# A Very Compact FPGA Implementation of LED and PHOTON

N. Nalla Anandakumar[1,2] Thomas Peyrin[2]
Axel Poschmann[2,3]

[1]Society for Electronic Transactions and Security (SETS), India
[2]Nanyang Technological University (NTU), Singapore
[3]NXP Semiconductors, Germany

Indocrypt - 2014

# Outline

- Introduction
- Algorithms Overview
- Implementations
- Results
- Conclusion

# Lightweight cryptographic algorithms

- Lightweight devices such as
  1. RFID tags
  2. Wireless sensor nodes
  3. Smart cards
- These smart lightweight devices might manipulate sensitive data and thus usually require some security
- Classical cryptographic algorithms are not very suitable for this type of applications
- Thus many lightweight cryptographic schemes have been recently proposed (block ciphers or hash functions)

# Lightweight cryptographic algorithms

- Lightweight devices such as
  1. RFID tags
  2. Wireless sensor nodes
  3. Smart cards
- These smart lightweight devices might manipulate sensitive data and thus usually require some security
- Classical cryptographic algorithms are not very suitable for this type of applications
- Thus many lightweight cryptographic schemes have been recently proposed (block ciphers or hash functions)

In this work we study:

1. LED (the lightweight block cipher)
2. PHOTON (the lightweight family of hash functions)

# Trade-offs

- The main focus of lightweight cryptography research has been on the trade-offs between
  - Cost
  - Security
  - Performance in terms of speed, area and computational power.
- These primitives can be implemented either in software or in hardware platforms such as
  - Field-Programmable Gate Array (**FPGA**)
  - Application Specific Integrated Circuit (**ASIC**)
- Compared to ASICs, FPGAs offer additional advantages in terms of
  1. Time-to-market
  2. Reconfigurability
  3. Cost

# Our contributions.

- In this article, we describe three different hardware architectures of the LED and PHOTON family optimized for FPGA devices

    1. Round-based architecture: computes one round per clock cycle
    2. Fully serialized architecture: performing operations on a single cell per clock cycle
    3. Serialized using SRL16: computations based on shift registers (SRL16)

# Our Goal.

- To cover a wide variety of new implementation trade-offs offered by crypto primitives using serialized MDS (Maximum Distance Separable) matrices
- For which LED and PHOTON are the main representatives
- Implemented on a wide variety of different Xilinx FPGA families, ranging from low-cost (**Spartan-3**) to high-end (**Artix-7**).

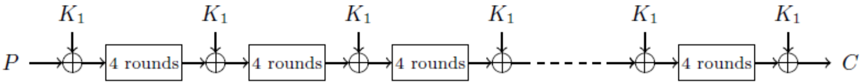
# LED Algorithm

- Substitution-Permutation Network,
- 64-bit block size,
- 64-128 bit key length, 32/48 rounds,
- No Keyschedule (Key repeated every four rounds),

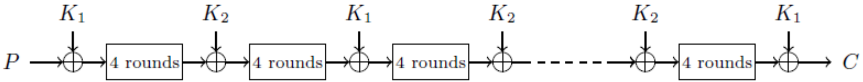
# LED Algorithm

- Substitution-Permutation Network,
- 64-bit block size,
- 64-128 bit key length, 32/48 rounds,
- No Keyschedule (Key repeated every four rounds),

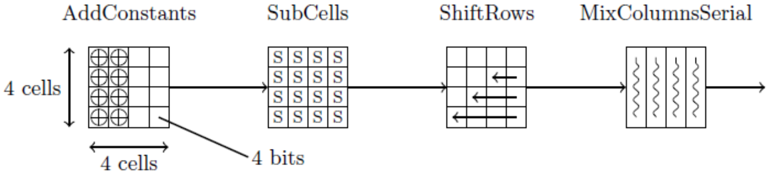
a 64-bit key array



a 128-bit key array



# A single round of LED

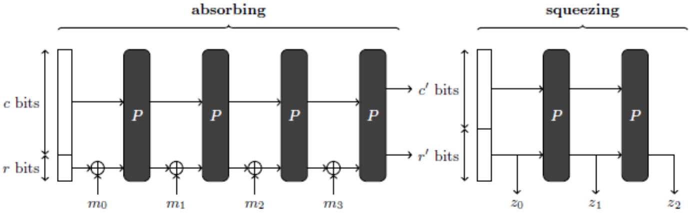


- *AddConstants*: xor round-dependent constants to the two first columns
- *SubCells*: apply the PRESENT 4-bit Sbox to each cell
- *ShiftRows*: rotate the i-th line by i positions to the left
- *MixColumnsSerial*: each nibble column of the internal state is transformed by multiplying it once with MDS matrix $\chi^4$ (or two times with matrix $\chi^2$, or four times with matrix $\chi$)

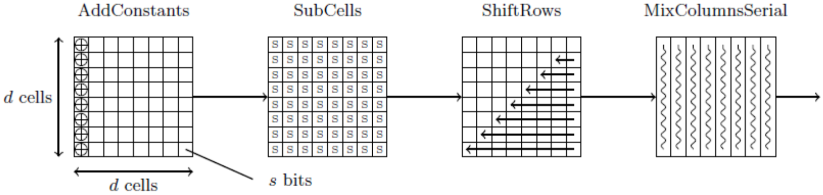
$$\chi = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{pmatrix}; \quad (\chi)^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \end{pmatrix}; \quad (\chi)^4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{pmatrix}^4 = \begin{pmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{pmatrix}$$

# PHOTON Algorithm

- PHOTON is a family of sponge functions, characterized by two parameters: a bitrate r, and a capacity c.
- Each PHOTON hash function is denoted by PHOTON-$n/r/r'$
- The (t=c + r)-bit, with c = n, internal state is viewed as a $(d \times d)$ matrix of s-bit cells.
- Two Phases:
  - absorbing phase: iteratively processes all the $r$-bit message chunks by XORing them to the bitrate part of the internal state and then applying the t-bit permutation P
  - squeezing phase: the extracting $r'$ bits from the bitrate part of the internal state and then applying the permutation P on it.



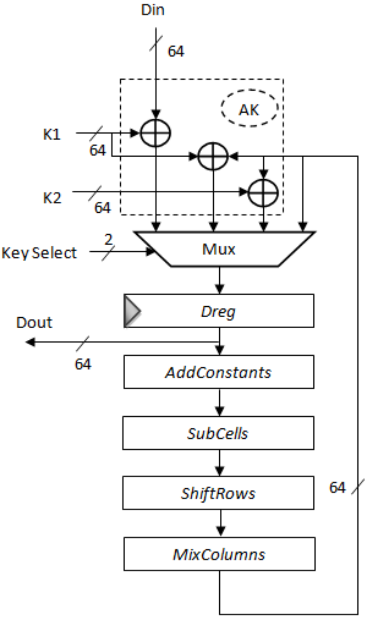
# One round of a PHOTON permutation



The internal permutations apply 12 rounds

- *AddConstants*: xor round-dependant constants to the first column
- *SubCells*: apply the PRESENT Sbox (when s = 4) or AES Sbox (when s = 8) to each cell
- *ShiftRows*: rotate the i-th line by i positions to the left
- *MixColumnsSerial*: each nibble column of the internal state is transformed by multiplying it once with MDS matrix

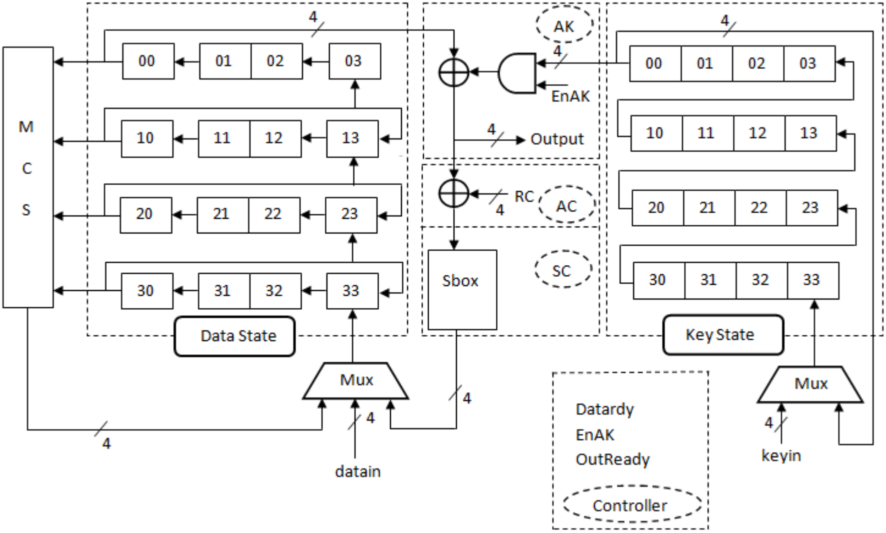
# LED round based encryption architecture



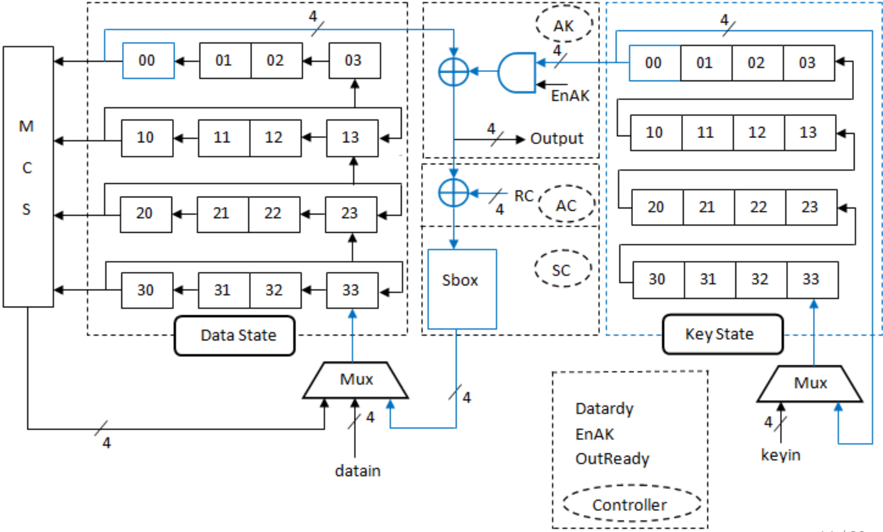
# FPGA round-based implementation results of LED.

| Design | MDS approach | Block Size (bits) | Key Size (bits) | Area (slices) | Clock Cycles | T/put (Mbps) | Eff. (Mbps/slices) | FPGA Device |
|---|---|---|---|---|---|---|---|---|
| LED Round−based | $(\chi)$ | 64 | 64 | 170 | 32 | 157.56 | 0.93 | Spartan-3 XC3S50-5 |
| | | | 128 | 199 | 48 | 104.8 | 0.53 | |
| | $(\chi)^2$ | 64 | 64 | 198 | 32 | 175.3 | 0.89 | |
| | | | 128 | 227 | 48 | 116.54 | 0.51 | |
| | $(\chi)^4$ | 64 | 64 | 204 | 32 | 197.35 | 0.97 | |
| | | | 128 | 233 | 48 | 131.2 | 0.56 | |
| | $(\chi)$ | 64 | 64 | 102 | 32 | 565.54 | 5.50 | Artix-7 XC7A100T-3 |
| | | | 128 | 158 | 48 | 376.57 | 2.39 | |
| | $(\chi)^2$ | 64 | 64 | 110 | 32 | 580.97 | 5.28 | |
| | | | 128 | 163 | 48 | 389.18 | 2.40 | |
| | $(\chi)^4$ | 64 | 64 | 136 | 32 | 669.7 | 4.92 | |
| | | | 128 | 168 | 48 | 444.97 | 2.65 | |
| PRESENT | | 64 | 128 | 202 | 32 | 508 | 2.51 | Spartan-3 XC3S400-5 |
| AES | | 128 | 128 | 17,425 | — | 25,107 | 1.44 | Spartan-3 XC3S2000-5 |
| AES | | 128 | 128 | 1800 | — | 1700 | 0.90 | Spartan 3 |
| ICEBERG | | 64 | 128 | 631 | — | 1016 | 1.61 | Virtex-II |
| SEA | | 126 | 126 | 424 | — | 156 | 0.37 | Virtex-II XC2V4000 |

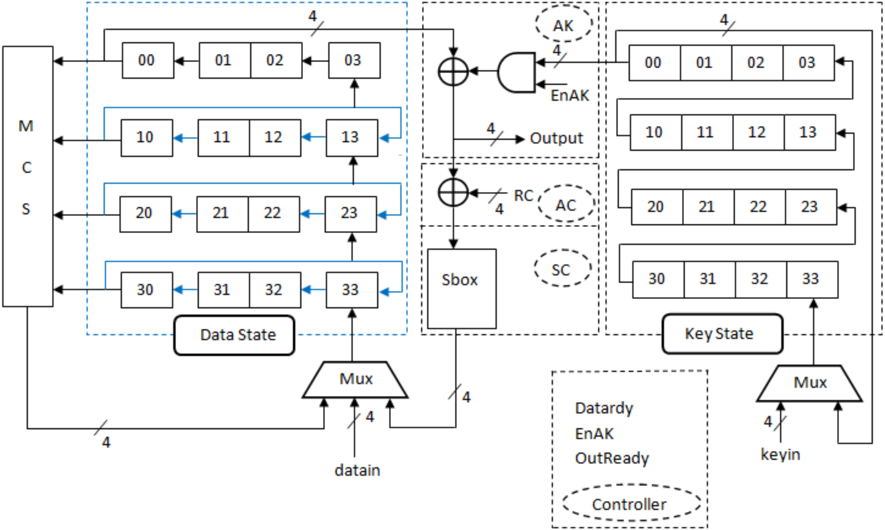
# Serialized LED encryption architecture



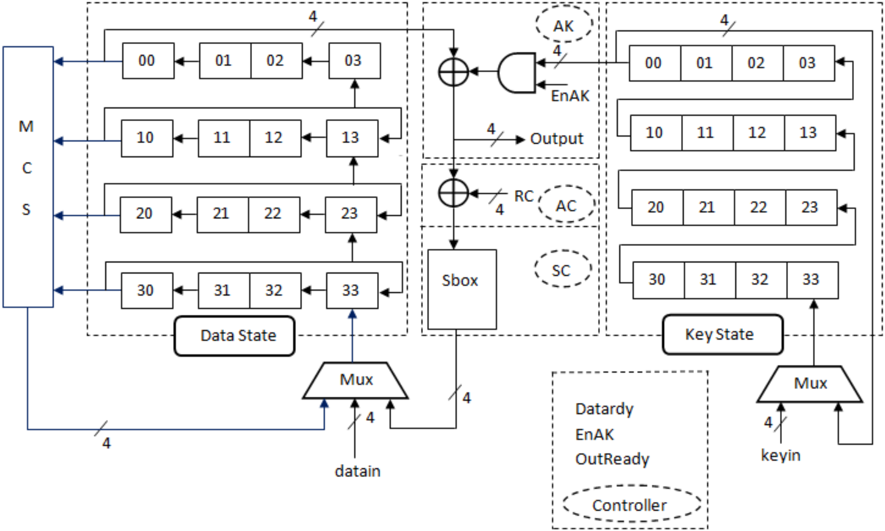
# Serialized LED encryption architecture



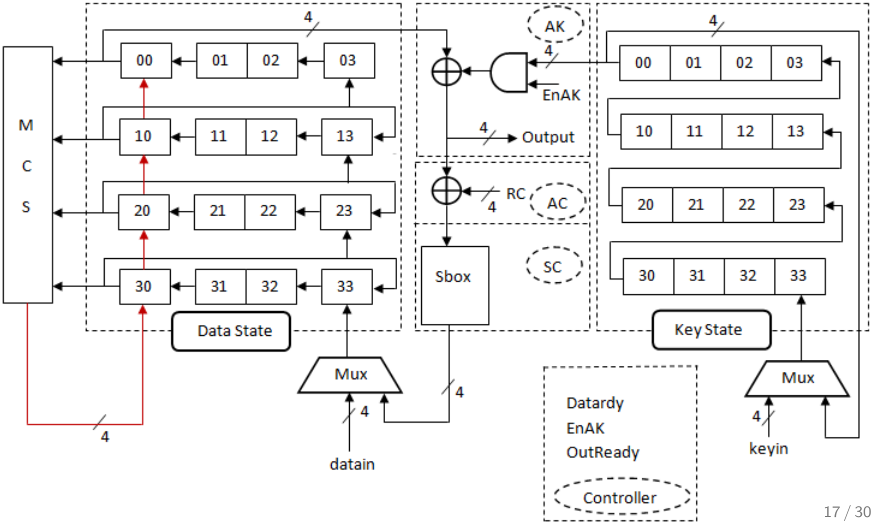
# Serialized LED encryption architecture



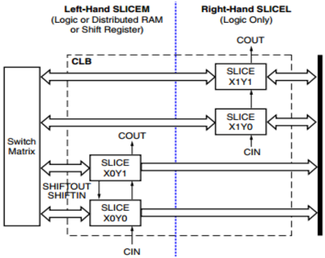
# Serialized LED encryption architecture

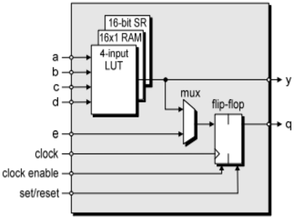


# Serialized LED architecture: original proposal for ASICs



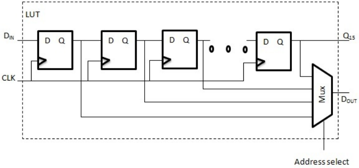
# SRL16s based implementation: Xilinx Shift Register





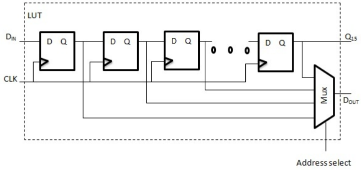
- **The CLB is the basic logic unit in a FPGA**.
  - Each CLB has four slices.
  - Only the two at the left of the CLB can be used as shift registers.
- **LUT can be configured as a 16-bit shift register (SRL16)**
  - 32 bit shift register normally requires 16 slices
  - Using SRL16 requires only 2 slices

# SRL16s based implementation of LED



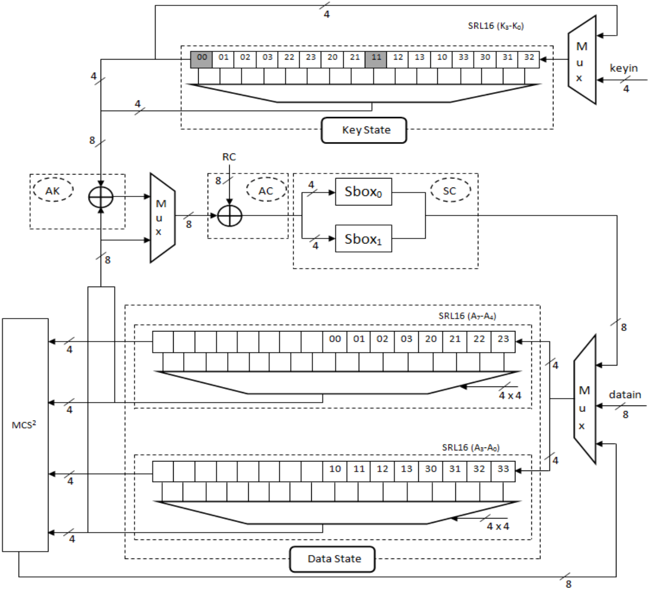
- Data read from SRL16s by two ways:
    - The last bit of its 16 stages (Q15) is always available.
    - A multiplexer allows to access one additional bit from any of its internal stages.

# SRL16s based implementation of LED



- Data read from SRL16s by two ways:
  - The last bit of its 16 stages (Q15) is always available.
  - A multiplexer allows to access one additional bit from any of its internal stages.
- Investigated possible area reductions using SRL16s:
  - 8-bit datapath when using $(\chi)^2$
  - 16-bit datapath when using $(\chi)$ and $(\chi)^4$
    - MixColumnsSerial requires 16-bit inputs (4 times 4-bit) in every clock cycle
    - Each SRL16 only allows access to 2 bits
    - We have to use eight and sixteen SRL16s to store the state, respectively.

# SRL16s based implementation of LED



# Content of SRL16s for one round of LED when using $(\chi)^2$ for the 8-bit datapath

| clk | content of SRL16s | clk | content of SRL16s |
|---|---|---|---|
| | *Init* | | *Re-update* |
| 1 | 00<br>10 | 17 | 01 02 03 20 21 22 23 00 **01** 02 03 22 23 20 21 00<br>11 12 13 30 31 32 33 11 **12** 13 10 33 30 31 32 11 |
| 2 | 00 01<br>10 11 | 18 | 02 03 20 21 22 23 00 01 **02** 03 22 23 20 21 00 01<br>12 13 30 31 32 33 11 12 **13** 10 33 30 31 32 11 12 |
| 3 | 00 01 02<br>10 11 12 | 19 | 03 20 21 22 23 00 01 02 **03** 22 23 20 21 00 01 02<br>13 30 31 32 33 11 12 13 **10** 33 30 31 32 11 12 13 |
| 4 | 00 01 02 03<br>10 11 12 13 | 20 | 20 21 22 23 00 01 02 03 **22** 23 20 21 00 01 02 03<br>30 31 32 33 11 12 13 10 **33** 30 31 32 11 12 13 10 |
| 5 | 00 01 02 03 20<br>10 11 12 13 30 | 21 | 21 22 23 00 01 02 03 22 **23** 20 21 00 01 02 03 22<br>31 32 33 11 12 13 10 33 **30** 31 32 11 12 13 10 33 |
| 6 | 00 01 02 03 20 21<br>10 11 12 13 30 31 | 22 | 22 23 00 01 02 03 22 23 **20** 21 00 01 02 03 22 23<br>32 33 11 12 13 10 33 30 **31** 32 11 12 13 10 33 30 |
| 7 | 00 01 02 03 20 21 22<br>10 11 12 13 30 31 32 | 23 | 23 00 01 02 03 22 23 20 **21** 00 01 02 03 22 23 20<br>33 11 12 13 10 33 30 31 **32** 11 12 13 10 33 30 31 |
| 8 | **00** 01 02 03 20 21 22 23<br>10 **11** 12 13 30 31 32 33 | 24 | **00** 01 02 03 22 23 20 21 00 01 02 03 **22** 23 20 21<br>**11** 12 13 10 33 30 31 32 11 12 13 10 **33** 30 31 32 |
| | *SrSc* | | *MCS* |
| 9 | 00 **01** 02 03 20 21 22 23 00<br>10 11 **12** 13 30 31 32 33 11 | 25 | **01** 02 03 22 23 20 21 00 01 02 03 22 **23** 20 21 00′<br>**12** 13 10 33 30 31 32 11 12 13 10 33 **30** 31 32 10′ |
| 10 | 00 01 **02** 03 20 21 22 23 00 01<br>10 11 12 **13** 30 31 32 33 11 12 | 26 | **02** 03 22 23 20 21 00 01 02 03 22 23 **20** 21 00′ 01′<br>**13** 10 33 30 31 32 11 12 13 10 33 30 **31** 32 10′ 11′ |
| 11 | 00 01 02 **03** 20 21 22 23 00 01 02<br>**10** 11 12 13 30 31 32 33 11 12 13 | 27 | **03** 22 23 20 21 00 01 02 03 22 23 20 **21** 00′ 01′ 02′<br>**10** 33 30 31 32 11 12 13 10 33 20 31 **32** 10′ 11′ 12′ |
| 12 | 00 01 02 03 20 21 **22** 23 00 01 02 03<br>10 11 12 13 30 31 32 **33** 11 12 13 10 | 28 | **22** 23 20 21 00 01 02 03 22 23 20 21 **00′** 01′ 02′ 03′<br>**33** 30 31 32 11 12 13 10 33 30 31 32 **10′** 11′ 12′ 13′ |
| 13 | 00 01 02 03 20 21 22 **23** 00 01 02 03 22<br>10 11 12 13 **30** 31 32 33 11 12 13 10 33 | 29 | **23** 20 21 00 01 02 03 22 23 20 21 00′ **01′** 02′ 03′ 20′<br>**30** 31 32 11 12 13 10 33 30 31 32 10′ **11′** 12′ 13′ 30′ |
| 14 | 00 01 02 03 **20** 21 22 23 00 01 02 03 22 23<br>10 11 12 13 30 **31** 32 33 11 12 13 10 33 30 | 30 | **20** 21 00 01 02 03 22 23 20 21 00′ 01′ **02′** 03′ 20′ 21′<br>**31** 32 11 12 13 10 33 30 31 32 10′ 11′ **12′** 13′ 30′ 31′ |
| 15 | 00 01 02 03 20 **21** 22 23 00 01 02 03 22 23 20<br>10 11 12 13 30 31 **32** 33 11 12 13 10 33 30 31 | 31 | **21** 00 01 02 03 22 23 20 21 00′ 01′ 02′ **03′** 20′ 21′ 22′<br>**32** 11 12 13 10 33 30 31 32 10′ 11′ 12′ **13′** 30′ 31′ 32′ |
| 16 | 00 01 02 03 20 21 22 23 **00** 01 02 03 22 23 20 21<br>10 11 12 13 30 31 32 33 **11** 12 13 10 33 30 31 32 | 32 | 00 01 02 03 22 23 20 21 **00′** 01′ 02′ 03′ 20′ 21′ 22′ 23′<br>11 12 13 10 33 30 31 32 10′ **11′** 12′ 13′ 30′ 31′ 32′ 33′ |

# FPGA serialized implementation results of LED

| Design | MDS approach | Data-path (bits) | Block Size (bits) | Key Size (bits) | Area (slices) | Clock Cycles | T/put (Mbps) | Eff. (Mbps/ slices) | FPGA Device |
|---|---|---|---|---|---|---|---|---|---|
| LED<br>Serialized | $(\chi)$ | 4 | 64 | 64 | 140 | 1120 | 9.11 | 0.07 | Spartan-3 XC3S50-5 |
| | | | | 128 | 167 | 1680 | 5.2 | 0.03 | |
| | $(\chi)^2$ | 8 | 64 | 64 | 169 | 608 | 16.6 | 0.10 | |
| | | | | 128 | 203 | 912 | 9.97 | 0.05 | |
| | $(\chi)^4$ | 16 | 64 | 64 | 180 | 352 | 24.99 | 0.14 | |
| | | | | 128 | 219 | 528 | 15.6 | 0.07 | |
| | $(\chi)$ | 4 | 64 | 64 | **37** | 1120 | 21.6 | 0.58 | Artix-7 XC7A100T-3 |
| | | | | 128 | **40** | 1680 | 14.02 | 0.35 | |
| | $(\chi)^2$ | 8 | 64 | 64 | 58 | 608 | 40.03 | 0.69 | |
| | | | | 128 | 61 | 912 | 25.02 | 0.41 | |
| | $(\chi)^4$ | 16 | 64 | 64 | 78 | 352 | **66.8** | **0.86** | |
| | | | | 128 | 82 | 528 | **45.53** | **0.56** | |
| LED<br>Serialized using SRL16s | $(\chi)$ | 16 | 64 | 64 | 111 | 640 | 11.96 | 0.11 | Spartan-3 XC3S50-5 |
| | | | | 128 | 122 | 960 | 7.88 | 0.06 | |
| | $(\chi)^2$ | 8 | 64 | 64 | 77 | 768 | 9.93 | 0.13 | |
| | | | | 128 | 86 | 1152 | 6.71 | 0.08 | |
| | $(\chi)^4$ | 16 | 64 | 64 | 119 | 256 | 29.82 | 0.25 | |
| | | | | 128 | 127 | 384 | 19.65 | 0.15 | |
| | $(\chi)$ | 16 | 64 | 64 | 51 | 640 | 30.39 | 0.60 | Artix-7 XC7A100T-3 |
| | | | | 128 | 59 | 960 | 20.57 | 0.35 | |
| | $(\chi)^2$ | 8 | 64 | 64 | **40** | 768 | 22.93 | 0.57 | |
| | | | | 128 | **50** | 1152 | 16.81 | 0.34 | |
| | $(\chi)^4$ | 16 | 64 | 64 | 63 | 256 | **71.21** | **1.13** | |
| | | | | 128 | 69 | 384 | **47.75** | **0.70** | |
| PRESENT | | | 64 | 128 | 117 | 256 | 28.46 | 0.24 | Spartan-3 XC3S50-5 |
| HIGHT | | | 64 | 128 | 91 | 160 | 65.48 | 0.72 | Spartan-3 XC3S50-5 |
| xTEA | | | 64 | 128 | 254 | 112 | 35.78 | 0.14 | Spartan-3 XC3S50-5 |
| PRESENT | | | 64 | 80 | 271 | — | — | — | Spartan-3E XC3S500 |
| SIMON | | | 128 | 128 | 36 | — | 3.60 | 0.10 | Spartan-3E XC3S500 |
| AES | | | 128 | 128 | 184 | 160 | 36.5 | 0.20 | Spartan-3 XC3S50-5 |
| AES | | | 128 | 128 | 393 | 534 | 16.86 | 0.04 | Spartan-3 XC3S50-5 |

# SRL16s based implementation of LED

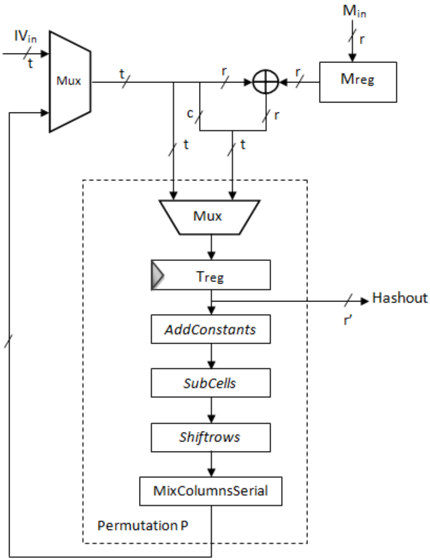
- One can see in the previous table that our SRL16 implementation technique both saves area and increases throughput compared to a classical optimized serial implementation.

# SRL16s based implementation of LED

- One can see in the previous table that our SRL16 implementation technique both saves area and increases throughput compared to a classical optimized serial implementation.

- We believe this technique is very interesting in order to implement serial-matrix based cryptographic primitives in FPGA technology.

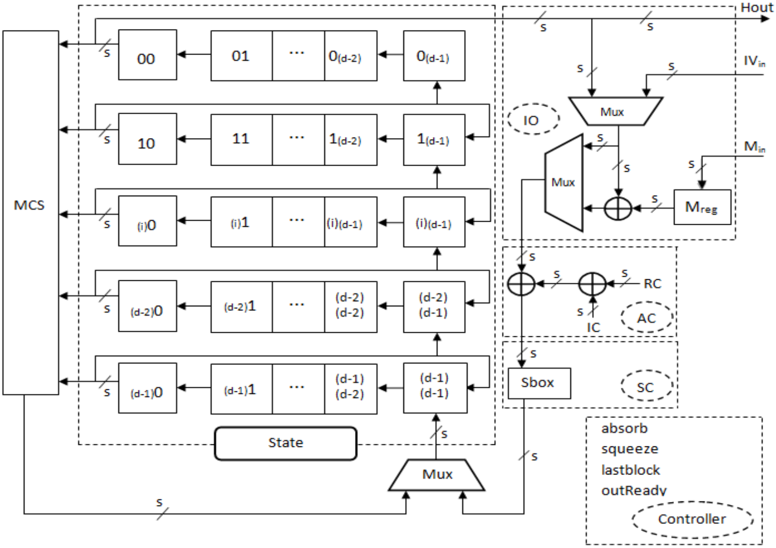
# A round based architecture of the PHOTON



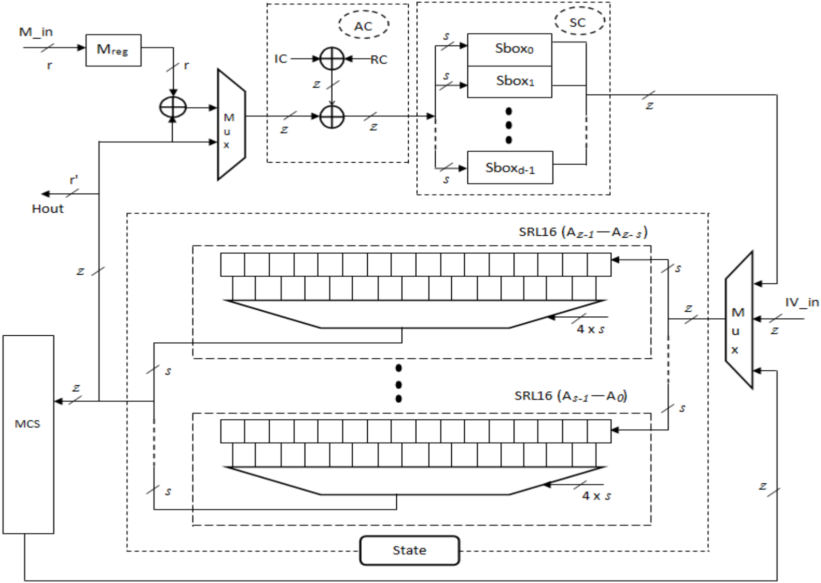
# Round-based implementation results of PHOTON

| Design | MDS approach | Data-path (bits) | Area (slices) | Clock Cycles | T/put (Mbps) | Eff. (Mbps/ slices) | FPGA Device |
|---|---|---|---|---|---|---|---|
| PHOTON-80/20/16 | $(\chi)$ | 100 | 285 | 12 | 130.88 | 0.46 | Spartan-3 XC3S50-5 |
| | $(\chi)$ | 100 | 142 | 12 | 387.75 | 2.73 | Artix-7 XC7A100T-3 |
| SPONGENT-88 | | 88 | 157 | 45 | 17.78 | 0.11 | Spartan-3 |
| PHOTON-128/16/16 | $(\chi)$ | 144 | 549 | 12 | 87.19 | 0.16 | Spartan-3 XC3S50-5 |
| | $(\chi)$ | 144 | 204 | 12 | 252.04 | 1.24 | Artix-7 XC7A100T-3 |
| SPONGENT-128 | | 136 | 208 | 70 | 11.43 | 0.06 | Spartan-3 |
| PHOTON-160/36/36 | $(\chi)$ | 196 | 846 | 12 | 183.09 | 0.22 | Spartan-3 XC3S400-5 |
| | $(\chi)$ | 196 | 429 | 12 | 467.25 | 1.10 | Artix-7 XC7A100T-3 |
| SPONGENT-160 | | 176 | 264 | 90 | 8.89 | 0.03 | Spartan-3 |
| PHOTON-224/32/32 | $(\chi)$ | 256 | 1235 | 12 | 137.95 | 0.11 | Spartan-3 XC3S400-5 |
| | $(\chi)$ | 256 | 616 | 12 | 402.11 | 0.65 | Artix-7 XC7A100T-3 |
| SPONGENT-224 | | 240 | 322 | 120 | 6.67 | 0.02 | Spartan-3 |
| PHOTON-256/32/32 | $(\chi)$ | 288 | 2067 | 12 | 94.24 | 0.05 | Spartan-3 XC3S400-5 |
| | $(\chi)$ | 288 | 865 | 12 | 299.81 | 0.35 | Artix-7 XC7A100T-3 |
| SPONGENT-256 | | 272 | 357 | 140 | 5.71 | 0.02 | Spartan-3 |
| CUBEHASH-256 | | — | 2883 | — | 50 | 0.017 | Spartan-3 XC3S5000-5 |

# A serialized architecture of the PHOTON



# The SRL16s based implementation of PHOTON



# Serialized implementation results of PHOTON

| Design | impl. approach | MDS approach | Data-path (bits) | Area (slices) | Clock Cycles | T/put (Mbps) | Eff. (Mbps/ slices) | FPGA Device |
|---|---|---|---|---|---|---|---|---|
| PHOTON-80/20/16 | serial | $(\chi)$ | 4 | 146 | 648 | 3.10 | 0.02 | Spartan-3 XC3S50-5 |
| | SRL16 | $(\chi)$ | 20 | 112 | 360 | 6.57 | 0.06 | |
| | serial | $(\chi)$ | 4 | 67 | 648 | 10.17 | 0.15 | Artix-7 XC7A100T-3 |
| | SRL16 | $(\chi)$ | 20 | 58 | 360 | 18.33 | 0.32 | |
| | serial | $(\chi)$ | 4 | **82** | 648 | 9.34 | **0.11** | Virtex-5 XC5VLX50-1 |
| | SRL16 | $(\chi)$ | 20 | **69** | 360 | 15.84 | **0.22** | |
| PHOTON-80/20/16 | | | 4 | 149 | 708 | 7 | 0.05 | Virtex-5 |
| SPONGENT-88 | | | 4 | 116 | 900 | .81 | 0.01 | Spartan-3 |
| PHOTON-128/16/16 | serial | $(\chi)$ | 4 | 183 | 924 | 1.76 | 0.01 | Spartan-3 XC3S50-5 |
| | SRL16 | $(\chi)$ | 24 | 137 | 504 | 3.67 | 0.03 | |
| | serial | $(\chi)$ | 4 | 84 | 924 | 6.24 | 0.07 | Artix-7 XC7A100T-3 |
| | SRL16 | $(\chi)$ | 24 | 72 | 504 | 10.87 | 0.20 | |
| PHOTON-128/16/16 | | | 4 | 469 | 948 | .551 | 0.001 | Spartan-3 |
| SPONGENT-128 | | | 4 | 144 | 2380 | .34 | 0.002 | Spartan-3 |
| PHOTON-160/36/36 | serial | $(\chi)$ | 4 | 233 | 1248 | 2.01 | 0.01 | Spartan-3 XC3S50-5 |
| | SRL16 | $(\chi)$ | 28 | 164 | 672 | 6.58 | 0.04 | |
| | serial | $(\chi)$ | 4 | 117 | 1248 | 9.47 | 0.08 | Artix-7 XC7A100T-3 |
| | SRL16 | $(\chi)$ | 28 | 89 | 672 | 17.58 | 0.20 | |
| SPONGENT-160 | | | 4 | 193 | 3960 | .2 | 0.001 | Spartan-3 |
| PHOTON-224/32/32 | serial | $(\chi)$ | 4 | 274 | 1620 | 1.36 | 0.005 | Spartan-3 XC3S50-5 |
| | SRL16 | $(\chi)$ | 32 | 176 | 864 | 4.57 | 0.03 | |
| | serial | $(\chi)$ | 4 | 130 | 1620 | 7.55 | 0.06 | Artix-7 XC7A100T-3 |
| | SRL16 | $(\chi)$ | 32 | 96 | 864 | 12.12 | 0.13 | |
| SPONGENT-224 | | | 4 | 225 | 7200 | .11 | 0.0005 | Spartan-3 |
| PHOTON-256/32/32 | serial | $(\chi)$ | 8 | 327 | 924 | 1.47 | 0.004 | Spartan-3 XC3S50-5 |
| | SRL16 | $(\chi)$ | 48 | 416 | 504 | 3.74 | 0.009 | |
| | serial | $(\chi)$ | 8 | 157 | 924 | 4.59 | 0.03 | Artix-7 XC7A100T-3 |
| | SRL16 | $(\chi)$ | 48 | 159 | 504 | 10.75 | 0.07 | |
| SPONGENT-256 | | | 4 | 241 | 9520 | 0.08 | .0003 | Spartan-3 XC3S200-5 |
| SHABAL-256 | | | — | 499 | — | .8 | 1.60 | Spartan-3 XC3S200-5 |
| BLAKE-256 | | | — | 631 | — | 216.3 | 0.34 | Spartan-3 XC3S50-5 |
| GRØSTL | | | — | 766 | — | 192.6 | 0.25 | Spartan-3 XC3S50-5 |
| JH | | | — | 558 | — | 63.7 | 0.11 | Spartan-3 XC3S50-5 |
| KECCAK | | | — | 766 | — | 46.2 | 0.06 | Spartan-3 XC3S50-5 |
| SKEIN | | | — | 766 | — | 16.6 | 0.02 | Spartan-3 XC3S50-5 |
| SHA-2 | | | — | 745 | — | 137.8 | 0.19 | Spartan-3 XC3S50-5 |

# Conclusion

- In this paper, we have analyzed the feasibility of creating a very compact, low cost FPGA implementation of LED and PHOTON.
- For both primitives, we studied round-based and serial architectures.
- We implemented several possible trade-offs when computing the diffusion matrix.
- Our results show that LED and PHOTON are very good candidates for lightweight applications.
- Our implementations yield for example the best area of all lightweight hash functions implementations published so far.

Thank you!

Any questions?