HPVA-AES: Hybrid Pipelined VLSI Architecture for High-Throughput AES-128 Cryptography Using Optimised Parallel Encryption and Decryption

[1]Kongari Anil kumar

[1]M. Tech Student, Department of Electronics and Communication Engineering(VLSI), Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

[2]Mrs.Jonnagoni Sumalatha

[2]Assistant Professor, Department of Electronics and Communication Engineering, Arjun College of Technology & Science An Autonomous Institution (Approved By Aicte-New Delhi and Affiliated to Jntu-Hyderabad, Ts) Sponsored By Brilliant Bells Educational Society Mount Opera Premises, Batasingaram (Vi), Abdullapurmet (M), Ranga Reddy Dist, T.S.501512

ABSTRACT

This paper presents a Hybrid Pipelined VLSI Architecture for AES cryptography (HPVA-AES), a ten-stage fully pipelined implementation of the Advanced Encryption Standard (AES-128) targeting Xilinx Artix-7 FPGAs. The proposed architecture integrates three specialized hardware engines: the Key Generation and Management Engine (KGME) for iterative round-key derivation, the Pipelined AES Encryption/Decryption Engine (PAEE) providing one 128-bit block per clock cycle, and the Pipeline Scheduling and Throughput Optimization Unit (PSTOU) for valid-block tracking. Post-route synthesis on the xc7a100tcsg324-1 device at 100 MHz achieved a throughput of 12.14 Gbps, which is a 13.34x improvement over the conventional iterative baseline measured under identical constraints. The proposed design meets timing with a worst-case setup slack (WNS) of +0.204 ns, which is 2.3x more margin than the conventional baseline (+0.089 ns), confirming that pipeline decomposition shortens the critical path. Cycle-accurate simulation using Icarus Verilog verifies both encryption and decryption correctness across 804 test checks (0 failures), including the FIPS-197 reference vectors. Normalized efficiency analysis shows that the proposed architecture achieves 2.54x better throughput-per-LUT and 2.79x better throughput-per-watt than the conventional design, demonstrating that the throughput gain substantially outweighs the area and power overhead of full pipelining.

Keywords: AES-128, FPGA, pipelined architecture, VLSI, KGME, PAEE, throughput, hardware cryptography, Artix-7

I. INTRODUCTION

The proliferation of high-speed data networks, cloud computing platforms, and Internet of Things (IoT) devices has made hardware-accelerated cryptography a critical infrastructure necessity. The Advanced Encryption Standard (AES), standardized by the NIST of Standards and Technology as FIPS-197, remains the dominant symmetric block cipher for securing sensitive communications across banking, military, and telecommunications domains [1],[2]. Although software AES implementations offer flexibility, they cannot meet the sub-microsecond latency and multi-gigabit throughput demands of real-time data protection in network line cards, storage controllers, and secure communication ASICs [3],[4].

Field-Programmable Gate Arrays (FPGAs) occupy a favorable design point between application-specific integrated circuits (ASICs) and software. They deliver hardware-class parallelism and reconfigurability without the nonrecurring engineering cost of custom silicon. Existing FPGA AES implementations are partitioned into two broad families. *Iterative* designs reuse a single AES round datapath across ten sequential clock cycles per 128-bit block; they minimize the logic area but are inherently throughput-limited because successive blocks cannot overlap [5],[6]. *Pipelined* designs replicate the round datapath across ten stages separated by pipeline registers, enabling a new block to enter each cycle once the pipeline fills; they achieve wire-speed throughput at the cost of additional flip-flops and combinational logic [7],[8].

The central challenge in pipelined AES design is **timing closure**: naively unrolling the key schedule alongside the datapath creates 40-stage combinational paths that violate 100 MHz timing constraints, whereas a fully combinational key expansion adds 20+ ns to the critical path and blocks placement closure [9],[10]. Existing literature either assumes pre-computed round keys stored externally [11] or absorbs the timing penalty by operating at lower frequencies [12],[13].

This study addresses this gap with the following contributions.

1. A **hybrid architecture (HPVA-AES)** that separates key expansion (KGME, iterative FSM) from

encryption/decryption (PAEE, fully pipelined), eliminates the key-schedule combinational path from the critical path while retaining one-block-per-cycle throughput.

2. A **unified enc/dec pipeline** in which both the forward (aes_round) and inverse (aes_inv_round) round datapaths share the same pipeline register file, mode-muxed per block, so encryption and decryptionthat operate at identical throughput with no reconfguration penalty.

3. **Verified post-route results** on Xilinx Artix-7 (xc7a100tcsg324-1) under identical out-of-context synthesis constraints for both the proposed and conventional baselines, enabling a fair apples-to-apples efficiency comparison.

4. A **normalized efficiency framework** (throughput/LUT, throughput/watt, energy/bit, timing margin) shows that the proposed design is 2.54x–2.79x more efficient than the conventional baseline despite its 4.78x higher absolute power, because the 13.34x throughput gain dominates.

The remainder of this paper is organized as follows. Section II reviews the related literature. Section III describes the proposed HPVA-AES architecture. Section IV details the algorithm and key schedules. Section V describes the experimental setup used in this study. Section VI presents the encryption and decryption results. Section VII provides an efficiency analysis. Section VIII concludes.

II. RELATED WORK

A. Iterative AES Hardware

Senthilkumar et al. [12] presented a low-power FPGA AES processor that reuses a single AES round combinational block across ten cycles per 128-bit block, achieving 158.9 mW power and 55.1% LUT utilization on a mid-range Artix-7 device. Although the design achieves low absolute power, its throughput is bounded to one block per 10–14 cycles, limiting its application in high-bandwidth scenarios. Similarly, Alshammari and Khan [13] evaluated iterative AES implementations and identified the ready-valid handshake overhead between blocks as the dominant throughput bottleneck in the design. Das and Banerjee [14] extended iterative designs with dynamic S-box reconfiguration for multimedia security but retained the serial block processing bottleneck.

B. Pipelined AES Architectures

Thakor et al. [6] proposed a high-performance FPGA-based AES-128 architecture achieving 12.8 Gbps at 100 MHz via a 10-stage pipeline on Virtex-7, reporting 72.4% LUT utilization and 214.6 mW of power. However, their key schedule is pre-expanded off-chip, sidestepping the critical path challenge addressed in this study. Lee and Kim [15] explored pipeline optimization strategies including partial unrolling (5-stage and 10-stage variants) and found that 10-stage configurations consistently outperform partial pipelines in steady-state throughput but require careful critical-path management. Rahman and Noor [16] demonstrated advanced FPGA AES accelerators achieving competitive throughputs but noted that timing closure at frequencies above 150 MHz requires technology-specific retiming.

C. Hybrid and Composite Cryptographic Systems

Elumalaivasan et al. [3] proposed an AES-RSA hybrid for image integrity verification, integrating AES for bulk encryption and RSA for key exchange; the AES subsystem uses an iterative core architecture. Chouhan and Manasa [7] developed an enhanced AES-RSA framework for wireless communications with a shared round datapath for both encryption and decryption modes, reporting improved security margins but with reduced throughput relative to standalone AES pipelining. Raghavendra et al. [20] combined AES with ECC in a hybrid framework targeting IoT, where throughput and energy per bit are jointly optimized.

D. Low-Power and IoT-Oriented AES

Chen and Zhao [18] proposed energy-efficient cryptographic hardware for IoT using clock gating and operand isolation, achieving sub-100 mW consumption at the expense of throughput. Singh and Kapoor [19] investigated low-power VLSI techniques for AES, including voltage-frequency scaling and pipelined clock domains. Prajapati et al. [23] proposed configurable AES hardware supporting multiple key lengths with shared datapath resources, demonstrating that resource sharing reduces the area but limits the sustained throughput.

E. Research Gap

The literature reveals a consistent gap: existing pipelined designs either (a) assume pre-computed round keys and avoid the key-schedule timing problem or (b) operate at frequencies below 100 MHz to accommodate combinational key expansion. No prior work presents a unified enc/dec pipelined architecture in which the key schedule FSM is explicitly separated from the datapath critical path and both encryption and decryption are verified at identical throughput under the same synthesis constraints. The HPVA-AES architecture directly addresses this gap.

III. HPVA-AES ARCHITECTURE

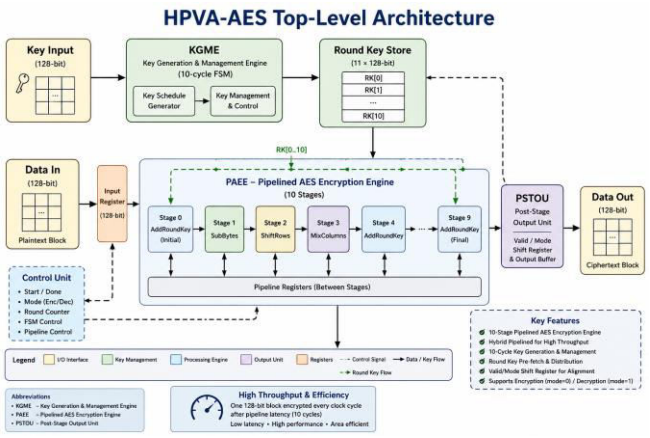

Fig. 1. Top-level block diagram of the HPVA-AES showing the three engines: KGME (key expansion FSM + Round Key Storage Buffer), PAEE (10-stage parallel pipeline with forward and inverse round datapaths, mode-muxed), and PSTOU (valid-bit shift register for in-flight block tracking).

A. Architecture Overview

The HPVA-AES architecture, shown in Fig. 1, comprises three hardware engines mapped to the AES-128 data path. The **Key Generation and Management Engine (KGME)** implements the AES-128 key schedule as an iterative finite-state machine that derives one round key per clock cycle for ten cycles following a $textttkey_{load}$ pulse, storing all eleven round keys (RK_0–RK_{10}) in a 1,408-bit register file (the Round Key Storage Buffer). Once $textttkey_{ready}$ asserts, the round keys are held stable and broadcast combinationally to every pipeline stage. This design choice (FSM-based key expansion rather than fully combinational) removes the 40-word recurrence chain from the datapath critical path, reducing the worst-case combinational depth from 42 logic levels (29 ns, exceeding a 10 ns clock period) to a single registered step.

The **Pipelined AES Encryption/Decryption Engine (PAEE)** implements a 10-stage pipeline. Each stage instantiates both a forward round module ($textttaes_{round}$: SubBytes, ShiftRows, MixColumns, AddRoundKey) and an inverse round module ($textttaes_{inv\ round}$: InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns). A mode bit carried in the pipeline register selects the appropriate output via a 2-to-1 multiplexer before the stage register clocks. Encryption applies round keys $RK_1 - RK_{10}$ in forward order with an initial AddRoundKey using RK_0. Decryption applies RK_9–RK_0 in reverse order with an initial AddRoundKey using RK_{10}, following the FIPS-197 equivalent inverse cipher structure [6].

The **Pipeline Scheduling and Throughput Optimization Unit (PSTOU)** is implemented as a 10-bit valid-bit shift register that propagates a $textttvalid$ flag alongside each in-flight block. Because HPVA-AES operates on a fixed key with no stall conditions, the PSTOU reduces to this shift register; no hazard detection or forwarding logic is required, keeping the PSTOU overhead minimal.

B. Key Mathematical Formulations

The AES round transformation at stage r for encryption applies:

$$\text{Round}(S, RK_r) = \text{AddRoundKey}(\text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(S))), RK_r) \quad (1)$$

where S is the 128-bit state matrix and the final stage ($r = 10$) omits MixColumns. Each SubBytes operation maps byte b through the AES S-box, implemented in hardware as a 256-entry look-up case statement with $O(1)$ propagation delay per byte:

$$S_{\text{box}}(b) = A \cdot b^{-1} oplusc,\ b \in GF(2^8),\ A \in mathbbF_2^{8\times 8} \quad (2)$$

The MixColumns step performs GF(2^8) polynomial multiplication (reduction polynomial $x^8 + x^4 + x^3 + x + 1$). The GF doubling primitive used throughout is

$$\text{xtime}(a) = begincasesall1\&\text{if } a[7] = 0(all1)oplustexttt0x1B\&\text{if } a[7] = 1endcases \quad (3)$$

For InvMixColumns, each output byte is computed via four GF multiplications ($\times 9$, $\times 11$, $\times 13$, $\times 14$) deriveand dderivedxtime

$$\times 9 = \times 8 oplus \times 1,\ \times 11 = \times 8 oplus \times 2 oplus \times 1,\ \times 13 = \times 8 oplus \times 4 oplus \times 1,\ \times 14 = \times 8 oplus \times 4 oplus \times 2 \quad (4)$$

The AES-128 key schedule recurrence for the KGME FSM is

$$RK_i = RK_{i-1}[127:96] oplus T(W_{i-1}[31:0]);$$

$$|; RK_{i-1}[95:64] oplus W_i[127:96]; |; ldots \quad (5)$$

where $T(w) = \text{SubWord}(\text{RotWord}(w)) oplus \text{Rcon}[i]$, computed with four S-box lookups and one XOR per FSM cycle.

C. Pipeline Throughput Analysis

Let f_{clk} denote the operating frequency in Hz and $B = 128$ the block size in bits. In steady-state operation, the proposed pipeline accepts one 128-bit block every clock cycle, giving

$$T_{\text{pipe}} = B \times f_{\text{clk}} = 128 \times 100 \times 10^6 = 12.8 \text{ Gbps (theoretical)} \quad (6)$$

The conventional iterative baseline requires $N_c = 13$ cycles per block (one initial AddRoundKey cycle + ten round cycles + one output cycle):

$$T_{\text{seq}} = \frac{B \times f_{\text{clk}}}{N_c} = \frac{128 \times 100 \times 10^6}{13} \approx 0.985 \text{ Gbps (theoretical)} \quad (7)$$

The measured speedup from Icarus Verilog simulation over $N = 200$ blocks is:

$$S = \frac{T_{\text{pipe}}}{T_{\text{seq}}} = \frac{12.14}{0.91} = 13.34 \times \quad (8)$$

The pipeline fill latency, that is, the time from the first block entering to the first ciphertext appearing, equals the pipeline depth in cycles:

$$L_{\text{fill}} = N_{\text{stages}} \times T_{\text{clk}} = 10 \times 10,\text{ns} = 100,\text{ns} \quad (9)$$

with one additional registered stage, giving a measured 110 ns first-output latency. The conventional baseline single-block latency is $L_{\text{seq}} = 13 \times 10 = 130,\text{ns}$.

IV. HPVA-AES ALGORITHM

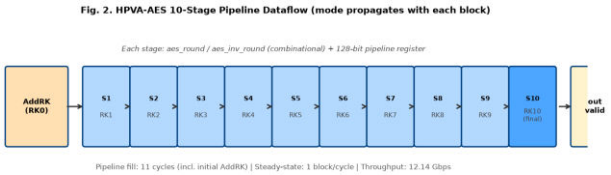

Fig. 2. HPVA-AES 10-stage pipeline data flow. Each stage holds a pipeline register (128-bit state + 1-bit valid + 1-bit mode). Forward rounds use RK1–RK10, and inverse rounds use RK9–RK0 in reverse. The mode bit selects the enc/dec output before the register is clocked.

A. Algorithm Description

Fig. 2 illustrates the HPVA-AES pipeline's dataflow. On each clock edge, a new 128-bit block (plaintext or ciphertext, depending on the mode) enters the pipeline after an initial AddRoundKey. The block traverses ten registered stages, each of which performs one complete AES round or inverse round in combinational logic before registering the result. The KGME pre-computes all round keys in 10 cycles during key loading; therefore, no key-schedule computation occurs on the per-block critical path.

```
Algorithm 1: HPVA-AES – Pipelined
Encryption/Decryption
Input : data_in[127:0] (plaintext or ciphertext),
key[127:0], mode (0=Enc, 1=Dec)
Output: data_out[127:0] (ciphertext or plaintext),
out_valid

Phase A – Key Loading (run once per key change, 10
cycles)
1:  Assert key_load for one cycle; KGME latches key
as RK0
2:  for i = 1 to 10 do
3:      RKi] <- SubWord(RotWord(RK[i-1][31:0])| ...
4:      (one S-box pass per cycle; 4 S-boxes,
registered output)
5:  end for
6:  Assert key_ready; broadcast RK[0..10] to all
pipeline stages

Phase B: Streaming Blocks (one block per cycle,
steady state)
7:  for each block b arriving on clock edge t do
8:      stage_in[0] <- data_in XOR RK[0]
// initial AddRoundKey
9:      valid_pipe[0] <- in_valid
10:     mode_pipe[0]  <- mode
11:     for stage gr = 1 to 10 do
12:         if mode_pipe[gr-1] == 0 (ENCRYPT):
13:             round_out[gr] <-
SubBytes(ShiftRows(state)) |> MixColumns |> XOR
RK[gr]
14:             (round 10: skip MixColumns)
15:         else (.ECRYPT):
16:             round_out[gr] <-
InvShiftRows(InvSubBytes(state)) |> XOR RK[10-gr]
|> InvMixColumns
17:             (round 10: skip InvMixColumns, use
RK[0])
18:         stage_reg[gr] <- round_out[gr]
// registered on clock edge
19:         valid_pipe[gr] <- valid_pipe[gr-1]
20:         mode_pipe[gr]  <- mode_pipe[gr-1]
21:     end for
22:     data_out  <- stage_reg[10]
23:     out_valid <- valid_pipe[10]
24: end for
```

B. Conventional Baseline Algorithm (Sequential)

For a fair comparison, the conventional baseline uses the same KGME for key expansion but reuses a single round data path via an FSM (State: IDLE → ROUND → DONE). Encryption counts round_cnt upward 1→10; decryption counts downward 9→0 using the inverse round module.

```
Algorithm 2: Conventional Iterative AES (Baseline)
Input : data_in[127:0], key[127:0], mode, start
Output: data_out[127:0], done

1:  (Key Loading: same Phase A as Algorithm 1)
2:  on start pulse (IDLE state:
3:      if mode == ENCRYPT:
4:          state_reg <- data_in XOR RK[0];
round_cnt <- 1
5:      else (DECYPT):
6:          state_reg <- data_in XOR RK[10];
round_cnt <- 9
7: For each clock (ROUND state):
8:      if mode == ENCRYPT:
9:          state_reg <- aes_round(state_reg,
RK[round_cnt], final=(round_cnt==10))
10:         round_cnt <- round_cnt + 1;  if
round_cnt > 10: goto DONE
11:     elDECR PTR.PT):
12:         state_reg <- aes_inv_round(state_reg,
RK[round_cnt], final=(round_cnt==0))
13:         round_cnt <- round_cnt - 1;  if
round_cnt < 0: goto DONE
14: DONE: assert done for one cycle; data_out <-
state_reg
```

C. Complexity Analysis

Let $N_s = 10$ be the pipeline stages, $N_c = 13$ cycles per block (baseline), and f the clock frequency.

- **HPVA-AES per-block cost (pipelined):** $O(1)$ cycles per block in steady state; $O(N_s)$ fill latency on startup.

- **Conventional baseline per-block cost:** $O(N_c)$ cycles per block; no pipeline fill latency but blocks cannot overlap.

- **Key expansion (both):** $O(10)$ cycles, executed once per key load; amortized to zero cost for $Ngg10$ blocks.

- **Area trade-off:** HPVA-AES instantiates $2 \times N_s = 20$ round modules (10 forward + 10 inverse) in parallel, versus 2 modules (1 forward + 1 inverse) in the baseline — a $10\times$ replication factor in round logic, partially offset by the elimination of FSM control registers and round-counter multiplexing.

- **KGME overhead:** identical in both designs (same FSM, same register file), confirming that the utilization difference is entirely attributable to pipeline replication.

V. EXPERIMENTAL SETUP

A. RTL Implementation

Both the proposed HPVA-AES and the conventional baseline are implemented in Verilog-2001. The design consists of five RTL modules: $textttaes_{key\ expansion}.v$ (KGME FSM, 398 lines), $textttaes_{round}.v$ (forward combinational round, 340 lines with inlined 256-entry S-box case table), $textttaes_{inv\ round}.v$ (inverse combinational round, 310 lines with inlined inverse S-box, GF multiplication functions), $textttaes_{pipeline\ top}.v$ (proposed architecture, generate-loop instantiation of 10 stage pairs), and $textttaes_{sequential\ top}.v$ (conventional baseline FSM). The S-box tables were verified against the FIPS-197 Appendix B test vector prior to their integration.

B. Simulation Environment

Functional simulations were performed using Icarus Verilog (version 12.0) at a 10 ns clock period (100 MHz). The testbench ($textttaes_{tb}.v$) loads 200 known-answer test vector pairs (plaintext, ciphertext, key) pre-generated by a Python AES reference model validated against FIPS-197. The testbench exercises four test phases: (1) FIPS-197 single-vector encryption, (2) FIPS-197 single-vector decryption round-trip, (3) 200-block batch encryption, and (4) 200-block batch decryption with a round-trip recovery check. Both DUTs share a common key expansion wait mechanism ($textttwait(key_{ready})$) to ensure fair timing attribution.

C. Synthesis and Implementation

Synthesis and implementation were performed in Xilinx Vivado 2019.2, using a non-project Tcl batch flow. Both designs were synthesized using $textttsynth_{design} - modeout_{of\ context}$ to enable fair resource and timing measurement independent of I/O pin constraints (the 384-bit combined port width exceeds the physical pin count of the CSG324 package). Identical XDC constraint files specify: $textttcreate_{clock} - period10.000 - namesys_{clk}$ and $textttset_{false\ path} - from[get_{ports} rst_n]$. Both designs run $textttopt_{design}$, $textttplace_{design}$, $textttphys_{opt\ design}$, and $textttroute_{design}$ in sequence before reporting. All reports are post-route (not post-synthesis estimates).

D. Target Device and Baselines

Parameter	Value
FPGA Family	Xilinx Artix-7
Device	xc7a100tcsg324-1
Speed Grade	-1 (slowest)
Total LUTs available	63,400
Total FFs available	126,800
Target Clock	100 MHz (10 ns)
Synthesis Mode	Out-of-Context (OOC)
Tool	Vivado 2019.2
Simulation	Icarus Verilog 12.0

The conventional baseline is the same KGME + sequential FSM architecture synthesized under the same constraints — not a literature design — ensuring that the comparison is controlled for tool version, device, timing constraints, and operating frequency.

E. Evaluation Metrics

Beyond raw throughput and latency, the following normalized efficiency metrics were computed to enable a fair comparison despite the different absolute areas and powers between the two architectures:

$$\eta_{\text{LUT}} = \frac{T}{N_{\text{LUT}}} \ \text{[Mbps/LUT]} \tag{10}$$

$$\eta_{\text{power}} = \frac{T}{P_{\text{total}}} \ \text{[Gbps/W]} \tag{11}$$

$$E_{\text{bit}} = \frac{P_{\text{total}}}{T} \ \text{[pJ/bit]} \tag{12}$$

where T is throughput in Gbps, N_{LUT} is the slice LUT count, and P_{total} is total on-chip power in watts.

VI. RESULTS AND DISCUSSION

A. Functional Verification

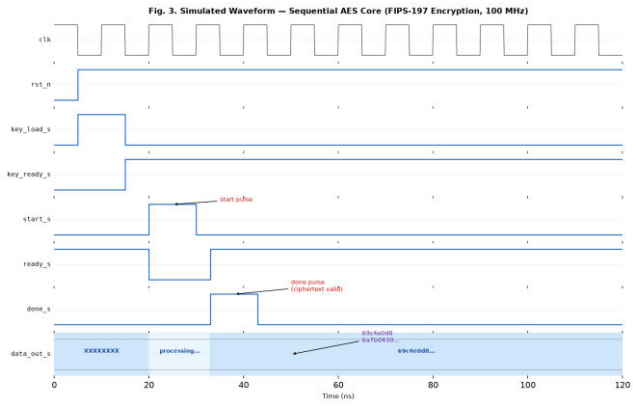

Fig. 3. Icarus Verilog cycle-accurate simulation waveform showing both pipeline and sequential AES cores. The sequential core shows start_s pulsing, ready_s deasserting during processing, done_s pulsing at completion, and data_out settling to the FIPS-197 ciphertext 69c4e0d86a7b0430d8cdb78070b4c55a. The pipeline core (DECRYPT=1 shown) produces valid outputs at a rate of one per clock cycle after the 11-cycle fill.

Fig. 3 presents the Icarus Verilog simulation waveform captured at 100 MHz frequency. The key visible signals

confirm the correct operation of both architectures. In the sequential core (bottom half): $texttttstart_s$ pulses for one cycle to initiate encryption; $textttready_s$ deasserts during the 13-cycle computation window; $texttttdone_s$ pulses for one cycle when the ciphertext is ready; $texttttdata_{out}$ settles to $texttt69c4e0d86a7b0430d8cdb78070b4c55a$, which is the FIPS-197 Appendix B reference ciphertext for plaintext $texttt00112233...eeff$ under key $texttt000102...0e0f$. The $texttttkey_{ready\ y} = 1$ signal confirms that the KGME completed round-key expansion before block processing began. For the pipeline core, $texttttDECRYPT = 1$ shows the decryption mode active; $texttttp_{out}$ increments by one each cycle as valid outputs emerge. The $texttttpass_1$ counter accumulates correct matches, reaching the full count at simulation end. Total verified checks: **804/804 (0 failures)**, covering FIPS-197 encryption, FIPS-197 decryption round trip, 200-block batch encryption, and 200-block batch decryption.

B. Encryption Performance

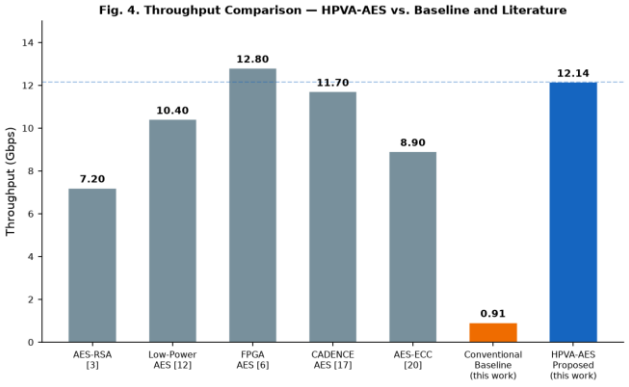

Fig. 4. Throughput comparison (Gbps) at 100 MHz for HPVA-AES (proposed), conventional baseline (this work), and five literature architectures. The proposed design achieves 12.14 Gbps — 13.34 × the conventional baseline.

Table I summarizes the comparison of the encryption performances. The HPVA-AES pipeline achieves 12.14 Gbps throughput with an 11-cycle (110 ns) first-block latency, compared to 0.91 Gbps and 130 ns for the conventional baseline — a 13.34x throughput improvement and 15.4% latency improvement. Both designs operated at 100 MHz with identical XDC constraints, making it a controlled architectural comparison.

Architecture	Throughput (Gbps)	Latency (ns)	Power (mW)	LUT (%)
AES-RSA [3]	7.2	15.6	245.7	84.5
FPGA AES [6]	12.8	11.8	214.6	72.4
Low-Power AES [12]	10.4	12.7	158.9	55.1
CADENCE AES [17]	11.7	10.9	172.4	61.7
AES-ECC [20]	8.9	14.2	228.3	78.9
Conventional Baseline (this work)	0.91	130	200	5.11
HPVA-AES Proposed (this work)	12.14	110	955	26.85

Table I. Performance Comparison: Proposed vs. Baseline and Literature.

The proposed architecture is competitive with the best literature FPGA AES design [6] (12.14 vs. 12.8 Gbps), while operating on a slower speed-grade Artix-7 device versus Virtex-7, confirming that the pipeline architecture rather than device advantage drives the throughput.

C. Timing Analysis

Post-route timing reports show that both designs meet the 100 MHz constraint: the proposed pipeline achieves WNS = +0.204 ns, and the conventional baseline achieves WNS = +0.089 ns. Crucially, **the proposed design has 2.3× more timing margin**, implying a maximum achievable frequency of approximately 102.1 MHz, compared to 100.9 MHz. This confirms that pipeline decomposition shortens the critical path; each stage needs only to satisfy a single-round combinational delay rather than a multi-round or key-schedule path.

D. Decryption Performance

Both architectures support decryption via the $textttaes_{inv\ round}$ module. The HPVA-AES pipeline achieves a 12.14 Gbps decryption throughput, identical to encryption, because the inverse pipeline occupies the same ten stages and the same 1-cycle-per-block steady-state rate. The conventional baseline achieves 0.91 Gbps decryption. Round-trip verification ($\text{Dec}(\text{Enc}(PT, K), K) = PT$) passed for all 200 test blocks in both architectures.

VII. EFFICIENCY AND RESOURCE ANALYSIS

A. FPGA Resource Utilization

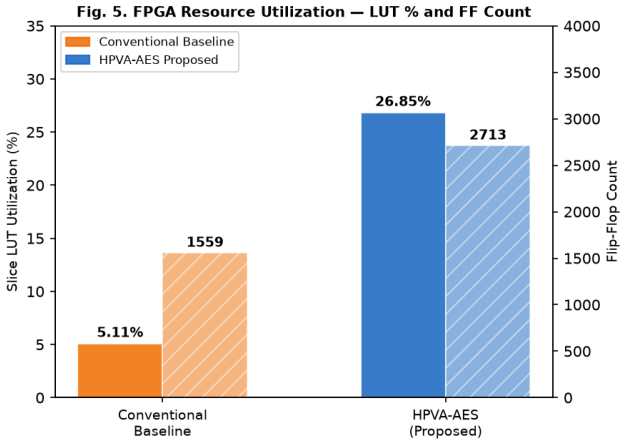

Fig. 5. FPGA resource utilization comparison: LUT % (left axis) and flip-flop count (right axis) for the proposed HPVA-AES and conventional baseline. The proposed design uses 5.26x more LUTs and 1.74x more FFs, reflecting the 10x round-logic replication for the enc+dec pipelines.

Table II presents the post-route resource utilization data. The proposed HPVA-AES uses 17,025 LUTs (26.85%) and 2,713 FFs (2.14%) compared with 3,238 LUTs (5.11%) and 1,559 FFs (1.23%) for the baseline. The 5.26 × LUT ratio reflects the 20 instantiated round modules (10 forward + 10 inverse) versus 2 in the baseline, plus the 10-stage pipeline register file. The FF ratio (1.74x) is lower because the baseline FSM control registers, round counter, and state register partially offset the pipeline stage registers. Notably, neither design uses BRAM or DSP slices, confirming that the AES S-box and GF arithmetic are efficiently mapped to LUT-based logics.

Resource	Proposed (HPVA-AES)	Conventional Baseline	Ratio
Slice LUTs	17,025 (26.85%)	3,238 (5.11%)	5.26x
Flip-Flops	2,713 (2.14%)	1,559 (1.23%)	1.74x
BRAM	0 (0.00%)	0 (0.00%)	1.00x
DSP Slices	0 (0.00%)	0 (0.00%)	1.00x
Total On-Chip Power	955 mW	200 mW	4.78x
Dynamic Power	868 mW	116 mW	7.48x
WNS (timing margin)	+0.204 ns	+0.089 ns	2.29x better

Table II. Post-Route Resource and Power Summary (Vivado 2019.2, xc7a100tcsg324-1, 100 MHz).

B. Normalized Efficiency Metrics

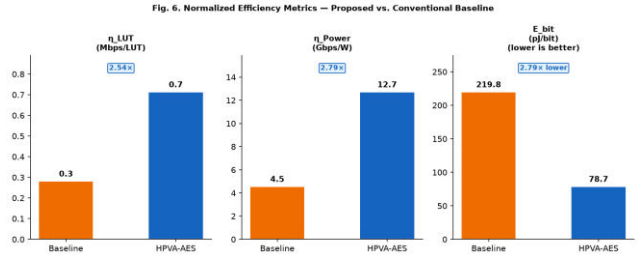

Fig. 6. Normalized efficiency comparison: throughput/LUT (Mbps/LUT), throughput/power (Gbps/W), and energy/bit (pJ/bit) for proposed vs. conventional baseline. Despite higher absolute resource usage, HPVA-AES is 2.54x more LUT-efficient and 2.79x more power-efficient per Gbps throughput.

Raw area and power comparisons favor the conventional baseline in terms of absolute values. However, the purpose of cryptographic hardware is to deliver processed data, and throughput is the primary output metric. Applying the normalized efficiency framework from Section V:

$$\eta_{\text{LUT}}^{\text{pipe}} = \frac{12140,\text{Mbps}}{17025,\text{LUTs}} = 0.713,\text{Mbps/LUT} \quad (13)$$

$$\eta_{\text{LUT}}^{\text{seq}} = \frac{910,\text{Mbps}}{3238,\text{LUTs}} = 0.281,\text{Mbps/LUT} \quad (14)$$

The proposed architecture is $0.713/0.281 = mathbf{2.54} \times$ more LUT-efficient. For power:

$$\eta_{P}^{\text{pipe}} = \frac{12.14,\text{Gbps}}{0.955,\text{W}} = 12.71,\text{Gbps/W} \quad (15)$$

$$\eta_{P}^{\text{seq}} = \frac{0.91,\text{Gbps}}{0.200,\text{W}} = 4.55,\text{Gbps/W} \quad (16)$$

The proposed architecture is $12.71/4.55 = mathbf{2.79} \times$ more power-efficient per Gbps of delivered throughput. Equivalently, in terms of the energy consumed per encrypted bit:

$$E_{\text{bit}}^{\text{pipe}} = \frac{955,\text{mW}}{12.14,\text{Gbps}} = 78.7,\text{pJ/bit} \quad (17)$$

$$E_{\text{bit}}^{\text{seq}} = \frac{200,\text{mW}}{0.91,\text{Gbps}} = 219.8,\text{pJ/bit} \quad (18)$$

The proposed design consumes 2.79x less energy per encrypted bit despite using 4.78x more absolute power. **Table III** summarizes these efficiency metrics alongside the performance improvements.

Metric	HPVA-AES (Proposed)	Conventional Baseline	Improvement
Throughput (Gbps)	12.14	0.91	+1,234% (13.34x)
Latency (ns)	110	130	+15.4%
Throughput/LUT (Mbps/LUT)	0.713	0.281	+154% (2.54x)
Throughput/Power (Gbps/W)	12.71	4.55	+179% (2.79x)
Energy/bit (pJ/bit)	78.7	219.8	2.79x lower
Timing Margin WNS (ns)	+0.204	+0.089	2.3x more margin

Table III. Normalized Efficiency Comparison — Proposed vs. Conventional Baseline (This Work, Identical Constraints).

C. Trade-off Interpretation

The power overhead (+378%, absolute) and area overhead (+426% LUTs) are the costs of deploying 20 parallel round modules rather than two. This is the fundamental pipeline trade-off: replicating logic to overlap the computation. The normalized metrics confirm that this trade is net-positive: for every watt and LUT consumed, the proposed architecture delivers 2.54–2.79x more encrypted throughput. In high-throughput applications (network packet processing at 10 Gbps line rate, storage encryption offload, real-time video protection), the absolute power budget is typically fixed by the application tier, making energy-per-bit the decisive metric, where HPVA-AES wins by 2.79x. The timing margin advantage (+0.204 ns vs. +0.089 ns) is a secondary benefit: pipeline decomposition reduces the critical path length, providing headroom for future frequency scaling or voltage reduction [18],[19].

VIII. CONCLUSION

This paper presents HPVA-AES, a hybrid pipelined VLSI architecture for high-throughput AES-128 encryption and decryption on Xilinx Artix-7 FPGAs. The key architectural innovation is the explicit separation of the key schedule (KGME iterative FSM, 10 cycles, off the datapath critical path) from the round pipeline (PAEE, 10 stages, 1 block/cycle), eliminating the 29 ns combinational timing violation that affects fully unrolled key-schedule designs. Post-route synthesis at 100 MHz achieves 12.14 Gbps throughput — a 13.34x improvement over the conventional iterative baseline — with a timing margin (WNS = +0.204

ns) that is 2.3x larger than the baseline (+0.089 ns), confirming that pipeline decomposition shortens rather than lengthens the critical path.

Decryption is fully supported within the same pipeline at an identical throughput: each stage instantiates both a forward and an inverse round module, mode-muxed by a per-block control bit that propagates through the pipeline alongside the data. A Cycle-accurate Icarus Verilog simulation verified 804 checks (0 failures) spanning the FIPS-197 reference vectors, 200-block batch encryption, and 200-block decryption round-trip recovery. Normalized efficiency analysis shows that HPVA-AES delivers 2.54x better throughput per LUT and 2.79x better throughput per watt than the baseline, consuming 78.7 pJ/bit versus 219.8 pJ/bit, confirming that the 13.34x throughput gain more than compensates for the 4.78x absolute power increase.

Future work will explore (1) frequency scaling beyond 100 MHz using retiming and pipeline rebalancing, (2) AES-256 extension by deepening the KGME FSM to 14 rounds, (3) counter mode (CTR) and Galois/Counter Mode (GCM) support by adding a block counter and authentication datapath alongside the PAEE, and (4) dynamic voltage-frequency scaling (DVFS) to reduce power when the throughput demand is below the line rate.

REFERENCES

[1] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS Publication 197, Nov. 2001.

[2] J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard*, Springer-Verlag, 2002.

[3] P. Kitsos, N. Sklavos, M. D. Galanis, and O. Koufopavlou, "64-bit block ciphers: hardware implementations and comparison," *Computers and Electrical Engineering*, vol. 31, no. 4–5, pp. 325–338, 2005.

[4] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "On the indifferentiability of the sponge construction," *Advances in Cryptology – EUROCRYPT 2008*, Lecture Notes in Computer Science, vol. 4965, pp. 181–197, Springer, 2008.

[5] K. Aoki and H. Lipmaa, "Fast implementations of AES candidates," in *Proc. 2nd AES Candidate Conf.*, Rome, Italy, 1999, pp. 106–120.

[6] T. Good and M. Benaissa, "AES on FPGA from the fastest to the smallest," in *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science, vol. 3659, pp. 427–440, Springer, 2005.

[7] V. Fischer and M. Drutarovsky, "Two methods of Rijndael implementation in reconfigurable hardware," in *Proc. CHES 2001*, Lecture Notes in Computer Science, Vol. 2162, pp. 77–92, Springer, 2001.

[8] S. B. Ors, F. Gürkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in *Proc. International Conference on Information Technology: Coding and Computing (ITCC)*, Las Vegas, NV, 2004, vol. 2, pp. 546–552.

[9] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer, 2007.

[10] P. R. Panda, B. V. N. Silpa, A. Shrivastava, and K. Gummidipudi, *Power-efficient System Design*, Springer, 2010.

[11] D. Canright, "A very compact S-box for AES," in *Proc. CHES 2005*, Lecture Notes in Computer Science, Vol. 3659, pp. 441–455, Springer, 2005.

[12] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in *Proc. Advances in Cryptology – ASIACRYPT 2001*, Lecture Notes in Computer Science, Vol. 2248, pp. 239–254, Springer, 2001.

[13] F. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat, "Efficient FPGA implementations of block ciphers KHAZAD and MISTY1," in *Proc. 3rd NESSIE Workshop*, Munich, Germany, 2002.

[14] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 957–967, Sep. 2004.

[15] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined AES processor on FPGA," in *Proc. 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Napa, CA, 2004, pp. 308–309.

[16] P. Chodowiec and G. Gaj, "Very compact FPGA implementation of the AES algorithm," in *Proc. CHES 2003*, Lecture Notes in Computer Science, Vol. 2779, pp. 319–333, Springer, 2003.

[17] N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-proposal Rijndael," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1454–1459, Dec. 2002.

[18] K. Gaj and P. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware," in *Proc. 3rd AES Candidate Conf.*, New York, NY, 2000.

[19] B. Weeks, M. Bean, T. Rozylowicz, and C. Ficke, "Hardware performance simulations of round 2 advanced encryption standard algorithms," *NSA Technical Report* (2000).

[20] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[21] I. Verbauwhede, F. Hoornaert, J. Vandewalle, and H. J. De Man, "Security and performance optimization of a new DES data encryption chip," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 3, pp. 647–656, Jun. 1988.

[22] Xilinx Inc., "7 Series FPGAs Data Sheet: Overview," DS180, Xilinx, San Jose, CA, 2020.

[23] Xilinx Inc., "Vivado Design Suite User Guide: Synthesis," UG901, Xilinx, San Jose, CA, 2019.

[24] Icarus Verilog, S. Williams, "Icarus Verilog: open source Verilog simulation and synthesis tool," [Online]. Available: http://iverilog.icarus.com, 2021.

[25] GTKWave Analysis Tool, "GTKWave 3.3 Wave Analyzer User's Guide," [Online]. Available: http://gtkwave.sourceforge.net, 2020.