Marwadi University
Marwadi Chandarana Group

NAAC A+

Implementation of RSA Algorithm on FPGA (Nexys-4 DDR)

NAME: HARSHVARDHAN SONI
CLASS: 7TK1
ENROLLMENT NO: 92200133028
SEMESTER: 7TH

Marwadi University
Marwadi Chandarana Group
NAAC A+

Overview

Motivation to Implement this Algorithm:

- To demonstrate the hardware acceleration of the computationally expensive RSA algorithm using a Xilinx Artix-7 FPGA.
- Data security requires low-latency cryptographic checks for embedded systems.
- Moving the RSA core operation, modular exponentiation, from vulnerable software to fast, customized Verilog hardware

Introduction to RSA (Rivest-Shamir-Adleman)

- The RSA algorithm is named after those who invented it in 1978: Ron **Rivest**, Adi **Shamir**, and Leonard **Adleman**.
- The RSA algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

Marwadi University
Marwadi Chandarana Group

NAAC A+

RSA Algorithm Steps (Theory)

1: **Input Values:** p and q

2: **Compute:**

3: $n = p \times q$

4: $(n) = (p-1)(q-1)$

5: **Select Integer values:** e $[(\gcd(), e) - 1; 1 < e < \phi(n)]$

6: **Compute:** d de $mod\ \phi(n) = 1$

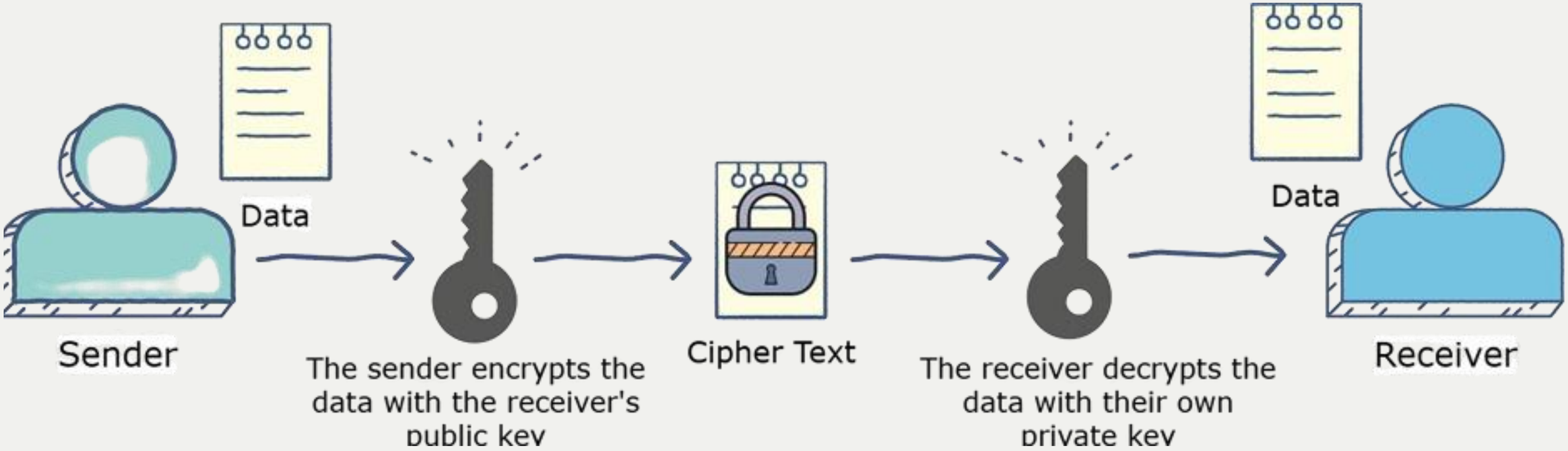
7: $C = Cg\ 1\ mod\ (z)$

8: **Encryption:** $M < n\ C = M\ (mod\ n)$

9: **Decryption:** $CM = C(mod\ n)$

Marwadi University
Marwadi Chandarana Group
NAAC A+

RSA Algorithm Steps (Theory)



Marwadi University
Marwadi Chandarana Group
NAAC A+

RSA Algorithm Steps (Theory)

RSA Algorithm core logic (cube + mod):

```
    // RSA Encrypt Core (inside always block)
plain  <= message;                    // ASCII input ('H' = 72)
temp   <= message * message * message;  // M^e (e=3)
cipher <= temp % n;                   // C = M^e mod n
```

Example of Implementation with Wi-Fi password

- Project Data Source: Wi-Fi Key * Message (M): Extracted Wi-Fi key: "rishit15".
- Packed Data: The gen_wifi_keyrom.py script packs this into a 64-bit Big-Endian constant: 64 ' h7269736869743135
- Core Input: The rsa_core uses the lower 16 bits of this constant for operation.
- Demonstration: The hardware output displays the Encrypted (C) and Decrypted (M dec) hex values on the 7-segment display, allowing for visual validation of the math

Marwadi University
Marwadi Chandarana Group

NAAC A+

FPGA Board – Nexys-4 DDR

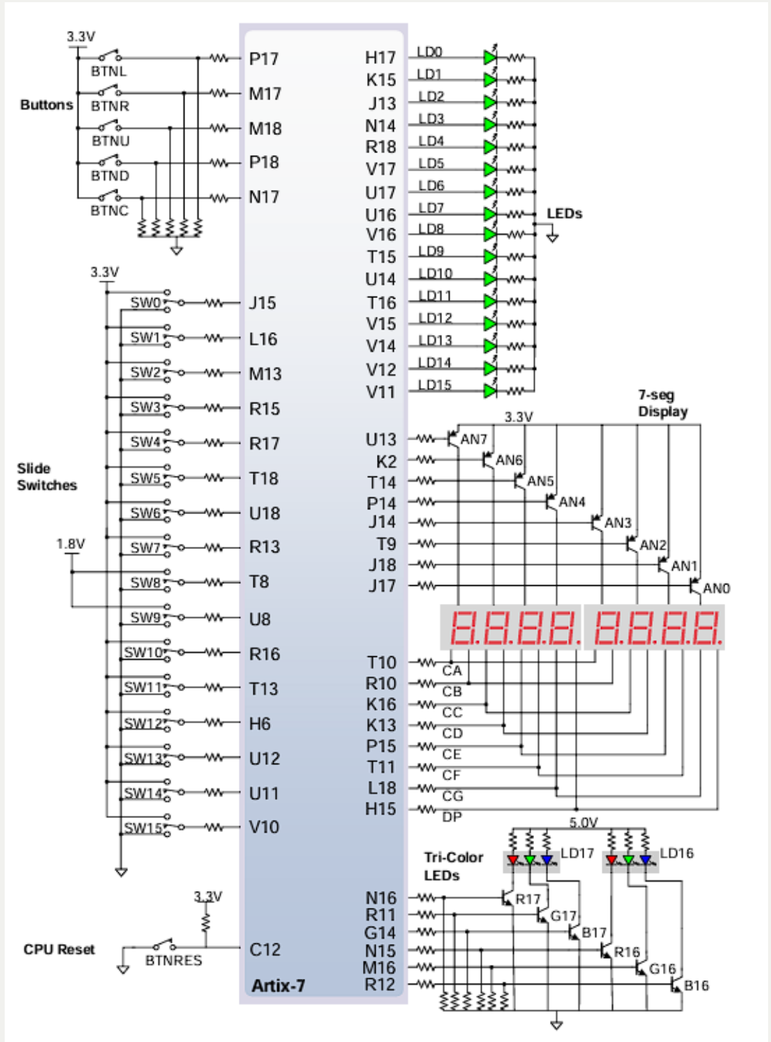
- FPGA: Xilinx Artix-7 XC7A100T-CSG324
- Onboard 100 MHz oscillator (clk on E3 pin)
- 7-segment display (8 digits)
- USB-JTAG programmer

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type
All ports (17)										
a_to_g (7)	OUT			☑	(Multip	LVCMOS33*	3.300		12	
a_to_g[6]	OUT		T10	☑	14	LVCMOS33*	3.300		12	
a_to_g[5]	OUT		R10	☑	14	LVCMOS33*	3.300		12	
a_to_g[4]	OUT		K16	☑	15	LVCMOS33*	3.300		12	
a_to_g[3]	OUT		K13	☑	15	LVCMOS33*	3.300		12	
a_to_g[2]	OUT		P15	☑	14	LVCMOS33*	3.300		12	
a_to_g[1]	OUT		T11	☑	14	LVCMOS33*	3.300		12	
a_to_g[0]	OUT		L18	☑	14	LVCMOS33*	3.300		12	
an (8)	OUT			☑	(Multip	LVCMOS33*	3.300		12	
an[7]	OUT		U13	☑	14	LVCMOS33*	3.300		12	
an[6]	OUT		K2	☑	35	LVCMOS33*	3.300		12	
an[5]	OUT		T14	☑	14	LVCMOS33*	3.300		12	
an[4]	OUT		P14	☑	14	LVCMOS33*	3.300		12	
an[3]	OUT		J14	☑	15	LVCMOS33*	3.300		12	
an[2]	OUT		T9	☑	14	LVCMOS33*	3.300		12	
an[1]	OUT		J18	☑	15	LVCMOS33*	3.300		12	
an[0]	OUT		J17	☑	15	LVCMOS33*	3.300		12	
Scalar ports (2)										
clk	IN		E3	☑	35	LVCMOS33*	3.300			
dp	OUT		H15	☑	15	LVCMOS33*	3.300		12	

Marwadi University
Marwadi Chandarana Group

NAAC A+

FPGA Board – Nexys-4 DDR



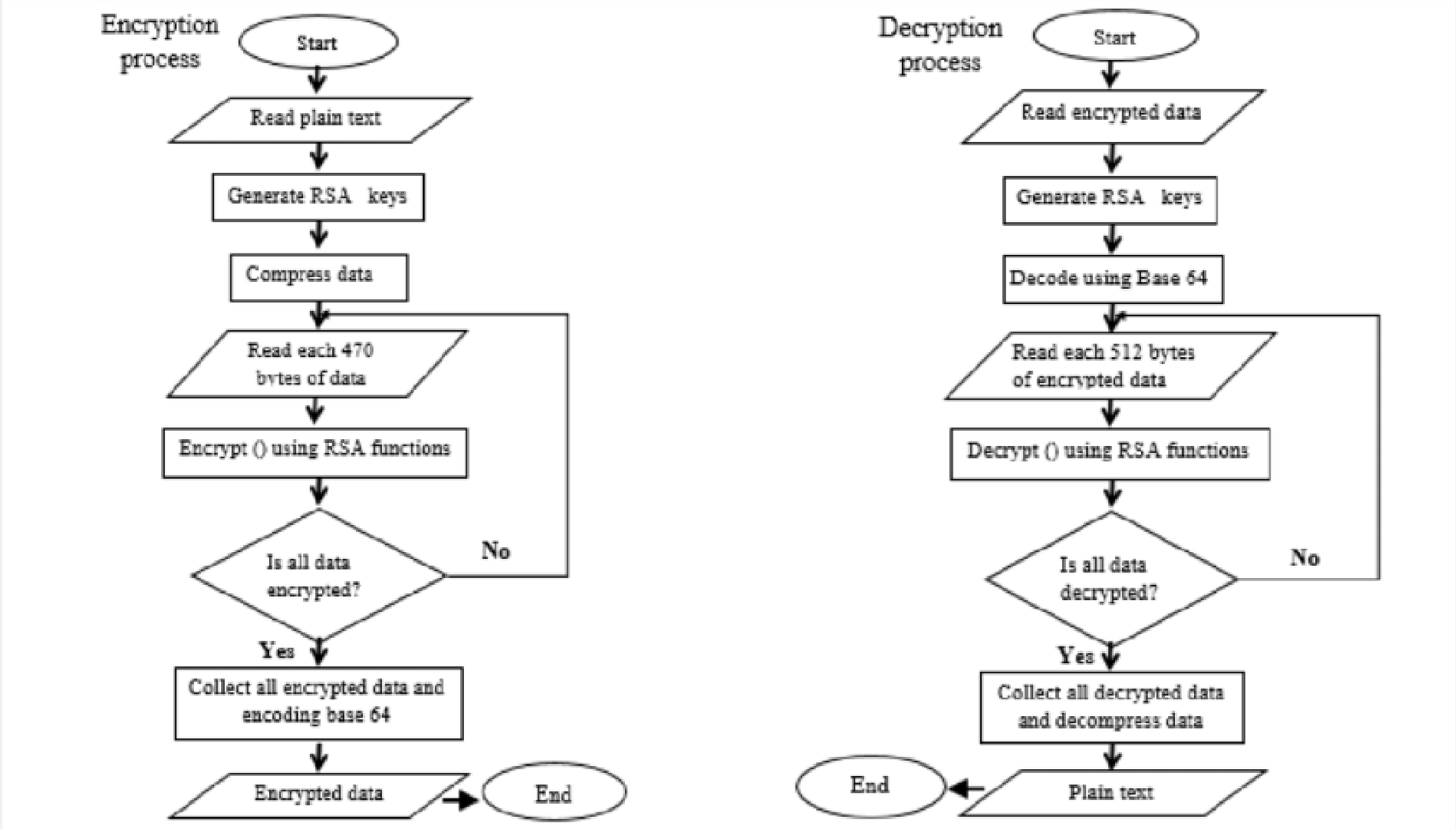
Marwadi University
Marwadi Chandarana Group

NAAC A+

Design Flow in Vivado

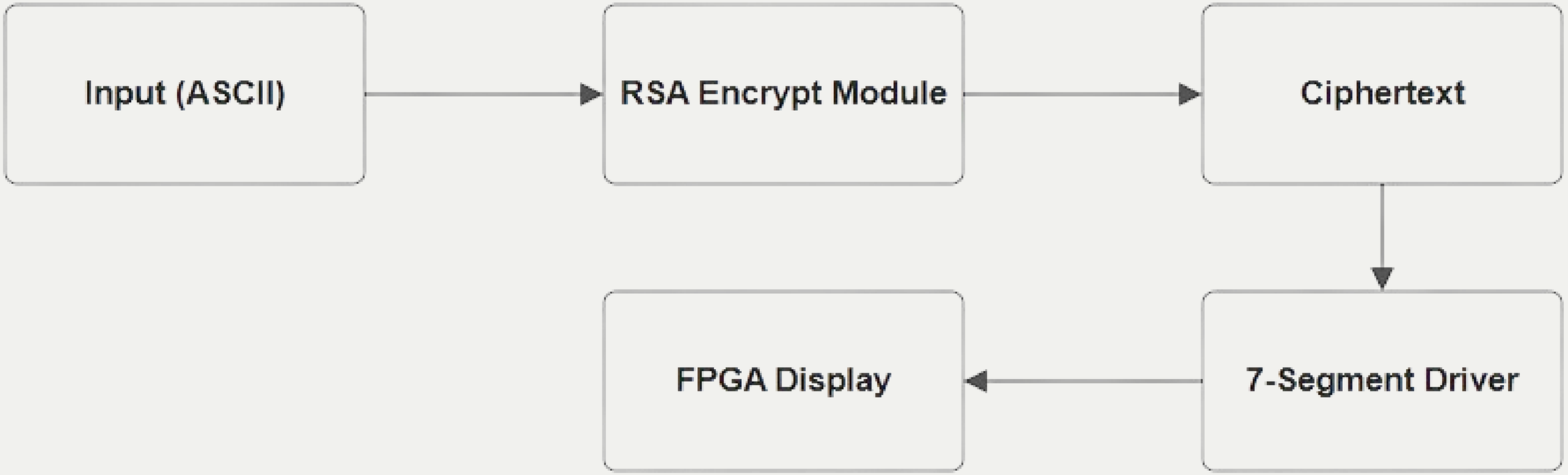
1. Write Verilog (RSA + hex7seg + top).
2. Run synthesis (check logic).
3. Assign I/O pins (clock = E3, segments, anodes, dp).
4. Run implementation + generate bitstream.
5. Load bitstream into FPGA via Hardware Manager.
6. Observe output on 7-seg.

Design Flow in Vivado



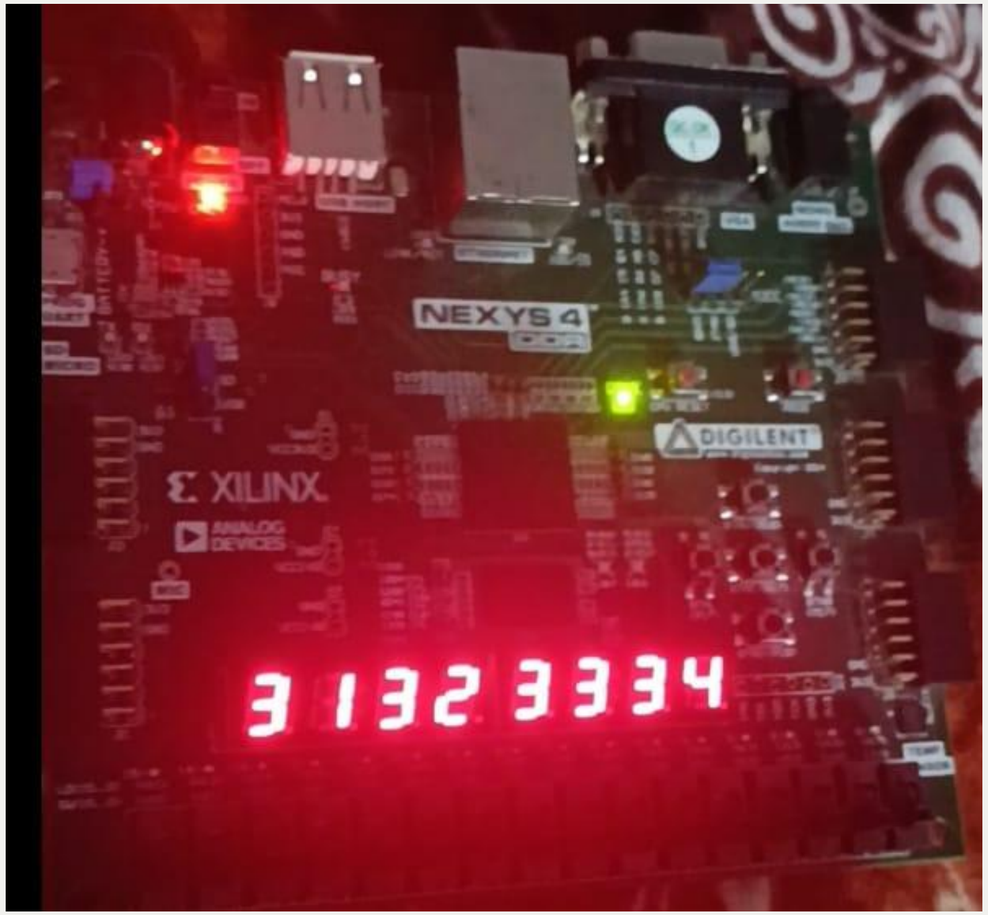
Marwadi University
Marwadi Chandarana Group
NAAC A+

Block Diagram of Implementation



Demo Output

- This Image Display is showing the Cyphertext of Wi-Fi Password



Marwadi University
Marwadi Chandarana Group

NAAC A+

Testing & Debugging Insights

Key Deployment Challenges:

- Clock Fixation: Had to generate an exact 48 MHz clock using the Clocking Wizard to overcome the USB core's timing strictness (prevented BUSY LED blinking from being useless).
- Compatibility: Required using a basic USB keyboard supporting the legacy PS/2 protocol as dictated by the Nexys-4 DDR port

Marwadi University
Marwadi Chandarana Group
NAAC A+

Testing & Debugging Insights

System Validation:

- Functional Proof: Simulation verified the correct output (M dec =M) for known inputs (Unit Tests)
- Performance Metrics (Latency): The ILA was used to measure the cryptographic speed, confirming the rsa_core operates in approximately 0.5μs per 17-bit operation (validating hardware acceleration).
- On-Board Debugging: Used the ILA and mapped signals to LEDs to monitor internal values like scan_code during debugging.

Marwadi University
Marwadi Chandarana Group

NAAC A+

Conclusion

- RSA Algorithm on the Nexys-4 DDR FPGA, fulfilling the goal of demonstrating a complete cryptographic cycle—encryption (C=M e (modn)) and decryption (M dec =C d (modn))—in custom hardware. The core contribution is the utilization of the FPGA to perform the computationally intensive.
- modular exponentiation using the Square-and-Multiply method, thereby moving this critical security function from vulnerable, slow software to low-latency, accelerated hardware. The modular architecture, consisting of dedicated.
- rsa_core instances and display_driver logic, was validated to ensure mathematical correctness and performance efficiency, confirming that FPGAs are a viable and powerful platform for tackling modern data security and embedded systems challenges.

Thank You

FOR LISTENING!

ANY QUESTIONS?