Goal: Our goal was to create a quasi-synthesizable Verilog model of a Polynomial Evaluation Accelerator (PEA) for streamlining polynomial computations. This model was designed with 2 input buffers, containing the control commands and input data, and two output buffers containing the result and status of the system. In addition, it needs to handle 16-bit signed, two's compliment numbers, and produce up to 32-bit signed two's compliment answers. The PEA supports equations up to and including the 10^{th} degree.

Procedure and Implementation Decisions: For this design, we utilized fifo buffers to read and store data from the input streams, until they were ready to be processed by the Polynomial Evaluator. The PEA itself can interpret one of four instructions during a single clock cycle. To ensure these were accurately interpreted, we chose to design a uniform layout for these instructions as 21-bit entries; 2 bits to determine the instruction, 3 bits to determine the Coefficient Address, and the remaining 16 bits to hold either a 16-bit x-value, a 4-bit polynomial degree for STP, or a 5-bit iteration count for EVB. The data inputs are all 16-bit values, to represent either coefficients or x-values. The table below shows the ideal operation of the system, subject to conditions of buffer fullness or emptiness in each case. In case of buffers are full or empty, the clock cycle is just skipped idly.

	Read	Execute	Status (Correct and Incorrect)
RST	Reset PEA actor. Current state is reset. Next state is Read	-	-
STP A n	Output result = 0 and a status. Next state is execute	Read & store n data values. No results.	CORRECT_STP_INSTRUCTION
			INVALID_A, INVALID_N
EVP	Output result of computation. Next state is read	-	EVP_CORRECT
			INVALID_A, A_NOT_SET
EVB A b	Output result = 0. Next state is execute	Read and produce b results	CORRECT_EVB_INSTRUCTION, EVB_CORRECT
			INVALID_B, A_NOT_SET, INVALID_A

Our design decision to handle the assignment of the values inside of the combinational block allows us to ensure that everything is calculated accurately and together within one clock cycle per execution. The sequential portion of the system was limited with minimal code and focuses on controlling the state machines and the state of the buffers. When all information is available to the PEA, it is simultaneously loaded and executed to maintain accuracy, and pushed to the output buffer if write is enabled. Another design implementation with PEA logic was to maintain it in a quasi-synthesizable state by using (*) and (**) operands to accurately calculate products and exponential results without designing the gate level description. To ensure the state of the system was also being properly managed, we implemented a status state which maintains a distinct value based on the state of the machine and any unique errors it has encountered. Under normal circumstances, it will output a code for correct instructions being received in correspondence with the execution of that instruction. If an error occurs through an

invalid entry, the system produces a 0 for its result, returns to the "read instruction" state, and updates its status with the corresponding error code. This allows the functionality of the machine to be fluid throughout errors, and limit any further negative effects on the remainder of the system.

Test Roster

Test Name	Function
Test00	Unit test of PEA combinatorial block that evaluates polynomials
Test01	Test Basic Functionality of PEA with STP and EVP
Test02	Test EVP Functionality with negative coefficients
Test03	Test EVP Functionality with negative x input
Test04	Test RST's ability to clear stored coefficients
Test05	Test for unique error status when a polynomial equation has a power greater than 10
Test06	Test Basic Functionality of PEA with STP and EVB
Test07	Test for unique error status when EVB wants to streamline more than 31 x inputs
Test08	Test for unique error status when Coefficients have not been set before using EVB.
Test09	Ensure functionality is correct even when sum passes 31 bits, (This result is user error)
Test10	Testing EVB functionality with negative x values in data input
Test11	Test for unique error when EVB B value is negative.
Test12	Test complete functionality with multiple control statements.

Test Results: The goal of our test plan was to ensure all possible inputs from the control and data ports were properly handled and executed as planned. Based on our design, errors should not interrupt program progress, but rather flag the system with a status update, and return to the "Read Instruction" state, ready to interpret the next instruction. In addition, our interaction with negative numbers with signed registers allowed us to accurately sum the total result within the Evaluator Combinational Logic block. To ensure this was being handled accurately, we designed several tests around negative numbers to ensure they would not introduce new errors. Additional error potentials existed mainly in commands desiring the use of invalid values, out of range of the system design such as an 11^{th} degree polynomial, or negative B value for EVB.

Conclusion: In conclusion, our design was focused around separating the system into a larger combinational block with state machines controlling the proper actions of calculations. As a result, we were able to ensure that values were always synchronized and loaded together properly when passing to the calculation of the total summation. Each clock cycle a calculation was executed and a summation produced or a coefficient stored. However, this also required that the results use an additional clock cycle to propagate the result to the output buffer. Our buffers allowed for proper handshaking between the instruction/data sources and the PEA. While a longer PEA function was occurring, such as an EVB with a large B value, the instruction buffer could continue to store future instructions and signal to the PEA that it was ready to load more information. Overall, this design worked as desired and allowed us to accurately depict the PEA.