Parallel generation of ℓ–sequences

Cédric Lauradoux
Princeton University
USA
currently at UCL/INGI, Belgium

Andrea Röck
INRIA Paris-Rocquencourt
France

SETA, September 17, 2008

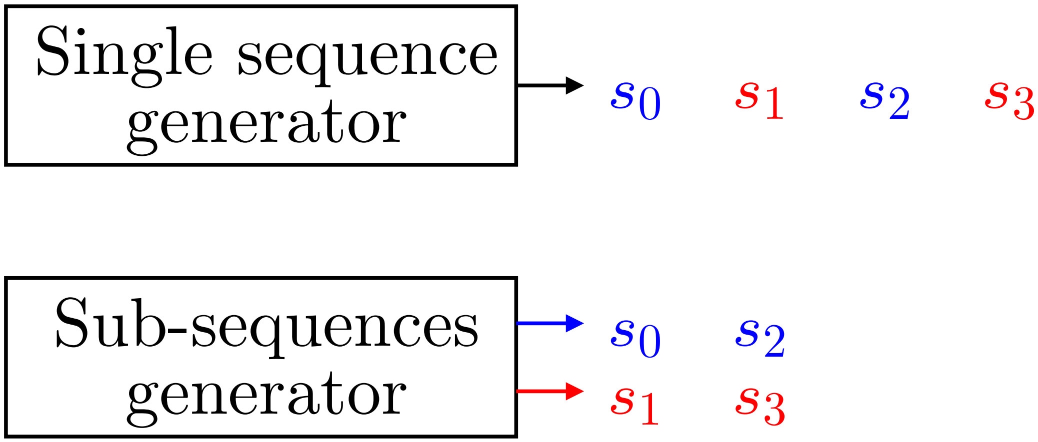
Outline

- **Introduction**
- **Parallel generation of m-sequences (LFSRs)**
  - Synthesis of sub-sequences
  - Multiple steps LFSR
- **Parallel generation of ℓ-sequences (FCSRs)**
  - Synthesis of sub-sequences
  - Multiple steps FCSR
- **Conclusion**

Part 1

Introduction

Sub-sequences generator



- Goal: **parallelism**
  - better throughput
  - reduced power consumption

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.
- $S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots)$: Decimated sequence, with $0 \leq i \leq d-1$.
  - $S_d^0 = (s_0, s_d, \cdots)$, $\cdots$, $S_d^{d-1} = (s_{d-1}, s_{2d-1}, \cdots)$

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.
- $S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots)$: Decimated sequence, with $0 \leq i \leq d-1$.
  - $S_d^0 = (s_0, s_d, \cdots)$, $\cdots$, $S_d^{d-1} = (s_{d-1}, s_{2d-1}, \cdots)$
- x_j: Memory cell.

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.
- $S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots)$: Decimated sequence, with $0 \leq i \leq d-1$.
  - $S_d^0 = (s_0, s_d, \cdots), \cdots, S_d^{d-1} = (s_{d-1}, s_{2d-1}, \cdots)$
- x_j: Memory cell.
- $(x_j)_t$: Content of the cell x_j.

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.
- $S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots)$: Decimated sequence, with $0 \leq i \leq d-1$.
  - $S_d^0 = (s_0, s_d, \cdots), \cdots, S_d^{d-1} = (s_{d-1}, s_{2d-1}, \cdots)$
- x_j: Memory cell.
- $(x_j)_t$: Content of the cell x_j.
- X_t: Entire internal state of the automaton.

Notations

- $S = (s_0, s_1, s_2, \cdots)$: Binary sequence with period T.
- $S_d^i = (s_i, s_{i+d}, s_{i+2d}, \cdots)$: Decimated sequence, with $0 \leq i \leq d-1$.
  - $S_d^0 = (s_0, s_d, \cdots), \cdots, S_d^{d-1} = (s_{d-1}, s_{2d-1}, \cdots)$
- x_j: Memory cell.
- $(x_j)_t$: Content of the cell x_j.
- X_t: Entire internal state of the automaton.
- $next^d(x_j)$: Cell connected to the output of x_j.

LFSRs

- Automaton with linear update function.

LFSRs

- Automaton with linear update function.
- Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ be the power series of $S = (s_0, s_1, s_2, \ldots)$. There exists two polynomials $p(x), q(x)$:

$$s(x) = \frac{p(x)}{q(x)}.$$

LFSRs

- Automaton with linear update function.
- Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ be the power series of $S = (s_0, s_1, s_2, \ldots)$. There exists two polynomials $p(x), q(x)$:

$$s(x) = \frac{p(x)}{q(x)}.$$

- $q(x)$: Connection polynomial of degree m.

LFSRs

- Automaton with linear update function.
- Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ be the power series of $S = (s_0, s_1, s_2, \ldots)$. There exists two polynomials $p(x), q(x)$:

$$s(x) = \frac{p(x)}{q(x)}.$$

- $q(x)$: Connection polynomial of degree m.
- $Q(x) = x^m q(1/x)$: Characteristic polynomial.

LFSRs

- Automaton with linear update function.
- Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ be the power series of $S = (s_0, s_1, s_2, \ldots)$. There exists two polynomials $p(x), q(x)$:

$$s(x) = \frac{p(x)}{q(x)}.$$

- $q(x)$: Connection polynomial of degree m.
- $Q(x) = x^m q(1/x)$: Characteristic polynomial.
- m–sequence: S has maximal period of $2^m - 1$. (*iff* $q(x)$ is a primitive polynomial)

LFSRs

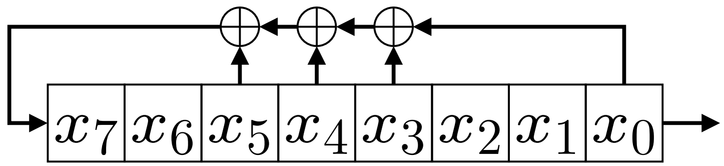
- Automaton with linear update function.
- Let $s(x) = \sum_{i=0}^{\infty} s_i x^i$ be the power series of $S = (s_0, s_1, s_2, \dots)$. There exists two polynomials $p(x), q(x)$:

$$s(x) = \frac{p(x)}{q(x)}.$$

- $q(x)$: Connection polynomial of degree m.
- $Q(x) = x^m q(1/x)$: Characteristic polynomial.
- m–sequence: S has maximal period of $2^m - 1$. (*iff* $q(x)$ is a primitive polynomial)
- Linear complexity: Size of smallest LFSR which generates S.

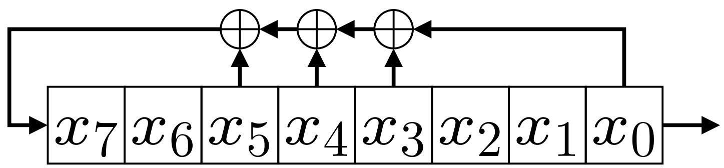
Fibonacci/Galois LFSRs

Fibonacci setup.

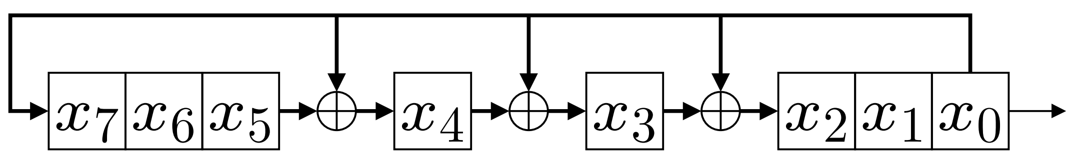


Fibonacci/Galois LFSRs

Fibonacci setup.



Galois setup.



FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.

FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.

FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.
  - Additional memory to store the carry.

FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.
  - Additional memory to store the carry.

- S is the 2–adic expansion of the rational number: $\frac{h}{q} \leq 0$.

FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.
  - Additional memory to store the carry.

- S is the 2–adic expansion of the rational number: $\dfrac{h}{q} \leq 0$.

- Connection integer q: Determines the feedback positions.

FCSRs
[Klapper Goresky 93]

- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.
  - Additional memory to store the carry.

- S is the 2–adic expansion of the rational number: $\frac{h}{q} \leq 0$.
- Connection integer q: Determines the feedback positions.
- ℓ–*sequences*: S has maximal period $\varphi(q)$.
  (*iff* q is odd and a prime power and $ord_q(2) = \varphi(q)$.)

FCSRs
[Klapper Goresky 93]

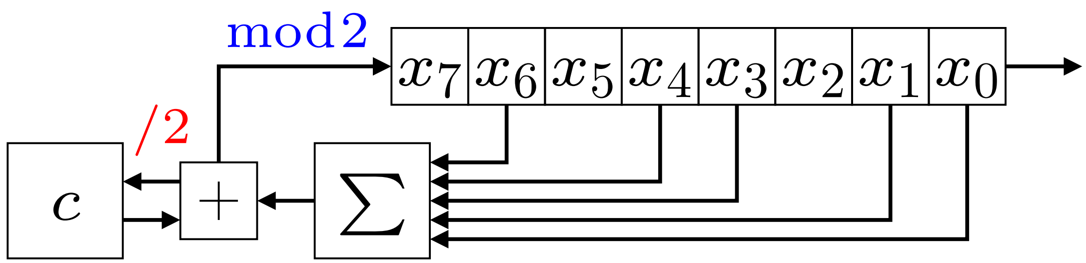
- Instead of XOR, FCSRs use additions with carry.
  - Non-linear update function.
  - Additional memory to store the carry.

- S is the 2–adic expansion of the rational number: $\frac{h}{q} \leq 0$.
- Connection integer q: Determines the feedback positions.
- *ℓ–sequences*: S has maximal period $\varphi(q)$.
  (*iff* q is odd and a prime power and $ord_q(2) = \varphi(q)$.)
- 2–adic complexity: size of the smallest FCSR which produces S.

Fibonacci/Galois FCSRs

[Klapper Goresky 02]

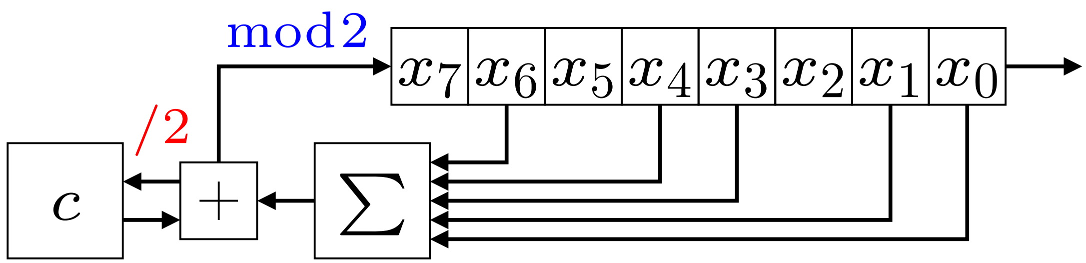
Fibonacci setup.



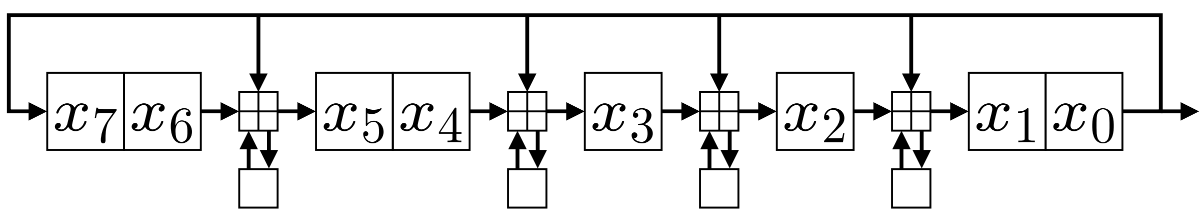
Fibonacci/Galois FCSRs

[Klapper Goresky 02]

Fibonacci setup.



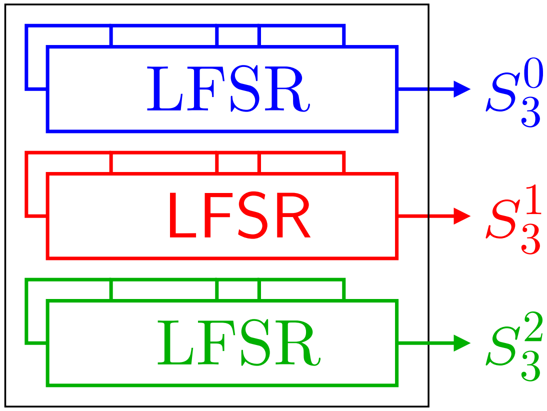
Galois setup.



Part 2

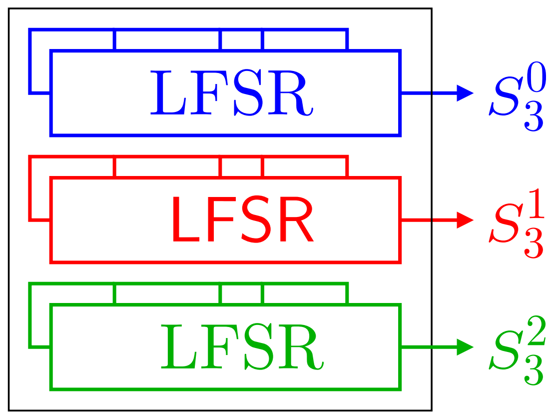
Parallel generation of m-sequences (LFSRs)

Synthesis of Sub-sequences (1)



- Use Berlekamp-Massey algorithm to find the smallest LFSR for each sub-sequence.

Synthesis of Sub-sequences (1)



- Use Berlekamp-Massey algorithm to find the smallest LFSR for each sub-sequence.
- All sub-sequences are generated using d LFSRs defined by $Q^\star(x)$ but initialized with different values.

Synthesis of Sub-sequences (2)

Theorem [Zierler 59]: Let S be produced by an LFSR whose characteristic polynomial $Q(x)$ is irreducible in $\mathbf{F}_2$ of degree m. Let α be a root of $Q(x)$ and let T be the period of S. For $0 \leq i < d$, S_d^i can be generated by an LFSR with the following properties:

- The minimum polynomial of α^d in $\mathbf{F}_{2^m}$ is the characteristic polynomial $Q^\star(x)$ of the new LFSR with:
- Period $T^\star = \frac{T}{gcd(d,T)}$.
- Degree $m^\star$ is the multiplicative order of 2 in $\mathbf{Z}_{T^\star}$.

Multiple steps LFSR
[Lempel Eastman 71]

- Clock d times the register in one cycle.

Multiple steps LFSR
[Lempel Eastman 71]

- Clock d times the register in one cycle.

- Equivalent to partition the register into d sub-registers

$$x_i x_{i+d} \cdots x_{i+kd}$$

such that $0 \leq i < d$ and $i + kd < m$.

Multiple steps LFSR
[Lempel Eastman 71]

- Clock d times the register in one cycle.

- Equivalent to partition the register into d sub-registers

$$x_i x_{i+d} \cdots x_{i+kd}$$

such that $0 \leq i < d$ and $i + kd < m$.

- Duplication of the feedback:

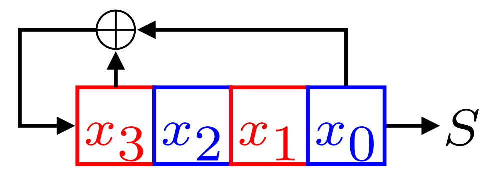
The sub-registers are linearly interconnected.

Fibonacci LFSR

$$next^1(x_0) = x_3$$
$$next^1(x_i) = x_{i-1} \text{ if } i \neq 0$$

$$(x_3)_{t+1} = (x_3)_t \oplus (x_0)_t$$
$$(x_i)_{t+1} = (x_{i-1})_t \text{ if } i \neq 3$$

1-decimation

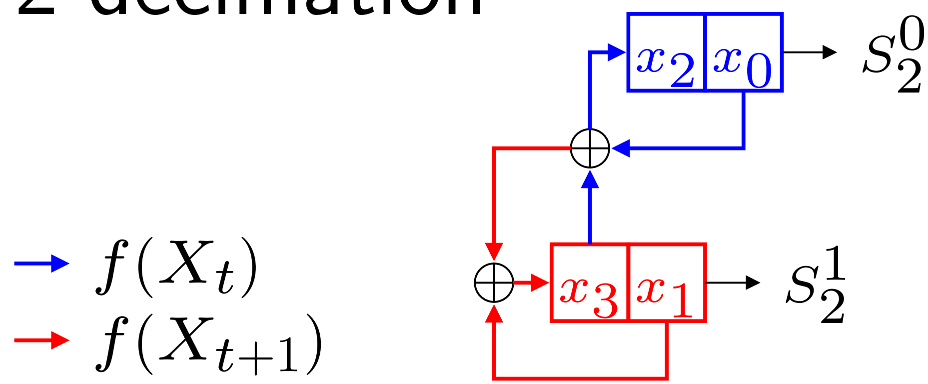


$$\Downarrow$$

$$next^2(x_0) = x_2$$
$$next^2(x_1) = x_3$$
$$next^2(x_i) = x_{i-2} \text{ if } i > 1$$

$$(x_i)_{t+2} = (x_{i-2})_t \text{ if } i < 2$$
$$(x_2)_{t+2} = (x_3)_t \oplus (x_0)_t$$
$$(x_3)_{t+2} = \underbrace{(x_3)_t \oplus (x_0)_t}_{(x_3)_{t+1}} \oplus (x_1)_t$$

2-decimation



Comparison

- **Synthesis of Sub-sequences:**
  - Larger memory size: $d \times m^\star$
  - More logic gates: $d \times wt(Q^\star)$

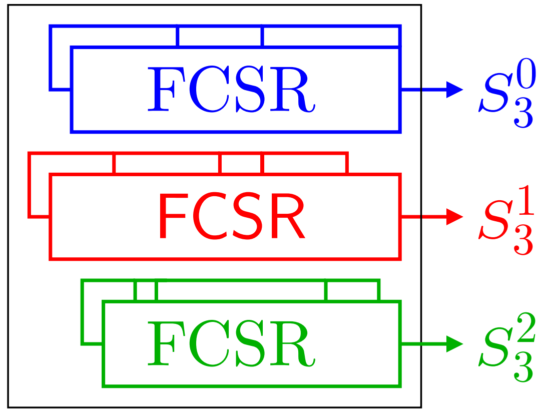
Comparison

- **Synthesis of Sub-sequences:**
  - Larger memory size: $d \times m^\star$
  - More logic gates: $d \times wt(Q^\star)$
- **Multiple steps LFSR:**
  - Same memory size: m
  - More logic gates: $d \times wt(Q)$

Part 3

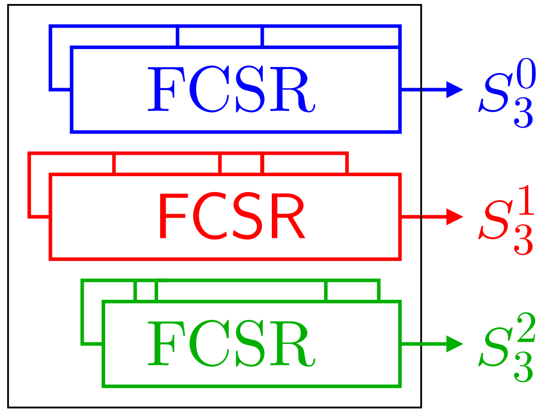
Parallel generation of ℓ-sequences (FCSRs)

Synthesis of Sub-sequences (1)



- We use an algorithm based on Euclid's algorithm **[Arnault Berger Necer 04]** or on lattice approximation **[Klapper Goresky 97]** to find the smallest FCSR for each sub-sequence.

Synthesis of Sub-sequences (1)



- We use an algorithm based on Euclid's algorithm **[Arnault Berger Necer 04]** or on lattice approximation **[Klapper Goresky 97]** to find the smallest FCSR for each sub-sequence.

- The sub-sequences do **not** have the same q.

Synthesis of Sub-sequences (2)

- A given S_d^i has period $T^\star$ and minimal connection integer $q^\star$.

Synthesis of Sub-sequences (2)

- A given S_d^i has period $T^\star$ and minimal connection integer $q^\star$.
- **Period:** (True for all periodic sequences)
  - $T^\star \,\Big|\, \frac{T}{\gcd(T,d)}$,

Synthesis of Sub-sequences (2)

- A given S_d^i has period $T^\star$ and minimal connection integer $q^\star$.
- **Period:** (True for all periodic sequences)
  - $T^\star \,\middle|\, \frac{T}{\gcd(T,d)}$,
  - If $\gcd(T, d) = 1$ then $T^\star = T$.

Synthesis of Sub-sequences (2)

- A given S_d^i has period $T^\star$ and minimal connection integer $q^\star$.
- **Period:** (True for all periodic sequences)
  - $T^\star \left| \frac{T}{\gcd(T,d)} \right.$,
  - If $\gcd(T,d) = 1$ then $T^\star = T$.
- If $\gcd(T,d) > 1$: $T^\star$ might depend on i!
  E.g. for $S = -1/19$ and $d = 3$: $T/gcd(T,d) = 6$.
  - S_3^0: The period $T^\star = 2$.
  - S_3^1: The period $T^\star = 6$.

Princeton University INRIA

Synthesis of Sub-sequences (3)

- **2-adic complexity [Goresky Klapper 97]:**
  - General case: $q^\star | 2^{T^\star} - 1$.

Synthesis of Sub-sequences (3)

- **2-adic complexity [Goresky Klapper 97]:**
  - General case: $q^{\star} | 2^{T^{\star}} - 1$.
  - $\gcd(T, d) = 1$: $q^{\star} | 2^{T/2} + 1$.

Synthesis of Sub-sequences (3)

- **2-adic complexity [Goresky Klapper 97]:**
  - General case: $q^\star | 2^{T^\star} - 1$.
  - $\gcd(T, d) = 1$: $q^\star | 2^{T/2} + 1$.

- **Conjecture [Goresky Klapper 97]:** Let S be an ℓ–sequence with connection integer $q = p^e$ and period T. Suppose p is prime and $q \notin \{5, 9, 11, 13\}$. For any d_1, d_2 relatively prime to T and incongruent modulo T and any i, j:

  $S^i_{d_1}$ and $S^j_{d_2}$ are cyclically distinct.

Synthesis of Sub-sequences (3)

- **2-adic complexity [Goresky Klapper 97]:**
  - General case: $q^\star | 2^{T^\star} - 1$.
  - $\gcd(T, d) = 1$: $q^\star | 2^{T/2} + 1$.

- **Conjecture [Goresky Klapper 97]:** Let S be an ℓ–sequence with connection integer $q = p^e$ and period T. Suppose p is prime and $q \notin \{5, 9, 11, 13\}$. For any d_1, d_2 relatively prime to T and incongruent modulo T and any i, j:

  $S^i_{d_1}$ and $S^j_{d_2}$ are cyclically distinct.

- **Based on Conjecture:**
  - If q is prime and $gcd(T, d) = 1$ then $q^\star > q$.

Synthesis of Sub-sequences (3)

- **2-adic complexity [Goresky Klapper 97]:**
  - General case: $q^\star | 2^{T^\star} - 1$.
  - $\gcd(T, d) = 1$: $q^\star | 2^{T/2} + 1$.

- **Conjecture [Goresky Klapper 97]:** Let S be an ℓ–sequence with connection integer $q = p^e$ and period T. Suppose p is prime and $q \notin \{5, 9, 11, 13\}$. For any d_1, d_2 relatively prime to T and incongruent modulo T and any i, j:

  $S^i_{d_1}$ and $S^j_{d_2}$ are cyclically distinct.

- **Based on Conjecture:**
  - If q is prime and $gcd(T, d) = 1$ then $q^\star > q$.
  - Let q, p be prime and $T = q - 1 = 2p$:

    $1 \leq d < T$, and $d \neq p$ then $q^\star > q$.

Multiple steps FCSR

- Clock d times the register in one cycle.

Multiple steps FCSR

- Clock d times the register in one cycle.
- Equivalent to partition the register into d sub-registers

$$x_i x_{i+d} \cdots x_{i+kd}$$

such that $0 \leq i < d$ and $i + kd < m$.

Multiple steps FCSR

- Clock d times the register in one cycle.
- Equivalent to partition the register into d sub-registers

$$x_i x_{i+d} \cdots x_{i+kd}$$

  such that $0 \leq i < d$ and $i + kd < m$.

- Interconnection of the sub-registers.

Multiple steps FCSR

- Clock d times the register in one cycle.
- Equivalent to partition the register into d sub-registers

$$x_i x_{i+d} \cdots x_{i+kd}$$

such that $0 \leq i < d$ and $i + kd < m$.

- Interconnection of the sub-registers.
- Propagation of the carry computation.

Fibonacci FCSR (1)

- Let the feedback function be defined by

$$g(X_t, c_t) = \sum_{j=0}^{m-1} (x_j)_t a_j + c_t$$

Fibonacci FCSR (1)

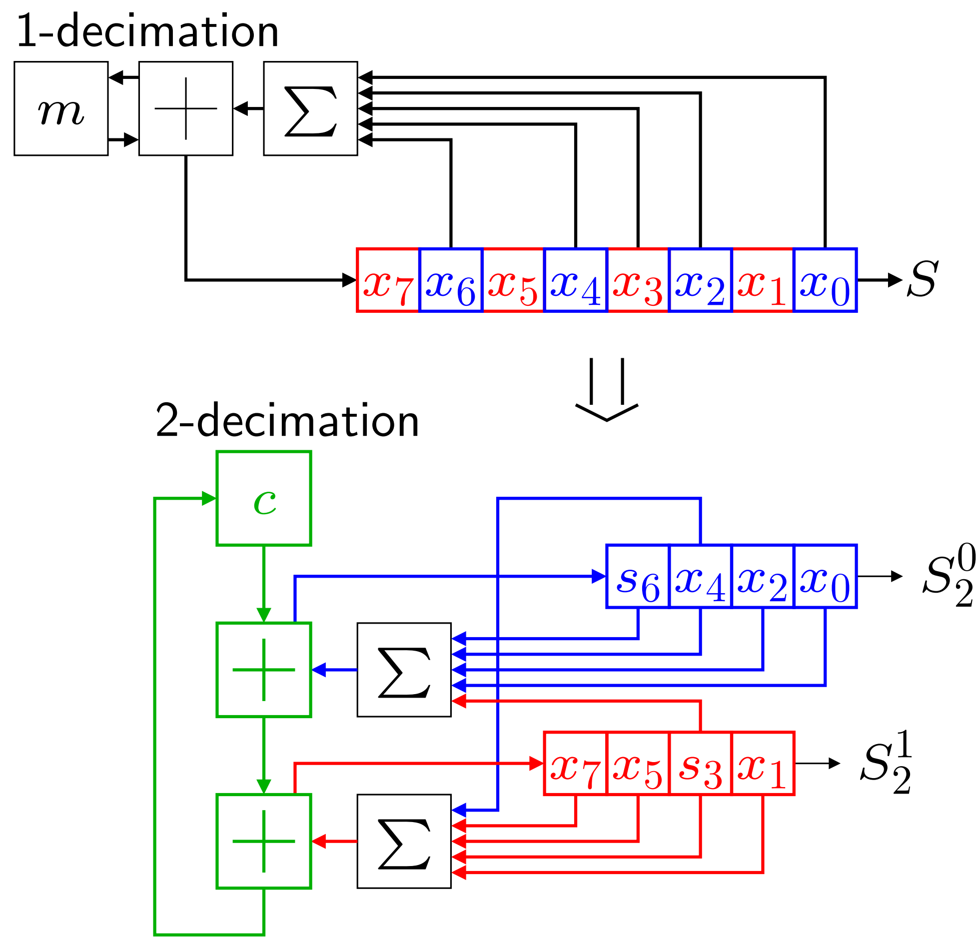
- Let the feedback function be defined by

$$g(X_t, c_t) = \sum_{j=0}^{m-1} (x_j)_t a_j + c_t$$

- We can use the following equations:

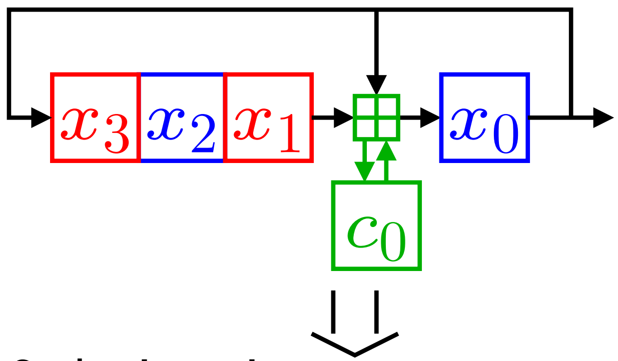
$$\begin{aligned}
(x_i)_{t+d} &= \begin{cases} g(X_{t+d-m+i}, c_{t+d-m+i}) \bmod 2 & \text{if } m-d < i < m \\ (x_{i+d})_t & \text{if } i \leq m-d \end{cases} \\
c_{t+d} &= g(X_{t+d-1}, c_{t+d-1})/2
\end{aligned}$$

Fibonacci FCSR (2)

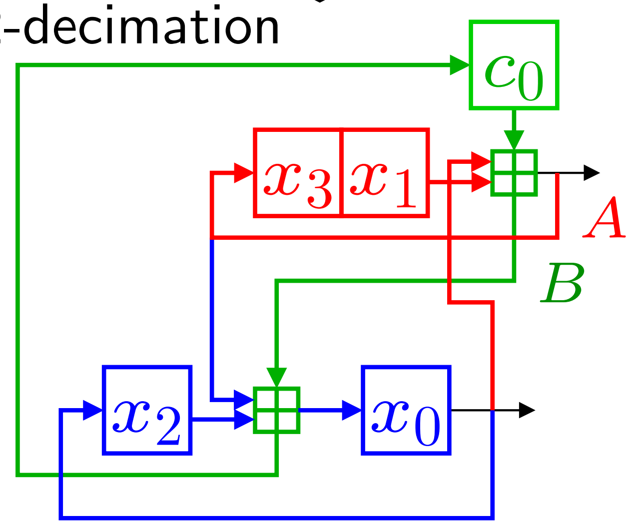


Galois FCSR

1-decimation



2-decimation



$$A = \boxplus\left[(x_0)_t, (x_1)_t, (c_0)_t\right] \bmod 2$$
$$B = \boxplus\left[(x_0)_t, (x_1)_t, (c_0)_t\right]_{\div 2}$$

$$(x_0)_{t+2} = \boxplus\left[A, B, (x_2)_t\right] \bmod 2$$
$$(c_0)_{t+2} = \boxplus\left[A, B, (x_2)_t\right]_{\div 2}$$

$$(x_1)_{t+2} = (x_3)_t$$
$$(x_2)_{t+2} = (x_0)_t$$
$$(x_3)_{t+2} = A$$

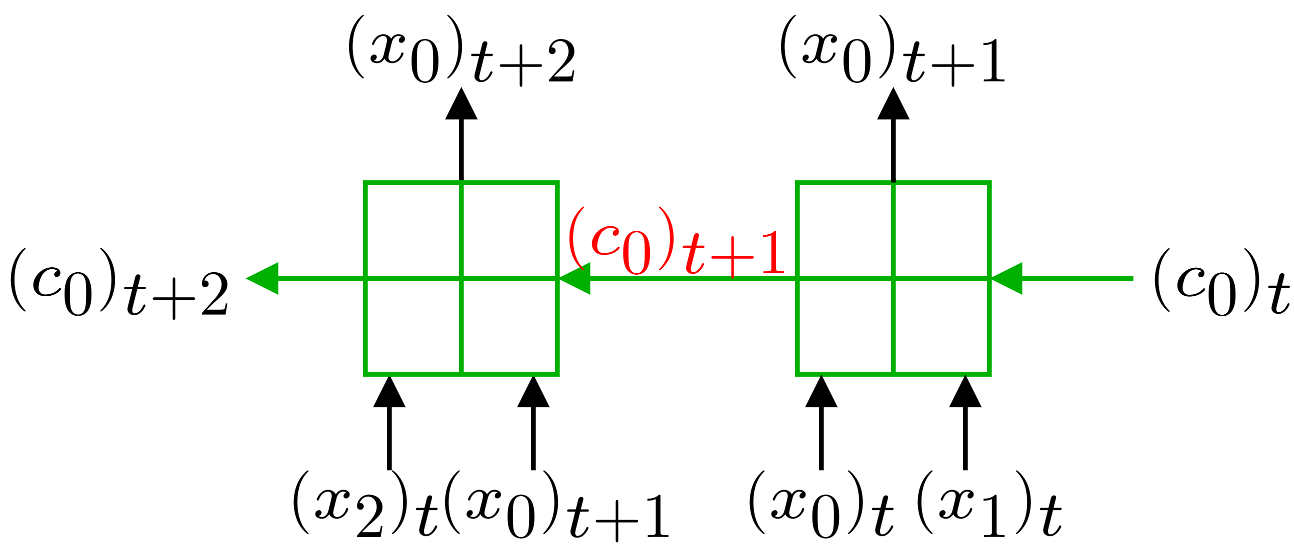
Carry Propagation

- Efficient implementation by means of n-bit ripple carry adder:

Carry Propagation

- Efficient implementation by means of n-bit ripple carry adder:

2-bit ripple carry adder



Comparison

- **Synthesis of Sub-sequences:**
  - Period: If $gcd(T, d) > 1$ it might depend on i.
  - 2-adic complexity: $q^\star$ can be much bigger than q.

Comparison

- **Synthesis of Sub-sequences:**
  - Period: If $gcd(T, d) > 1$ it might depend on i.
  - 2-adic complexity: $q^\star$ can be much bigger than q.

- **Multiple steps FCSR:**
  - Same memory size.
  - Propagation of carry by well-known arithmetic circuits.

Part 4

Conclusion

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.
- A separated FCSR for each sub–sequence is not satisfying.

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.
- A separated FCSR for each sub–sequence is not satisfying. However, the multiple steps FCSR works fine (even with carry).

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.
- A separated FCSR for each sub–sequence is not satisfying. However, the multiple steps FCSR works fine (even with carry).
- Efficient software implementation: 14-bit FCSR with $q = 18433$.

Implementation	Throughput
classic	2.7 MByte/s
decimated ($d = 8$)	19 MByte/s

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.
- A separated FCSR for each sub–sequence is not satisfying. However, the multiple steps FCSR works fine (even with carry).
- Efficient software implementation: 14-bit FCSR with $q = 18433$.

Implementation	Throughput
classic	2.7 MByte/s
decimated ($d = 8$)	19 MByte/s

- Future Work: How to find the best q for hardware/software implementation?

Watermill generator

Conclusion

- The decimation of an ℓ–sequence can be used to increase the throughput or to reduce the power consumption.
- A separated FCSR for each sub–sequence is not satisfying. However, the multiple steps FCSR works fine (even with carry).
- Efficient software implementation: 14-bit FCSR with $q = 18433$.

Implementation	Throughput
classic	2.7 MByte/s
decimated ($d = 8$)	19 MByte/s

- Future Work: How to find the best q for hardware/software implementation?

Watermill generator