KU LEUVEN

DDP – Final Presentation – Group 31

A co-design approach for implementing the RSA algorithm

Our implementation

- ~22 millions of CPU cycles for decrypting
- 33.61 % LUTs usage
- 19.85 % of Registers

Optimizing Area impacts positively

The speed

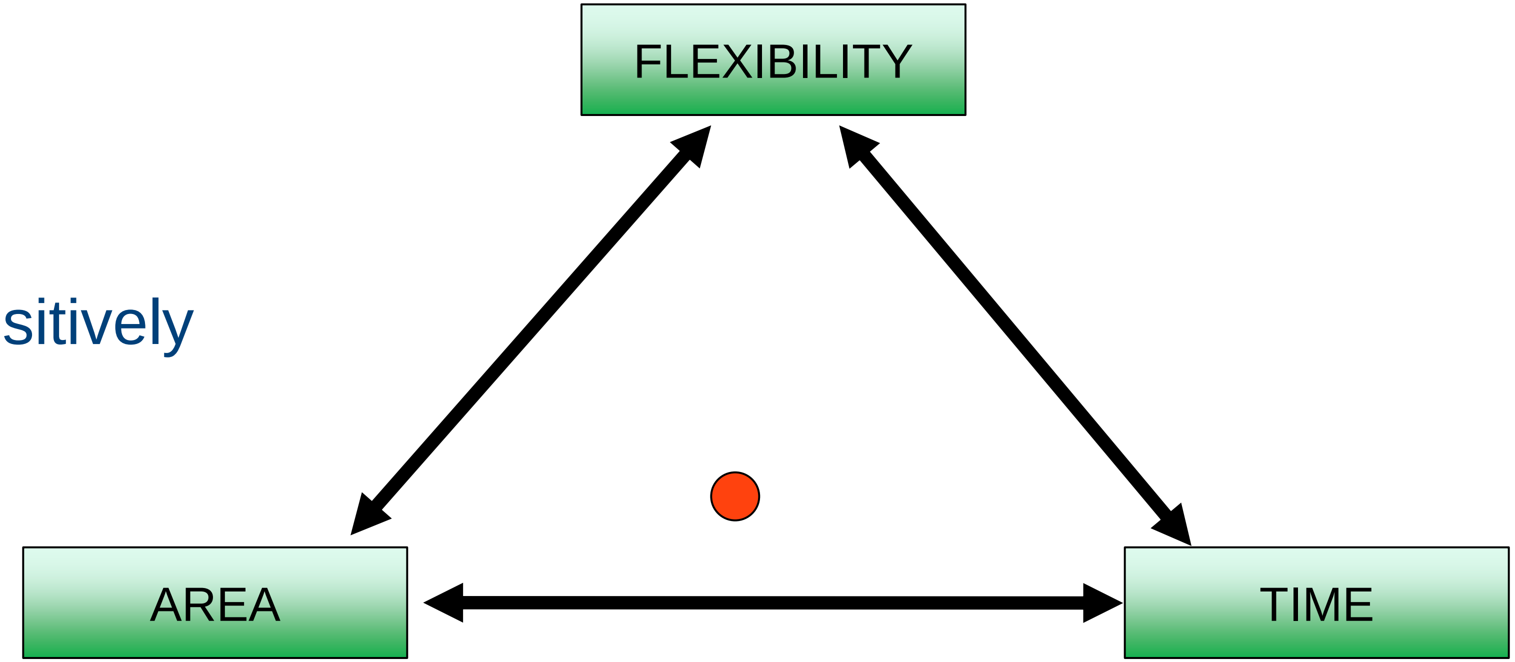


Fig.1 : Performance triangle

HW/SW boundaries

- Loading phase for R2_N and N
- Power Ladder Algorithm
  4 possible stages
  1. First X_tilde
  2. Bit = 1
  3. Bit = 0
  4. Mont(A,1,N)

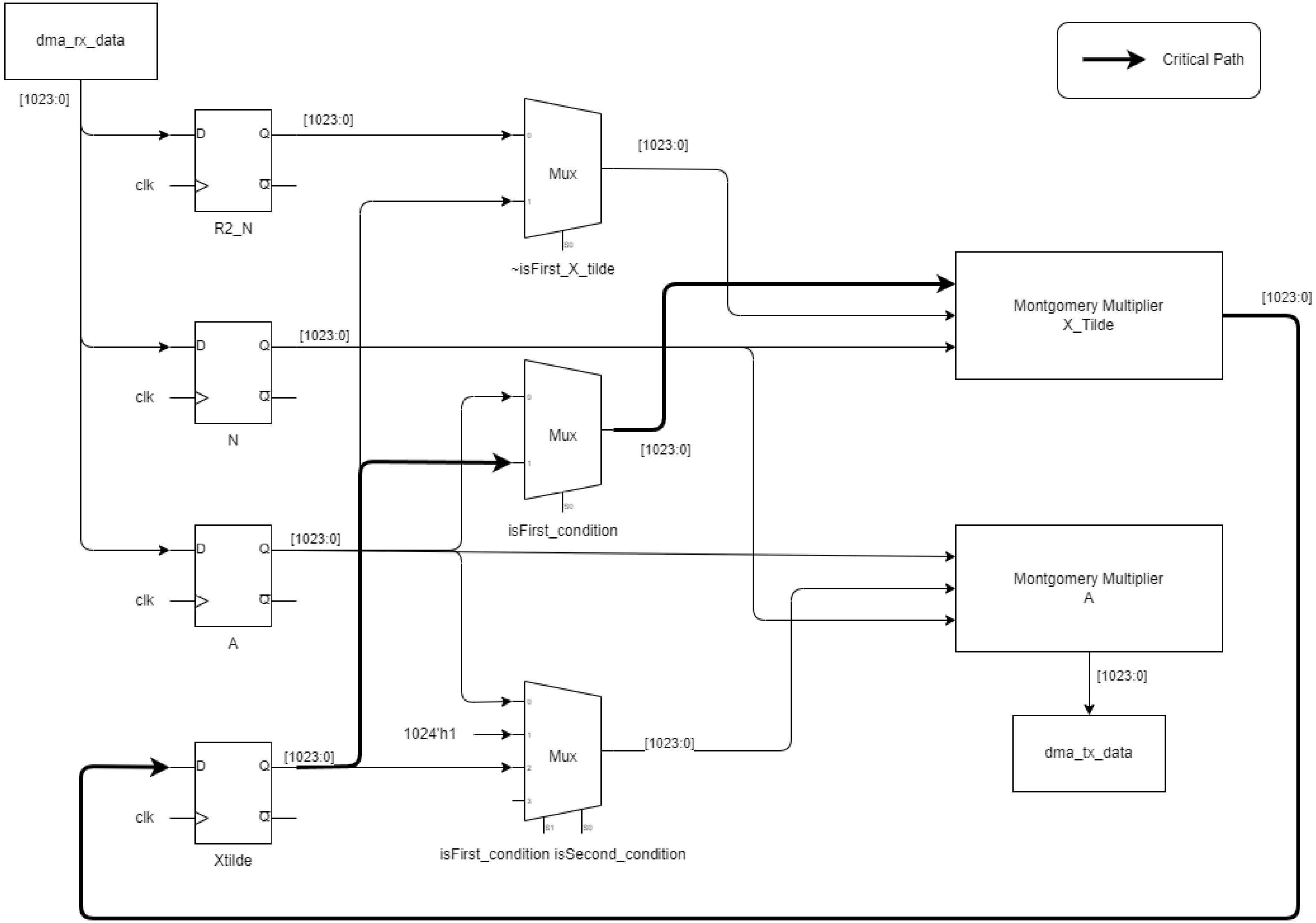
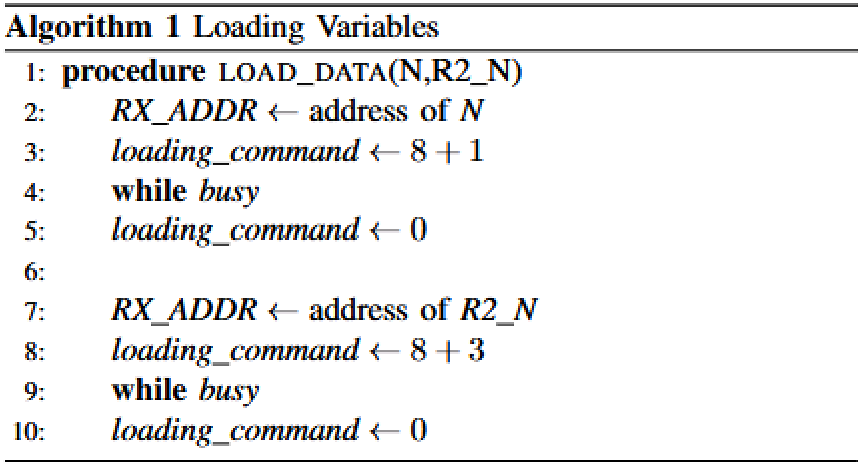


Fig.2 : Hardware of the Power Ladder implementation

HW/SW interface

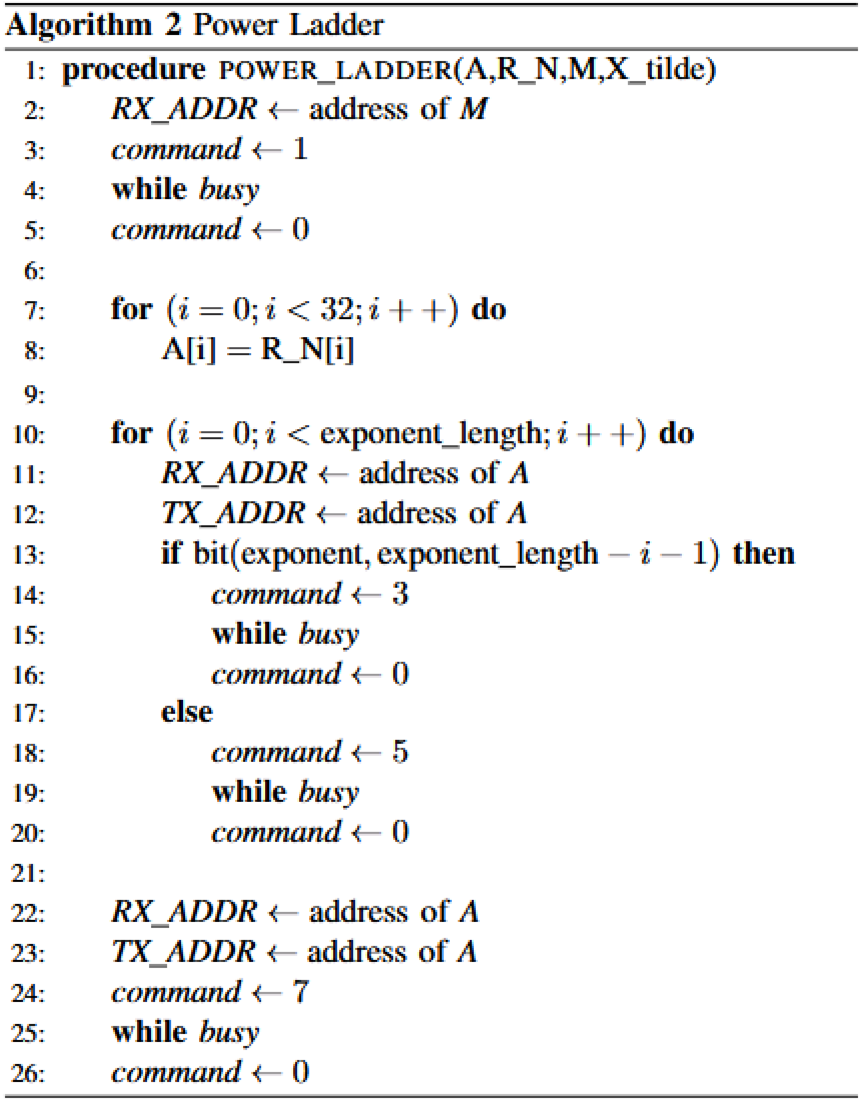


Algorithm 1 Loading Variables

```
procedure LOAD_DATA(N,R2_N)
    $RX\_ADDR \leftarrow$ address of $N$
    $loading\_command \leftarrow 8 + 1$
    while $busy$
    $loading\_command \leftarrow 0$

    $RX\_ADDR \leftarrow$ address of $R2\_N$
    $loading\_command \leftarrow 8 + 3$
    while $busy$
    $loading\_command \leftarrow 0$
```

Fig.3 : Loading stage



Algorithm 2 Power Ladder

```
procedure POWER_LADDER(A,R_N,M,X_tilde)
    $RX\_ADDR \leftarrow$ address of $M$
    $command \leftarrow 1$
    while $busy$
    $command \leftarrow 0$

    for $(i = 0; i < 32; i++)$ do
        A[i] = R_N[i]

    for $(i = 0; i < \mathrm{exponent\_length}; i++)$ do
        $RX\_ADDR \leftarrow$ address of $A$
        $TX\_ADDR \leftarrow$ address of $A$
        if $\mathrm{bit}(\mathrm{exponent}, \mathrm{exponent\_length} - i - 1)$ then
            $command \leftarrow 3$
            while $busy$
            $command \leftarrow 0$
        else
            $command \leftarrow 5$
            while $busy$
            $command \leftarrow 0$

    $RX\_ADDR \leftarrow$ address of $A$
    $TX\_ADDR \leftarrow$ address of $A$
    $command \leftarrow 7$
    while $busy$
    $command \leftarrow 0$
```

Fig.4 : Power Ladder stage

- + 2 distinctive stages : added modularity
- - Hard coding the algorithm : reduces flexibility

Results

- Hardware
  - CPU Cycles : 381 100
  - LUTs : 17 614
  - REGs : 21 123
  - WNS : 0.102 ns
- Software
  - Works with all test vectors

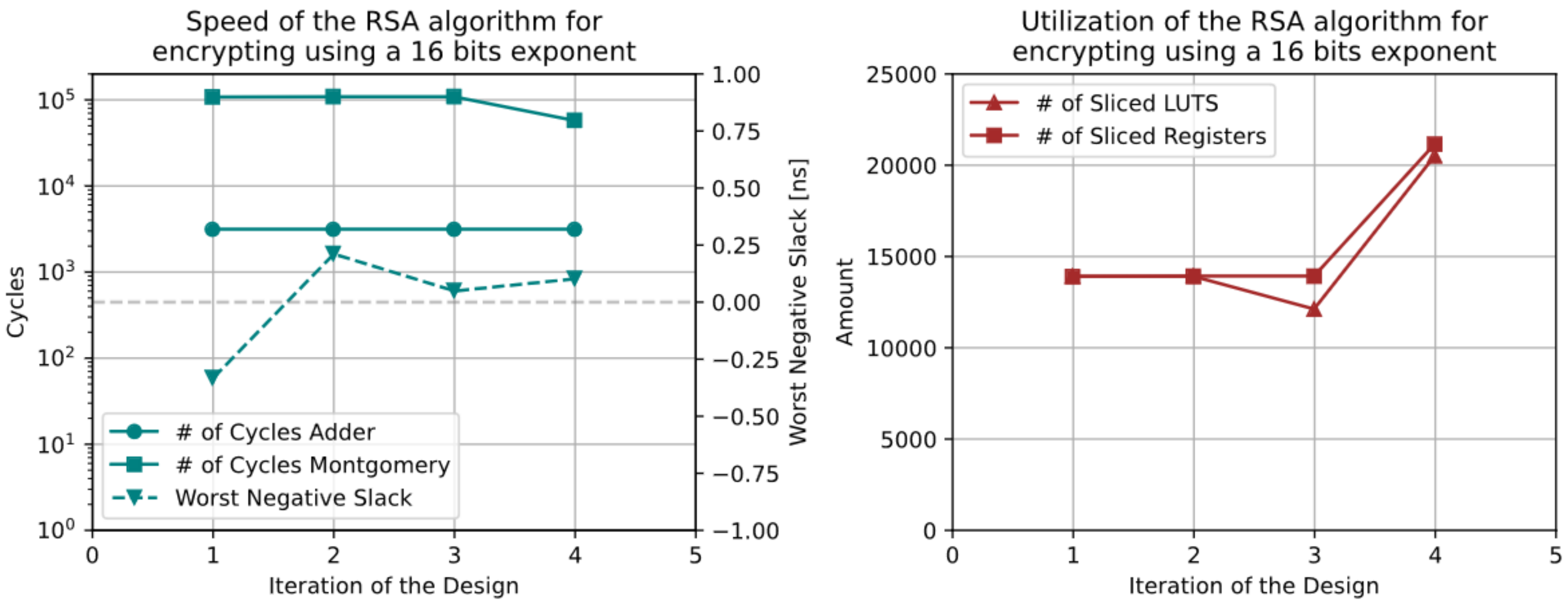


Fig.5 : Performance throughout the iterations

Going Beyond

- Better User Experience
  - Can encrypt user provided string of text
  - Small library of functions to add layer of abstraction
- Chinese Remainder Theorem [1]
  - Decrypting is expensive
    → reduce computation time
  - Divide and Conquer approach

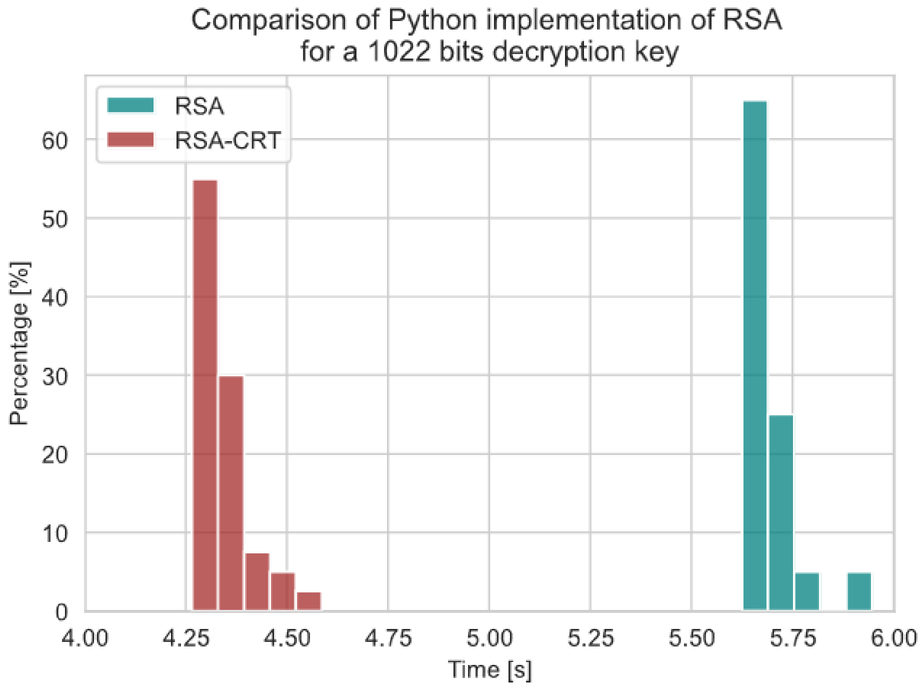


Fig.6 : Speed up of RSA decryption using the CRT

[1] : Special thanks to Artur Folwarczny for introducing us this piece of mathematics

Thank you

Any Questions ?

KU LEUVEN