Multiplication Modulaire de Montgomery avec une Architecture Systolique

MRABET Amine LASHERME Ronan et EL MRABET Nadia

LIASD Paris 8

SAS - CMP - Gardanne

RAIM 2015, avril 7-9, Rennes

Plan

1. Introduction
2. Multiplication Motgomery (CIOS)
3. Architecture
4. Résultat
5. Conclusion et Perspectives

Plan

Contexte Générale

- Ce travail s'inscrit dans le cadre général de l'implémentation Hardware des primitives de cryptographie asymétrique, tel que le couplage d'Optimal-Ate à base des courbes elliptiques, les systèmes cryptographiques à base de courbes elliptiques et RSA,

Contexte Générale

- Ce travail s'inscrit dans le cadre général de l'implémentation Hardware des primitives de cryptographie asymétrique, tel que le couplage d'Optimal-Ate à base des courbes elliptiques, les systèmes cryptographiques à base de courbes elliptiques et RSA,

- qui sont les méthodes les plus connues en chiffrement asymétrique.

Rappel

Algorithm 1: Montgomery Modular Multiplication

Input: p an odd prime, $n = \lceil log_2(p) \rceil$, $R = 2^n$, $p' = -p^{-1} mod\ R$, $M(a)$, $M(b) \in \mathbb{F}_p$

Output: $M(ab) \mod p$

1 $\gamma \leftarrow M(a) \times M(b)$

2 $\delta \leftarrow \gamma \times p' \mod R$

3 $T \leftarrow \frac{\gamma + \delta \times p}{R}$

4 **If** $T \geq p$ **then** $T \leftarrow T - p$

5 **return** T

La méthode Coarsely Integrated Operand Scanning [1] ?

- La méthode CIOS permet d'améliorer l'algorithme de Montgomery en intégrant la multiplication et la réduction.

Comment?

[1] Analyzing and Comparing Montgomery Multiplication Algorithms, IEEE Micro. , juin1996
Cetin Kaya Koç, Tolga Acar and Burton S. Kaliski Jr.

La méthode Coarsely Integrated Operand Scanning [1] ?

- La méthode CIOS permet d'améliorer l'algorithme de Montgomery en intégrant la multiplication et la réduction.

Comment?

- Au lieu de multiplier axb puis passer à la réduction, elle permet d'alterner entre les itérations de la multiplication et de la réduction.

[1] Analyzing and Comparing Montgomery Multiplication Algorithms, IEEE Micro. , juin1996
Cetin Kaya Koç, Tolga Acar and Burton S. Kaliski Jr.

Qu'est ce qu'une architecture systolique ?

- C'est un réseau composé d'un grand nombre de cellules , Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines.

Qu'est ce qu'une architecture systolique ?

- C'est un réseau composé d'un grand nombre de cellules , Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines.
- Une architecture systolique fournit des cellules élémentaires très simplifiées. Par conséquent, cette architecture diminue les besoins en ressources dans les implémentations matérielles.

Qu'est ce qu'une architecture systolique ?

- C'est un réseau composé d'un grand nombre de cellules , Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines.
- Une architecture systolique fournit des cellules élémentaires très simplifiées. Par conséquent, cette architecture diminue les besoins en ressources dans les implémentations matérielles.
- Notre contribution dans ce travail est de combiner une architecture systolique, qui est supposé être la meilleure solution pour les implémentations FPGA, avec la méthode CIOS de la multiplication modulaire Montgomery.

Plan

1. Introduction
2. Multiplication Motgomery (CIOS)
3. Architecture
4. Résultat
5. Conclusion et Perspectives

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p'$ mod 2^w
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $C \leftarrow 0$;
4 **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $(C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $T[j] \leftarrow S$
7 $(C, S) \leftarrow T[s] + C$
8 $T[s] \leftarrow S$
9 $T[s + 1] \leftarrow C$
10 $C \leftarrow 0$;
11 $m \leftarrow T[0] \cdot p'$ mod 2^w
12 $(C, S) \leftarrow T[0] + m \cdot p[0]$
13 **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $(C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $T[j] \leftarrow S$
16 $(C, S) \leftarrow T[s] + C$
17 $T[s - 1] \leftarrow S$
18 $T[s] \leftarrow T[s + 1] + C$
19 **return** T;

α — alpha : les lignes 5 et 6

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p'$ mod 2^w
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

α — alpha : les lignes 5 et 6

α_2 — alpha : les lignes 7,8 et 9

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

```
1  T ← Null;
2  for i ← 0 to s − 1 do
3      C ← 0;
4      for j ← 0 to s − 1 do
5          (C, S) ← T[j] + a[i] · b[j] + C
6          T[j] ← S
7      (C, S) ← T[s] + C
8      T[s] ← S
9      T[s + 1] ← C
10     C ← 0;
11     m ← T[0] · p' mod 2^w
12     (C, S) ← T[0] + m · p[0]
13     for j ← 1 to s − 1 do
14         (C, S) ← T[j] + m · p[j] + C
15         T[j] ← S
16     (C, S) ← T[s] + C
17     T[s − 1] ← S
18     T[s] ← T[s + 1] + C
19 return T;
```

α — alpha : les lignes 5 et 6

α_2 — alpha : les lignes 7,8 et 9

β — beta: les lignes 11 et 12

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p'$ mod 2^w
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

α — alpha : les lignes 5 et 6

α_2 — alpha : les lignes 7,8 et 9

β — beta: les lignes 11 et 12

γ — gamma: les lignes 14 et 15

Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p'$ mod 2^w
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

α — alpha : les lignes 5 et 6

α_2 — alpha : les lignes 7,8 et 9

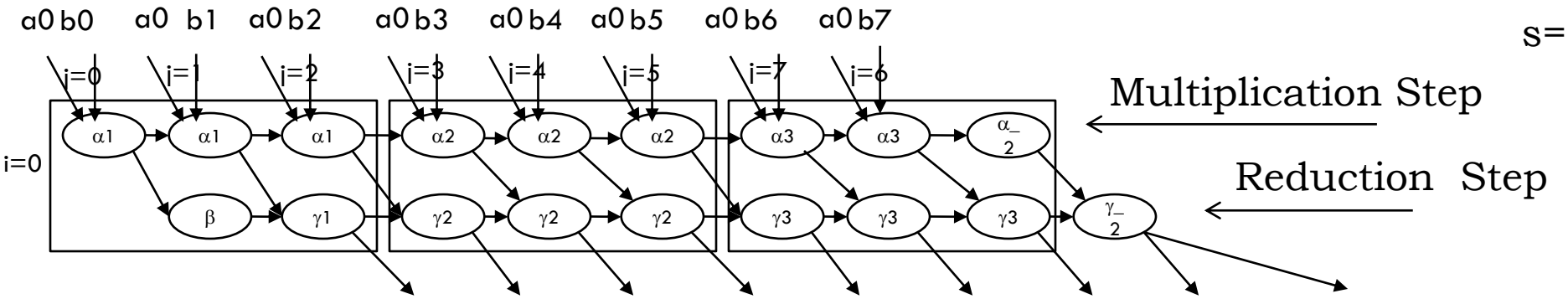
β — beta: les lignes 11 et 12

γ — gamma: les lignes 14 et 15

γ_2 — gamma: les lignes 16,17 et 18

Plan

1. Introduction
2. Multiplication Motgomery (CIOS)
3. Architecture
4. Résultat
5. Conclusion et Perspectives



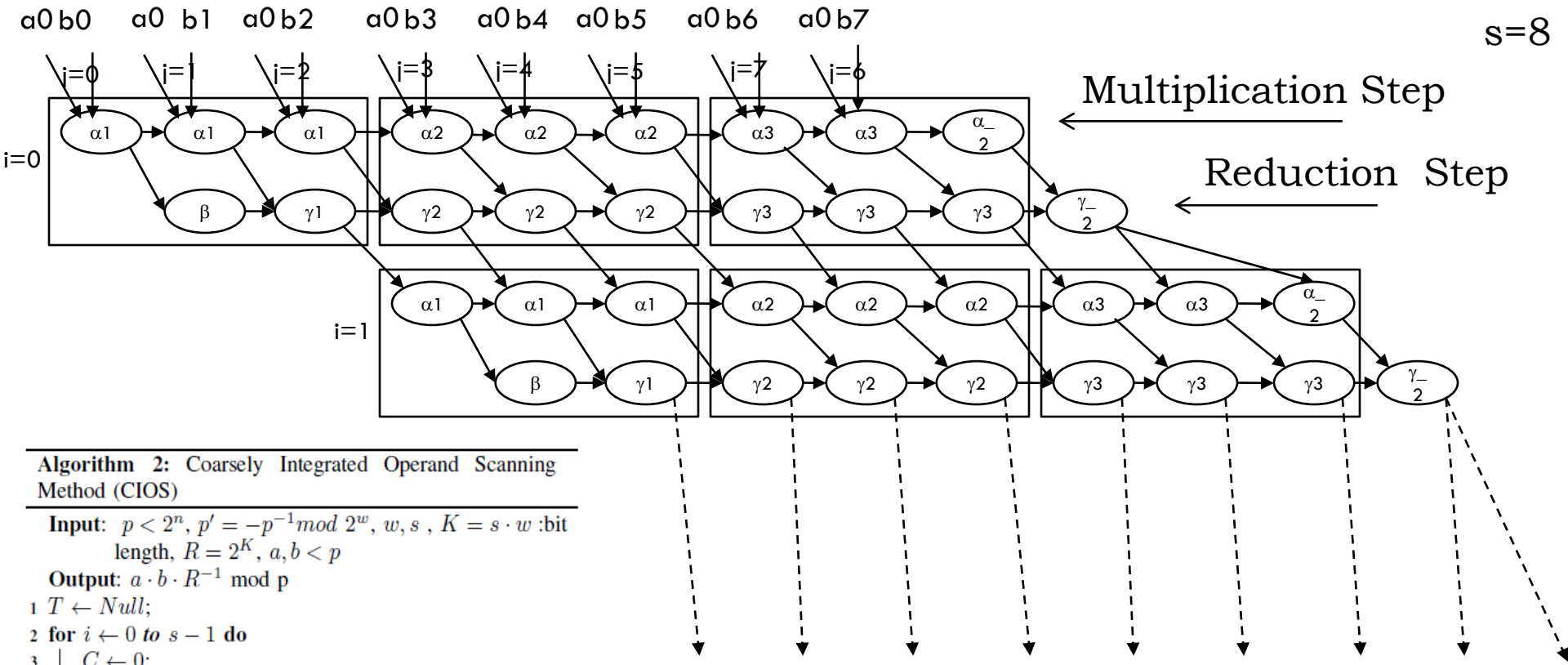
Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$

Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do**
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p' \bmod 2^w$
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do**
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

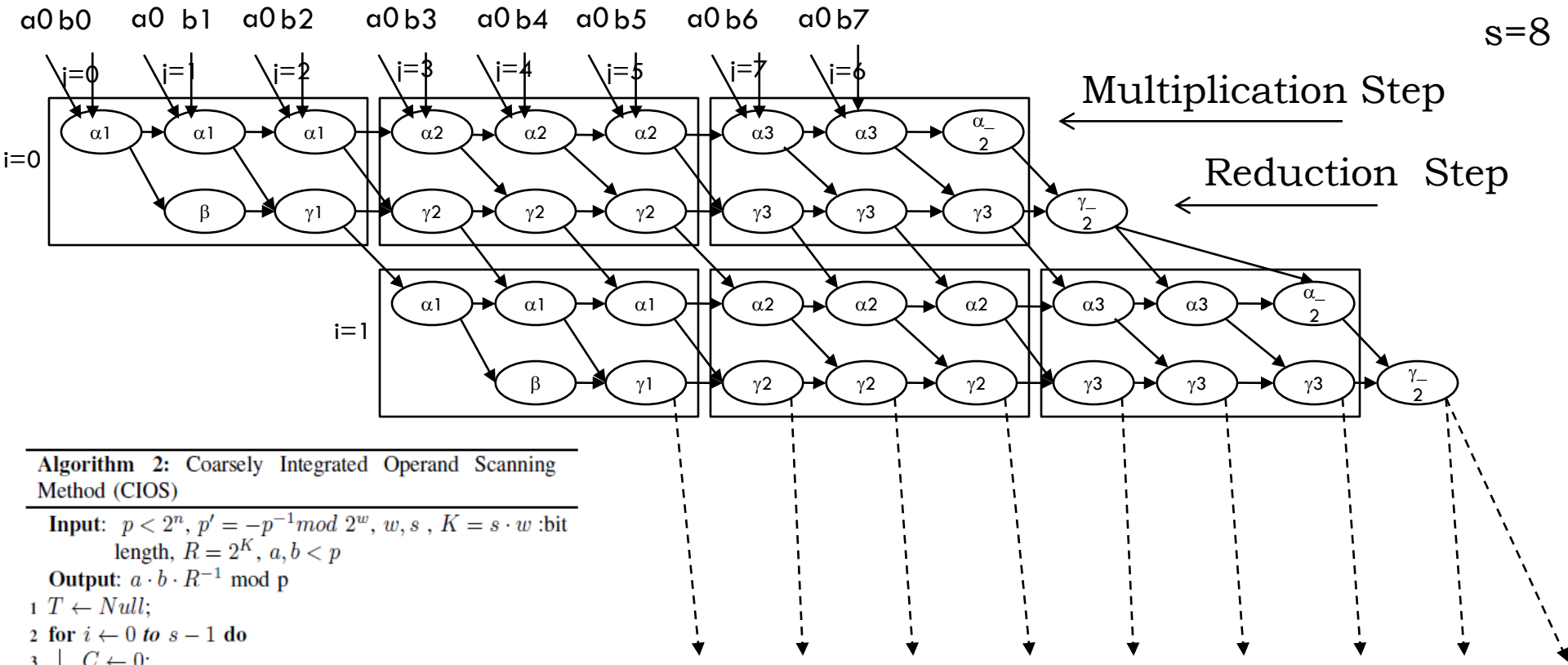
α
α_2
β
γ
γ_2



Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$
Output: $a \cdot b \cdot R^{-1}$ mod p

1 $T \leftarrow Null$;
2 **for** $i \leftarrow 0$ ***to*** $s - 1$ **do**
3 $\quad C \leftarrow 0$;
4 $\quad$ **for** $j \leftarrow 0$ ***to*** $s - 1$ **do** — α
5 $\quad\quad (C, S) \leftarrow T[j] + a[i] \cdot b[j] + C$
6 $\quad\quad T[j] \leftarrow S$
7 $\quad (C, S) \leftarrow T[s] + C$ — α_2
8 $\quad T[s] \leftarrow S$
9 $\quad T[s + 1] \leftarrow C$
10 $\quad C \leftarrow 0$;
11 $\quad m \leftarrow T[0] \cdot p' \bmod 2^w$ — β
12 $\quad (C, S) \leftarrow T[0] + m \cdot p[0]$
13 $\quad$ **for** $j \leftarrow 1$ ***to*** $s - 1$ **do** — γ
14 $\quad\quad (C, S) \leftarrow T[j] + m \cdot p[j] + C$
15 $\quad\quad T[j] \leftarrow S$
16 $\quad (C, S) \leftarrow T[s] + C$ — γ_2
17 $\quad T[s - 1] \leftarrow S$
18 $\quad T[s] \leftarrow T[s + 1] + C$
19 **return** T;

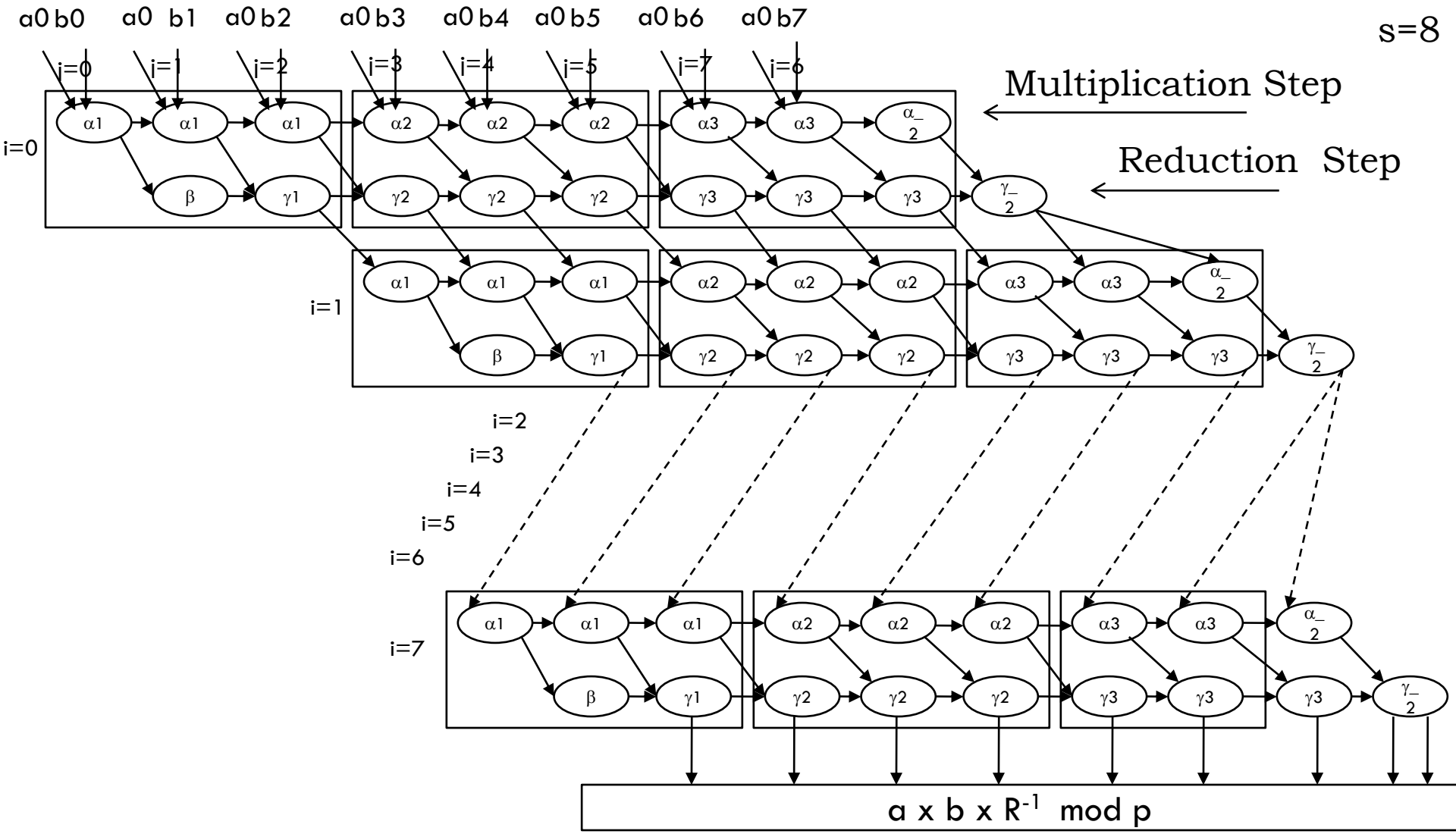


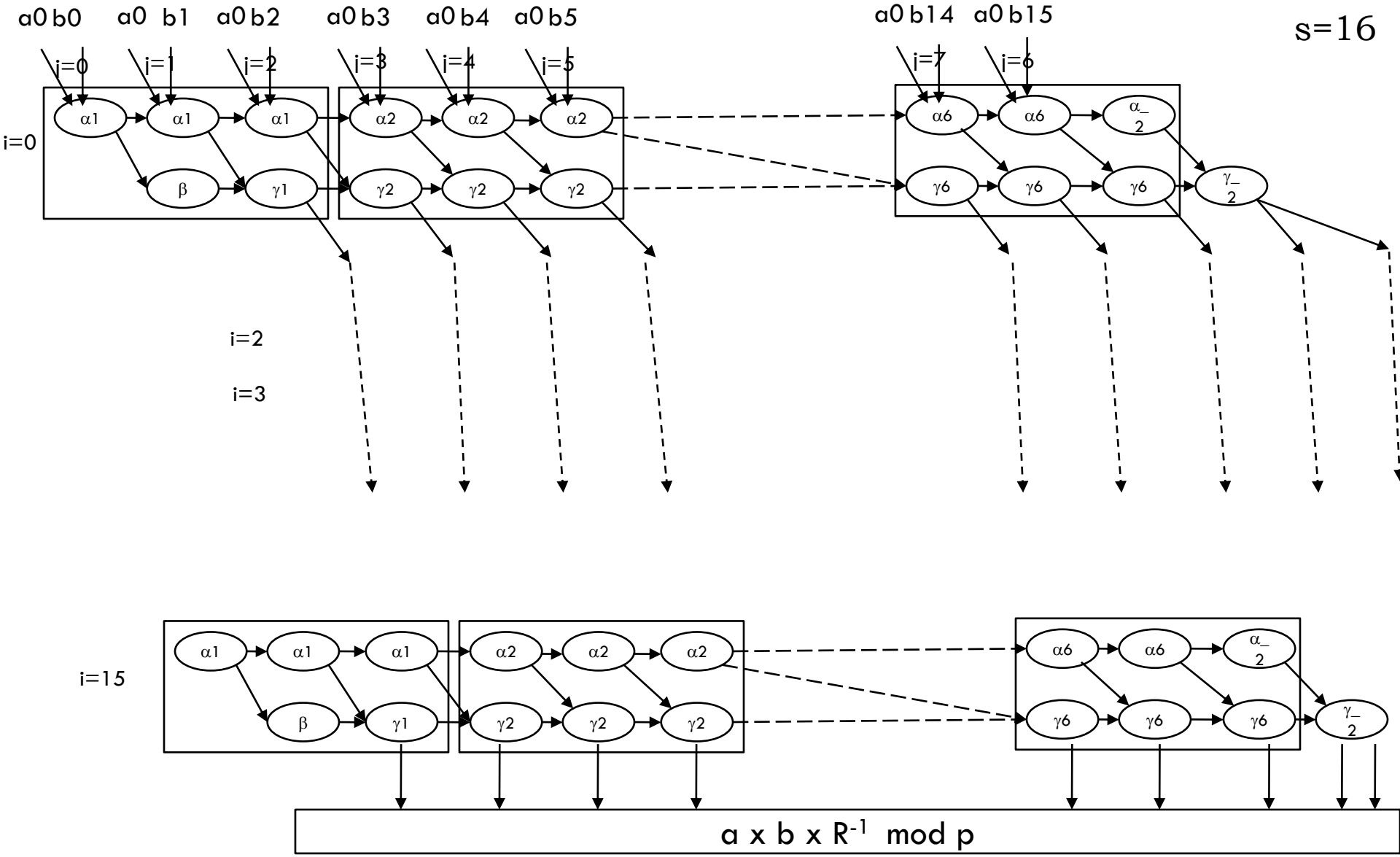
Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$
Output: $a \cdot b \cdot R^{-1}$ mod p

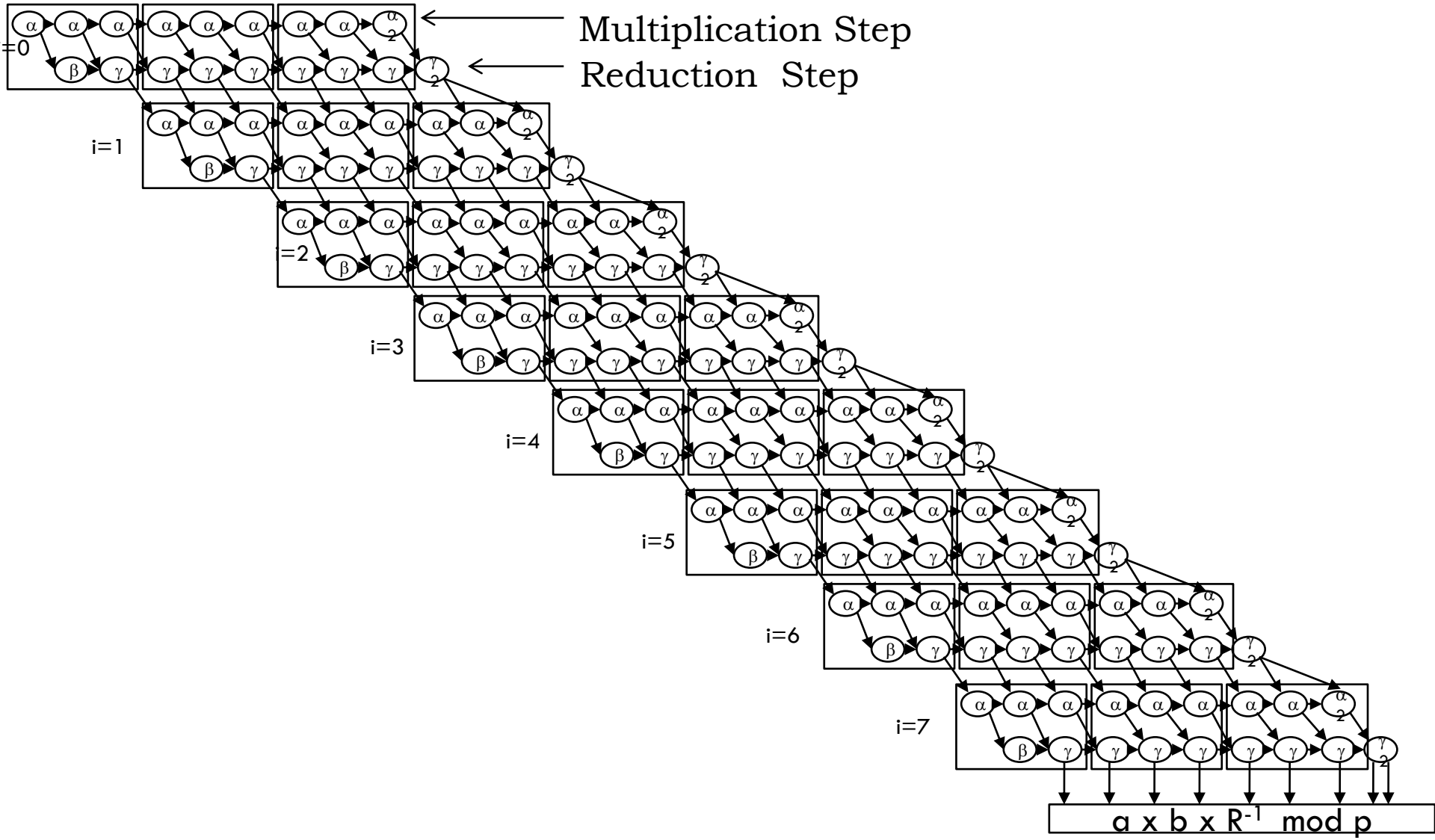
```
1  T ← Null;
2  for i ← 0 to s − 1 do
3      C ← 0;
4      for j ← 0 to s − 1 do                          ┐
5          (C, S) ← T[j] + a[i] · b[j] + C            │ α
6          T[j] ← S                                   ┘
7      (C, S) ← T[s] + C                              ┐
8      T[s] ← S                                       │ α_2
9      T[s + 1] ← C                                   ┘
10     C ← 0;                                         ┐
11     m ← T[0] · p′ mod 2^w                          │ β
12     (C, S) ← T[0] + m · p[0]                       ┘
13     for j ← 1 to s − 1 do                          ┐
14         (C, S) ← T[j] + m · p[j] + C               │ γ
15         T[j] ← S                                   ┘
16     (C, S) ← T[s] + C                              ┐
17     T[s − 1] ← S                                   │ γ_2
18     T[s] ← T[s + 1] + C                            ┘
19 return T;
```

Dans cette architecture on a aussi une intégration entre les différentes itérations qui bouclent sur i.
Dans notre cas on a 3 itérations de i qui peuvent s'exécuter en même temps.

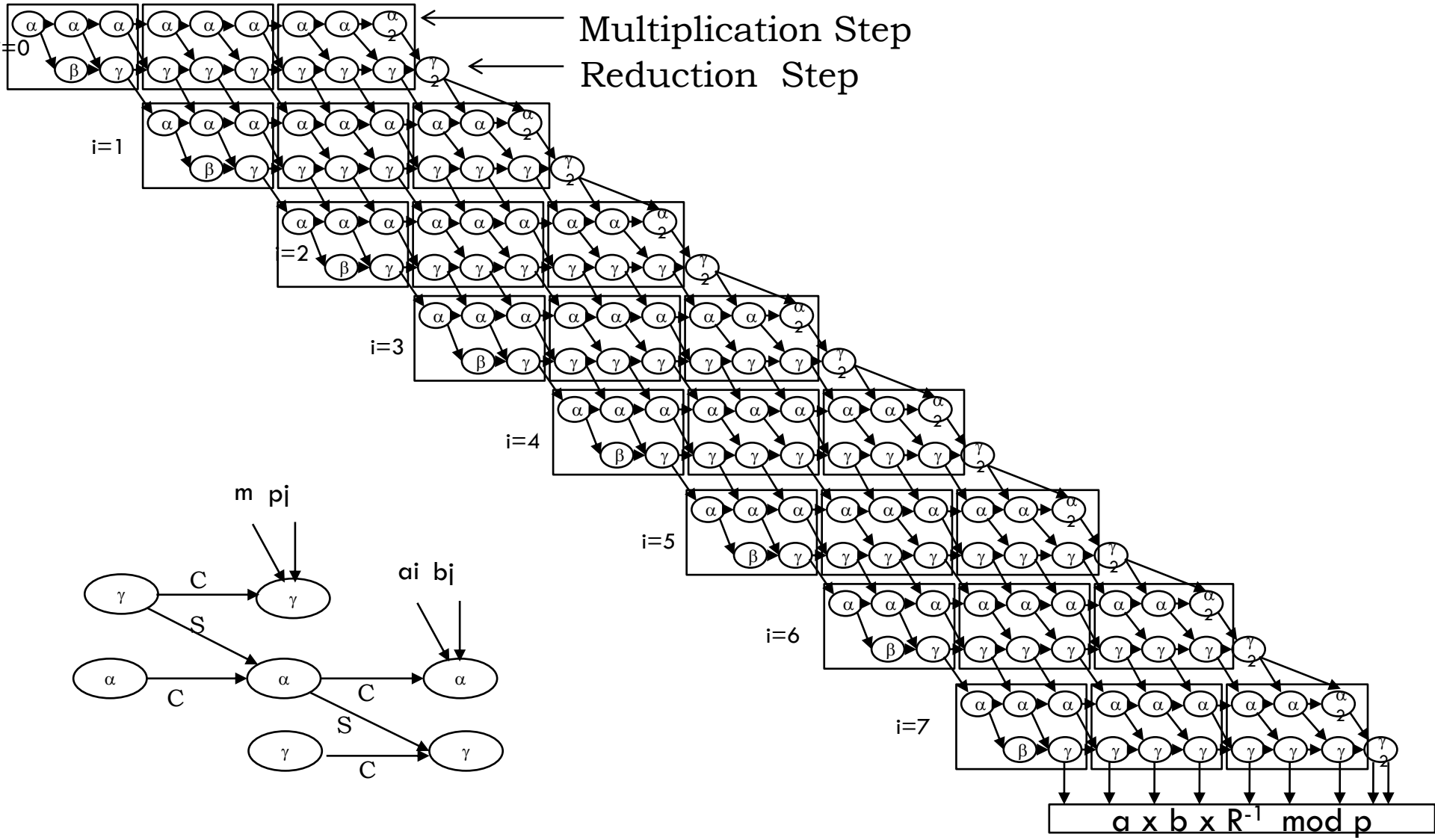

a0 b0
a0 b1
a0 b2
a0 b3
a0 b4
a0 b5
a0 b6
a0 b7
s=8
i=0
i=1
i=2
i=3
i=4
i=5
i=7
i=6
Multiplication Step
Reduction Step
α1
β
γ1
α2
γ2
α3
γ3
α_2
γ_2
a x b x R-1 mod p


a0 b0
a0 b1
a0 b2
a0 b3
a0 b4
a0 b5
a0 b14
a0 b15
s=16
i=0
i=1
i=2
i=3
i=4
i=5
i=7
i=6
α1
β
γ1
α2
γ2
α6
γ6
α_2
γ_2
i=0
i=2
i=3
i=15
a x b x R-1 mod p

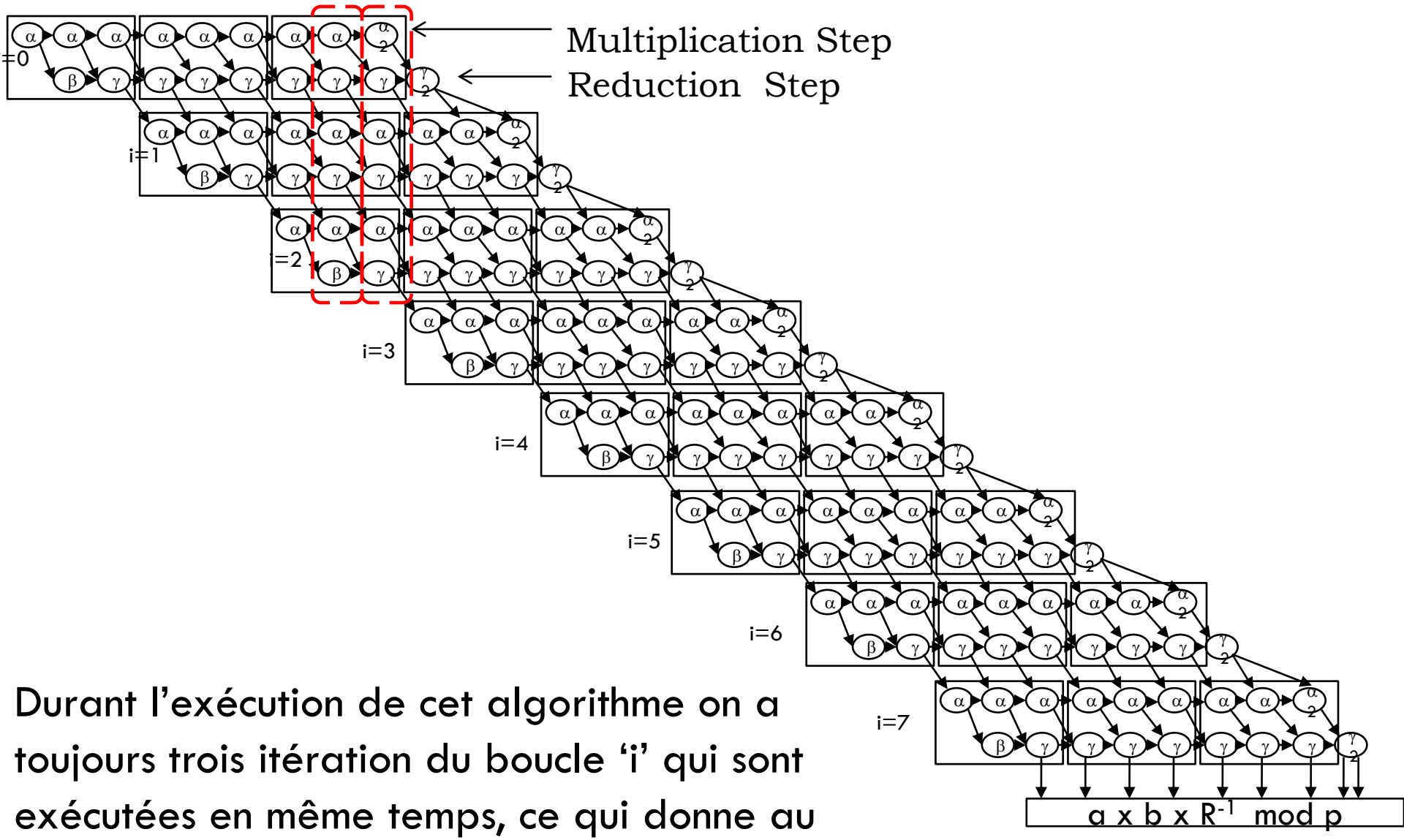
s=8


Multiplication Step
Reduction Step
i=0
i=1
i=2
i=3
i=4
i=5
i=6
i=7
a x b x R-1 mod p

s=8

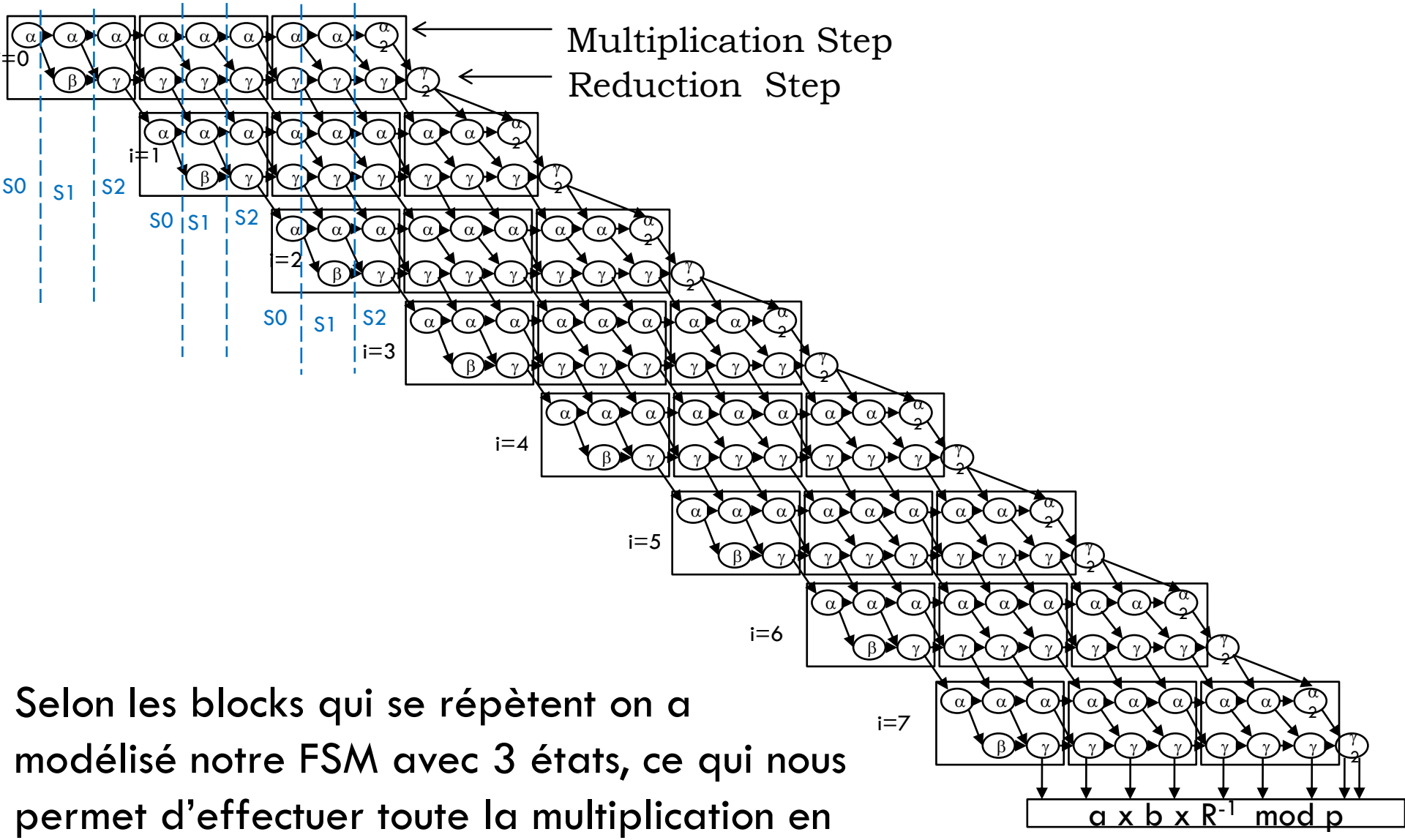

Multiplication Step
Reduction Step
i=0
i=1
i=2
i=3
i=4
i=5
i=6
i=7
m pj
ai bj
C
S
a x b x R-1 mod p

s=8



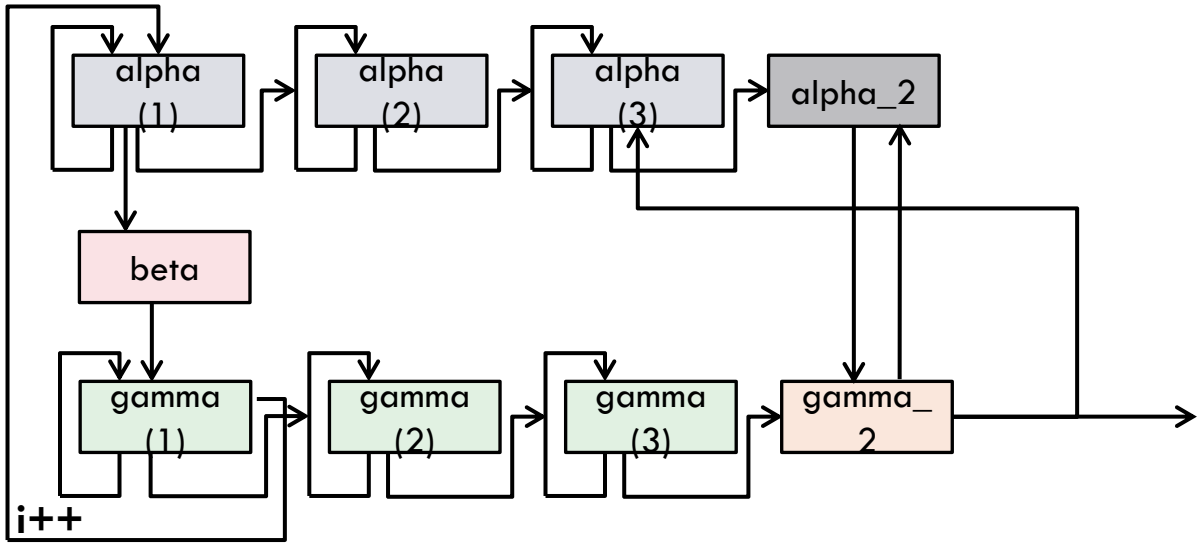
Durant l'exécution de cet algorithme on a toujours trois itération du boucle 'i' qui sont exécutées en même temps, ce qui donne au maximum trois alphas et trois gammas qui sont exécutés en parallèle.

s=8



Selon les blocks qui se répètent on a modélisé notre FSM avec 3 états, ce qui nous permet d'effectuer toute la multiplication en juste 33 cycles.

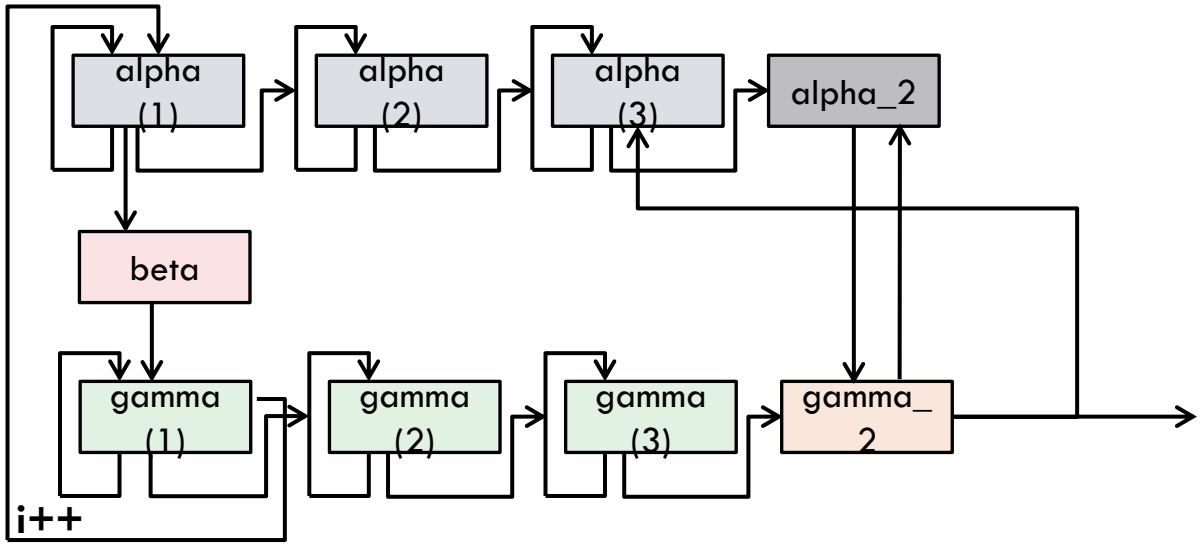
(8+3)*3=33



K=256, w=32, s=8

K=512, w=64, s=8

33 cycles d'horloge



K=256, w=32, s=8

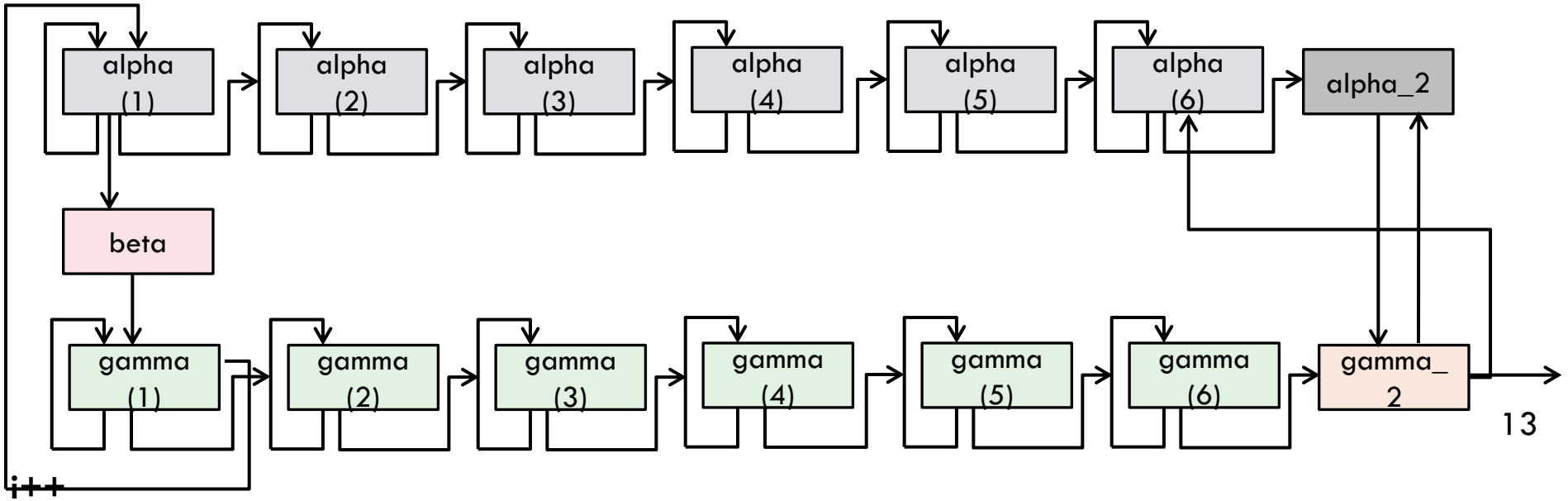
K=512, w=64, s=8

33 cycles d'horloge

K=256, w=16, s=16

K=512, w=32, s=16

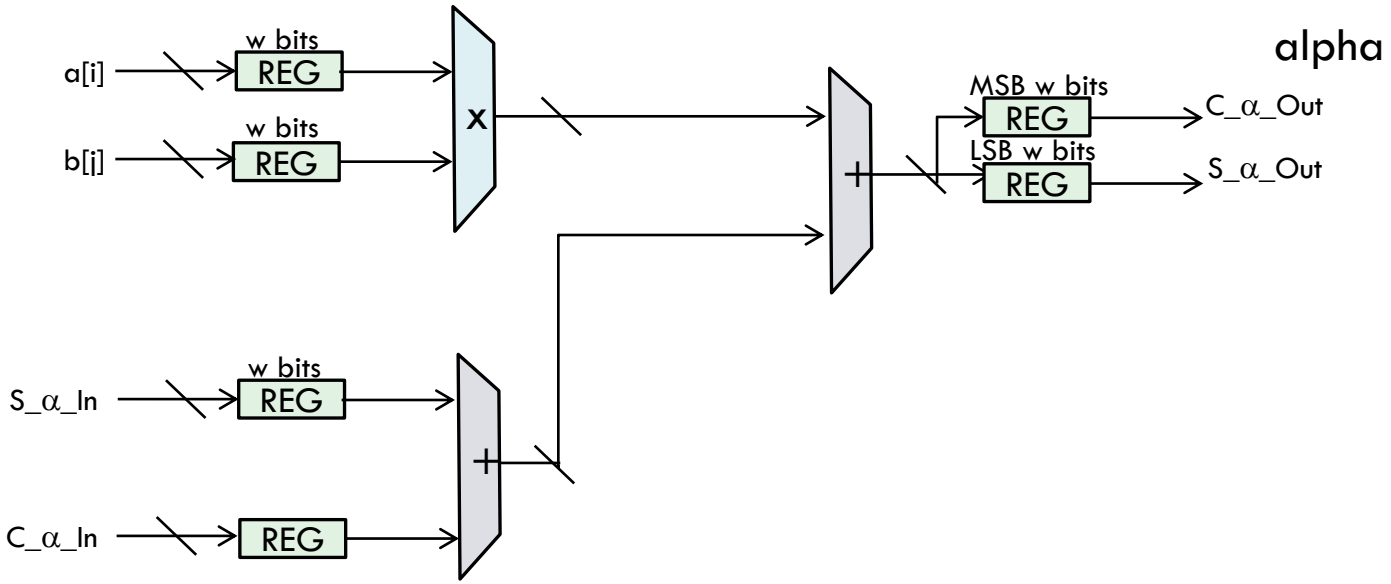
66 cycles d'horloge



Intérêt de chaque architecture

	S=8	S=16	S=32
K=256	32	16	8
K=512	64	32	16
K=1024	128	64	32
Nb cycle	33	66	132

L'intérêt de chaque architecture dépend de notre besoin niveau de sécurité, ressource, vitesse ainsi que la méthode utilisée.

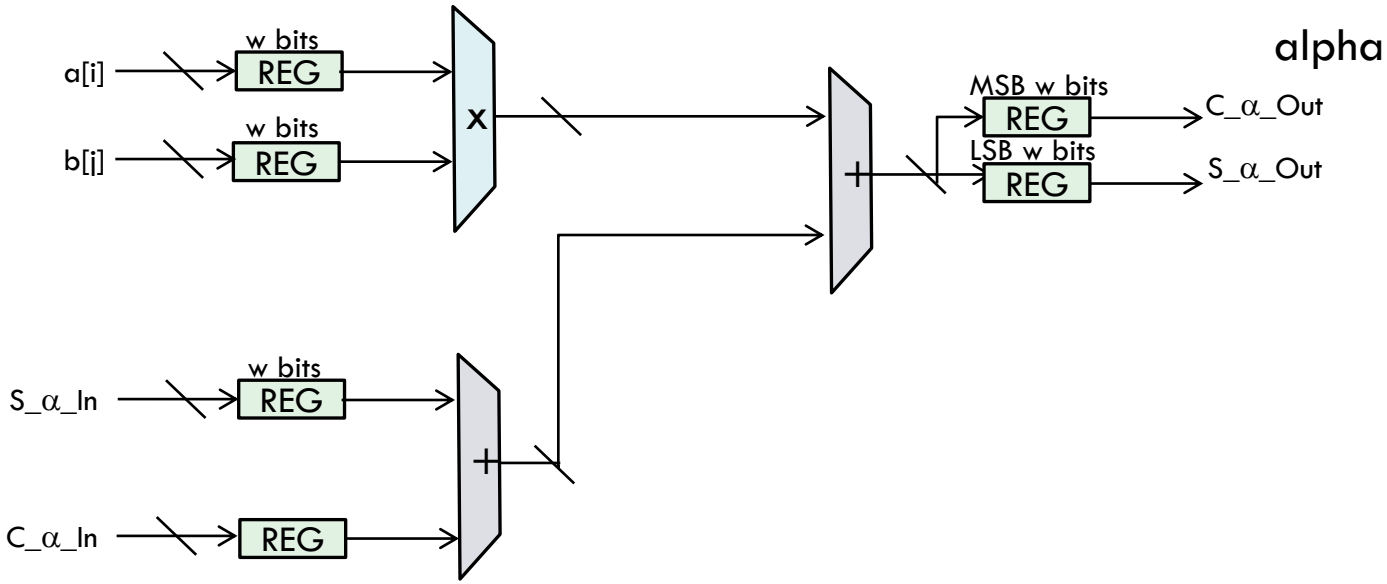


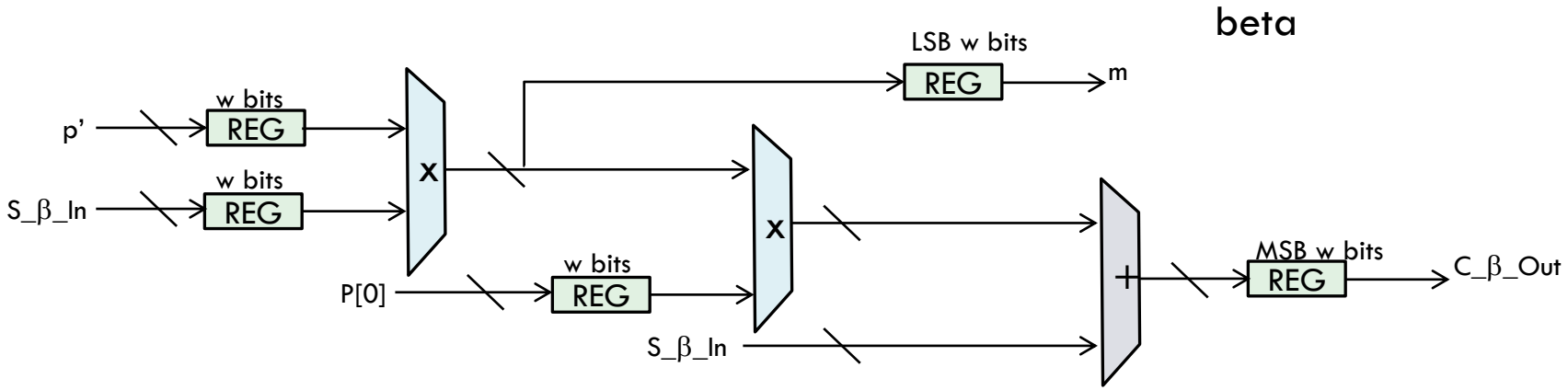
Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

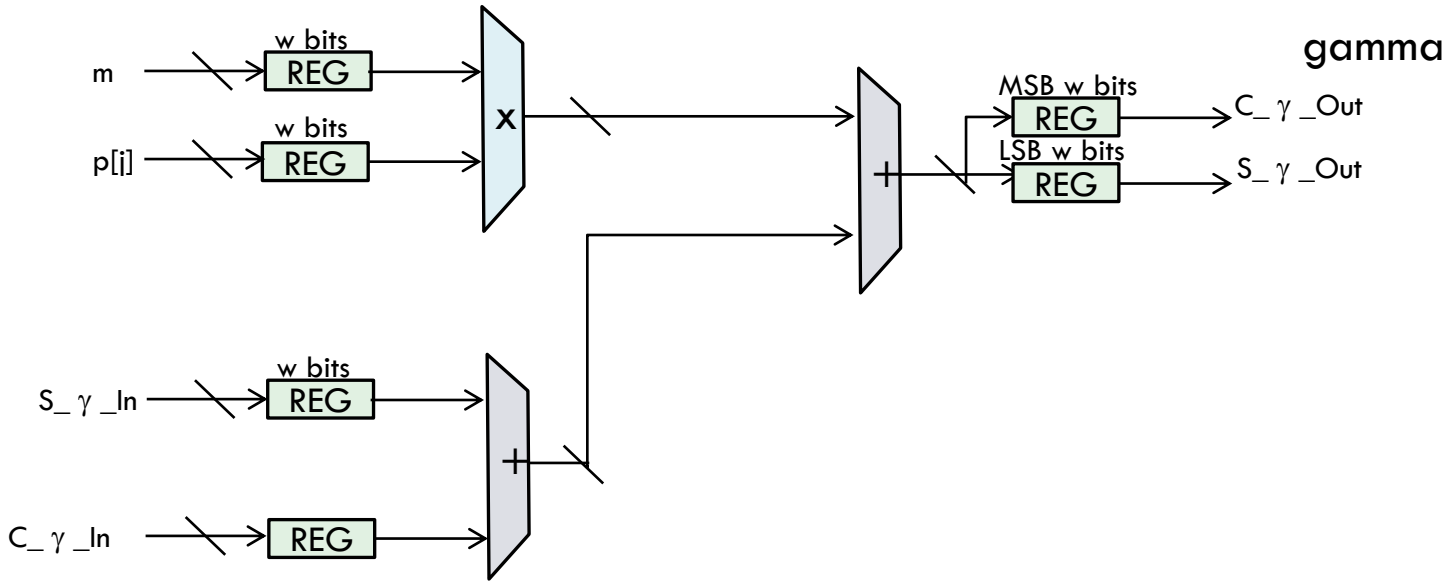
Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$
Output: $a \cdot b \cdot R^{-1}$ mod p

```
1  T ← Null;
2  for i ← 0 to s − 1 do
3      C ← 0;
4      for j ← 0 to s − 1 do                          ┐
5          (C, S) ← T[j] + a[i] · b[j] + C            │ α
6          T[j] ← S                                   ┘
7      (C, S) ← T[s] + C                              ┐
8      T[s] ← S                                       │ α_2
9      T[s + 1] ← C                                   ┘
10     C ← 0;
11     m ← T[0] · p' mod 2^w                          ┐ β
12     (C, S) ← T[0] + m · p[0]                       ┘
13     for j ← 1 to s − 1 do                          ┐
14         (C, S) ← T[j] + m · p[j] + C               │ γ
15         T[j] ← S                                   ┘
16     (C, S) ← T[s] + C                              ┐
17     T[s − 1] ← S                                   │ γ_2
18     T[s] ← T[s + 1] + C                            ┘
19 return T;
```

Les opérations arithmétiques de chaque cellule, sont conçue afin d'utiliser le maximum des DSPs.


alpha
a[i]
b[j]
w bits
REG
x
+
MSB w bits
LSB w bits
C_α_Out
S_α_Out
S_α_In
C_α_In

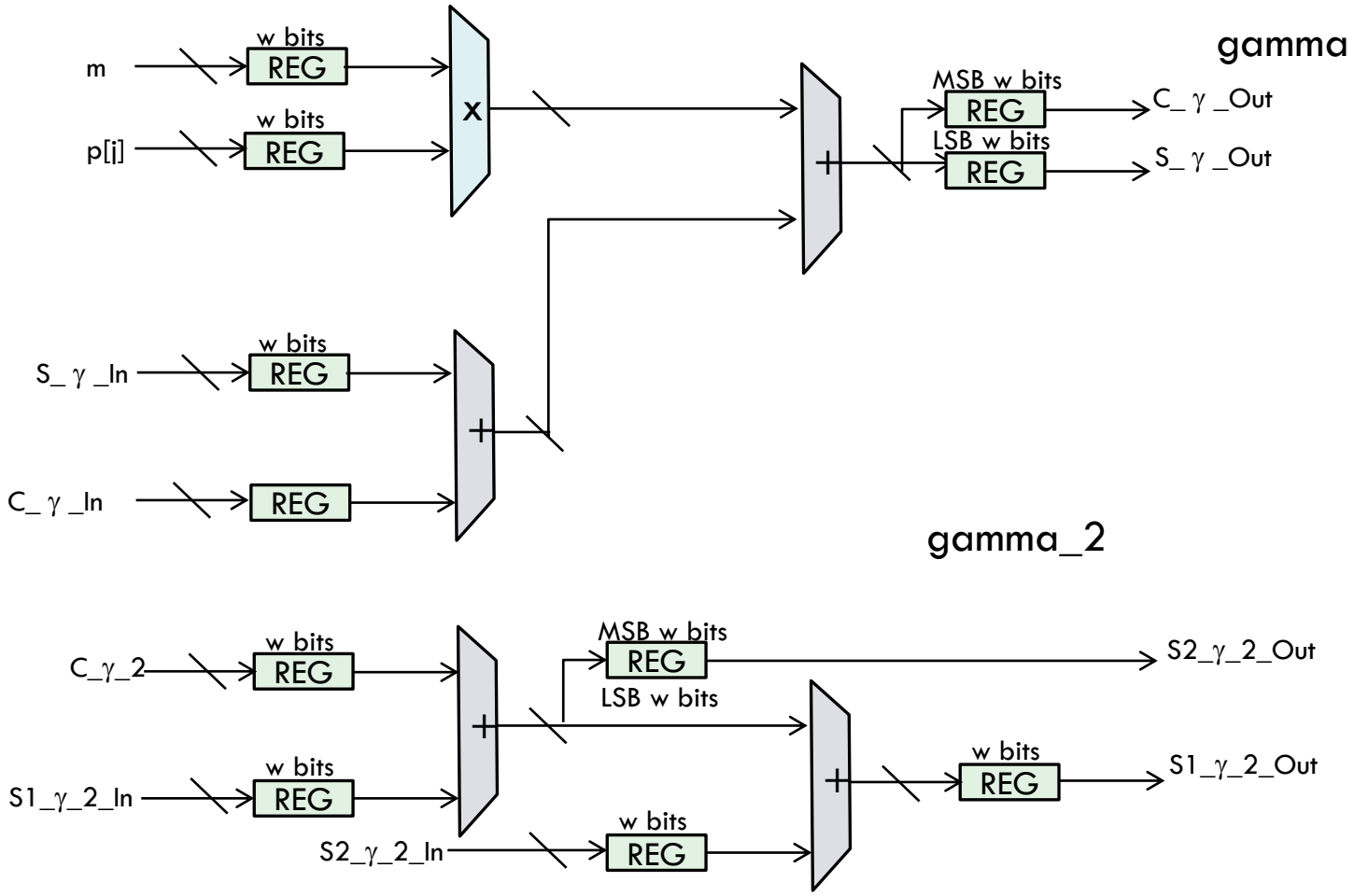

beta
p'
S_β_In
P[0]
w bits
REG
x
+
LSB w bits
m
MSB w bits
C_β_Out

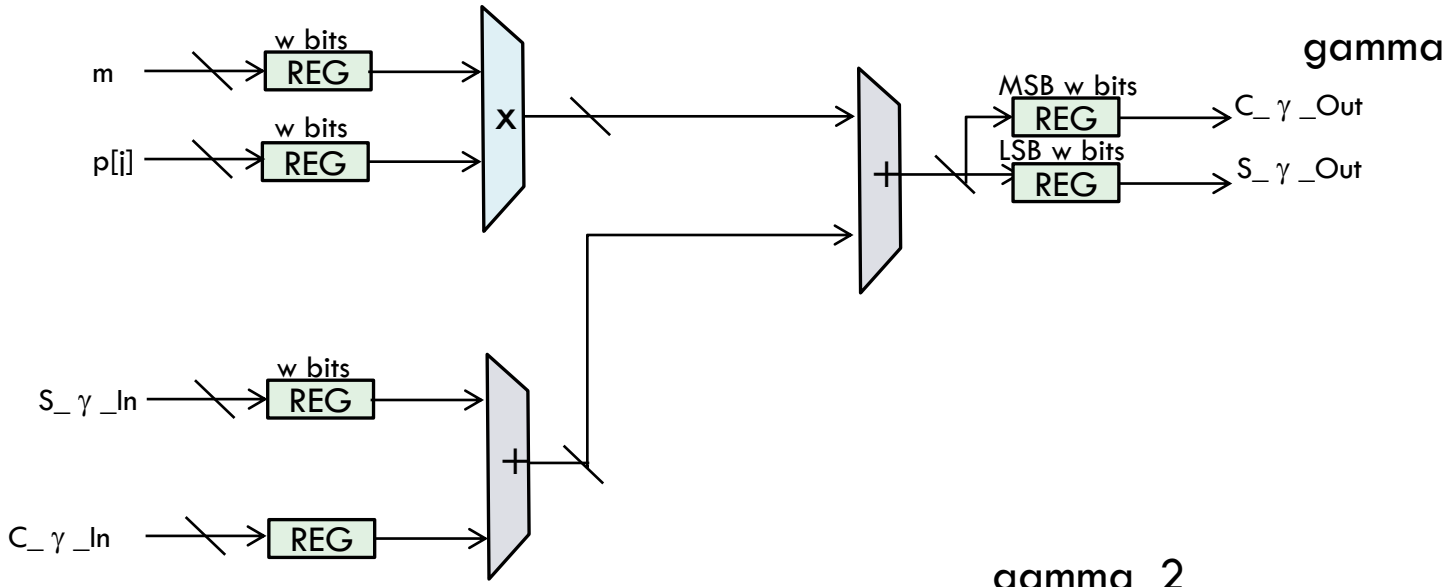


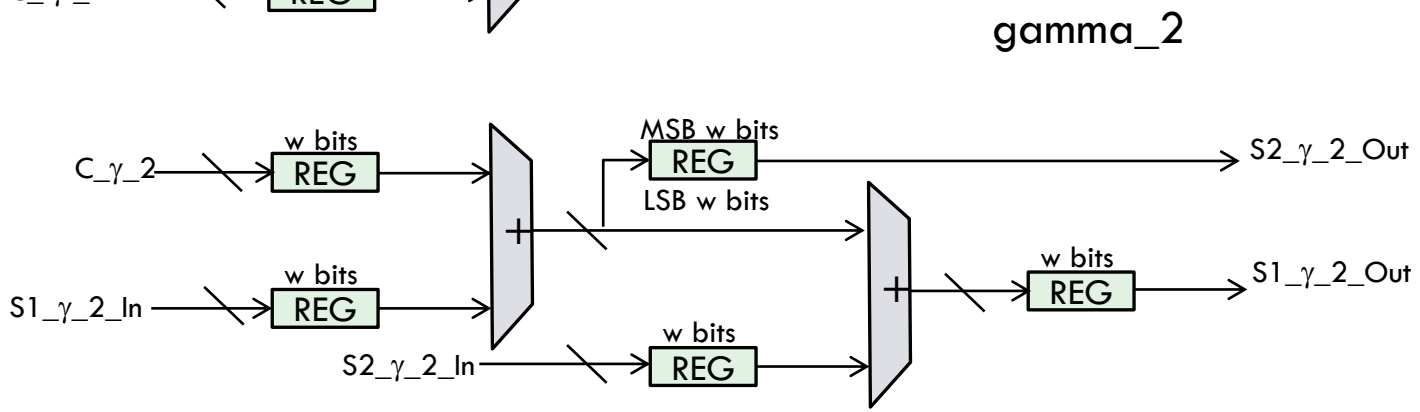
Algorithm 2: Coarsely Integrated Operand Scanning Method (CIOS)

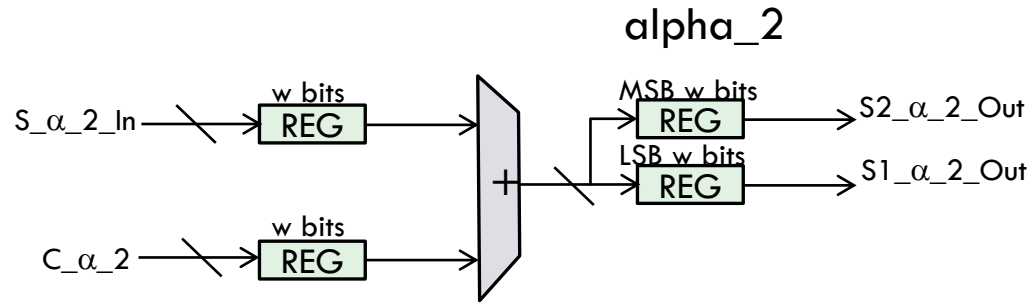
Input: $p < 2^n$, $p' = -p^{-1} mod\ 2^w$, w, s , $K = s \cdot w$:bit length, $R = 2^K$, $a, b < p$
Output: $a \cdot b \cdot R^{-1}$ mod p

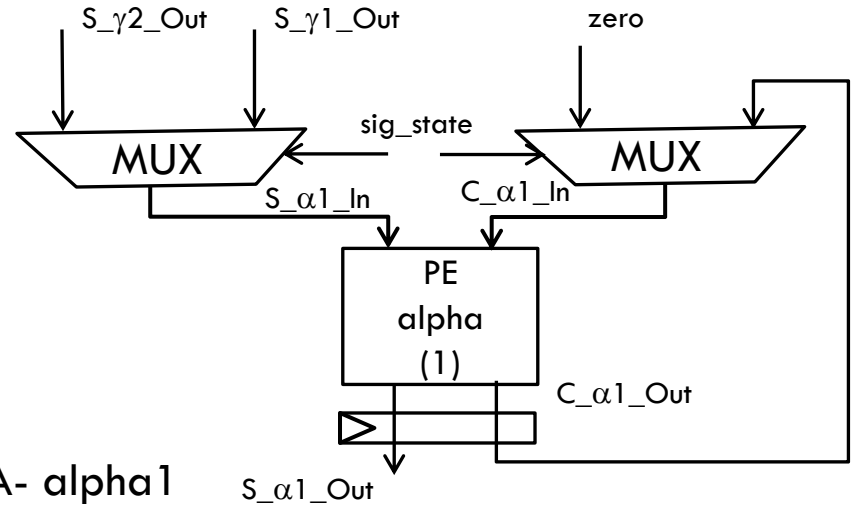
```
1  T ← Null;
2  for i ← 0 to s − 1 do
3      C ← 0;
4      for j ← 0 to s − 1 do                          ┐
5          (C, S) ← T[j] + a[i] · b[j] + C            │ α
6          T[j] ← S                                   ┘
7      (C, S) ← T[s] + C                              ┐
8      T[s] ← S                                       │ α_2
9      T[s + 1] ← C                                   ┘
10     C ← 0;                                         ┐
11     m ← T[0] · p' mod 2^w                          │ β
12     (C, S) ← T[0] + m · p[0]                       ┘
13     for j ← 1 to s − 1 do                          ┐
14         (C, S) ← T[j] + m · p[j] + C               │ γ
15         T[j] ← S                                   ┘
16     (C, S) ← T[s] + C                              ┐
17     T[s − 1] ← S                                   │ γ_2
18     T[s] ← T[s + 1] + C                            ┘
19 return T;
```


gamma
m
w bits
REG
p[i]
w bits
REG
x
MSB w bits
REG
C_ γ _Out
LSB w bits
REG
S_ γ _Out
S_ γ _In
w bits
REG
C_ γ _In
REG
gamma_2
C_γ_2
w bits
REG
S1_γ_2_In
w bits
REG
S2_γ_2_In
w bits
REG
MSB w bits
REG
LSB w bits
S2_γ_2_Out
w bits
REG
S1_γ_2_Out

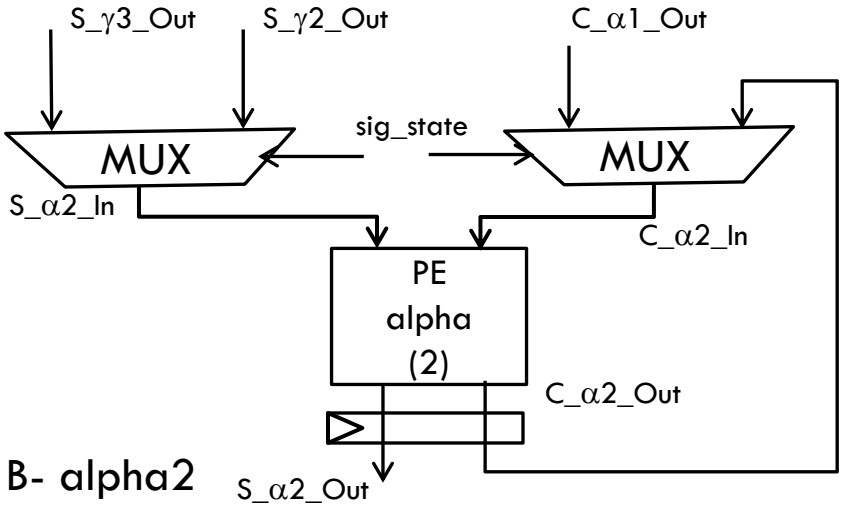

gamma
m
w bits
REG
p[j]
w bits
REG
x
MSB w bits
REG
C_ γ _Out
LSB w bits
REG
S_ γ _Out
S_ γ _In
w bits
REG
C_ γ _In
REG


gamma_2
C_γ_2
w bits
REG
S1_γ_2_In
w bits
REG
S2_γ_2_In
w bits
REG
MSB w bits
REG
LSB w bits
S2_γ_2_Out
w bits
REG
S1_γ_2_Out

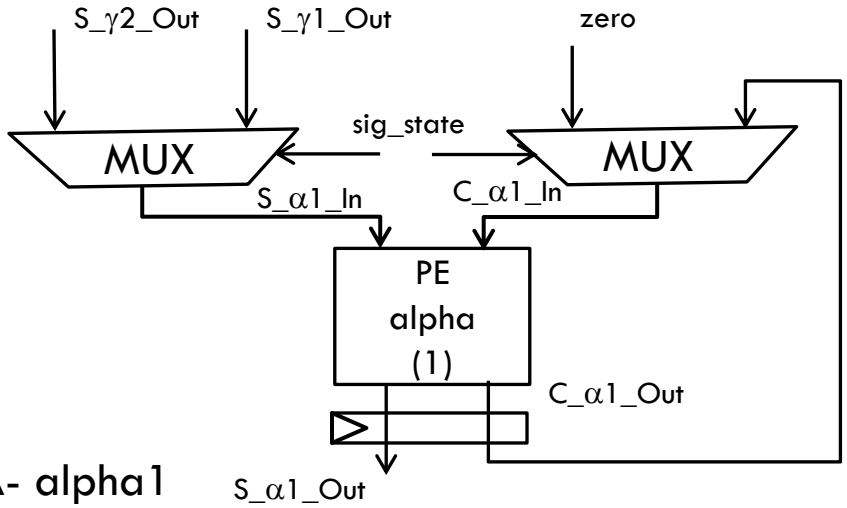

alpha_2
S_α_2_In
w bits
REG
C_α_2
w bits
REG
MSB w bits
REG
LSB w bits
REG
S2_α_2_Out
S1_α_2_Out


S_γ2_Out
S_γ1_Out
zero
sig_state
MUX
MUX
S_α1_In
C_α1_In
PE
alpha
(1)
C_α1_Out
S_α1_Out

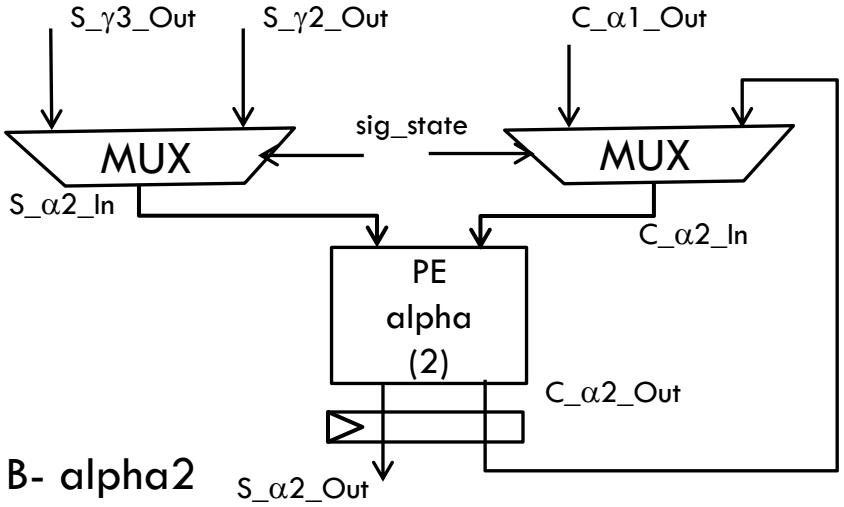
A- alpha1



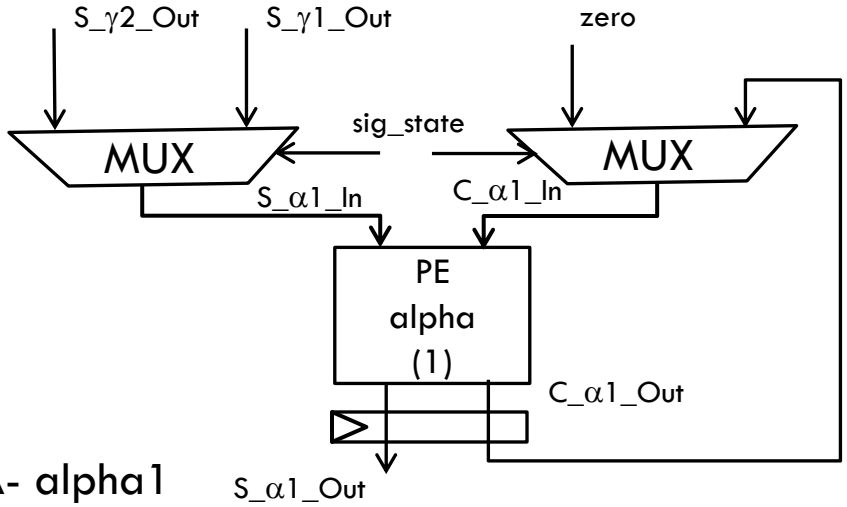
B- alpha2



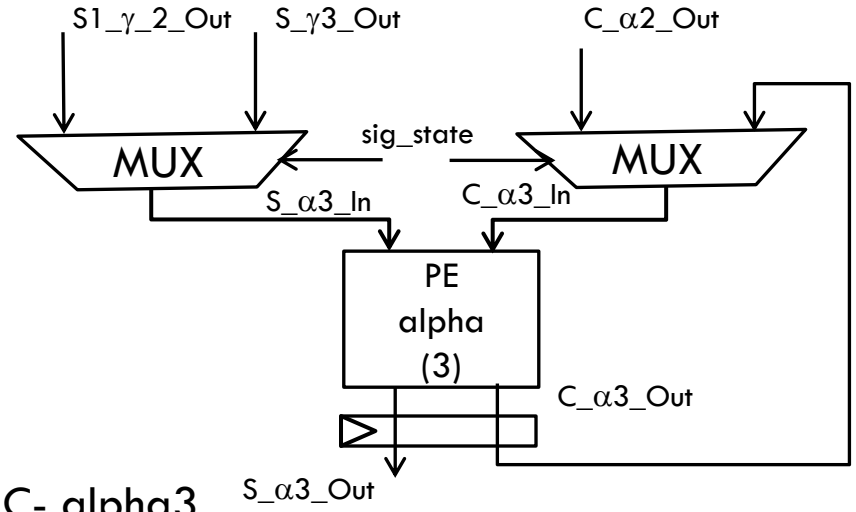
A- alpha1



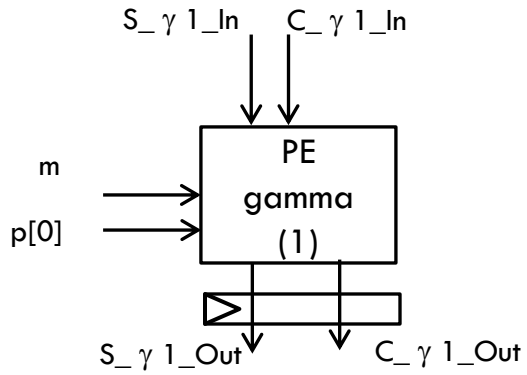
B- alpha2



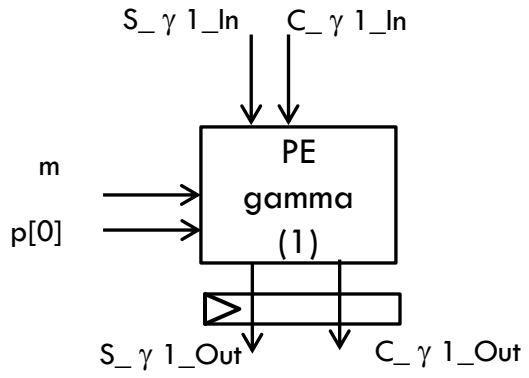
A- alpha1



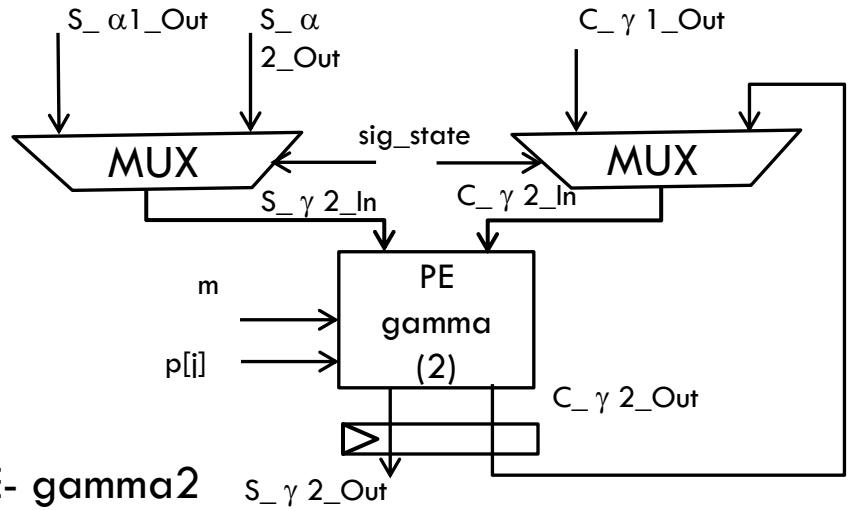
C- alpha3



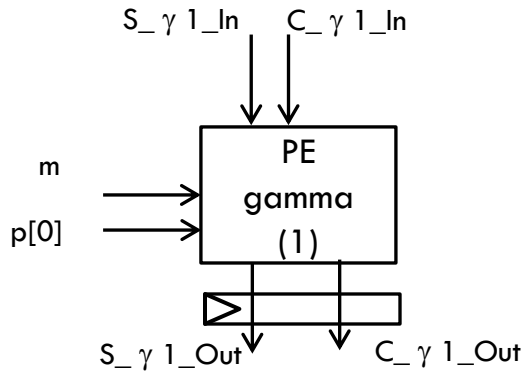
D- gamma1



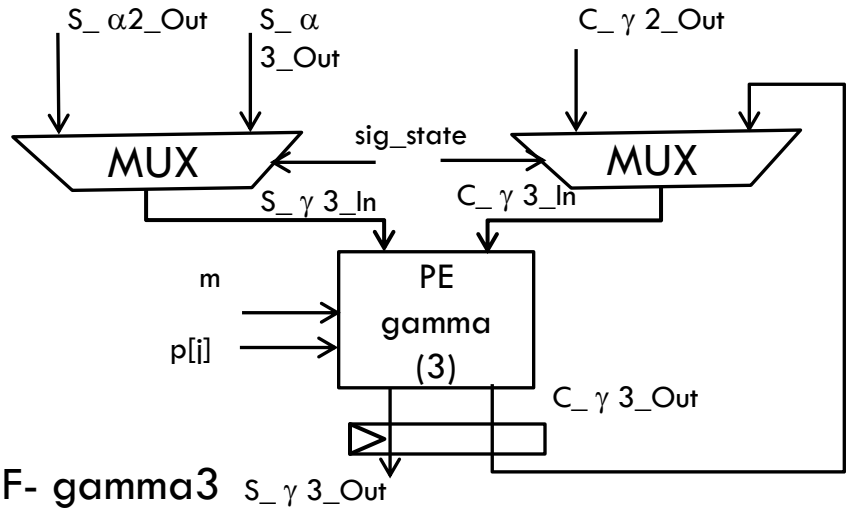
D- gamma1



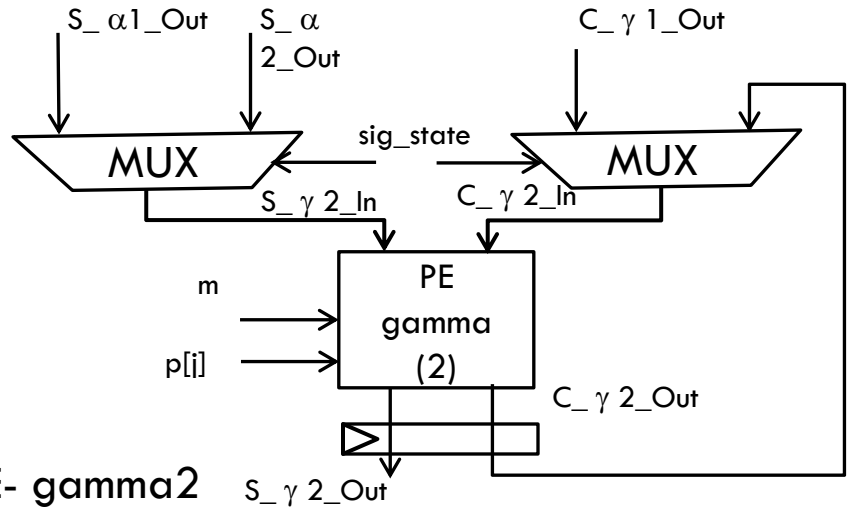
E- gamma2



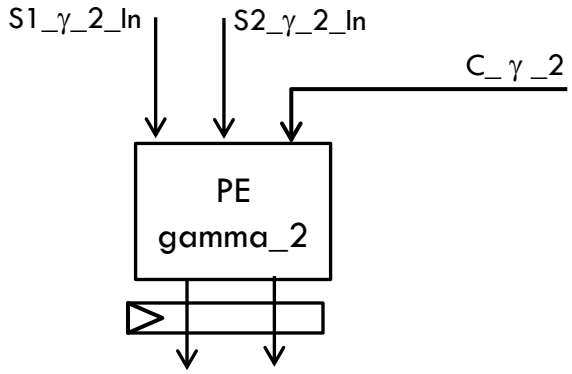
D- gamma1



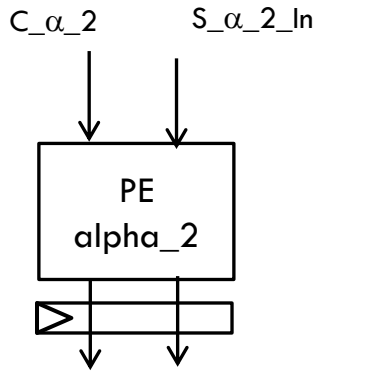
F- gamma3



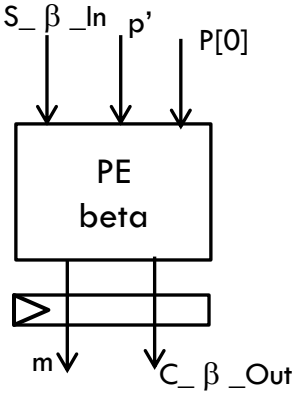
E- gamma2



H- gamma_2



G- alpha_2



I- beta

Plan

1. Introduction
2. Multiplication Motgomery (CIOS)
3. Architecture
4. **Résultat**
5. Conclusion et Perspectives

Résultat

Nexys 4	DSP	Fréquence	N° cycle
MMM(s=8)	31	105.275	33
Alpha	4	111.178	1
Beta	4	388.350	1
Alpha_2	1	459.918	1
Gamma_2	2	442.811	1

Résultat

Nexys 4	DSP	LUTs	Reg	Slice occupé	Fréquence	N° cycle
MMM S=8/k=256	31	809	870	352	105.275	33
MMM S=8/k=512	87	2650	1614	878	64.825	33
MMM S=16/k=256	33	846	1123	402	145.892	66
MMM S=16/k=512	57	1789	2164	798	105.594	66

Plan

1. Introduction
2. Multiplication Motgomery (CIOS)
3. Architecture
4. Résultat
5. Conclusion et Perspectives

conclusion et perspectives

Conclusion

Nous avons Implémenté la multiplication de Montgomery avec architecture systolique dans un nombre de cycles d'horloge fixe.

On a fait notre design afin d'utiliser le maximum des DSPs sur carte FPGA

Perspectives

Faire une implémentation mixte soft/hard pour la multiplication scalaire en utilisant cet algorithme de multiplication.

MERCI POUR VOTRE ATTENTION