EE4415 Lab 2 AES Encryption on Synopsys| David Lee Kwok Hung | A0168625W

Table of Contents

Overview of AES Encryption

The AES Encryption process consists of 11 total stages for a 128-bit word input and a 128-bit cipher key. During each stage, the input word passes through 4 steps of encryption, namely the byte substitution step, the shift rows step, the mix columns step, and the add round key step. The result of each stage is then used in the next stage of encryption. In the initial stage, the input is passed through the add round key step, after which the result is passed through 9 rounds of byte substitution, shift rows, mix columns, and add round keys. In the final round, the result is passed through all the steps except the mix columns step, producing one encrypted output from the word input and cipher key.

While the AES encryption process is not endian sensitive, for the purposes of verification, all inputs and cipher keys are manipulated as big-endian words. As such, the first byte of data is the 127th to the 120th bit. The order of the bits is as shown in Table 1.

Table 1: AES Encryption bit order for input text

	Column 1	Column 2	Column 3	Column 4
Row 1	127:120	95:88	63:56	31:24
Row 2	119:112	87:80	55:48	23:16
Row 3	111:104	79:72	47:40	15:8
Row 4	103:96	71:64	39:32	7:0

The byte substitution step replaces each byte of the word input with a database of values. Each byte is used as the input for a lookup table of cipher values, shown in Table 2, where the first 4 bits represent the row number and the next 4 bits represent the column number. Hence, a byte input of 8'hb6 would return a result of 8'h4e.

Table 2: Graphical representation of lookup table used for byte substitution step

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

The shift rows step shuffles the positions of each byte based on the row it is in. Table 3 depicts the positions of the shuffled bits. The first row is left unchanged, the second row is rotated left by one byte, the third row is rotated by two bytes, and the fourth row is rotated by three bytes.

Table 3: AES Encryption bit order for shift rows step

	Column 1	Column 2	Column 3	Column 4
Row 1	127:120	95:88	63:56	31:24
Row 2	87:80	55:48	23:16	119:112
Row 3	47:40	15:8	111:104	79:72
Row 4	7:0	103:96	71:64	39:32

The mix columns step multiplies each column of the input with Rijndael's Galois field, shown in Figure 1.

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Figure 1: Matrix multiplication of columns in mix columns step

The add round keys step combines the input from the previous steps and the subkey generated from the key schedule using an XOR operation on each byte.

The key schedule step generates a subkey from the input cipher, based on the current stage of encryption, using Rijndael's key expansion method. First, the last column of the cipher is rotated by one byte, then the column is replaced using the lookup table used in the byte substitution step, after which it is XOR-ed with a round counter matrix, and each column is XOR-ed with the columns to the left of itself.

For the lab project, the AES encryption process will be implemented using 3 methods: Single Input- Single Output (SISO), Multiple input - Multiple Output (MIMO), and N-Slowing. The implementations will be performed in Verilog and synthesized using Synopsys.

Part 1: SISO

In the SISO implementation, inputs are sent into the encryption machine one by one. When one word has completed encryption, the next input will then be sent in.

Overview of SISO AES implementation

The AES encryption machine consists of the AEStop module, which contains two smaller modules: AEScntx and AEScore. The AEScntx module performs the timing for each stage of the encryption process, while the AEScore module performs each step of the encryption based on the commands from the AEScntx module. The AEScore module consists of 5 sub modules for each step of the encryption process. The schematic of AEScore is shown in Figure 6, the schematic of AEScntx is shown in Figure 4, and the schematic of AEStop is shown in Figure 2.

AEStop Module

AEStop schematic:

The AEStop module shown in Figure 2 connects the AEScntx module with the AESCore module. It also sends in the plain text input to be encrypted, the encryption cipher key, the clock, reset, and start signals. The output of the AEStop module is the encrypted cipher text, as well as the done and completed round signal, which signals the testbench to send in the next input.

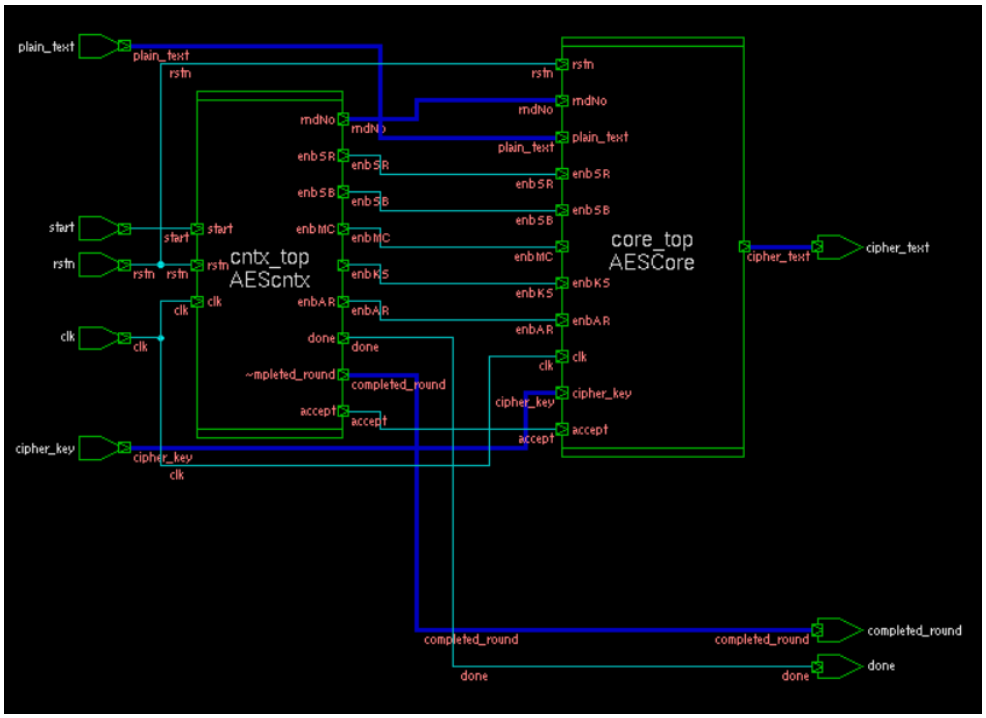

Figure 2: Schematic of AESTop

AEStop Timing Diagram:

Figure 3 shows the signal output from the AEStop module. When the start signal is sent, the AEScntx module begins to send control signals to the AESCore module. This increments the completed round signal and outputs the cipher text signal. When the done signal is sent at the end of an encryption sequence, the cipher text signal is stored in the testbench for comparison with a reference database. Then, a new plain text input is sent into the AEStop module for the next round of encryption.

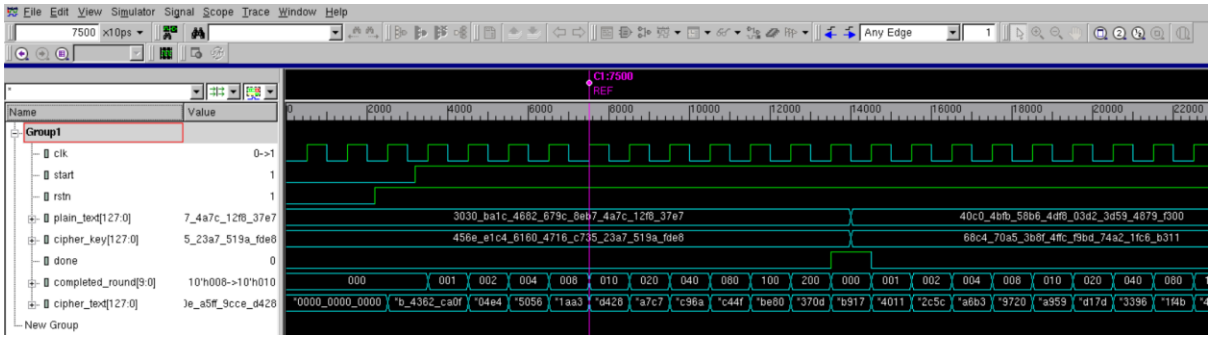

Figure 3: Timing Diagram for SISO testbench

AEScntx module

AEScntx schematic:

The AEScntx module controls the AEScore module based on input from AEStop. The clock and reset signals clk and rstn respectively are used to set the timing of each round, while the start signal is used to initialize the counting sequence of rndNo to set the stage of encryption. Table 4 shows the signal outputs and their purposes.

Table 4: Output Signal descriptions for AEScntx

Output	Purpose
Done	Shows if encryption has completed
Completed_round	Shows the current status of encryption
rndNo	Tells AEScore the current round number of encryptions
Accept	Tells AEScore to accept signal inputs from plaintext and cipherkey
enbSB	Tells AEScore to enable the subbytes module
enbSR	Tells AEScore to enable the shiftrows module
enbMC	Tells AEScore to enable the mixcolumns module
enbAR	Tells AEScore to enable the addroundkey module
enbKS	Tells AEScore to enable the keyschedule module

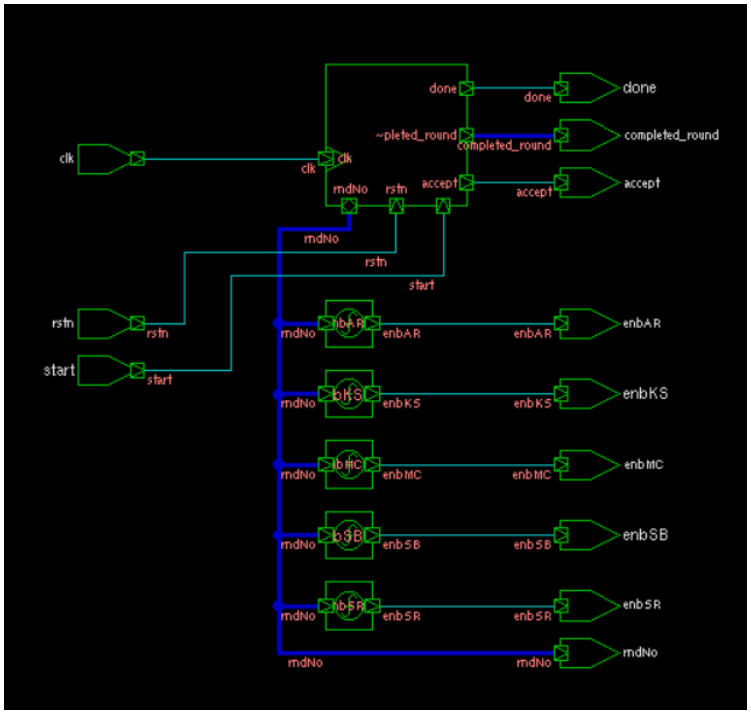

Figure 4: Schematic of AEScntx

AEScntx timing diagram:

Figure 5 shows the signal output from the AEScntx module. After the start signal is triggered, the rndNo and completed_round signals begin counting up, while the enable signals enbSB, enbSR, enbMC, enbAR, and enbKS are set to their respective levels depending on the current round number. When a cycle of encryption is complete the AEScntx module outputs a done signal to notify the testbench that a fully encrypted cipher text output has been produced.

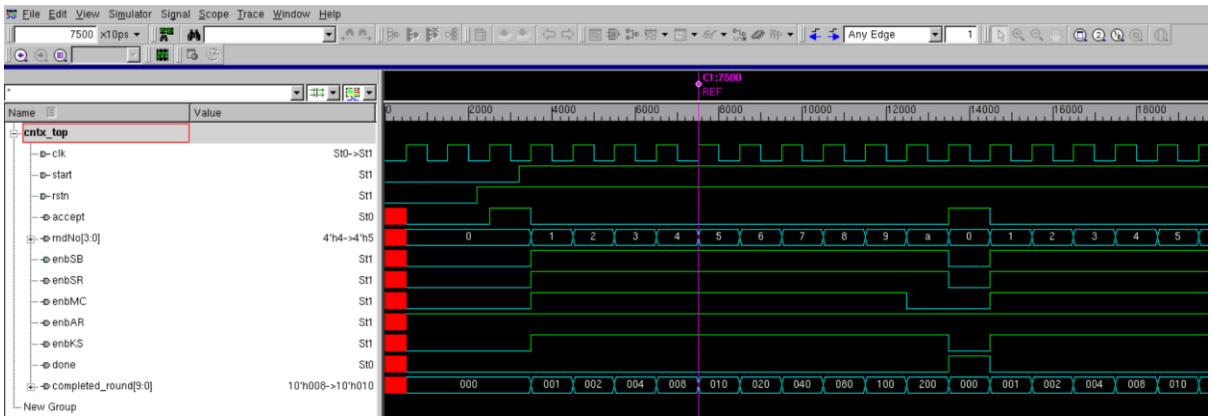

Figure 5: Timing Diagram for SISO AEScntx

AESCore module

AESCore schematic:

The AESCore module encrypts incoming input signals based on the current round and the enable signals sent to the submodules of the machine. Three registers, controlling the input from the Add round keys module to the subbytes module, the keyschedule input cipher, and the output cipher text are implemented to ensure the timing of the circuit corresponds to the commands from AESCntx. In addition, two

multiplexers, controlling the input of the plain text for encryption and the cipher key, are used to initialize the encryption process when the signal to begin is sent from AEScntx.

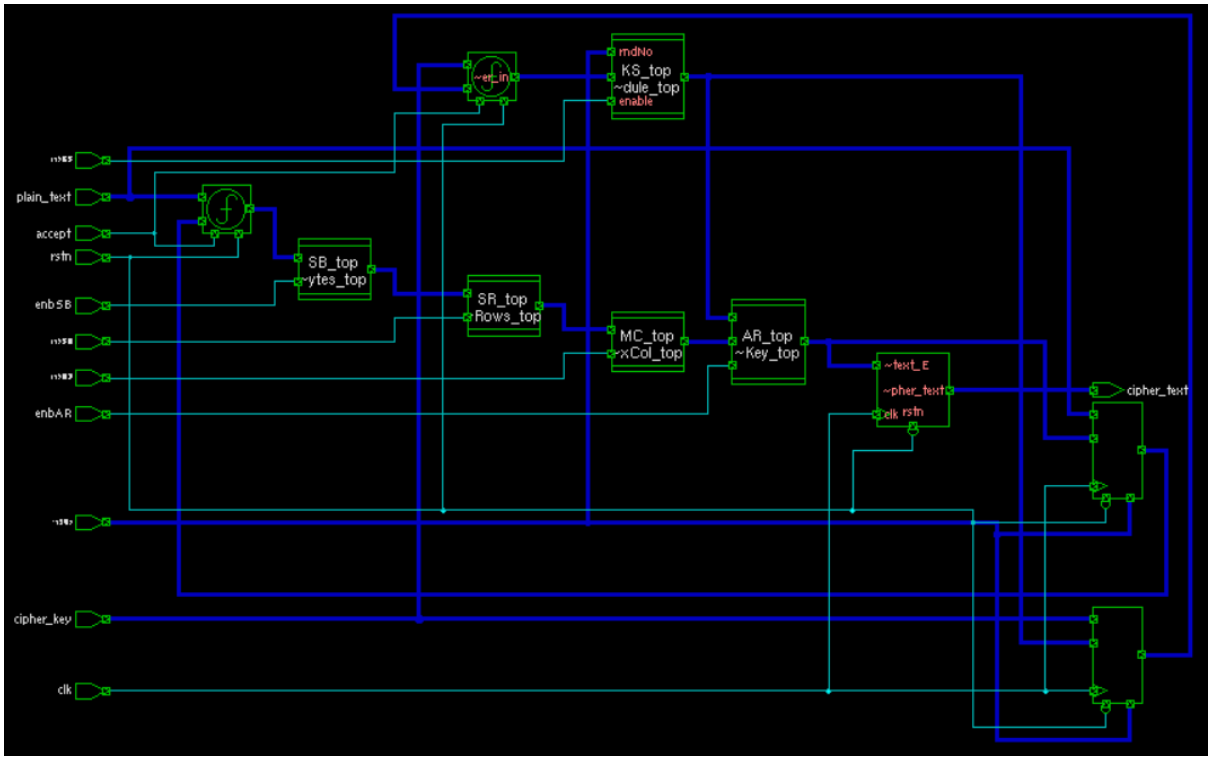

Figure 6: Schematic of AEScore

AESCore timing diagram:

Figure 7 shows the signal output from AESCore. Each connection between the main sub-modules performing the encryption is represented by the wires buffer_text_A, buffer_text_B, buffer_text_C, buffer_text_D, and buffer_text_E. The connections between the keyschedule module and the main sub-modules are represented using wires buffer_cipher_in and buffer_cipher_out. During each round, the signals pass through each of the wires and submodules and are depicted by the timing diagram. When the round is complete, the multiplexer notices the change in the input plain text and cipher key, updating the submodule inputs mid-clock, causing the submodules to show that 2 signals were processed in one clock cycle. However, the earlier signal is not propagated due to the behavior of the multiplexer, ensuring that the encryption process can continue smoothly.

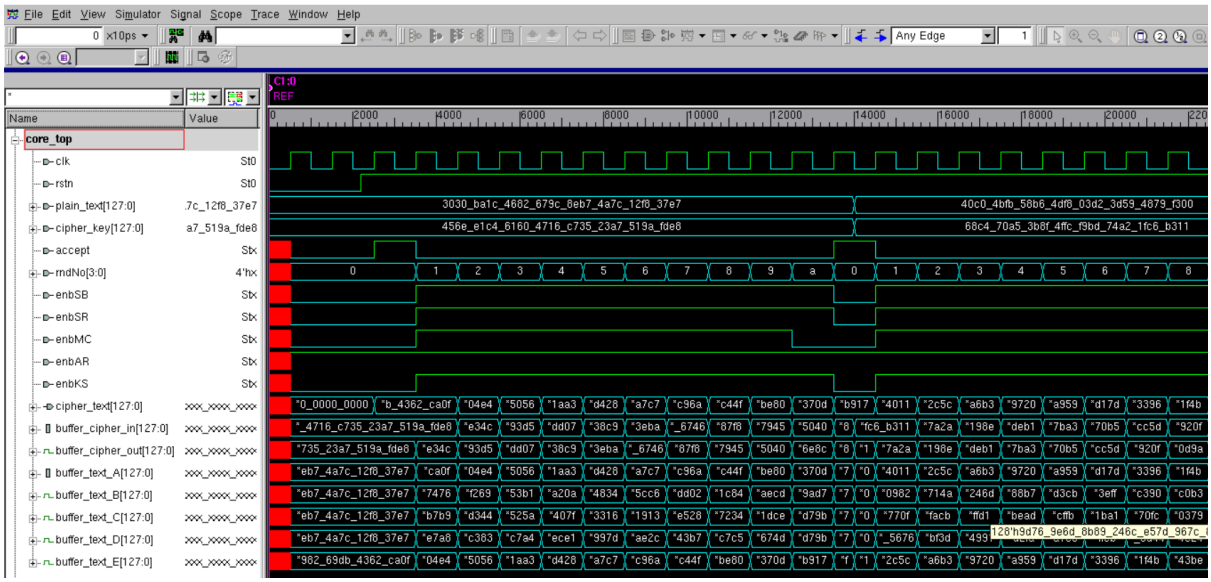

Figure 7: Timing Diagram for SISO AEScore

Subbytes schematic:

The subbytes module separates the 128-bit input into blocks of 8-bits and passes each block through the aes_sbox module, which processes the input using the lookup table used in Table 2. The result is then output to the next module depending on whether the module has been enabled. Placing the output multiplexer at the end of the cycle enables the input to buffer while waiting for an enable signal, speeding up the computation time.

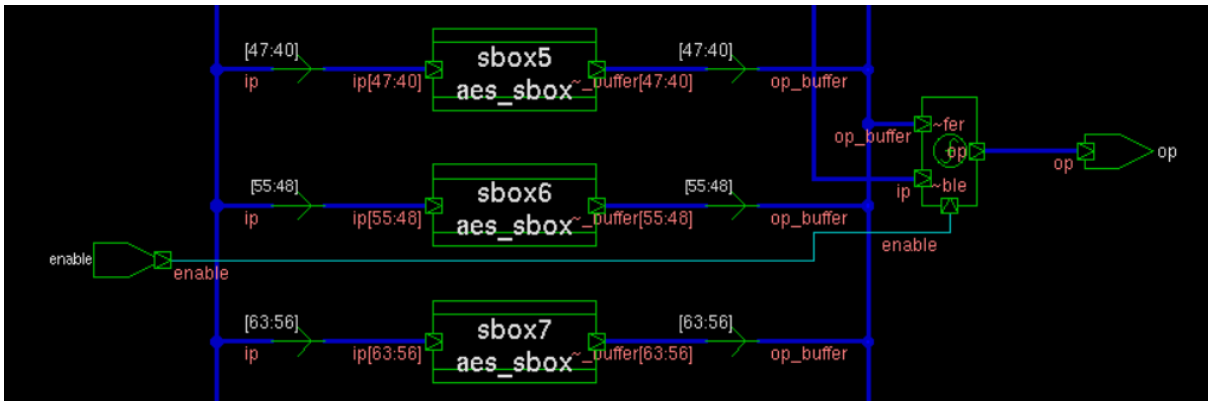

Figure 8: Schematic of Subbytes

Shiftrows schematic:

The shiftrows module assigns each byte of the 128-bit input to its new position based on the order shown in Table 3. The output is assigned to a multiplexer which determines the output based on the enable signal.

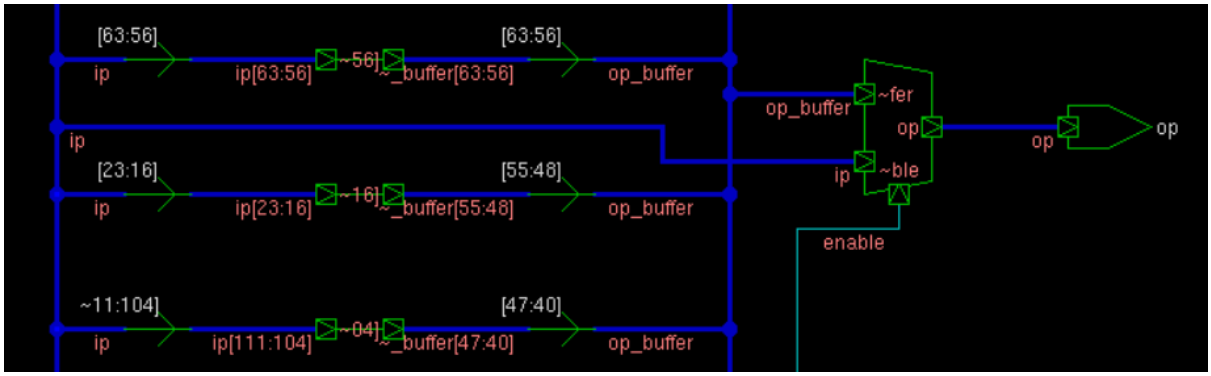

Figure 9: Schematic of Shift Rows

Mixcolumns schematic:

The mixcolumns module passes each column of 4 bytes into 4 smaller matrix_mult submodules which perform the rijndael galois matrix multiplication process on the input bytes. The result is then sent to the next module if the enable signal is high. Else, the output remains unchanged.

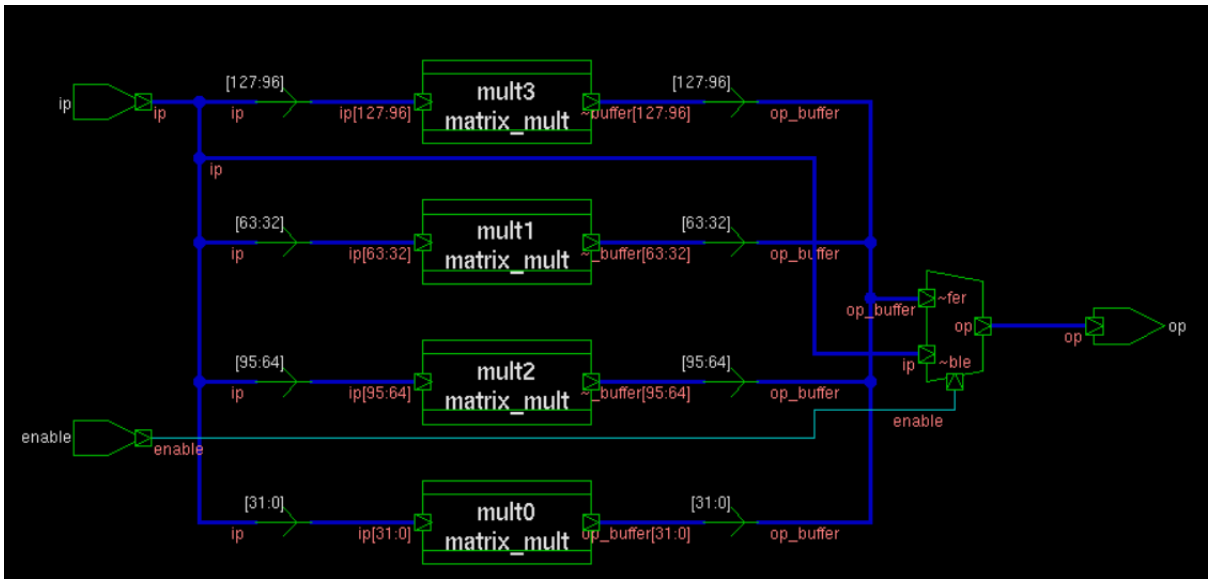

Figure 10: Schematic of Mix Columns

Addroundkey schematic:

The addroundkey module performs the XOR operation using the input and input key sent to it, and outputs the result to the buffer_text_E register for the computation of

the next round of encryption.

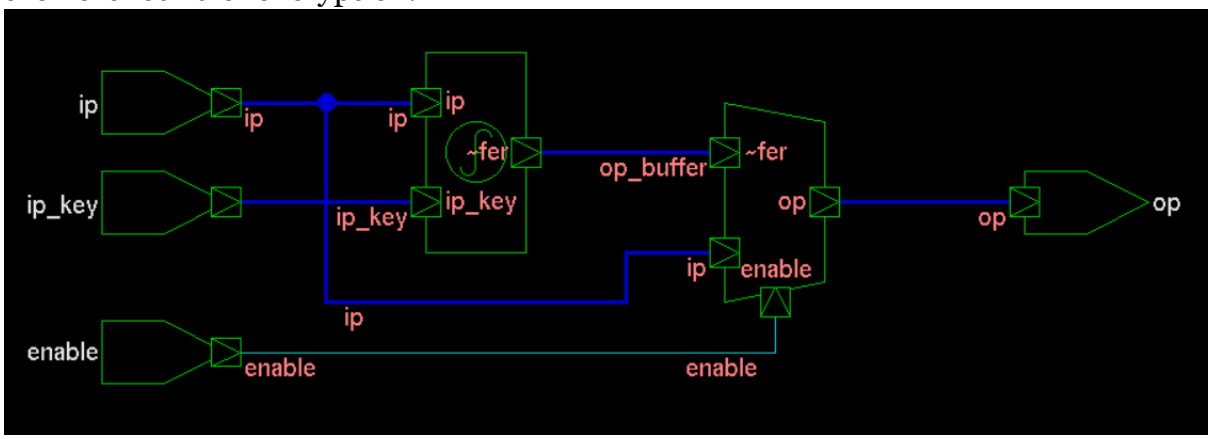

Figure 11: Schematic of Add Round Key

Keyschedule schematic:

The keyschedule module performs key expansion on the input cipher key and outputs a subkey based on the current round number. The subkey is then passed back to the input for the next round of computation with the addroundkey module.

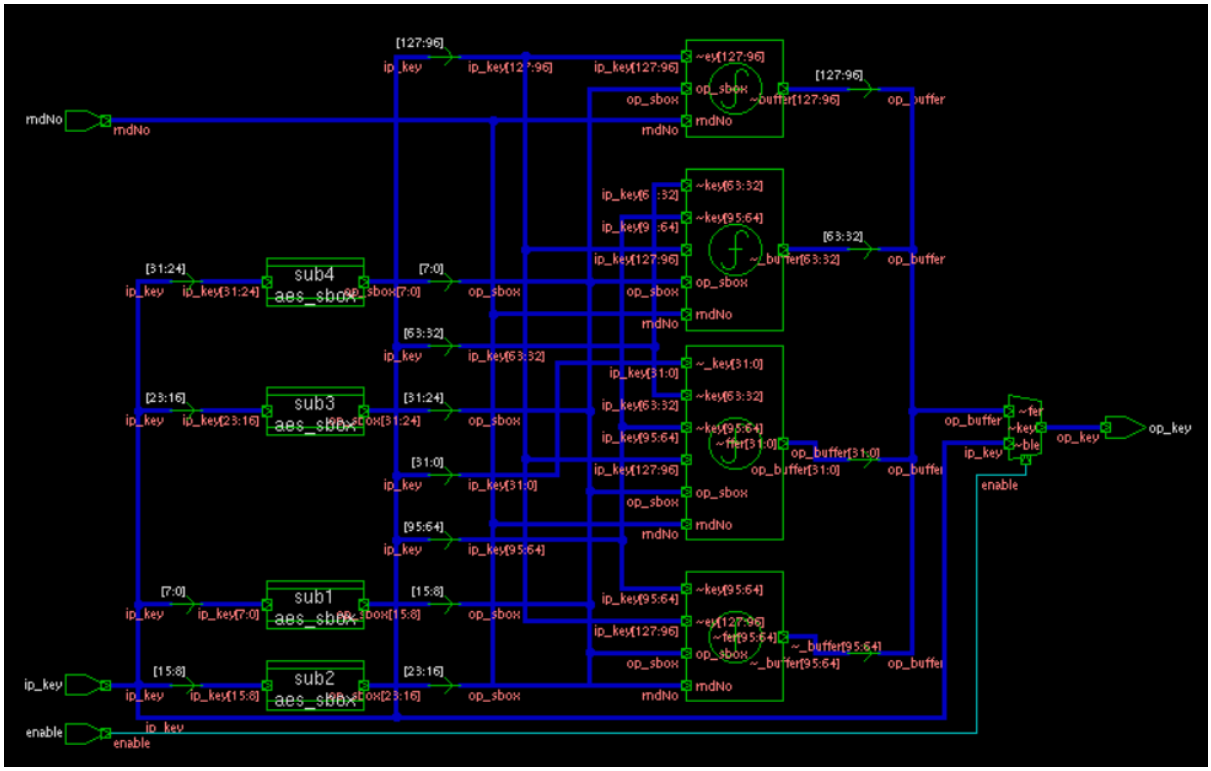

Figure 12: Schematic of Key Schedule

Speed and Throughput

From Figure 6, the critical path of the input signal is from the multiplexer at the subbytes module input, to the output of the addroundkeys module. The throughput of the machine is

$$\frac{1}{2ns} * \frac{1}{11} = 45.45MHz$$

Part 2: MIMO

In the MIMO implementation, a single stream of inputs is separated into multiple streams in parallel running concurrently. The output of the multiple streams is then recombined into a single output. This method greatly improves the throughput of the encryption machine.

Overview of MIMO AES implementation

In the MIMO implementation of the AES encryption machine, a single input is separated into N different streams, which are then sent to multiple copies of AEScore. They are controlled by one AEScntx module, producing one encrypted stream of encrypted cipher text at the end of every round of encryption.

AEStop Module

AEStop Schematic:

The AEStop module connects the AEScntx module with each AESCore module generated, as shown in Figure 13: Schematic for AESTop for MIMOFigure 13. The number of modules generated can be modified during synthesis using the code snippet shown in Figure 14. By varying N, the number of streams the input is separated into can be controlled increasing the throughput of the machine.

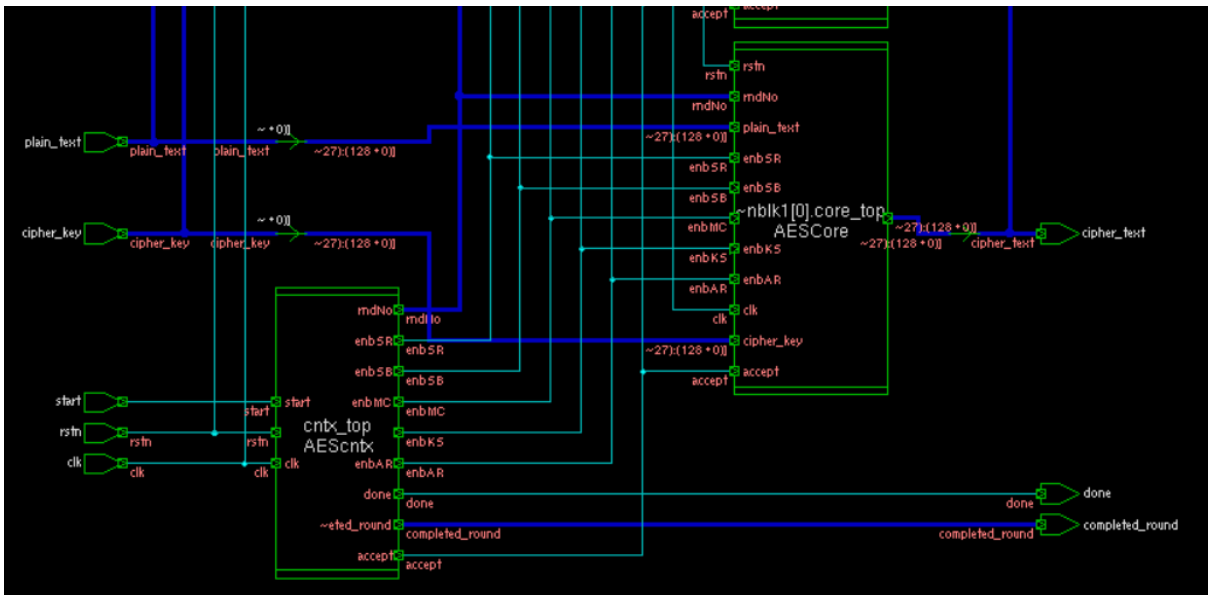

Figure 13: Schematic for AESTop for MIMO

```
genvar i;
generate for (i = 0; i < N; i = i + 1) begin
AESCore core_top(
    .clk(clk),
    .rstn(rstn),
    //from AEStop
    .plain_text(plain_text[128*i+127 : 128*i]),
    .cipher_key(cipher_key[128*i+127 : 128*i]),
    //from AEScntx
    .accept(accept),
    .rndNo(rndNo),
    .enbSB(enbSB),
    .enbSR(enbSR),
    .enbMC(enbMC),
    .enbAR(enbAR),
    .enbKS(enbKS),
    //to AEStop
    .cipher_text(cipher_text[128*i+127 : 128*i])
);
end endgenerate
endmodule
```

Figure 14: Code Snippet showing how multiple copies of AESCore are generated

AEStop timing diagram:

Figure 15 shows the timing diagram of the AEStop module. The input is a 1280-bit stream of words which are split into segments of 128-bits. Each 128-bit word is sent to each module of AESCore generated. At the output, the segments are recombined to form a 1280-bit encrypted stream.

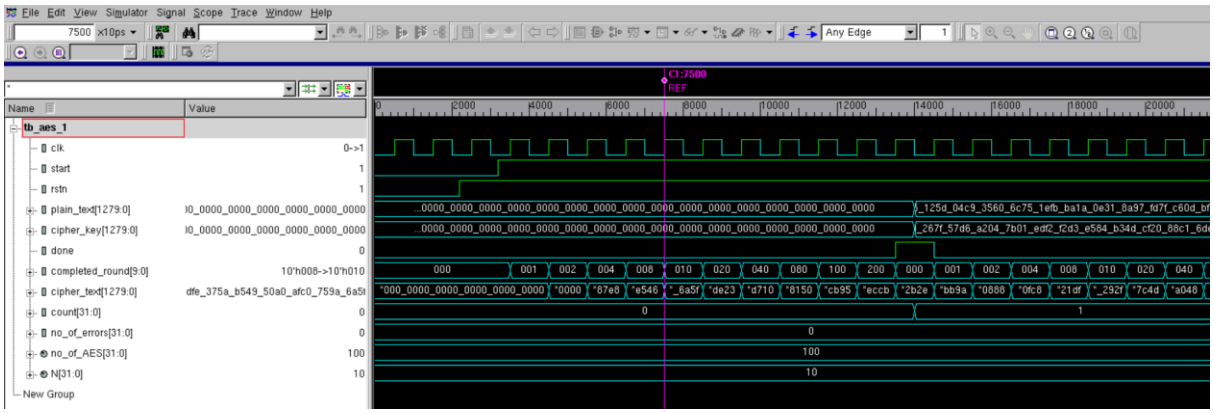

Figure 15: Timing Diagram for MIMO AES testbench

AEScntx Module

Schematic of AEScntx for MIMO:

The layout of AEScntx for the MIMO implementation is the same as the SISO implementation, as the timing of the machine does not need to be changed. Hence,

the same signals used in the SISO implementation can still be used in the MIMO case.

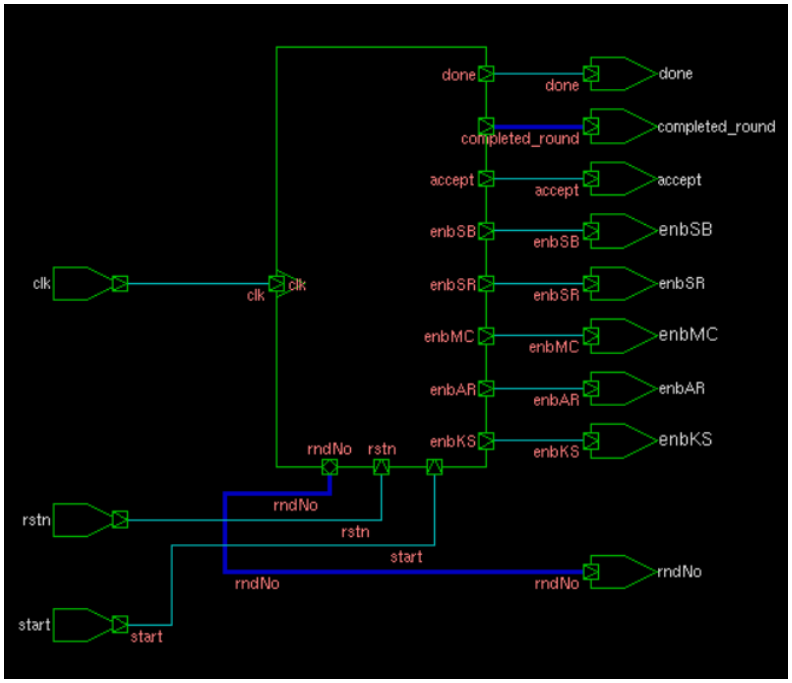

Figure 16: Schematic of AEScntx for MIMO

AESCore Module

Schematic of AESCore for MIMO:

The implementation of AESCore is the same for MIMO as the SISO case, as the input streams are separated at the AEStop module. Hence, the internal wiring of the AESCore module remains unchanged.

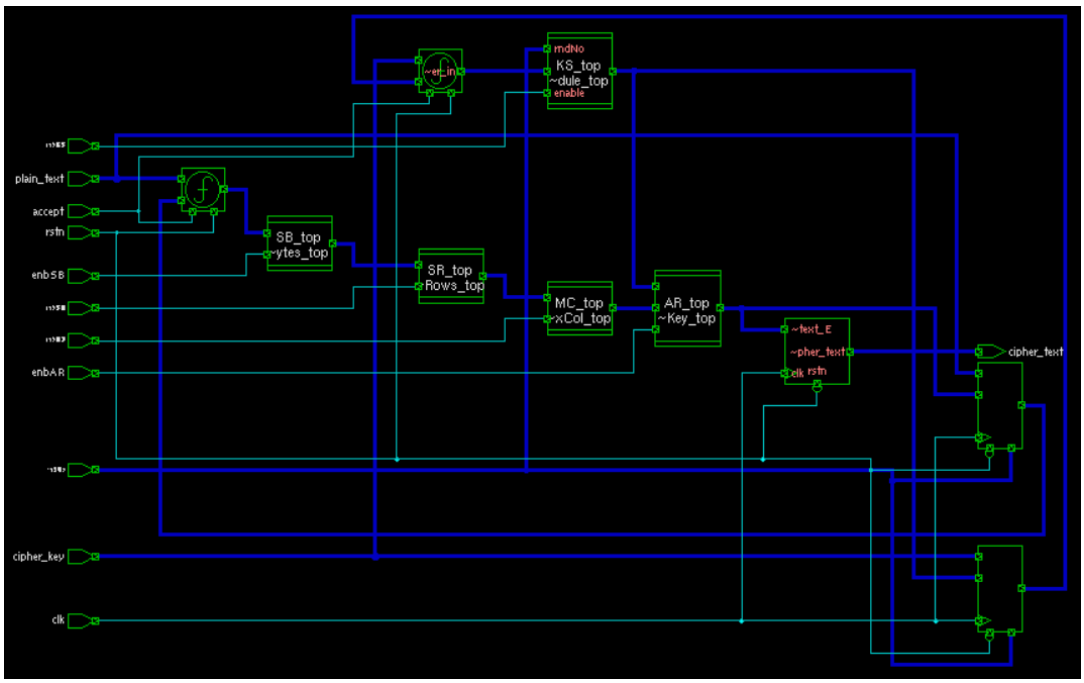

Figure 17: Schematic of AESCore for MIMO

Speed and Throughput

From Figure 17, the critical path for the signal is from the input of the multiplexer at the subbytes module, to the output of the addroundkeys module. The throughput of the machine is

$$\frac{1}{2ns} * \frac{4}{11} = 181.82MHz$$

Part 3: N-Slowing

In the N-slowed implementation, N inputs are sent in succession to generate N outputs in sequence. This is performed by adding registers to the machine to control the movement of the encrypted signal from one stage to the next, while allowing other signals in the midst of the encryption process to continue moving through the circuit simultaneously.

Overview of N-Slowed AES implementation:

In the N-slowed implementation of the AES machine, registers are added to the AESCore module to act as buffers for the input signal to travel through the circuit. In addition, the timing signals sent from the AEScntx modules are retimed to account for the additional register insertion. As a result, the movement of the encrypted signal in the AESCore module is slowed by N times.

AEStop Module

Schematic of AEStop for N-slowing:

The AEStop module is largely the same as the SISO implementation as the register insertion and retiming is mostly done within the AEScntx and AESCore modules. However, one difference between the AEStop implementation in SISO and N-slowing is the absence of the completed_round signal, which is no longer used by the testbench as it now relies on the done signal to capture data outputs from the cipher_text signal.

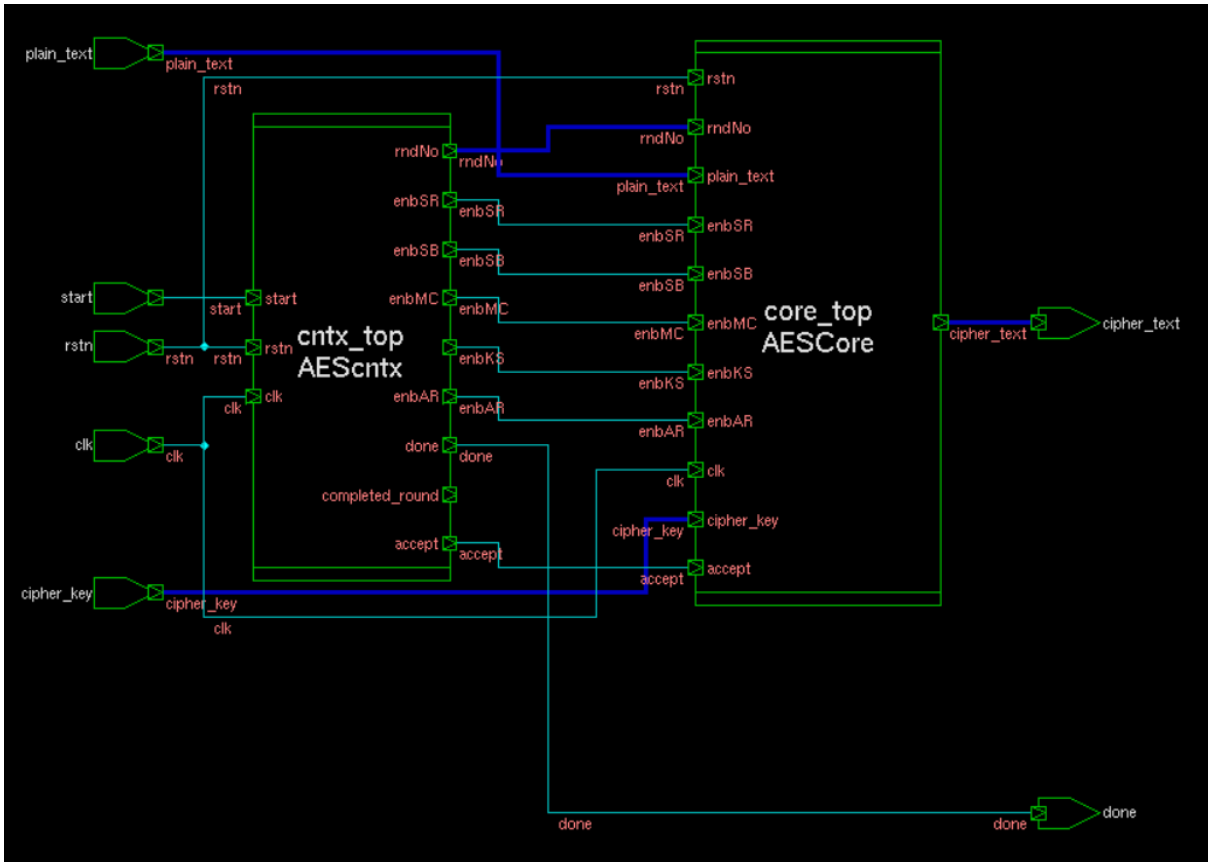

Figure 18: Schematic of AESTop for N-slowing

AEStop timing diagram:

Figure 19 shows the timing diagram for the AEStop module. When the signal has completed encryption, the done signal persists for N cycles to give time for each signal output to be stored by the testbench. In addition, new signals from plain_text and cipher_key arrive from the testbench for encryption.

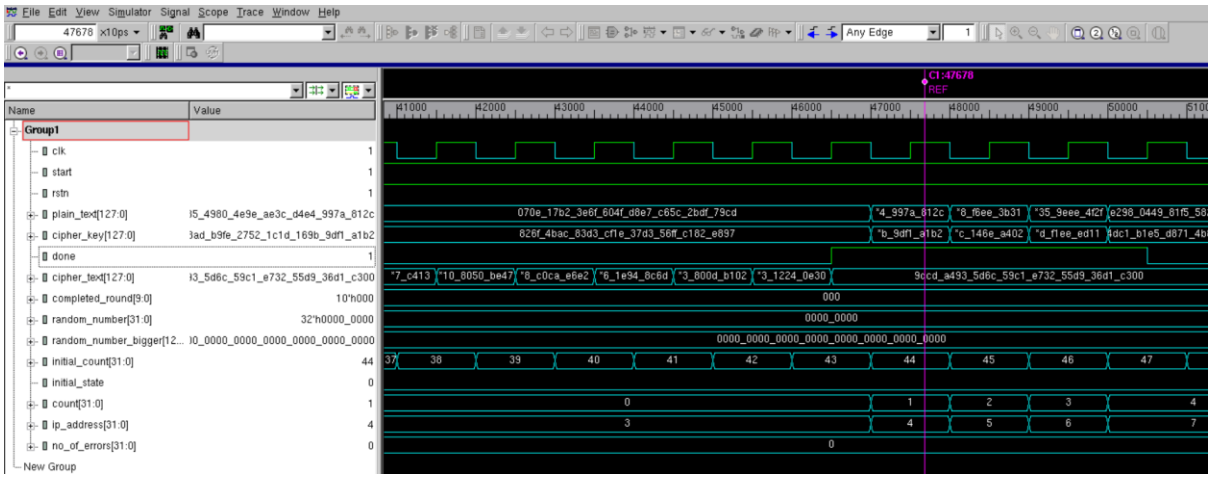

Figure 19: Timing Diagram of AES testbench for N-slowing

AEScntx Module

Schematic of AEScntx for N-slowing:

The AEScntx module adds in 2 sub-modules to the SISO implementation to implement the N-slowing of the clock cycles. The rndMult module implements an N counter to slow down the incrementing of the rndNo signal, while the rndNo_prev signal tracks the previous round number to time the transition of the final stage with the new stage.

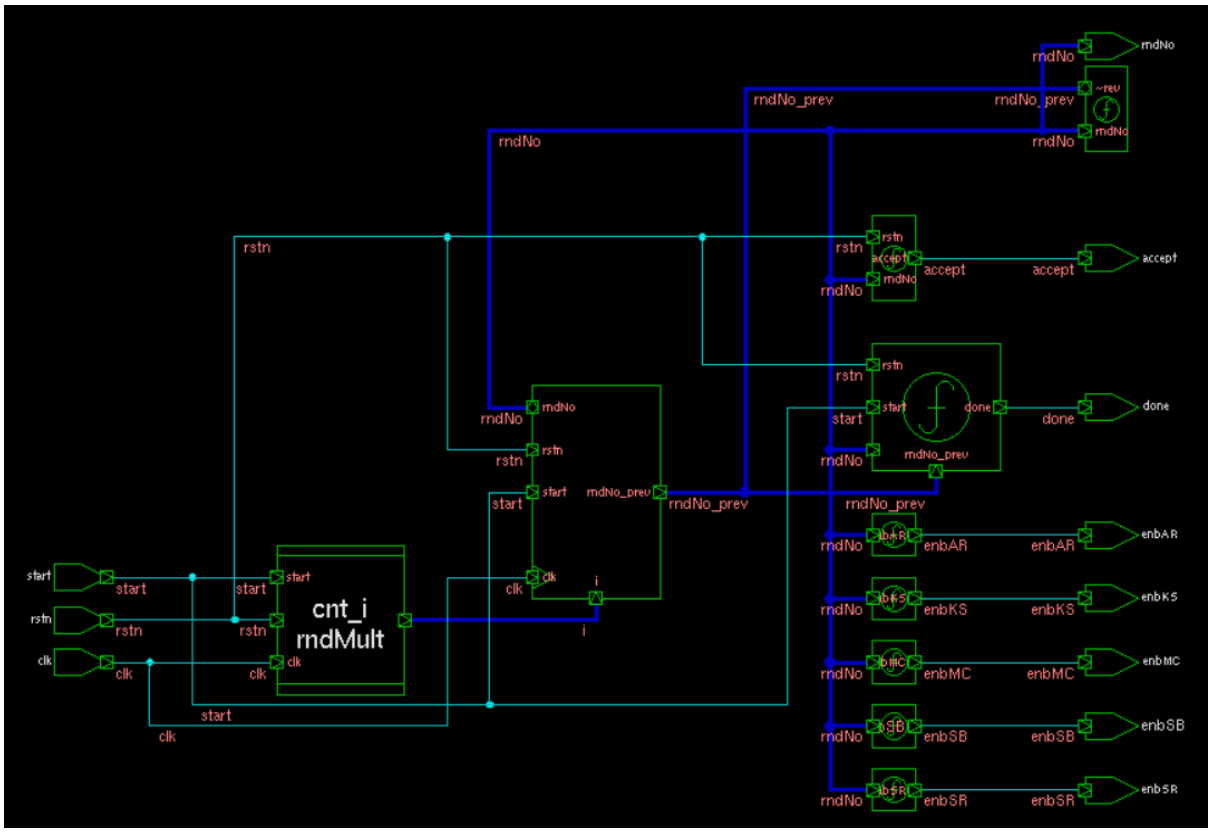

Figure 20: Schematic of AEScntx for N-slowing

AEScntx timing diagram:

Figure 21 shows the timing diagram for the AEScntx module. 2 additional counters, rndNo_prev and i, have been used to time the round transitions for the encryption. The counter i counts from 0 to 3 to track the current input being encrypted, while the rndNo_prev counter monitors the previous round number sent to the AESCore module.

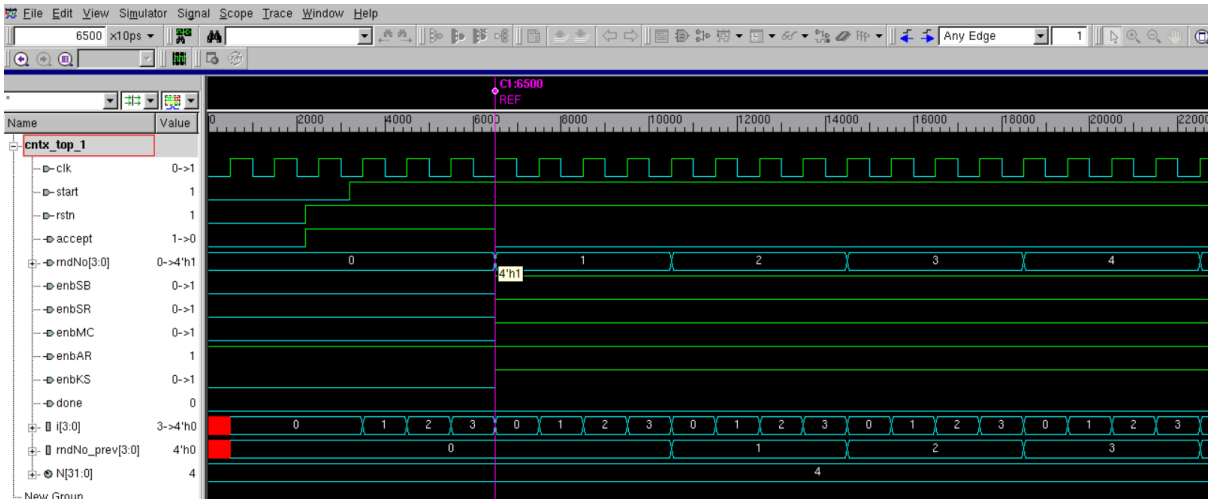

Figure 21: Timing Diagram of AEScntx for N-slowing

AESCore Module

Schematic of AESCore for N-slowing:

The AESCore module adds three additional registers to each register implemented in the SISO case to slow down the movement of the input plain_text signal and the cipher_key signal, so that several inputs can be present in the machine simultaneously. The registers are inserted at the Keyschedule module, the Subbytes module, and the cipher_text module.

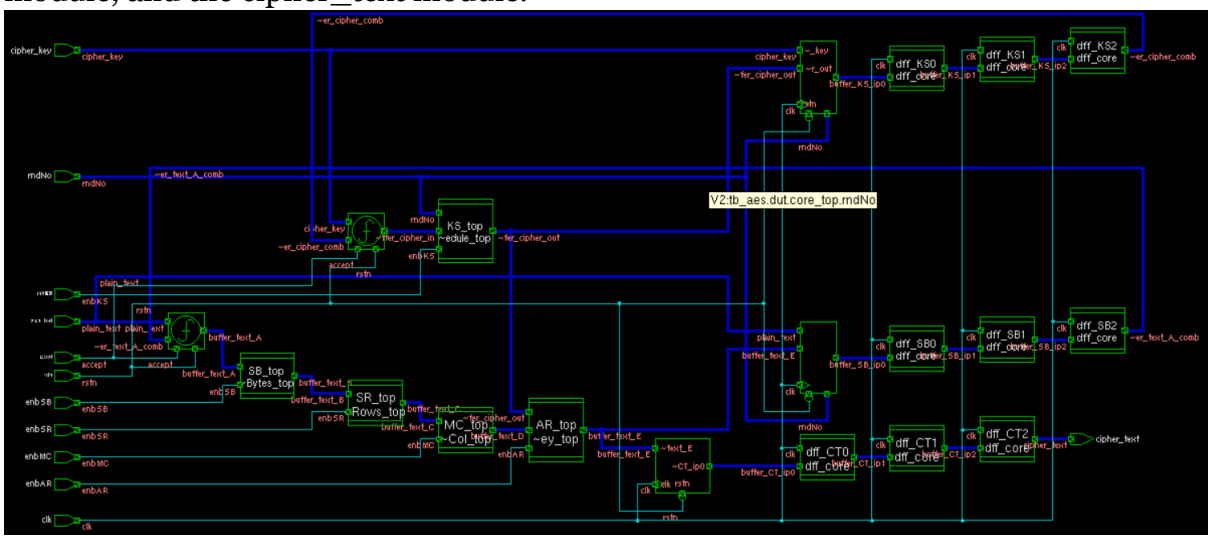

Figure 22: Schematic of AEScore for N-slowing

AESCore timing diagram:

Figure 23 shows the timing diagram of the AESCore module. When a signal is received at the multiplexers, they pass through several registers to store the signal during the encryption process. While moving from one register to the next, the timing of the enable signals coming from the AEScntx module is crucial to ensure that the signals are properly processed and are not lost during the transition between stages.

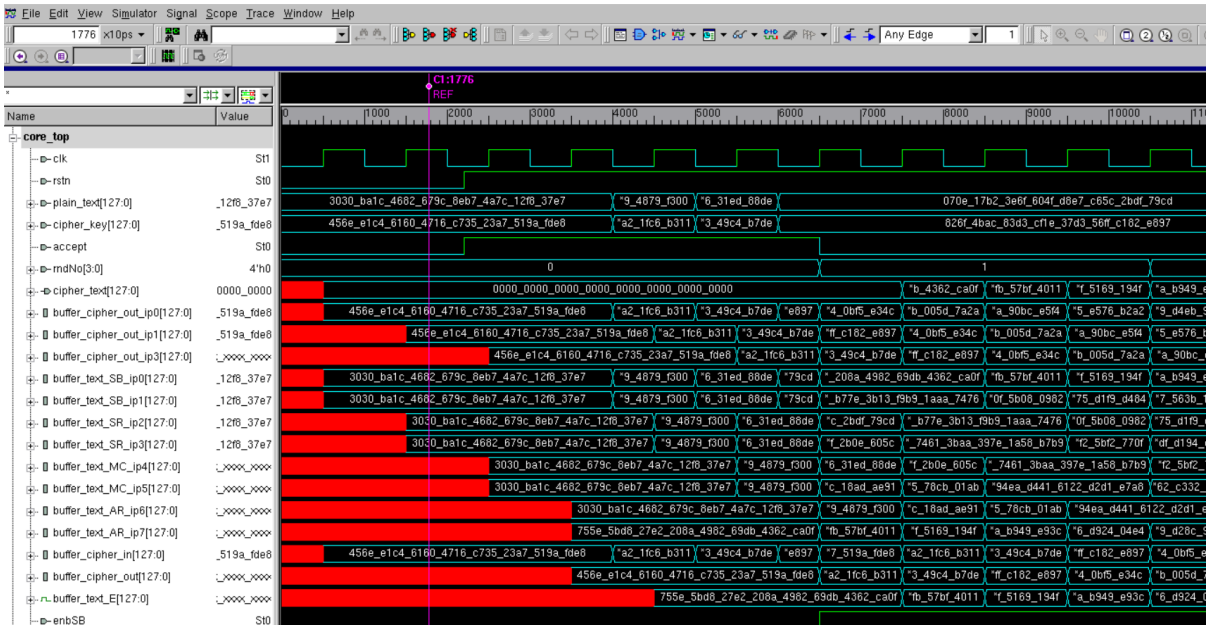

Figure 23: Timing Diagram of AEScore for N-slowing

Schematic of Keyschedule buffers for N-slowing:

Registers are inserted at the input of the multiplexer to the keyschedule module. This enables each register to hold one input, while one signal is sent into the keyschedule module on every positive clock edge.

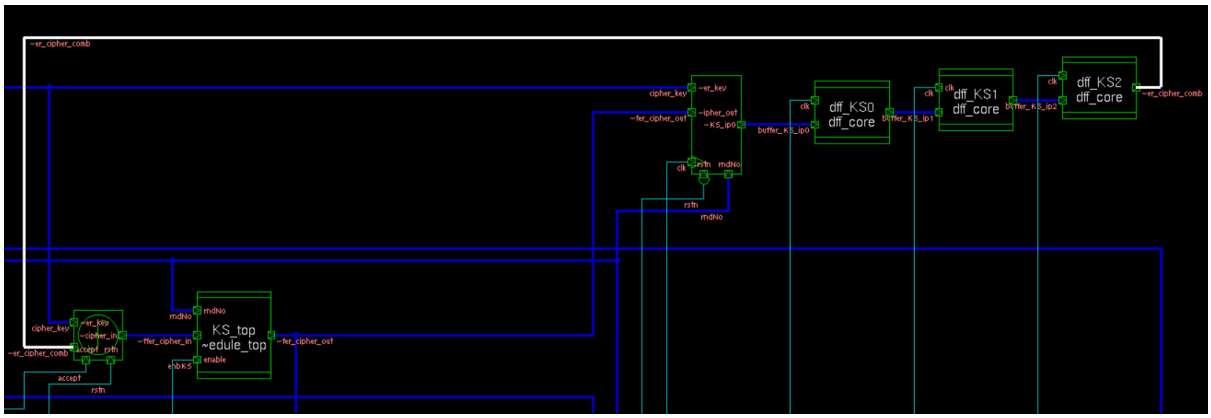

Figure 24: Schematic of Keyschedule buffers

Schematic of Subbytes buffers for N-slowing:

Registers are inserted at the input of the multiplexer to the Subbytes module as this module is the first module the input signal passes through in the encryption process. On every positive clock edge, the signal travels from one register to the next.

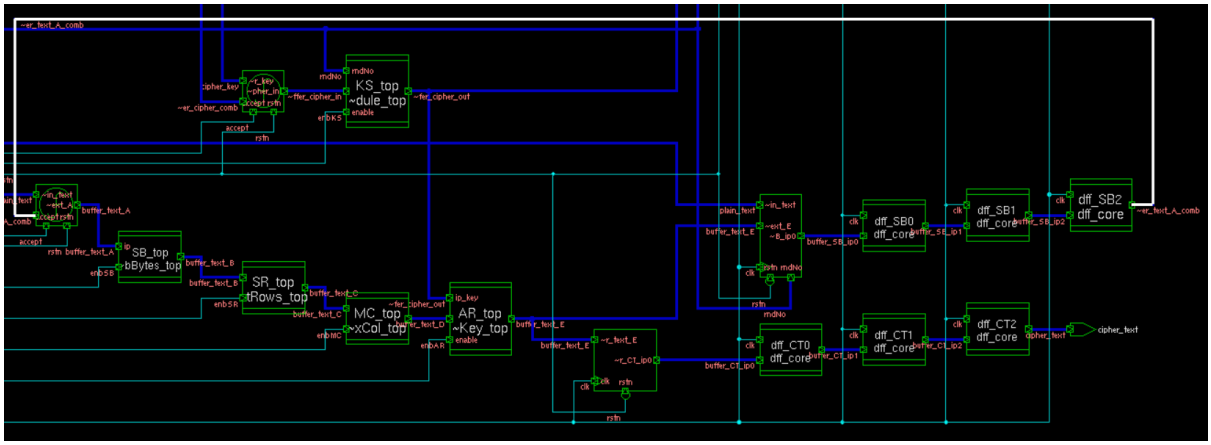

Figure 25: Schematic of Subbytes buffers

Schematic of Cipher text buffers:

The output of the AESCore module is also buffered, so that the internal timing of the signals as they travel within the circuit is also copied as the signals are output.

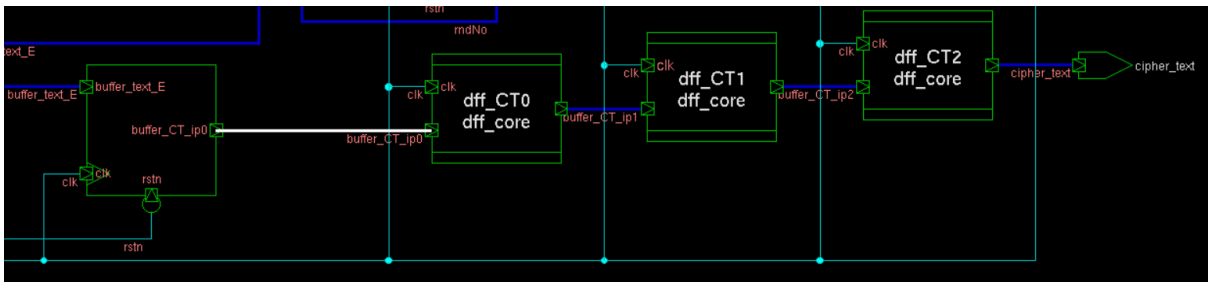

Figure 26: Schematic of Cipher Text buffers

Speed and Throughput

From Figure 22, the critical path for the signal is from the input of the multiplexer at the subbytes module, to the output of the addroundkeys module. The throughput of the machine is

$$\frac{1}{2ns} * \frac{4}{15} = 133.33MHz$$

Conclusion

In conclusion, AES encryption has been implemented using SISO, MIMO, and N-slowing. Of the three implementations, N-slowing was the most complex, but provided the most improvements in throughput, although latency was worse than MIMO and SISO.