RSA Cryptography

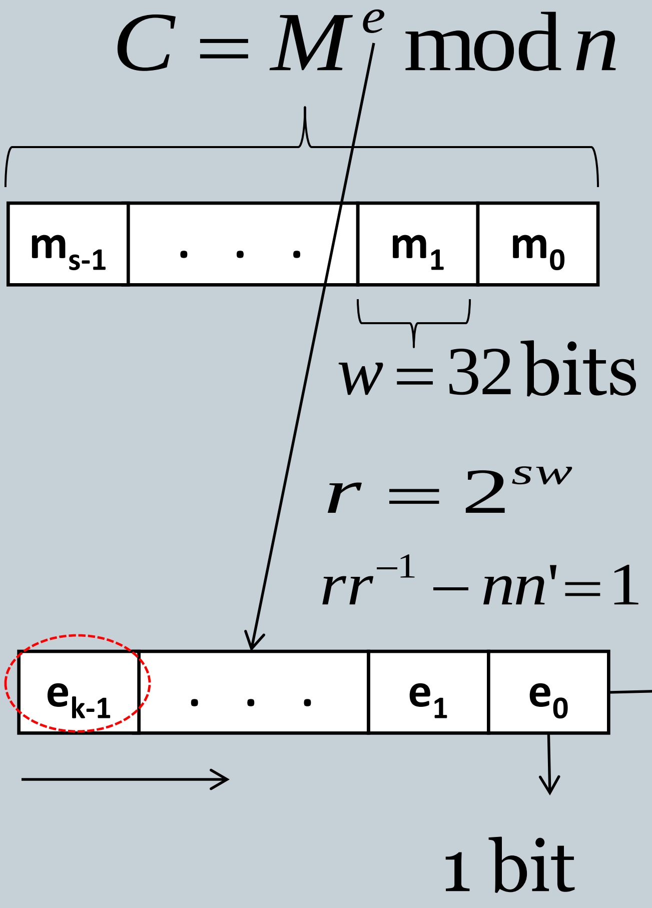
Nikola Bežanić
Jelena Popović-Božović
Veljko Milutinović
Ivan Popović

University of Belgrade
School of Electrical Engineering

Introduction

- Case study area: Public key cryptography acceleration
- Problem: RSA implementation on Maxeler
- Approach:
  - Accelerate multiplications
  - Analyze usability
- Conclusions:
  - Multiplication speedup: 70% (28% total)
  - Usability: Big data encryption
  - New perspective on RSA usage

RSA



$$C = M^e \bmod n$$

$$w = 32\,\text{bits}$$

$$r = 2^{sw}$$

$$rr^{-1} - nn' = 1$$

- Montgomery method: *n* -> *r*
- *r=2^sw -> power of 2*
- *Montgomery product (MonPro): modulo r arithmetic*

```
function ModExp(M, e, n)  {n is odd}
Step 1.   Compute n'.
Step 2.   $M_m := M \cdot r \bmod n$
Step 3.   $x_m := 1 \cdot r \bmod n$
Step 4.   for i = k – 1 down to 0 do
Step 5.       $x_m := MonPro(x_m, x_m)$
Step 6.       if $e_i = 1$ then $x_m := MonPro(M_m, x_m)$
Step 7.   $x := MonPro(x_m, 1)$
Step 8.   return x
```

Montgomery product

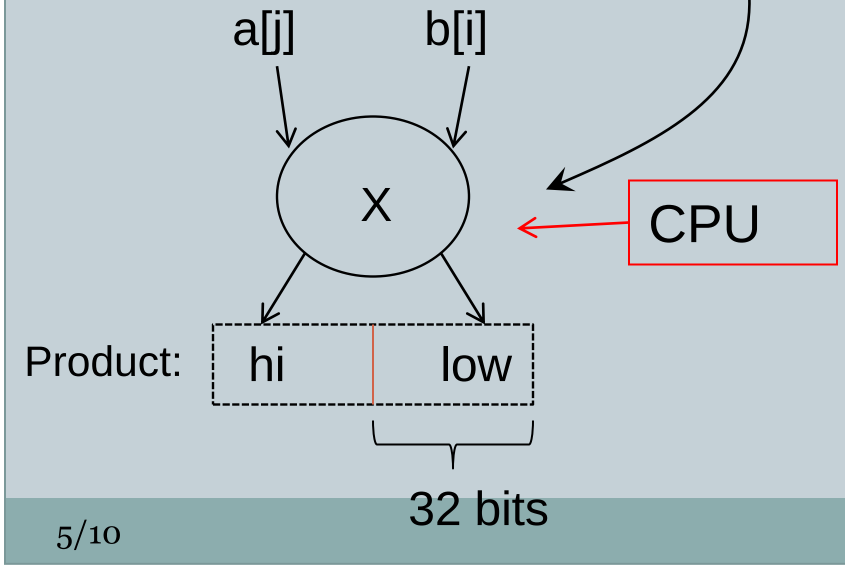
```
function MonPro(a, b)
Step 1.   $t := a \cdot b$
Step 2.   $m := t \cdot n' \bmod r$
Step 3.   $u := (t + m \cdot n) / r$
Step 4.   if $u \geq n$ then return $u - n$
                    else return $u$
```

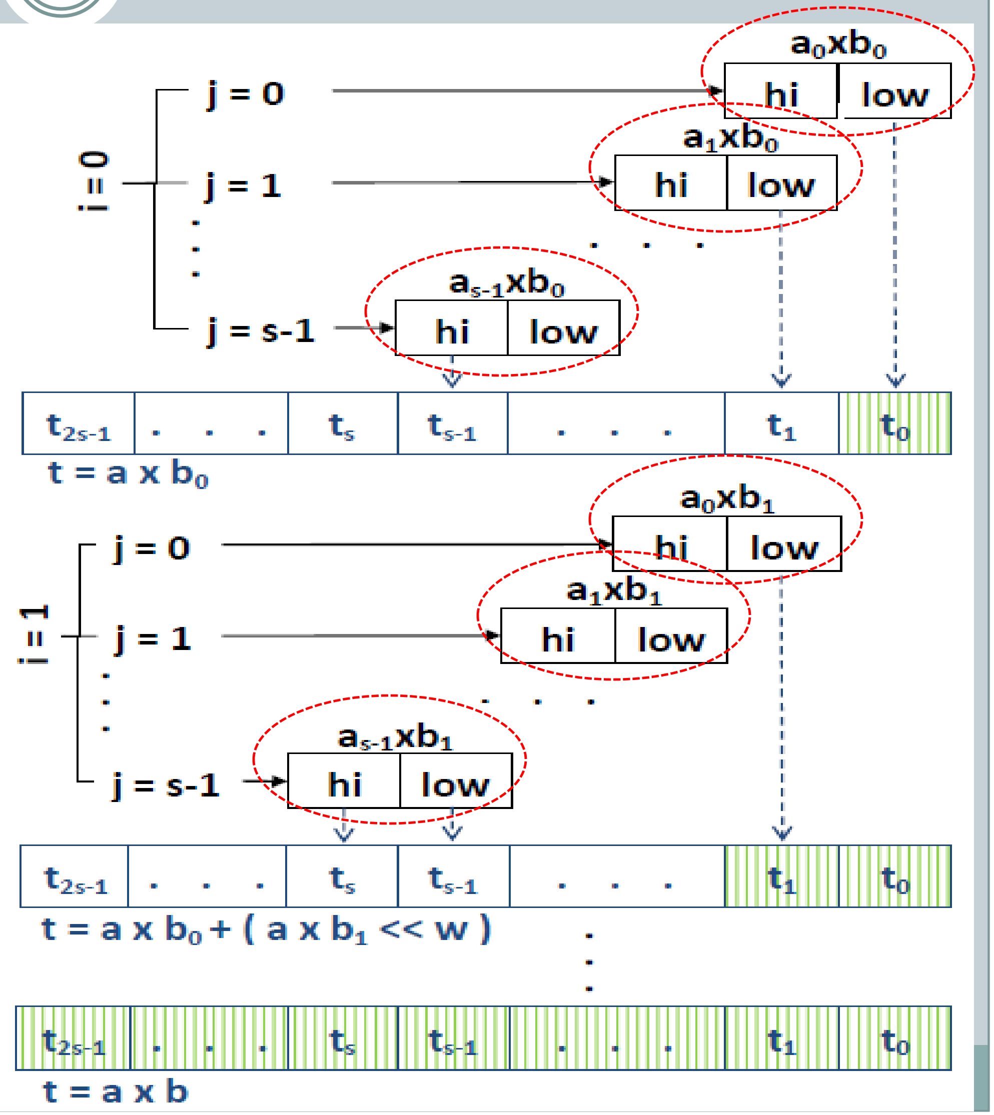
- *a* and *b* are big numbers
- Breaking them to digits:
  - ○ $b_{s-1} \ldots b_1 b_0$
  - ○ $a_{s-1} \ldots a_1 a_0$
- Processing on a word basis

```
for $i = 0$ to $s-1$
    $C := 0$
    for $j = 0$ to $s-1$
        $(C, S) := t[i + j] + a[j] \cdot b[i] + C$
        $t[i + j] := S$
    $t[i + S] := C$
```

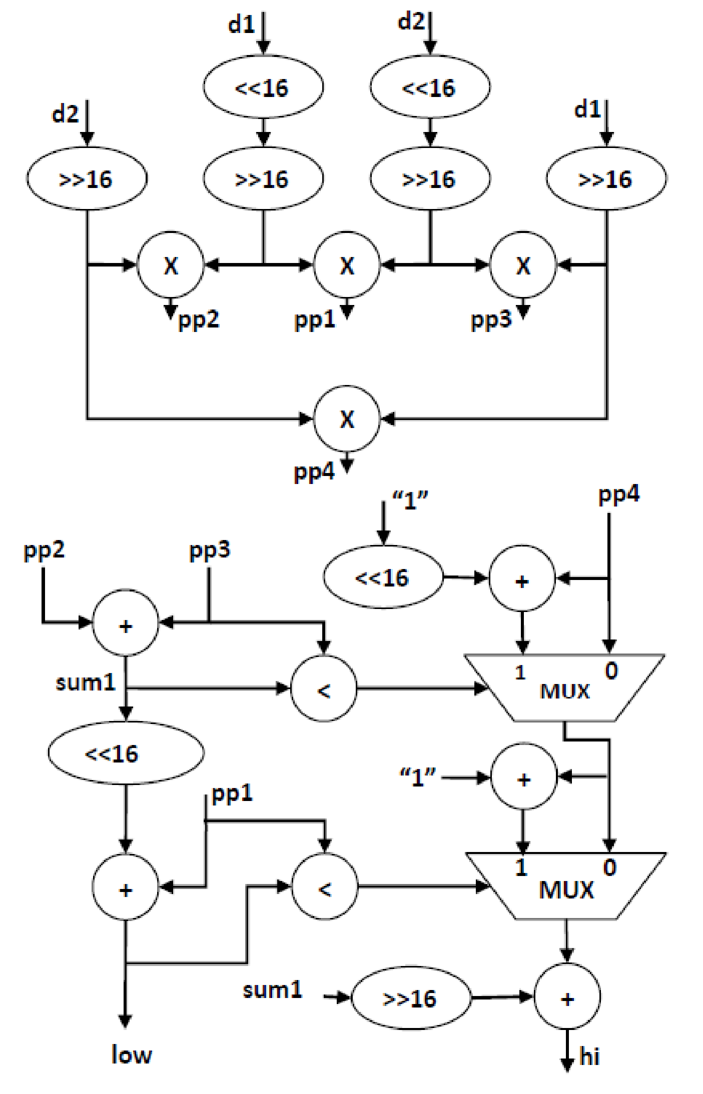
Montgomery product: Step 1

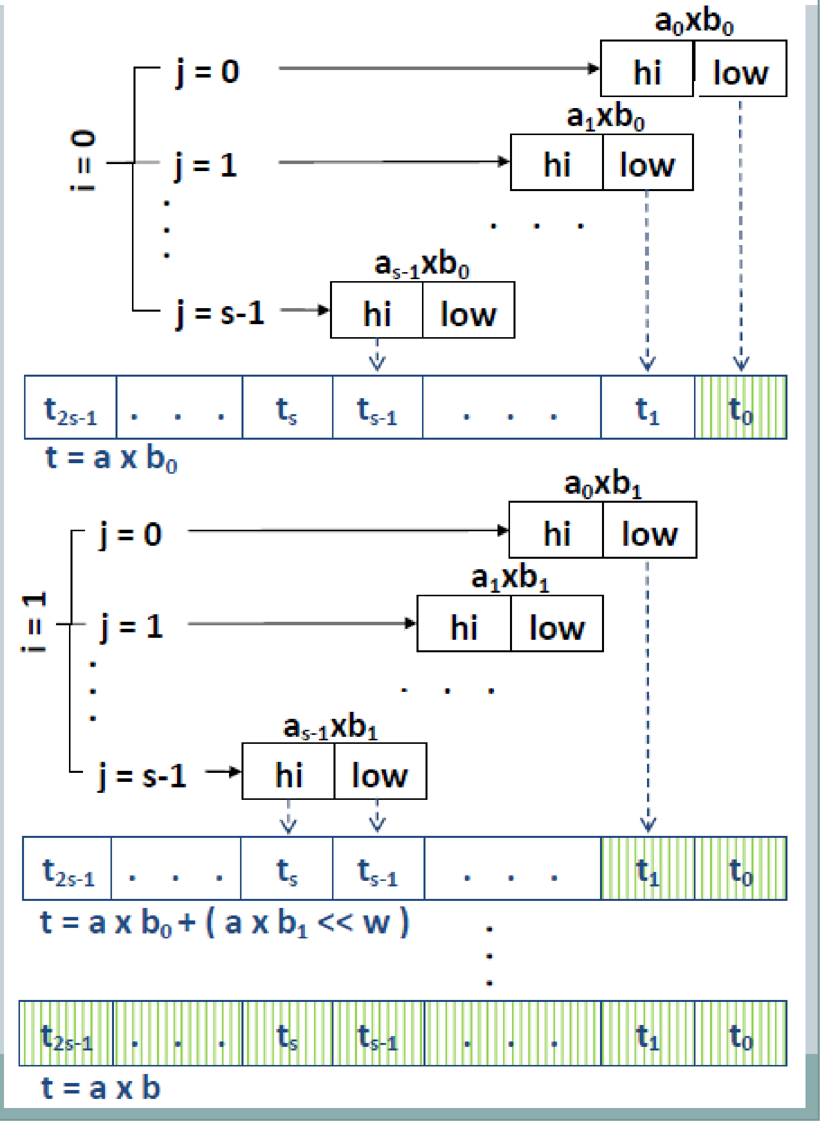
```
for i = 0 to s-1
    C := 0
    for j = 0 to s-1
        $(C, S) := t[i + j] + a[j] \cdot b[i] + C$
        $t[i + j] := S$
    $t[i + S] := C$
```



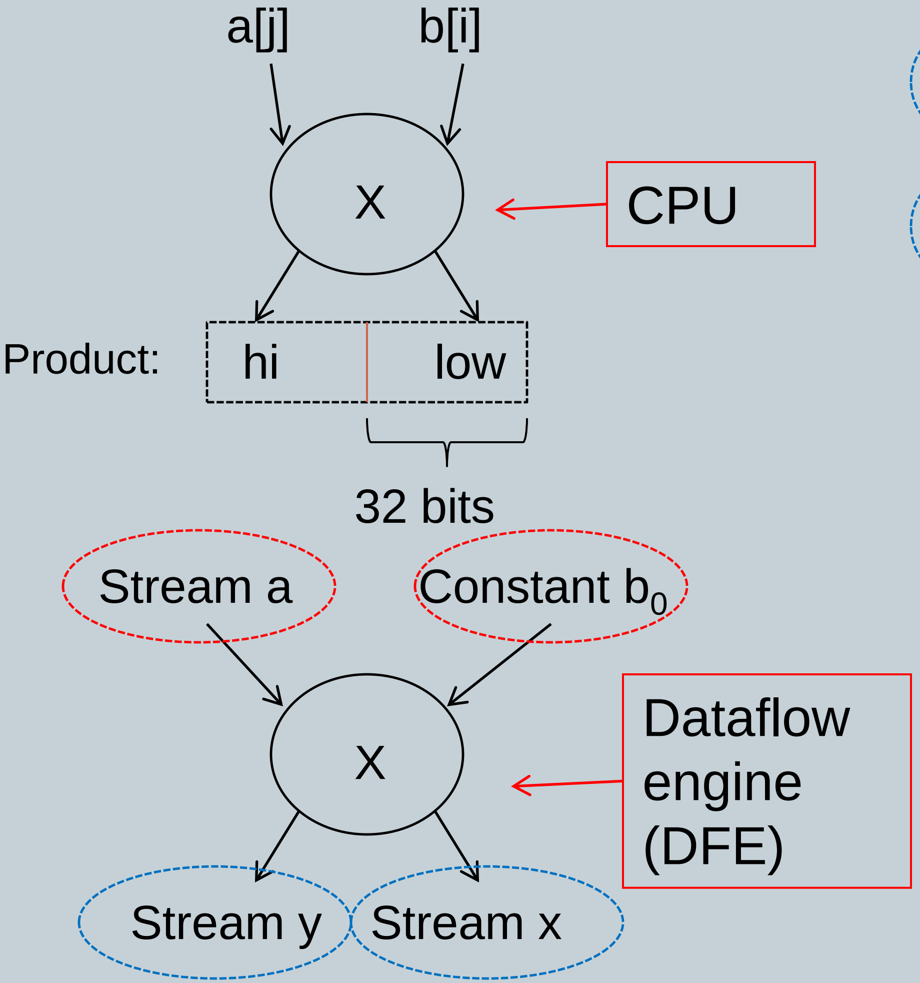


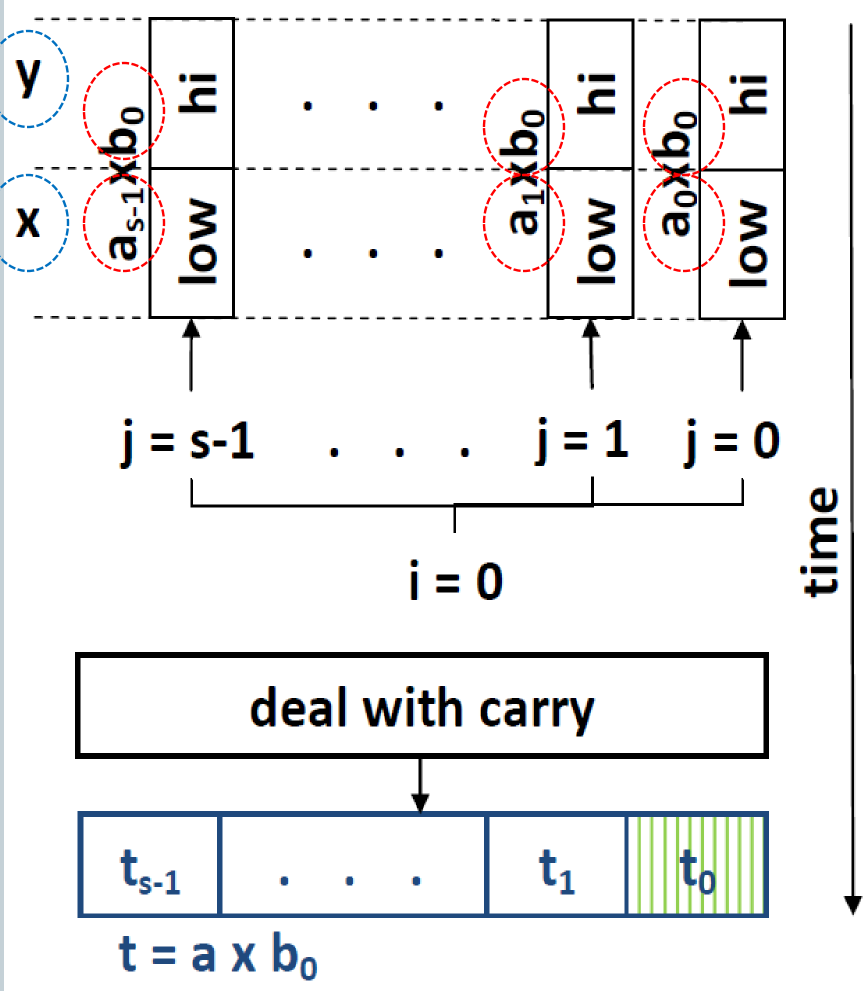
Kernel: *mult32x32*





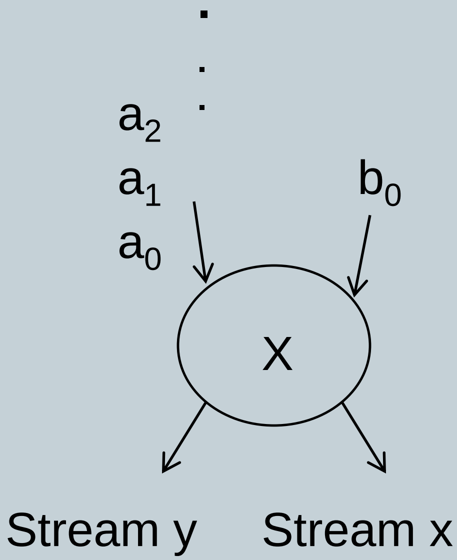
Dataflow multiplier





Dataflow multiplier: Pipeline problem

- Next iteration (next constant b_1) => new DFE run
- New DFE run => new pipeline fill-up overhead
- 1024-bits key requires only 32 digits (32 bits each)
- Not enough to fill-up the pipeline
- Result: CPU time < DFE time !
- Solution:
  - Work on blocks of data
  - Do not use constants, rather use a stream
  - Stream has redundant values: acts as a const.



Dataflow multiplier: Blocks of data

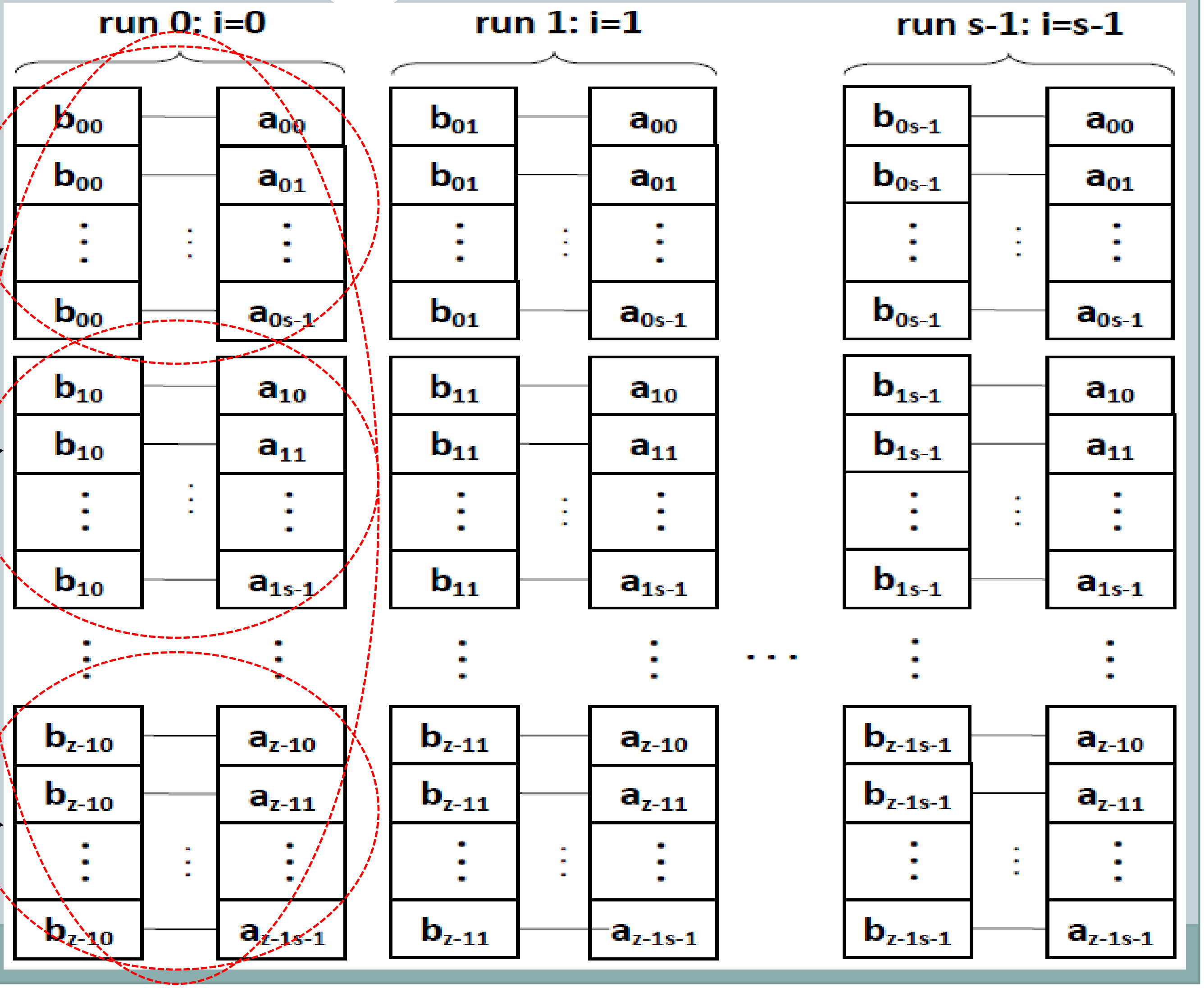
b_0 x a < = >

Block 0

Block 1

Big streams for each run

Block z-1



Results

- Using blocks pipeline is full
- Using one multiplier speed up is 10% for RSA
- Speedup is 70% for multiplication using 4 multiplers
- It leads to 28% for complete RSA (Amdahl’s law)
- Future work
  - Deal with carry at DFE or
  - Overlap carry propagation at CPU and multiplication at DFE

References

- Cetin Kaya Koc, High-Speed RSA Implementation, RSA Data Security, Inc., RSA Laboratories, Report, 1994.
- Peter L. Montgomery, Modular Multiplication Without Trial Division, Mathematics of Computation, 1985.
- http://home.etf.rs/~vm/os/vlsi/predavanja/maxeler.html
- Andrew Putnam, Aaron Smith, and Doug Burger, Dynamic Vectorization in the E2 Dynamic Multicore System, 1st International Workshop on Highly-efficient Accelerators and Reconfigurable Technologies (HEART), 2010.