# Pipelined AES Encryption

**Prepared by:**

Md Mahfooz Ansari

## Design Architecture

### Two-Stage Pipelined AES

The AES encryption is implemented as a fully pipelined architecture, in which each round is split into two synchronized stages. All intermediate values are stored in arrays of 128-bit registers so that multiple blocks can be processed concurrently, so that we get a ciphertext from the pipeline every clock cycle once it is filled.

At the input, the plaintext is first combined with the key to perform the initial key addition. The result of this operation forms the starting state for the first round. From this point onward, each round is decomposed into:

- **Stage 1 (Sub Byte and Shift rows stage):** The state from the previous round is passed through the standard AES operation Sub Bytes and Shift rows. The output of this combined transformation is written into a dedicated 128-bit register. This register forms the first pipeline boundary of the round. This helps in doing subsequent operation in next cycle , thus improving the achievable clock frequency.
- **Stage 2 (Mix column and Add Round Key stage):** In rounds 1 through 9, the registered state is then processed by the Mix column followed by an exclusive-OR with the corresponding round key. The result is stored into another 128-bit register, which becomes both:
  - the input state for the next round's Stage 1, and
  - a pipeline element that allows new plaintext blocks to enter the front of the pipeline every clock cycle.

  For the final (10th) round, the design preserves the same two-stage register structure, but the mix column is omitted. The second stage of the final round therefore consists only of the final key addition.

Every combinational block (Stage) between registers implements exactly one logically grouped step of the AES round function, and every transition between these groups (Stages) is registered. This creates a uniform two stage pipeline per round and enables high throughput: after an initial latency of 20 clock cycles, a new 128 bit ciphertext is produced every clock cycle.

### Key Expansion in the Two-Stage Pipeline

The round key generation is done in parallel with the data path so that each stage receives its corresponding round key in the correct cycle without additional control complexity.

The design uses two 128-bit registers to hold intermediate keys. For the first round, the next subkey is generated directly from the original key using the AES-128 key schedule operations. This derived key is immediately stored in a register that aligns with the first round's Stage 2, where the first mixing and key addition take place. In parallel, a second register bank captures the same subkey to serve as the starting point for computing the following round key.

For the later rounds, each subkey is derived solely from the previously stored subkey, independent of the data path. At round index $i$, the key schedule uses the registered key for that round, applies the AES key expansion to generate the key for round $i + 1$, and loads it into a new register. That new key is then:

1. Forwarded to the Stage 2 key input of the corresponding data round, runs parallel with Stage 1 of that round.

2. Stored into the second key register which is used for generating the next round key.

Because both the data path and the key schedule advance one stage per clock cycle and share the same register spacing, the correct round key naturally meets its associated data block at the appropriate pipeline stage.

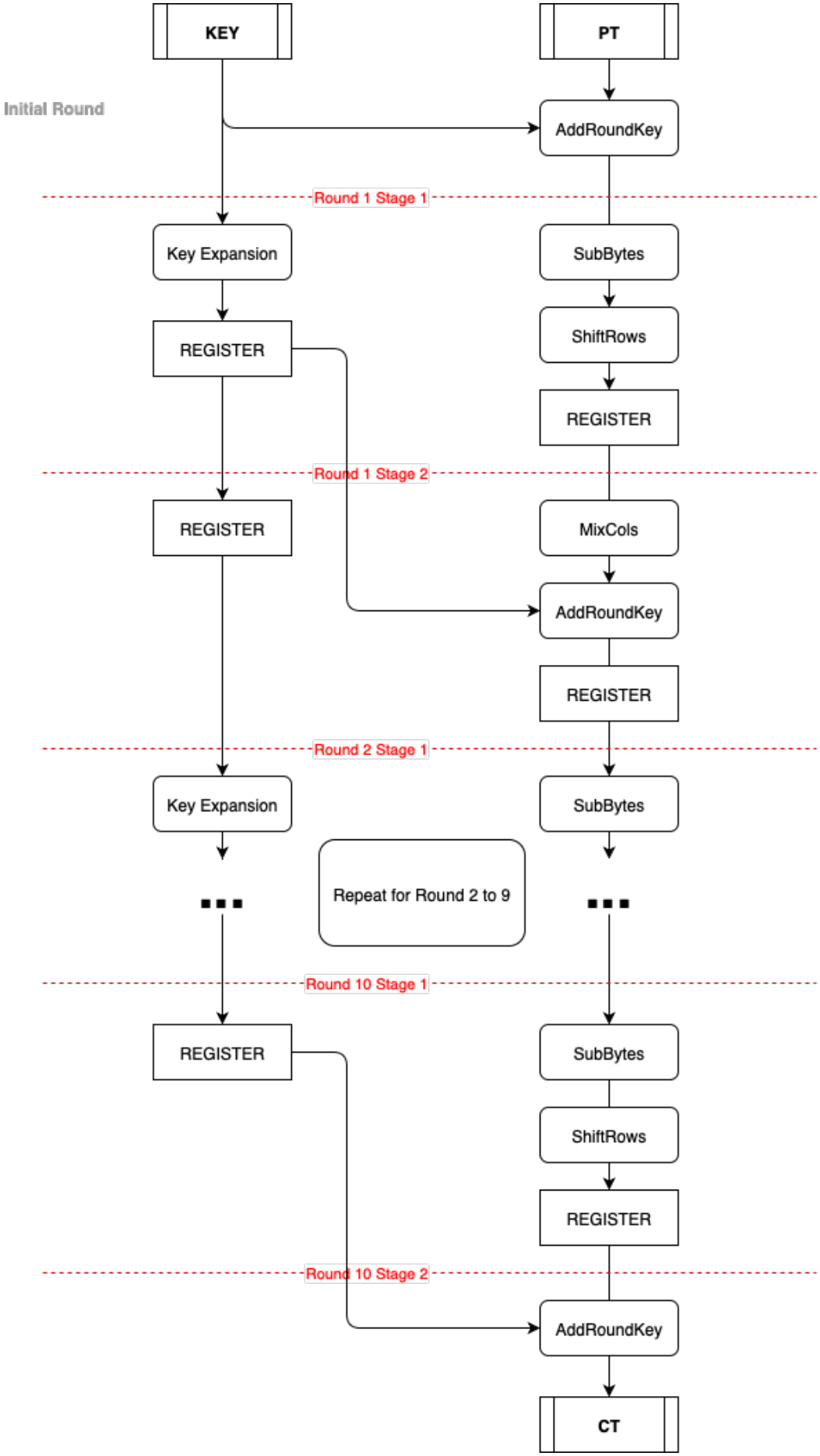


Figure 1: Implemented Architecture

## Performance Metrics:

### Throughput

In our pipelined architecture, after the initial setup period, the design processes one complete 128-bit block every clock cycle. The throughput is therefore directly proportional to the operating frequency we achieve on the PYNQ board:

$$\text{Throughput} = 128 \times f_{\max} \quad \text{bits/second} = 128 \times 214.286 \times 10^6 = 27.428 \text{ Gbps} \tag{1}$$

Where $f_{\max}$ is the maximum clock frequency in MHz.

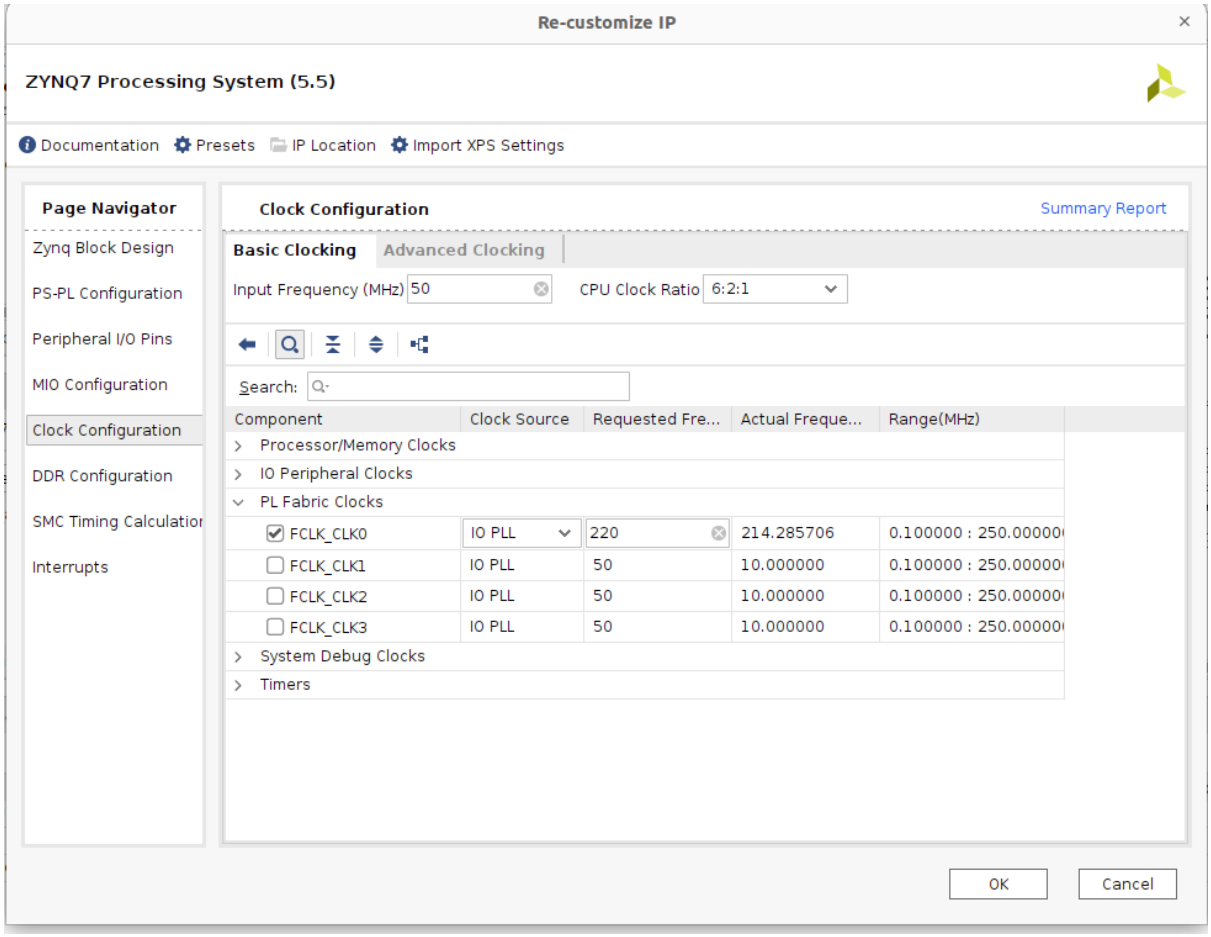


Figure 2: Clock Frequency

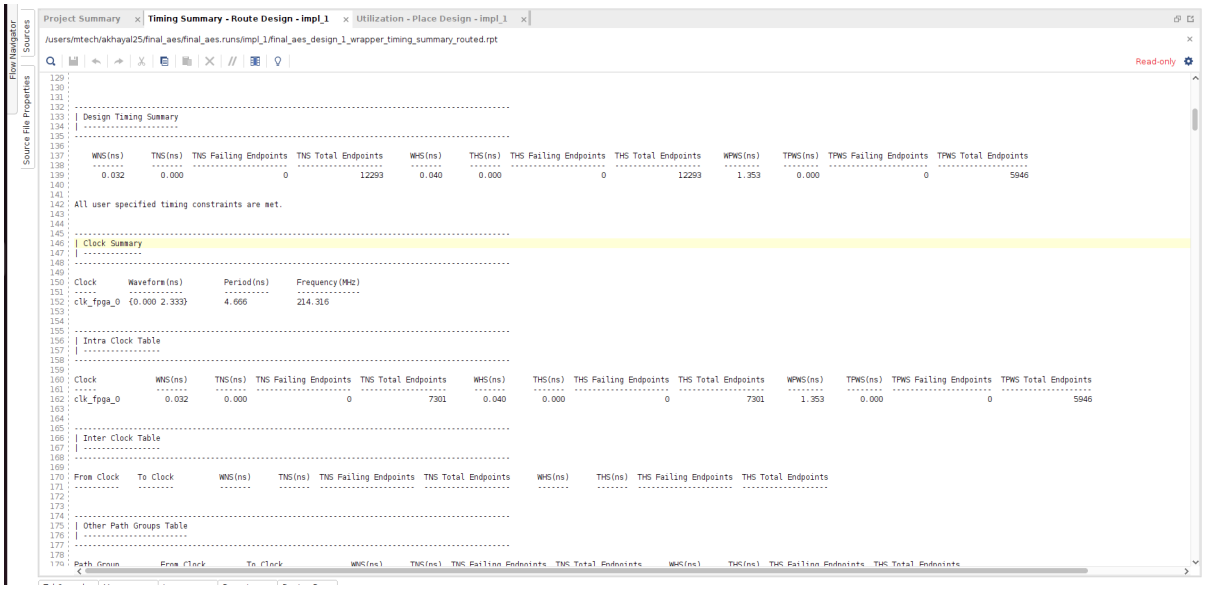


Figure 3: Timimg Summary

The timing summary for the pipelined AES design on the PYNQ-Z2 board shows successful timing closure with a positive slack at around **214 MHz**. This confirms that the design meets all setup and hold constraints and can operate reliably at high speed, demonstrating the efficiency of the pipelined AES architecture on FPGA.

## Area Utilization

Total number of slices: 2958

2. Slice Logic Distribution

| Site Type | Used | Fixed | Prohibited | Available | Util% |
|---|---|---|---|---|---|
| Slice | 2958 | 0 | 0 | 13300 | 22.24 |
| SLICEL | 2146 | 0 | | | |
| SLICEM | 812 | 0 | | | |
| LUT as Logic | 10389 | 0 | 0 | 53200 | 19.53 |
| using O5 output only | 0 | | | | |
| using O6 output only | 10191 | | | | |
| using O5 and O6 | 198 | | | | |
| LUT as Memory | 60 | 0 | 0 | 17400 | 0.34 |
| LUT as Distributed RAM | 0 | 0 | | | |
| LUT as Shift Register | 60 | 0 | | | |
| using O5 output only | 0 | | | | |
| using O6 output only | 56 | | | | |
| using O5 and O6 | 4 | | | | |
| Slice Registers | 5881 | 0 | 0 | 106400 | 5.53 |
| Register driven from within the Slice | 4171 | | | | |
| Register driven from outside the Slice | 1710 | | | | |
| LUT in front of the register is unused | 618 | | | | |
| LUT in front of the register is used | 1092 | | | | |
| Unique Control Sets | 82 | | 0 | 13300 | 0.62 |

Figure 4: Area: Number of slices used in implemented design

The **Throughput / Area** measures how efficiently the design uses hardware slices to achieve high performance.

$$\begin{aligned}\text{Throughput/Area} &= \frac{\text{Throughput}}{\text{Area (Slices)}} \\ &= \frac{27.428}{2,958} \\ &= 0.00927 \quad \text{Gbps per Slice} \\ &= 9.27 \quad \text{Mbps per Slice}\end{aligned}$$

## Initial Latency

The first ciphertext appears after 20 clock cycles, and subsequent ciphertexts are produced every cycle. Hence, the initial latency of the pipeline is 20 cycles. In the below image, we have given the first plain text at 25ns. Hence, we are getting the first cipher text at 225ns i.e. after 20 cycles.

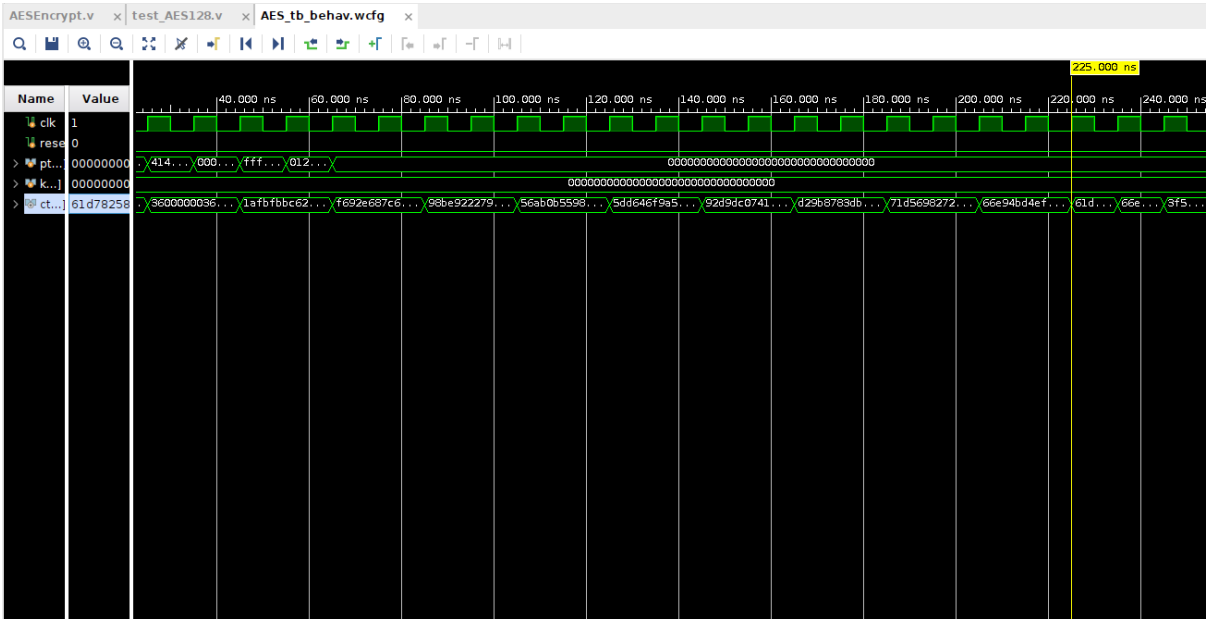


Figure 5: Gtkwave: First Ciphertext after 20 cycles

## Results

```
VCD info: dumpfile aes_waveform.vcd opened for output.
Plaintext : 41424344454647484a4b4c4d4e4f50
Key       : 00000000000000000000000000000000
Ciphertext: 61d78258eb1abd6fff479d1dabb6103b
Expected  : 61d78258eb1abd6fff479d1dabb6103b
>>> TEST 1 PASSED <<<
Plaintext : 00000000000000000000000000000000
Key       : 00000000000000000000000000000000
Ciphertext: 66e94bd4ef8a2c3b884cfa59ca342b2e
Expected  : 66e94bd4ef8a2c3b884cfa59ca342b2e
>>> TEST 2 PASSED <<<
Plaintext : ffffffffffffffffffffffffffffffff
Key       : 00000000000000000000000000000000
Ciphertext: 3f5b8cc9ea855a0afa7347d23e8d664e
Expected  : 3f5b8cc9ea855a0afa7347d23e8d664e
>>> TEST 3 PASSED <<<
Plaintext : 0123456789abcdef0123456789abcdef
Key       : 00000000000000000000000000000000
Ciphertext: 27ba3ada4e61bcb03ad82126fe4e0a23
Expected  : 27ba3ada4e61bcb03ad82126fe4e0a23
>>> TEST 4 PASSED <<<
test_AES128.v:93: $finish called at 265000 (1ps)
```

Figure 6: Simulation Result


Figure 7: Vitis Output when Plaintext and key set to zero

# References

[1] Y. Zhang and X. Wang, "Pipelined implementation of AES encryption based on FPGA," in *Proceedings of the 2010 IEEE International Conference on Information Theory and Information Security (ICITIS)*, Beijing, China, 2010, pp. 170–173, doi:10.1109/ICITIS.2010.5688757.

[2] `https://github.com/aneels3/AES-128`.