the 4 bits into a one-hot channel select value for the Multiplexer.
- **'Reset'** is a channel that feeds directly into the Multiplexer. It is a 32-bit integer consisting of all 0's. This channel will be selected in order to reset the circuit (i.e., make it all 0's).
- **'Pre-set'** is a channel that feeds directly into the Multiplexer. It is a 32-bit integer consisting of all 1's. This channel will be selected in order to pre-set the circuit (i.e., make it all 1's).
- **'No-Op'** is the 32-bit integer that will feed out of the memory register; however, it is also an input into the Multiplexer. The lower 16 bits of 'outMem' will feed directly into the Multiplexer. This will represent the No-Op command.
- **'feedback'** is a memory register that is used if subsequent operations are done

The List of Outputs:

- **'outALU'** is a 32-bit integer that will feed out of the multiplexer and into the 32-bit memory register.
- **'errorCode'** is the output of a 2-bit error code module. It will output a value if there is an error in the arithmetic operations (i.e., either an overflow error from the adder-subtractor or a divide by zero error from the division and modulo modules).
- 'outMem' is a 32-bit integer that will feed out of the memory register.

The List of Interfaces:

- **'outAddsub'** is a 32-bit channel that connects the sum of the adder-subtractor module into the Multiplexor.
- **'outAddsub'** is a 32-bit channel that connects the difference of the adder-subtractor module into the Multiplexor.
- **'outMult'** is a 32-bit channel that connects the product of the multiplication module into the Multiplexor.

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

- **‘outDiv’** is a 32-bit channel that connects the quotient of the division module into the Multiplexor.
- **‘outMod’** is a 32-bit channel that connects the remainder of the modulo module into the Multiplexor.
- **‘outAND’** is a 32-bit channel that connects the result of an AND module to the Multiplexor.
- ‘outNAND’ is a 32-bit channel that connects the result of a NAND module to the Multiplexor.
- **‘outNOR’** is a 32-bit channel that connects the result of a NOR module to the Multiplexor.
- ‘out NOT’ is a 32-bit channel that connects the result of a NOT module to the Multiplexor.
- **‘outOR’** is a 32-bit channel that connects the result of an OR module to the Multiplexor.
- ‘outXNOR’ is a 32-bit channel that connects the result of a XNOR module to the Multiplexor.
- **‘outXOR’** is a 32-bit channel that connects the result of an XOR module to the Multiplexor.
- **‘overflow’** is a 32-bit channel that connects the overflow of the adder-subtractor module to the least significant bit of ‘error’.
- **‘divZeroMod’** is a 32-bit channel that connects the division module to an OR gate, which will then connect to the most significant bit of ‘error’.
- **‘divZeroDiv’** is a 32-bit channel that connects the modulo module to an OR gate (the same OR gate as DivideByZeroLine1), which will then connect to the most significant bit of ‘error’.
- **‘hotselect’** is a 16-bit channel that connects the output of the Decoder to the select of the Multiplexor.
- **‘opCode’** is a 4-bit channel that connects ‘Command’ to the mode bit of the adder-subtractor module. If ‘Command’ is 0000, then the mode bit is 0 and the adder-subtractor will add. If ‘Command’ is 0001, then the mode bit is 1 and the adder-subtractor will subtract.
- **‘feedbackCheckOut’** is a 32 bit channel that determines if the feedback is zero or not. If feedback is zero feedbackCheck will equal inputQ. If feedbackCheck is nonzero, feedbackCheck will equal feedback

The List of Parts:

- **‘add_sub’** is an adder-subtractor module that uses two inputs and a mode bit. If the

System Design Description

---

mode bit is 0, then the module adds the two inputs. If the mode bit is 1, then the module subtracts the two inputs. Result ‘outAddsub’ is the sum of add-sub, result ‘carry’ is the carry of the add_sub, and result ‘overflow’ is the overflow warning. ‘outAddSub’ connects the sum of the module to the Multiplexor and ‘‘outAddsub’ also connects the difference of the module to the Multiplexor in a separate port..

- **‘Multiplication’** is the multiplication module that uses two inputs. It multiples the two inputs together and uses ‘outMult’ as output of the module and input to the Multiplexor.
- **‘Division’** is the division module that uses two inputs. It divides the two inputs and uses ‘outDiv’ as output of the module and input to the Multiplexor.
- **‘Modulo’** is the modulo module that uses two inputs. It takes the remainder of the two inputs and uses ‘outMod’ as output of the module and input to the Multiplexor.
- **‘ErrorModule’** is the error module that uses two inputs. One input will come from ‘overflow’, which detects if there is an overflow present in the adder-subtractor module. This input is connected to the most significant bit of the error output. The second output will come from the output of the OR gate (listed here last), which detects if there is a divide by zero error in either the division or modulo modules. This input is connected to the least significant bit of the error output.
- **‘AND’** is the AND module that uses two inputs. It ANDs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the Multiplexer.
- **‘NAND’** is the NAND module that uses two inputs. It NANDs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the Multiplexer.
- **‘NOR’** is the NOR module that uses two inputs. It NORs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the Multiplexer.
- **‘NOT’** is the NOT module that uses two 16-bit inputs. It will NOT the second input (i.e., InputQ) and it will concatenate/pad the first input (i.e., InputP) towards the front end so that it becomes 32 bits long and fits into the Multiplexer.
- **‘OR’** is the OR module that uses two inputs. It ORs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the Multiplexer.
- **‘XNOR’** is the XNOR module that uses two inputs. It XNORs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the Multiplexer.
- **‘XOR’** is the XOR module that uses two inputs. It XORs the two inputs together (both of which are 16-bit) and outputs a 32-bit answer which is padded on the front end to fit the

System Design Description

Multiplexer.

- **'feedback'** is a 32-bit memory register that is placed to the right of the Multiplexer. Its input will be the 32-bit output of the Multiplexer and it will have a 32-bit output whose lower 16 bits works as a feedback into the second input (InputQ).
- **'Multiplexer'** is the 16-channel, 32-bit, one-hot Multiplexor that chooses a value as its output. The single output of the Multiplexor which is outALU.
- **'Decoder'** is a 4-to-16-bit Decoder. It translates the 4-bit opcode into a 16-bit one-hot. That 16-bit one-hot is used as the channel selector of the Multiplexor.
- **'Error Module'** is a 2-input OR gate that will OR the 'divZeroMod' and 'DivZeroDiv' lines together. If there is a divide by zero error in either the division or modulo modules, then it will be fed into the most significant bit of the error module.
- **'FeedBackCheck'** is a module that checks if feedback is zero. If feedback is zero, then i

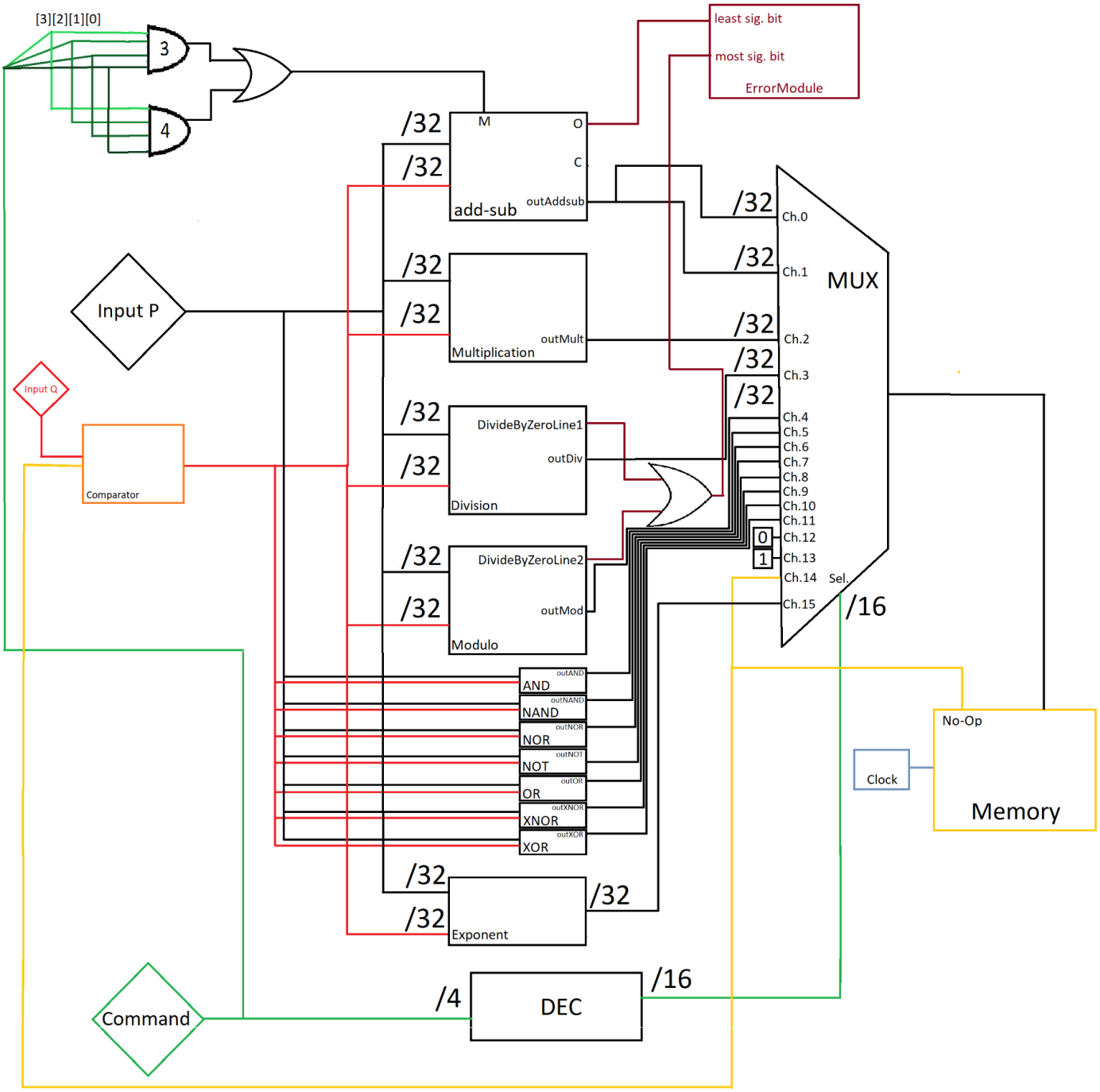
Assigned 4-bit codes to the arithmetic functions, logic operations, and memory/control:

- Addition - 0000
- Subtraction - 0001
- Multiplication - 0010
- Division - 0011
- Modulo - 0100
- AND - 0101
- NAND - 0110
- NOR - 0111
- NOT - 1000
- OR - 1001
- XNOR - 1010
- XOR - 1011
- Reset - 1100
- Preset - 1101
- No-Op - 1110
- Exponent - 1111

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

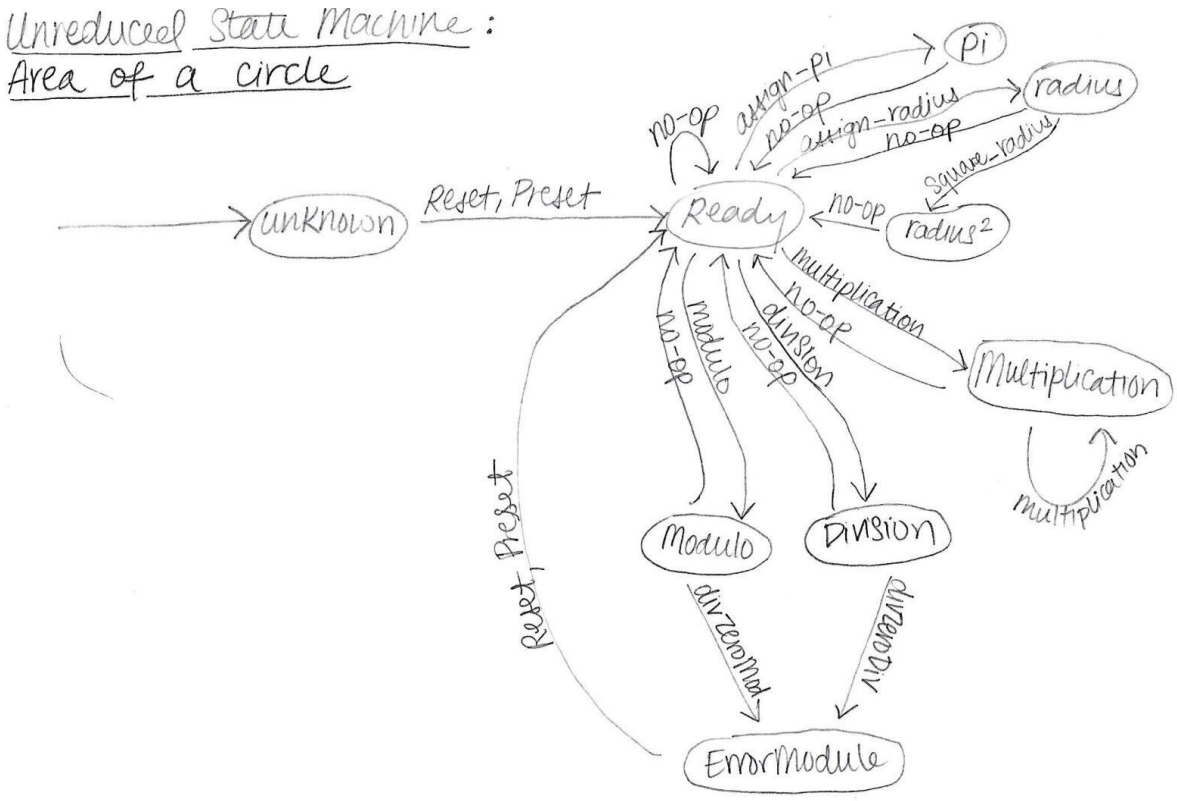
Top-Level Circuit Diagram:



Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Unreduced State Machine Drawing (Area of a Circle):



Unreduced State Table (Area of a Circle):

Current state	Trigger: opcode (operation)	Next State
Unknown	1100 (Reset), 1101 (Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign_pi	pi

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

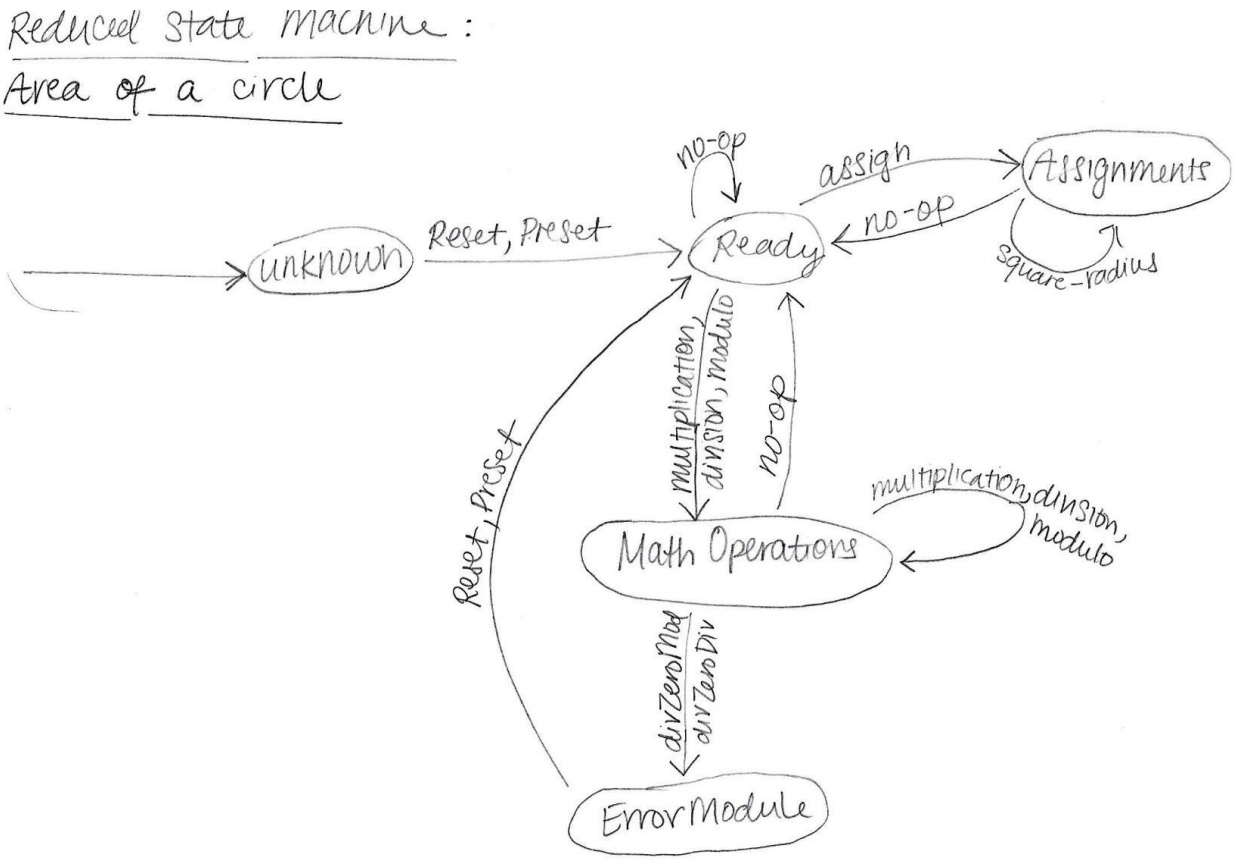
System Design Description

Ready	assign_radius	radius
Ready	0010 (multiplication)	Multiplication
Ready	0011 (division)	Division
Ready	0100 (modulo)	Modulo
pi	1110 (No-Op)	Ready
radius	1110 (No-Op)	Ready
Multiplication	1110 (No-Op)	Ready
Division	1110 (No-Op)	Ready
Modulo	1110 (No-Op)	Ready
radius	1111 (exponent: square_radius)	$radius^2$
$radius^2$	1110 (No-Op)	Ready
Multiplication	0010 (multiplication)	Multiplication
Modulo	divZeroMod	Error Module
Division	divZeroDiv	Error Module
Error Module	1100 (Reset), 1101 (Pre-set)	Ready

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Reduced State Machine Drawing (Area of a Circle):



Reduced State Table (Area of a Circle):

Current State	Trigger: opcode (operation)	Next State
Unknown	1100 (Reset), 1101(Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign	Assignments
Assignments	1111 (exponent:	Assignments

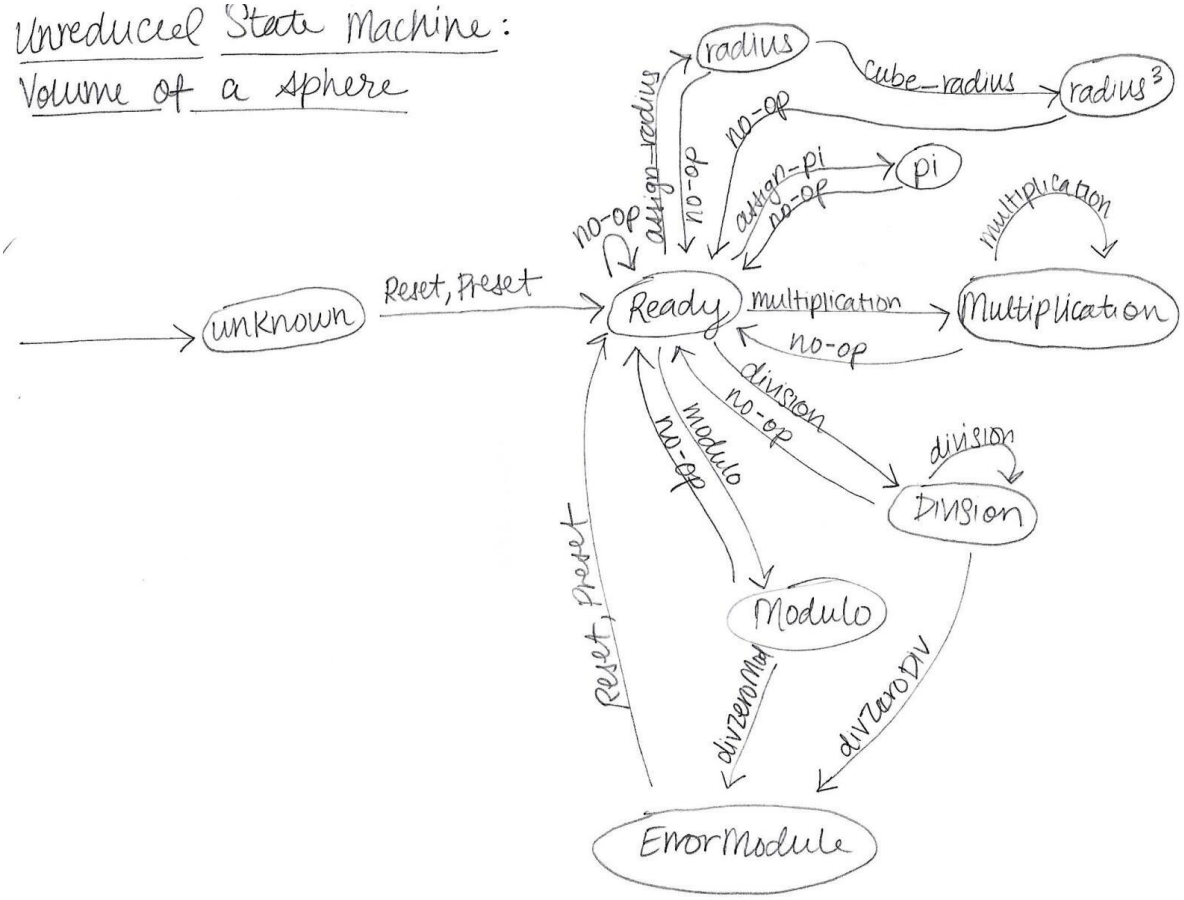
Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

---

	square-radius)	
Assignments	1110 (No-Op)	Ready
Ready	Multiplication, Division, Modulo (0010, 0011, 0100)	Math Operations
Math Operations	1110 (No-Op)	Ready
Math Operations	Multiplication, Division, Modulo (0010, 0011, 0100)	Math Operations
Math Operations	divZeroMod, divZeroDiv	Error Module
Error Module	1100(Reset), 1101(Pre-set)	Ready

Unreduced State Machine Drawing (Volume of a Sphere):



Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

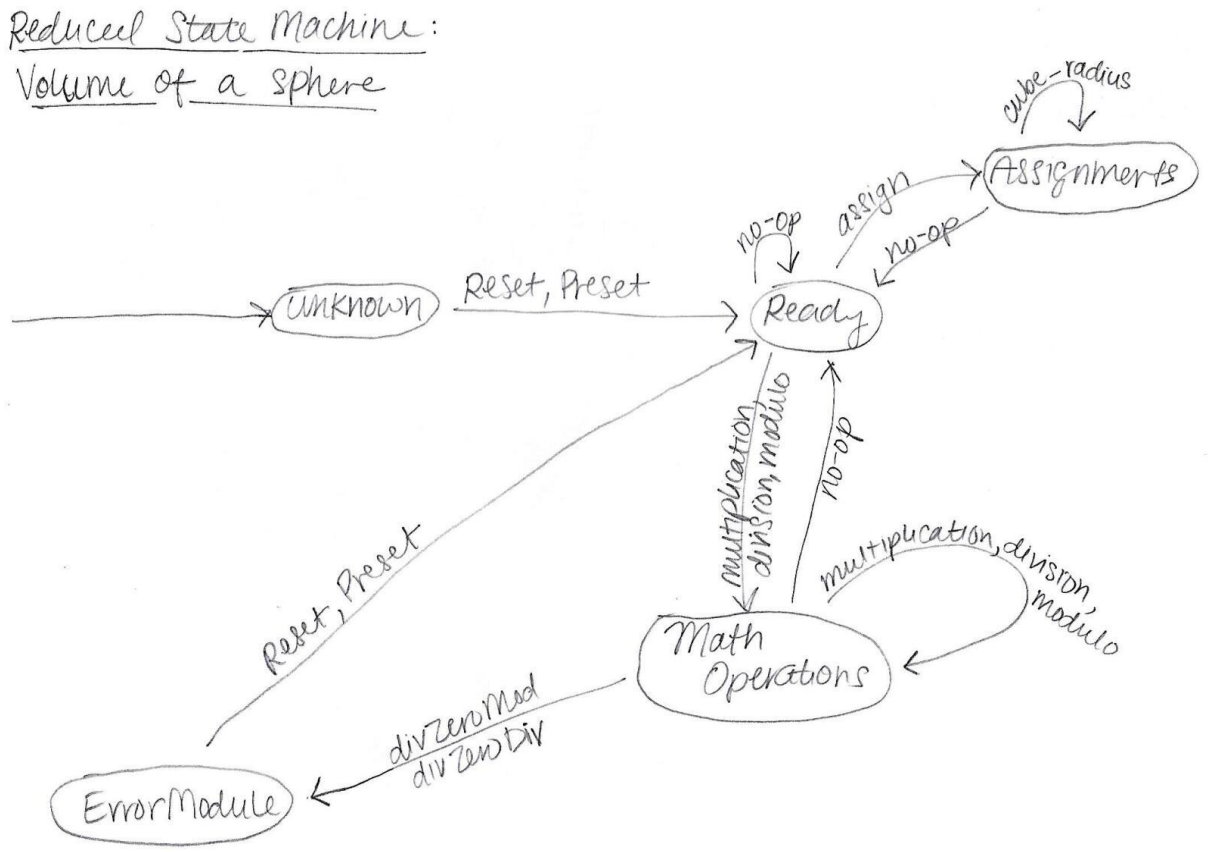
Unreduced State Table (Volume of a Sphere):

Current State	Trigger: opcode (operation)	Next State
Unknown	1100 (Reset), 1101 (Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign_radius	radius
Ready	assign_pi	pi
Ready	0010 (multiplication)	Multiplication
Ready	0011 (division)	Division
Ready	0100 (modulo)	Modulo
radius	1110 (No-Op)	Ready
$radius^3$	1110 (No-Op)	Ready
pi	1110 (No-Op)	Ready
Multiplication	1110 (No-Op)	Ready
Division	1110 (No-Op)	Ready
Modulo	1110 (No-Op)	Ready
radius	1111 (exponent: cube_radius)	$radius^3$
Multiplication	0010 (multiplication)	Multiplication
Division	0011 (division)	Division
Division	divZeroDiv	Error Module
Modulo	divZeroMod	Error Module
Error Module	1100 (Reset), 1101 (Preset)	Ready

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Reduced State Machine Drawing (Volume of a Sphere):



Reduced State Table (Volume of a Sphere):

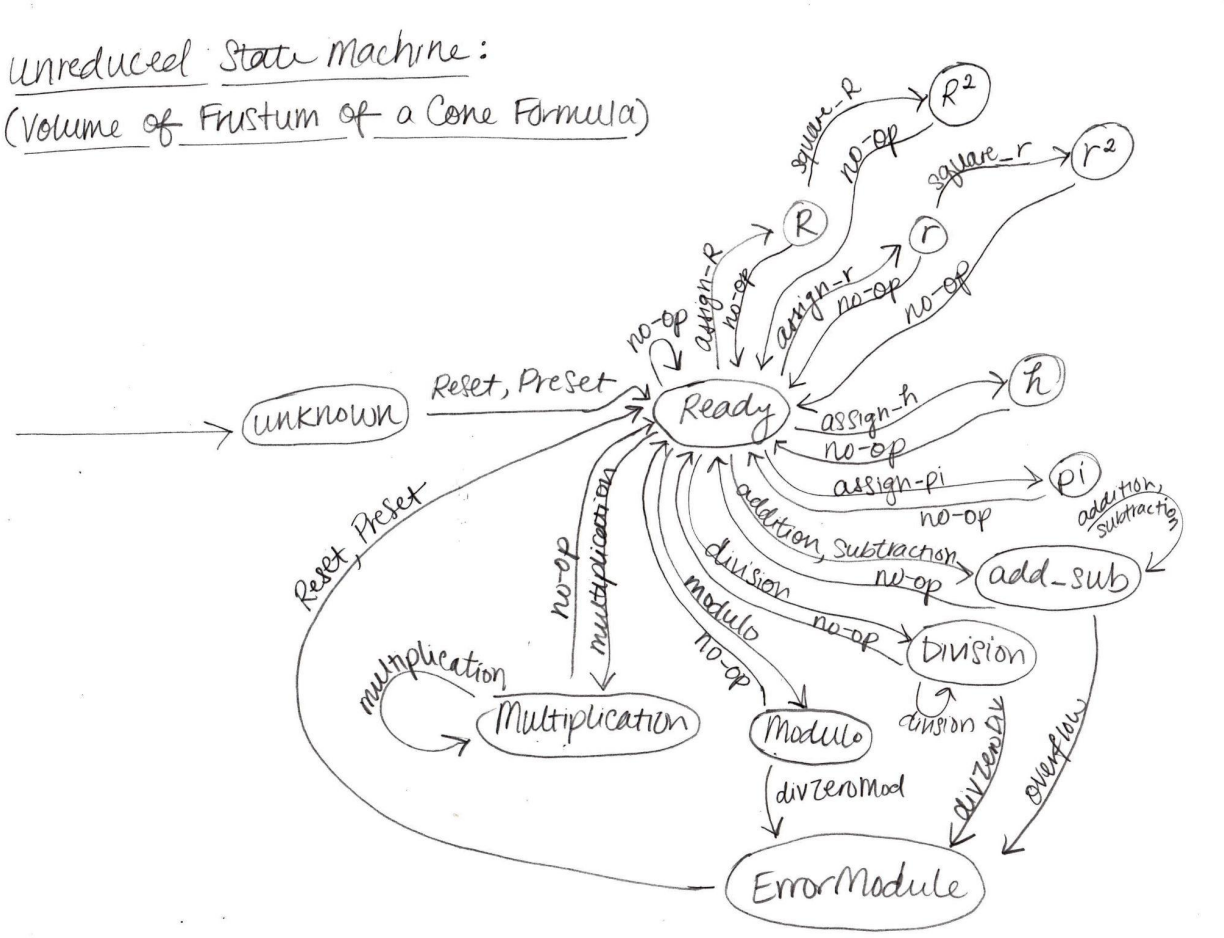
Current State	Trigger: opcode (operation)	Next State
Unknown	1100(Reset), 1101(Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign	Assignments
Assignments	1111 (exponent: cube_radius)	Assignments

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Assignments	1110 (No-Op)	Ready
Ready	Multiplication, Division, Modulo (0010, 0011, 0100)	Math Operations
Math Operations	Multiplication, Division, Modulo (0010, 0011, 0100)	Math Operations
Math Operations	1110 (No-Op)	Ready
Math Operations	divZeroMod, divZeroDiv	Error Module
Error Module	1100(Reset), 1101(Pre-set)	Ready

Unreduced State Machine Drawing (Volume of Frustum of a Cone):



Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Unreduced State Table (Volume of a Frustum of a Cone):

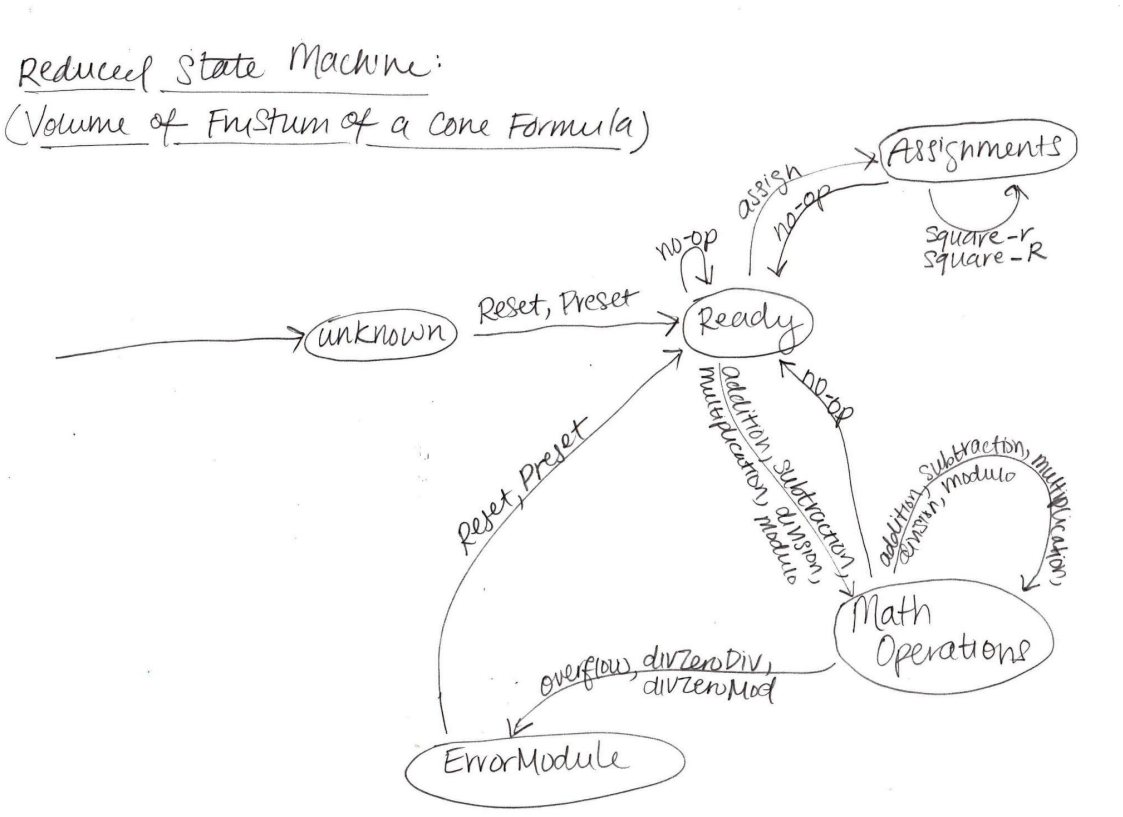
Current State	Trigger: opcode (operation)	Next State
Unknown	1100 (Reset), 1101 (Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign_R	R
Ready	assign_r	r
Ready	assign_h	h
Ready	assign_pi	pi
Ready	0000 (addition), 0001 (subtraction)	add_sub
Ready	0011 (division)	Division
Ready	0100 (modulo)	Modulo
Ready	0010 (multiplication)	Multiplication
R	1110 (No-Op)	Ready
R^2	1110 (No-Op)	Ready
r	1110 (No-Op)	Ready
r^2	1110 (No-Op)	Ready
h	1110 (No-Op)	Ready
pi	1110 (No-Op)	Ready
add_sub	1110 (No-Op)	Ready
Division	1110 (No-Op)	Ready
Modulo	1110 (No-Op)	Ready

Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Multiplication	1110 (No-Op)	ready
R	1111 (exponent: square_R)	R^2
r	1111 (exponent: square_r)	r^2
add_sub	0000 (addition), 0001 (subtraction)	add_sub
Division	0011 (division)	Division
Multiplication	0010 (multiplication)	Multiplication
add_sub	overflow	Error Module
Division	divZeroDiv	Error Module
Modulo	divZeroMod	Error module
Error Module	1100 (Reset), 1101 (Pre-set)	Ready

Reduced State Machine Drawing (Volume of Frustum of a Cone);



Cohort Members: Reg Gonzalez, Desh Padmakumar, Gabriela Saenz, Joshua Vardeleon, Matthew Lineberry
Cohort Name: DJMr.G
Class: CS 4341.502

System Design Description

Reduced State Table (Volume of Frustum of a Cone):

Current State	Trigger: opcode (operation)	Next State
Unknown	1100(Reset), 1101(Pre-set)	Ready
Ready	1110 (No-Op)	Ready
Ready	assign	Assignments
Assignments	1111 (exponent: square_r/square_R)	Assignments
Assignments	1110 (No-Op)	Ready
Ready	Addition, Subtraction, Multiplication, Division, Modulo (0000, 0001, 0010, 0011, 0100)	Math Operations
Math Operations	Addition, Subtraction, Multiplication, Division, Modulo (0000, 0001, 0010, 0011, 0100)	Math Operations
Math Operations	1110 (No-Op)	Ready
Math Operations	divZeroMod, divZeroDiv, overflow	Error Module
Error Module	1100(Reset), 1101(Pre-set)	Ready