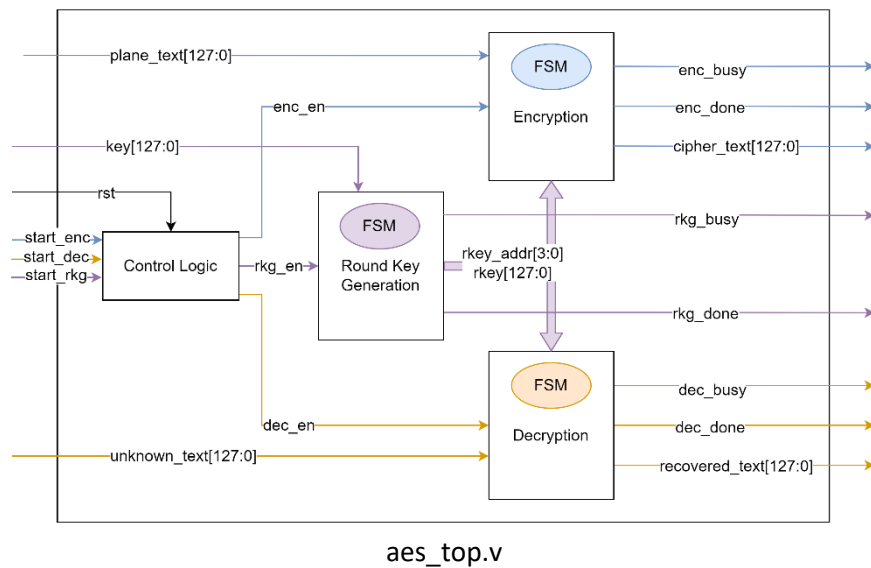


AES Hardware Implementation

Note: The architecture shown here corresponds to the design I have implemented in my code and the names of the variables are shown to be same as per the code for easy reference.

Design



The design consists of 3 modules

1. `aes_rkg()`
2. `aes_enc()`
3. `aes_dec()`

Each of the 3 modules have associated control signals and are controlled by the control logic in the `aes_top()` module. Each of the 3 modules run independent FSMs.

The `aes_rkg()` module computes the 10 round key values and these values are stored in both the `aes_enc()` and `aes_dec()` separately to keep the modules independent.

This design was targeted for avnet minized board which is not very resource intensive and hence LUT usage has been kept at the less as possible.

Eg. The column mixing operation in both encryption and decryption is implemented using **Galois Multiplication lookup tables** and is not an actual multiplication or shift operation!!!

All the 3 modules take 10 cycles each to complete one rkg generation/encryption/decryption.

The clock used in the implementation is 50MHz.

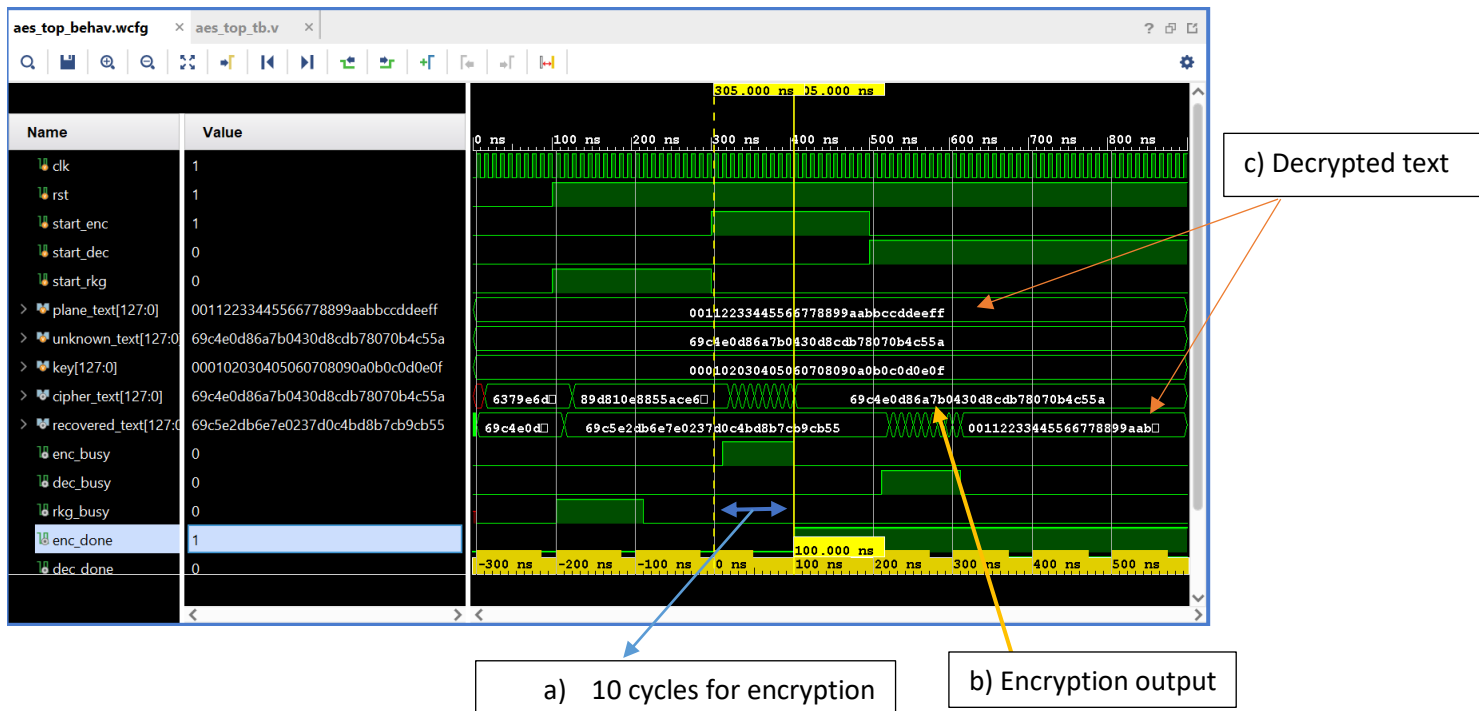
Simulation

The following screenshot shows the test example for following inputs

Key: **0x000102030405060708090a0b0c0d0e0f**

Plane Text: **0x00112233445566778899aabbccddeeff**

Cipher Text: **0x69c4e0d86a7b0430d8cdb78070b4c55a**



As can be seen from the waveform,

Throughput:

$$\text{Throughput} = \frac{50 \text{ Mhz}}{10 \text{ cycles}} = 5 * 10^6 \text{ enc/dec per second} \quad (\text{neglecting initial 10 cycles for rkg})$$

Latency:

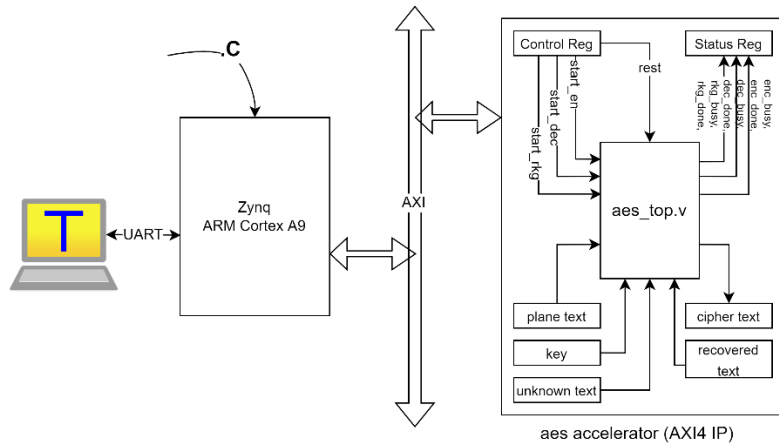
$$\text{Latency} = 10(\text{rkg}) + 10 \text{ cycles} = 20 \text{ cycles}$$

Area:

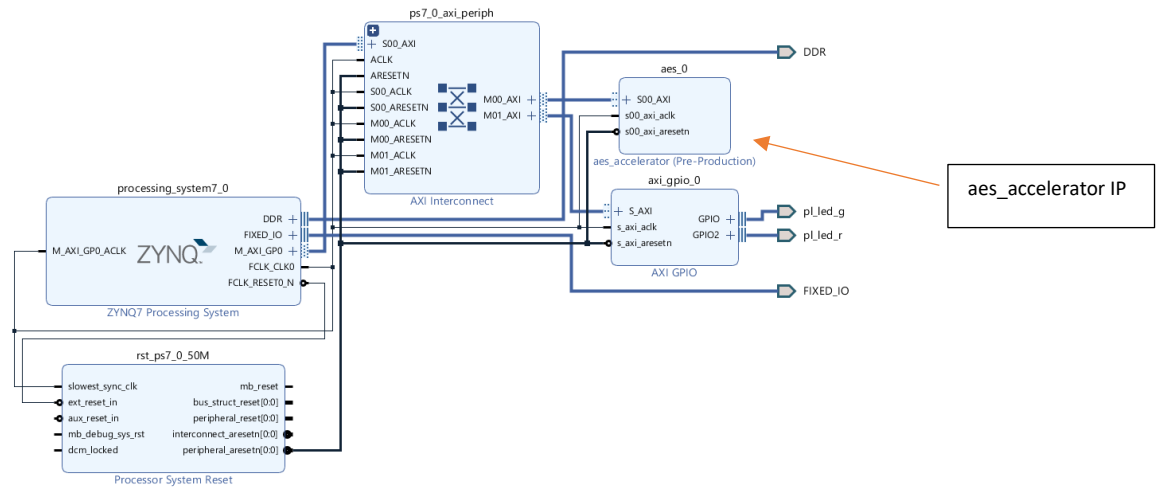
| Log | Reports | Design Runs | Utilization | Package Pins | I/O Ports | | |
|-----------------|--------------------|-------------------------|-----------------|-----------------|---------------------|-----------------|---------------|
| | | | | Hierarchy | | | |
| Name | Slice LUTs (14400) | Slice Registers (28800) | F7 Muxes (8800) | F8 Muxes (4400) | Block RAM Tile (50) | Bonded IOB (54) | BUFGCTRL (32) |
| ▼ aes_top | 3711 | 1834 | 336 | 128 | 1.5 | 651 | 1 |
| > DEC (aes_dec) | 1622 | 143 | 0 | 0 | 0 | 0 | 0 |
| > ENC (aes_enc) | 1921 | 1552 | 336 | 128 | 0 | 0 | 0 |
| RKG (aes_rkg) | 168 | 137 | 0 | 0 | 0.5 | 0 | 0 |

Testing

aes_top() module was then packaged into an AXI4 IP and interfaced with ARM Cortex A9 processor on the ZYNQ chip on Minized Board.



The block diagram of the entire system looks as follows



The inputs and outputs of the aes_top() module were mapped to registers of the AXI packaged IP.

| Register | Offset |
|----------------|-----------|
| Status | 0x00 |
| Control | 0x04 |
| Key | 0x08-0x20 |
| PlaneText | 0x24-0x36 |
| Unknown Cipher | 0x40-0x52 |
| CipherText | 0x56-0x68 |
| Recovered Text | 0x72-0x84 |

This design was then exported to Xilinx SDK and a simple test program was written in C to test the aes accelerator.

```

AES_MINIZED.sdk - C/C++ - aes/src/helloworld.c - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Window Help

Project Explorer
  Binaries
  Includes
  Debug
  src
    helloworld.c
    platform_config.h
    platform.c
    platform.h
    platformId
    Xilinx.spec
  aes_bsp
  AES_MINIZED_wrapper_hw_platform_0
  aes_software
    Binaries
    Includes
    Debug
    src
    aes_software_bsp

helloworld.c
  void display_status()
  void display_input()
  void display_output(uint8_t type)

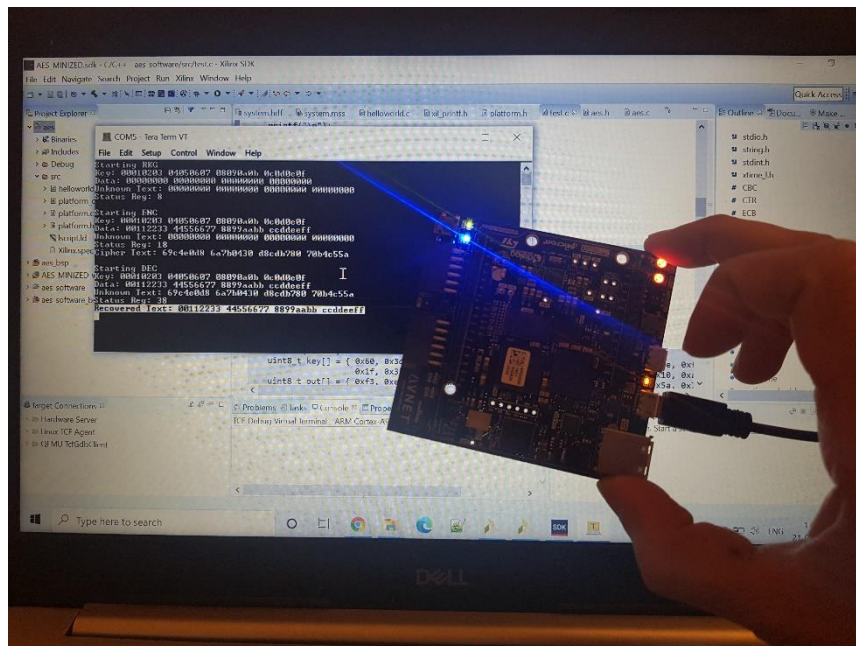
int main()
{
  init_platform();

  //Get system out of reset, start RKG
  print("Starting RKG\n");
  Xil_Out32(XPAR_AES_0_S00_AXI_BASEADDR+8, 0x0010203);
  Xil_Out32(XPAR_AES_0_S00_AXI_BASEADDR+12, 0x04050607);
  Xil_Out32(XPAR_AES_0_S00_AXI_BASEADDR+16, 0x08090a0b);
  Xil_Out32(XPAR_AES_0_S00_AXI_BASEADDR+20, 0xc0d0e0f);
  Xil_Out32(XPAR_AES_0_S00_AXI_BASEADDR+4, 0x03);
  sleep(2);
  display_input();
  display_status();

  //start ENC
  print("\n\nStarting ENC\n\n");
  Xil_Out32(XPAR_APS_0_S00_AXT_BASEADDR+24, 0x00112233);

```

Result



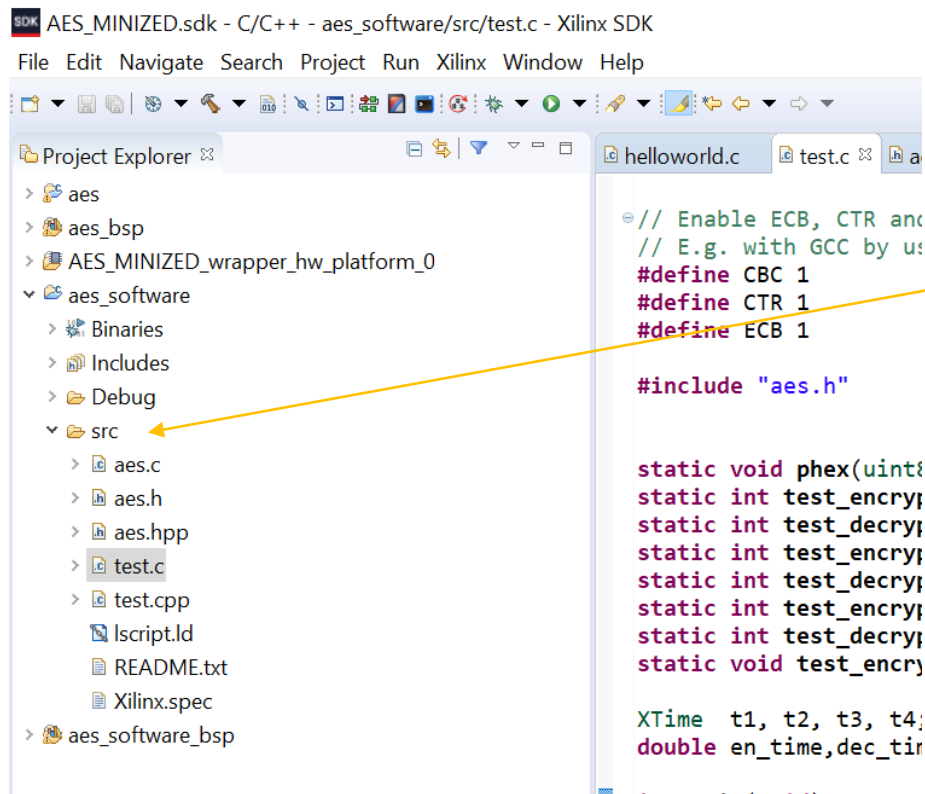
```
COM5 - Tera Term VT
File Edit Setup Control Window Help
Starting RKG
Key: 00010203 04050607 08090a0b 0c0d0e0f
Data: 00000000 00000000 00000000 00000000
Unknown Text: 00000000 00000000 00000000 00000000
Status Reg: 8

Starting ENC
Key: 00010203 04050607 08090a0b 0c0d0e0f
Data: 00112233 44556677 8899aabb ccddeeff
Unknown Text: 00000000 00000000 00000000 00000000
Status Reg: 18
Cipher Text: 69c4e0d8 6a7b0430 d8cdb780 70b4c55a

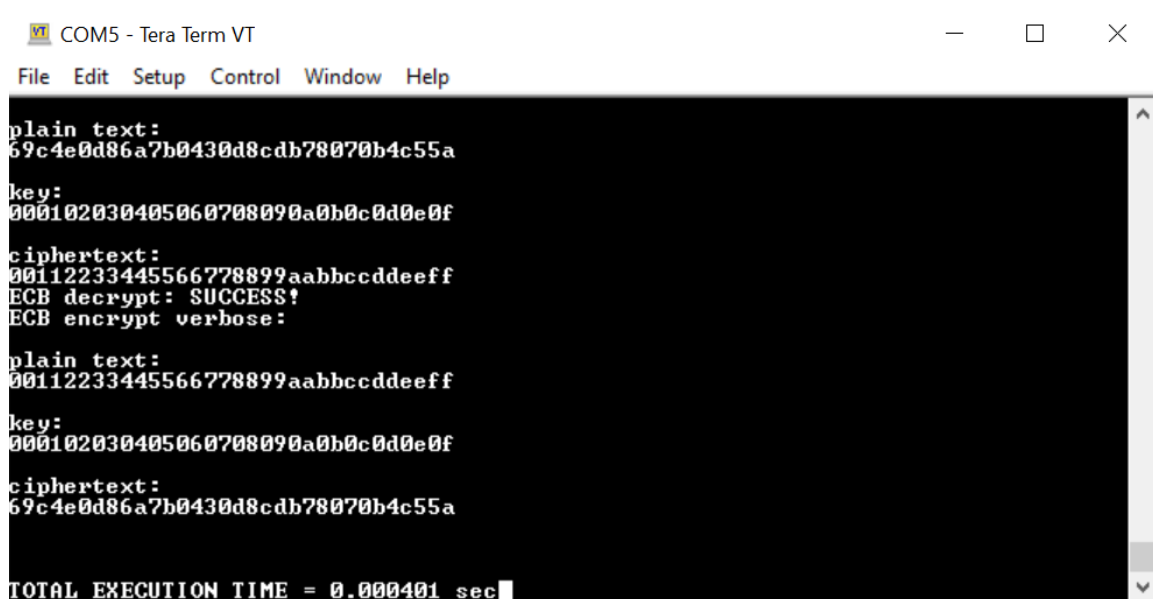
Starting DEC
Key: 00010203 04050607 08090a0b 0c0d0e0f
Data: 00112233 44556677 8899aabb ccddeeff
Unknown Text: 69c4e0d8 6a7b0430 d8cdb780 70b4c55a
Status Reg: 38
Recovered Text: 00112233 44556677 8899aabb ccddeeff
```

Appendix A: Hardware vs Software

Additionally the aes encryption was run on the Single Core ARM Cortex A9 processor entirely in software as well!



Result:



Comparison:

| | Software | Hardware |
|--------------------|----------|----------|
| Clock(MHz) | 666.67 | 50 |
| Execution Time(us) | 401 | 0.6 |

The hardware Implementation was almost 668.3 times faster.