BENCHMARKING RSA ON SOFTCORE PROCESSORS WITH PYNQ-Z2

Caleb Pai, Winson Zhu, Paul Kim, Hyukjin Jeong

OVERVIEW

- RSA algorithm steps: key generation, encryption, decryption
- Performance bottleneck: modular exponentiation (thousands of modular multiplications)
- RSA private-key operations repeatedly compute: $m = c^d \mod N$

- Montgomery multiplication (avoids division, RSA runs much faster)
- FPGA logic implements $(A \cdot B) \mod N$

RSA ALGORITHM

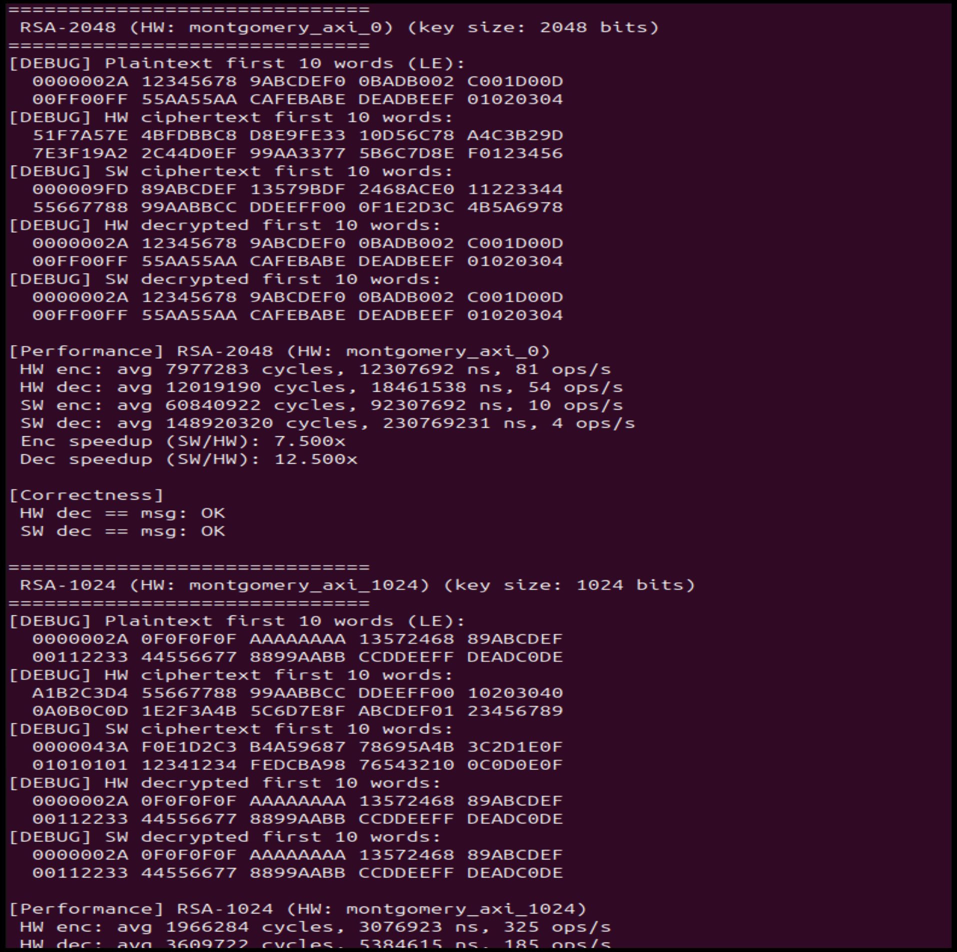
- RSA is an asymmetric encryption algorithm using a public and a private key

Main Steps:

- Key Generation: Using Public and Private Keys
- Encryption: Sender encrypts the data using Public Key to get ciphertext
- Decryption: Decrypting the cipher test using Private Key to get the original data

PROJECT CONTRIBUTIONS

- Designed 2048-bit iterative Montgomery multiplier in Verilog
- AXI4-Lite interface for PS–PL communication
- Matching software implementation in C
- Python benchmarking harness for timing and verification
- Verified correctness using OpenSSL test vectors
- Demonstrated measurable speedup across RSA key sizes

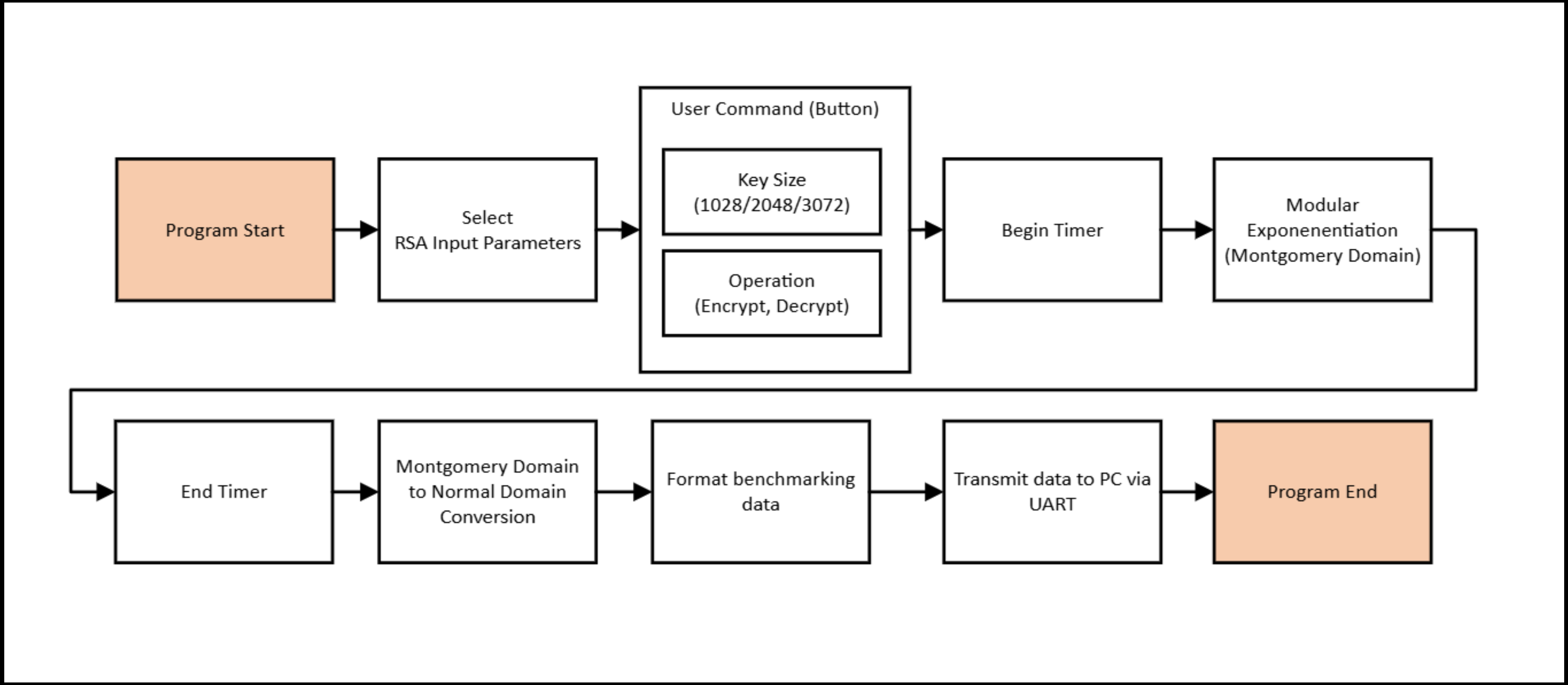


```
==============================
 RSA-2048 (HW: montgomery_axi_0) (key size: 2048 bits)
==============================
[DEBUG] Plaintext first 10 words (LE):
  0000002A 12345678 9ABCDEF0 0BADB002 C001D00D
  00FF00FF 55AA55AA CAFEBABE DEADBEEF 01020304
[DEBUG] HW ciphertext first 10 words:
  51F7A57E 4BFDBBC8 D8E9FE33 10D56C78 A4C3B29D
  7E3F19A2 2C44D0EF 99AA3377 5B6C7D8E F0123456
[DEBUG] SW ciphertext first 10 words:
  000009FD 89ABCDEF 13579BDF 2468ACE0 11223344
  55667788 99AABBCC DDEEFF00 0F1E2D3C 4B5A6978
[DEBUG] HW decrypted first 10 words:
  0000002A 12345678 9ABCDEF0 0BADB002 C001D00D
  00FF00FF 55AA55AA CAFEBABE DEADBEEF 01020304
[DEBUG] SW decrypted first 10 words:
  0000002A 12345678 9ABCDEF0 0BADB002 C001D00D
  00FF00FF 55AA55AA CAFEBABE DEADBEEF 01020304

[Performance] RSA-2048 (HW: montgomery_axi_0)
 HW enc: avg 7977283 cycles, 12307692 ns, 81 ops/s
 HW dec: avg 12019190 cycles, 18461538 ns, 54 ops/s
 SW enc: avg 60840922 cycles, 92307692 ns, 10 ops/s
 SW dec: avg 148920320 cycles, 230769231 ns, 4 ops/s
 Enc speedup (SW/HW): 7.500x
 Dec speedup (SW/HW): 12.500x

[Correctness]
 HW dec == msg: OK
 SW dec == msg: OK

==============================
 RSA-1024 (HW: montgomery_axi_1024) (key size: 1024 bits)
==============================
[DEBUG] Plaintext first 10 words (LE):
  0000002A 0F0F0F0F AAAAAAAA 13572468 89ABCDEF
  00112233 44556677 8899AABB CCDDEEFF DEADC0DE
[DEBUG] HW ciphertext first 10 words:
  A1B2C3D4 55667788 99AABBCC DDEEFF00 10203040
  0A0B0C0D 1E2F3A4B 5C6D7E8F ABCDEF01 23456789
[DEBUG] SW ciphertext first 10 words:
  0000043A F0E1D2C3 B4A59687 78695A4B 3C2D1E0F
  01010101 12341234 FEDCBA98 76543210 0C0D0E0F
[DEBUG] HW decrypted first 10 words:
  0000002A 0F0F0F0F AAAAAAAA 13572468 89ABCDEF
  00112233 44556677 8899AABB CCDDEEFF DEADC0DE
[DEBUG] SW decrypted first 10 words:
  0000002A 0F0F0F0F AAAAAAAA 13572468 89ABCDEF
  00112233 44556677 8899AABB CCDDEEFF DEADC0DE

[Performance] RSA-1024 (HW: montgomery_axi_1024)
 HW enc: avg 1966284 cycles, 3076923 ns, 325 ops/s
 HW dec: avg 3609722 cycles, 5384615 ns, 185 ops/s
```

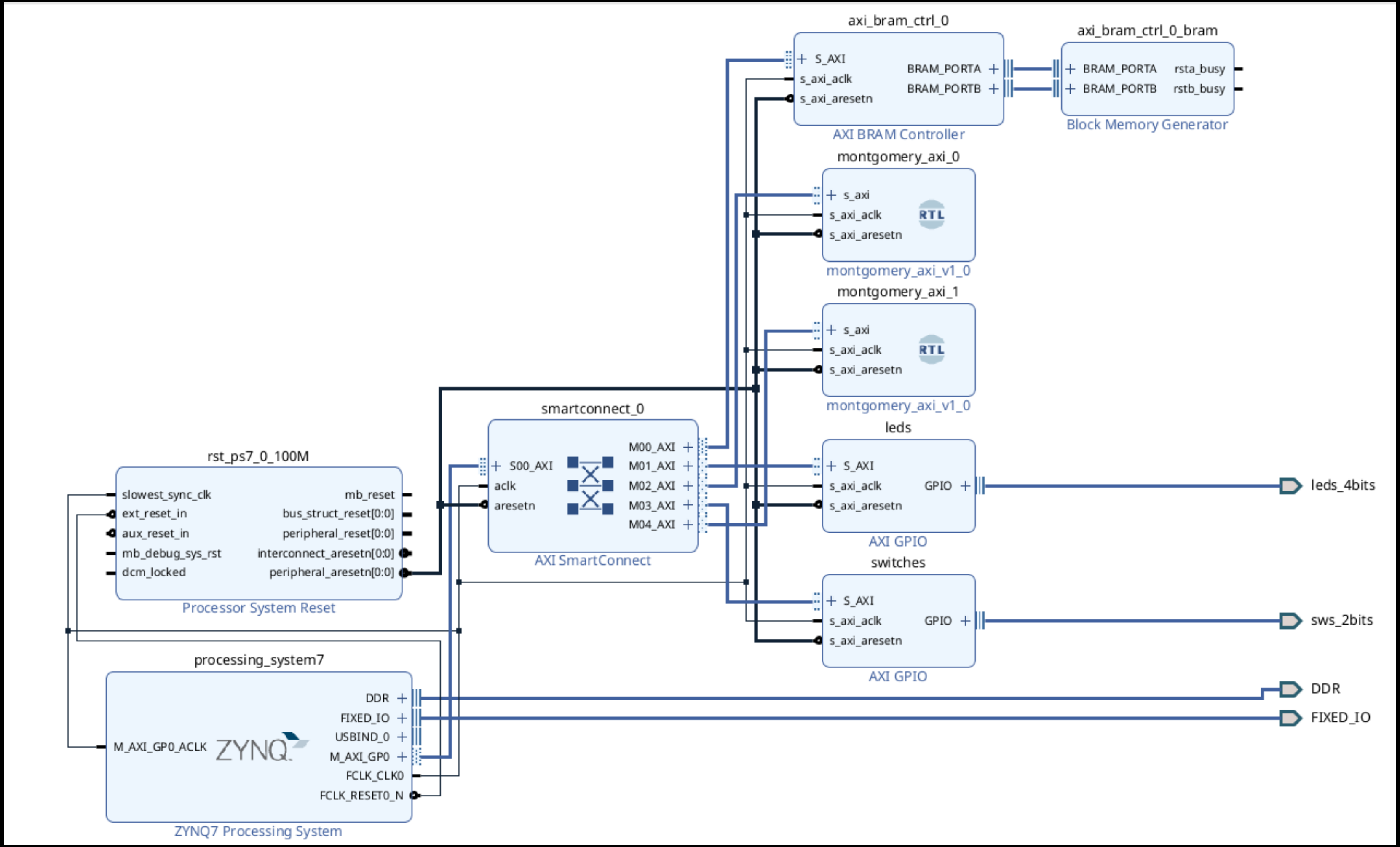
SOFTWARE BASELINE

- RSA executed entirely in software
- Platform: CPU only (no FPGA logic)
- Implemented in C and executed on Pynq-Z2 processing system
- Serves as reference to compare with hardware acceleration
- Measures encryption and decryption time for different key sizes

HIGH LEVEL BLOCK DIAGRAM

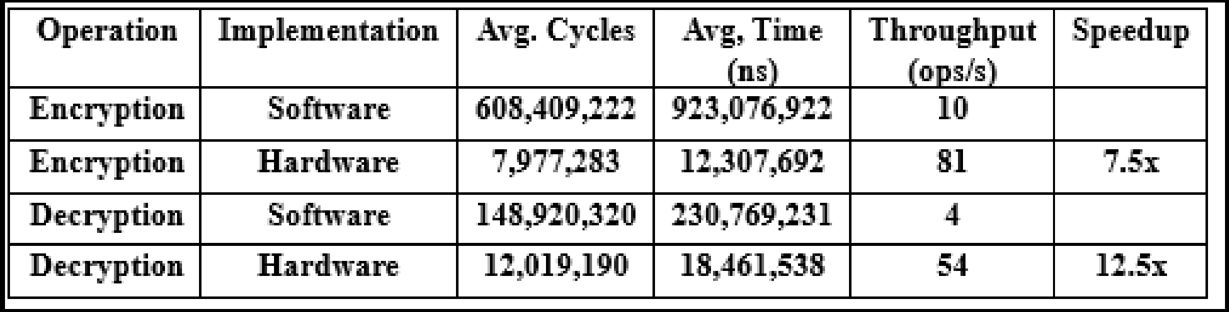

Program Start
Select RSA Input Parameters
User Command (Button)
Key Size (1028/2048/3072)
Operation (Encrypt, Decrypt)
Begin Timer
Modular Exponenentiation (Montgomery Domain)
End Timer
Montgomery Domain to Normal Domain Conversion
Format benchmarking data
Transmit data to PC via UART
Program End

FPGA IP CORE BLOCK DIAGRAM



BENCHMARK DATA – RSA 2048

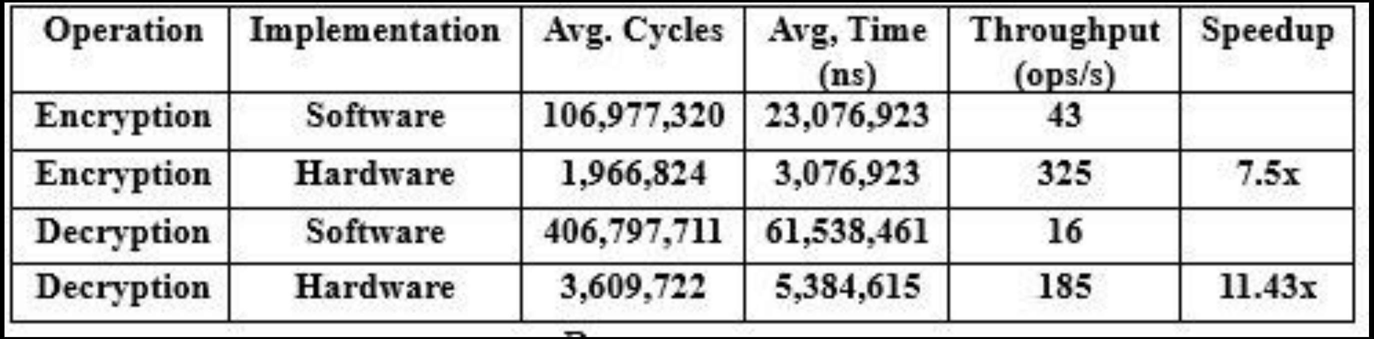
- Hardware speedup:
- ~7.5× for encryption
- ~12.5× for decryption
- Software: 6.08×10^8 cycles (enc) vs. 7.98×10^6 cycles (hw)
- Hardware significantly reduces runtime even though only Montgomery multiplication is accelerated



Operation	Implementation	Avg. Cycles	Avg, Time (ns)	Throughput (ops/s)	Speedup
Encryption	Software	608,409,222	923,076,922	10	
Encryption	Hardware	7,977,283	12,307,692	81	7.5x
Decryption	Software	148,920,320	230,769,231	4	
Decryption	Hardware	12,019,190	18,461,538	54	12.5x

BENCHMARK DATA – RSA-1024

- Hardware speedup:
- ~7.5× for encryption
- ~11.4× for decryption
- Similar trend to RSA-2048
- Higher decryption benefit due to more multiplications in exponentiation loop
- Verified bit-accurate correctness in all test cases



Operation	Implementation	Avg. Cycles	Avg, Time (ns)	Throughput (ops/s)	Speedup
Encryption	Software	106,977,320	23,076,923	43	
Encryption	Hardware	1,966,824	3,076,923	325	7.5x
Decryption	Software	406,797,711	61,538,461	16	
Decryption	Hardware	3,609,722	5,384,615	185	11.43x

ANALYSIS

- Iterative architecture provides strong area-time tradeoff
- Hardware parallelism accelerates most expensive arithmetic primitive
- AXI4-Lite overhead minimal relative to computation savings
- CPU load reduced; FPGA handles large-integer arithmetic efficiently

CONCLUSION

- Offloading Montgomery multiplication dramatically increases RSA performance
- Achieved 7.5×–12.5× speedup across key sizes
- Modular architecture supports scaling and future RSA datapath expansion
- Confirms viability of FPGA-based cryptographic acceleration on embedded SoCs

FUTURE IMPLEMENTATION

- Full modular exponentiation accelerator
- Pipelined Montgomery multiplier or parallel word blocks
- Support for 3072-bit and larger RSA keys
- Explore alternative softcore or RISC-V processors on FPGA
- Comparative energy-efficiency analysis