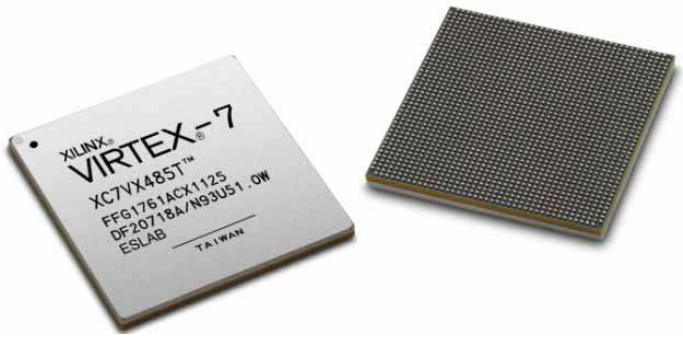
Noekeon

technology
from seed

Hardware Tutorial

Nele Mentens and Ricardo Chaves



Outline

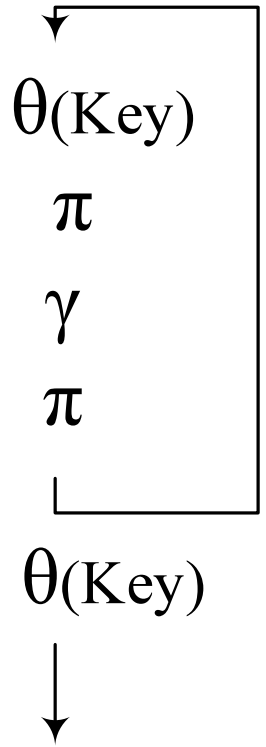
- Tutorial Tasks
  - Fully folded Noekeon loop – just simulate and get the results
  - Unroll the loop
  - Fully unroll the loop
  - Add pipelining to the fully unrolled loop
  - Design the decryption
  - Compute the round constant using LFSR
- Digesting VHDL
- Noekeon particularities
- FPGA

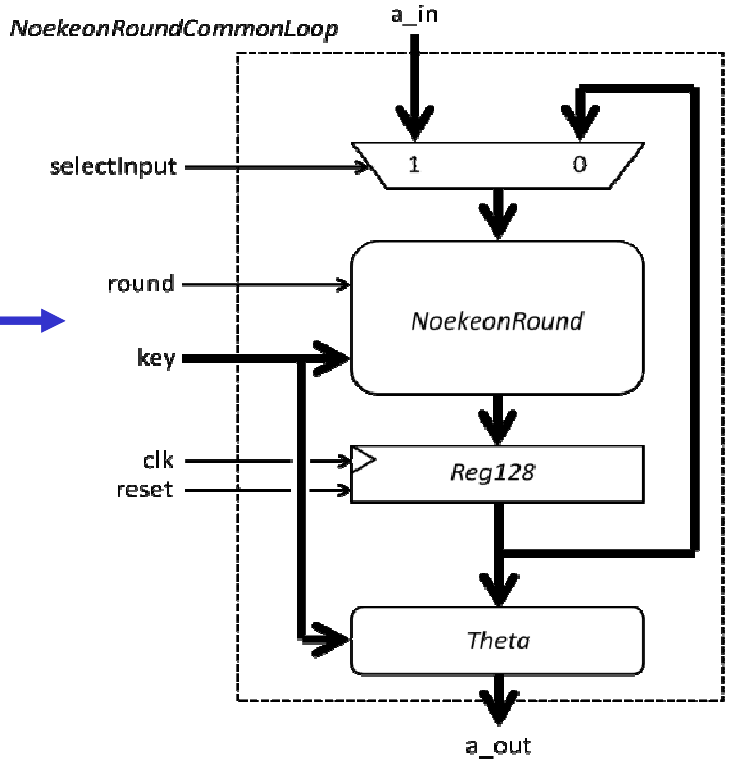
Tutorial Tasks

Description of Noekeon

technology
from seed

- *Describing the structure*
- *Noekeon loop*
  - *Algorithm*
  - *Structure*
  - *VHDL description*





Tutorial Tasks

Noekeon Wrapper

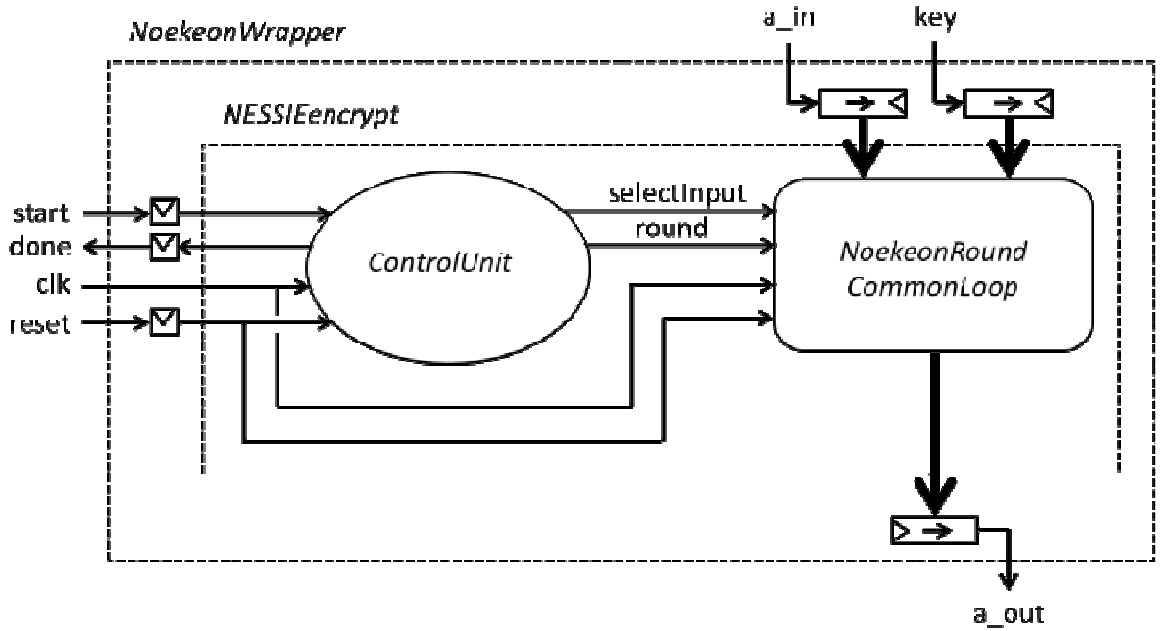
technology
from seed

inesc id lisboa

Do I map all in/out ports to the IOs of the device

- *128 x 3 + control = 388 bits!*

- *Lets wrap it in a more compact version with 32bit ports*



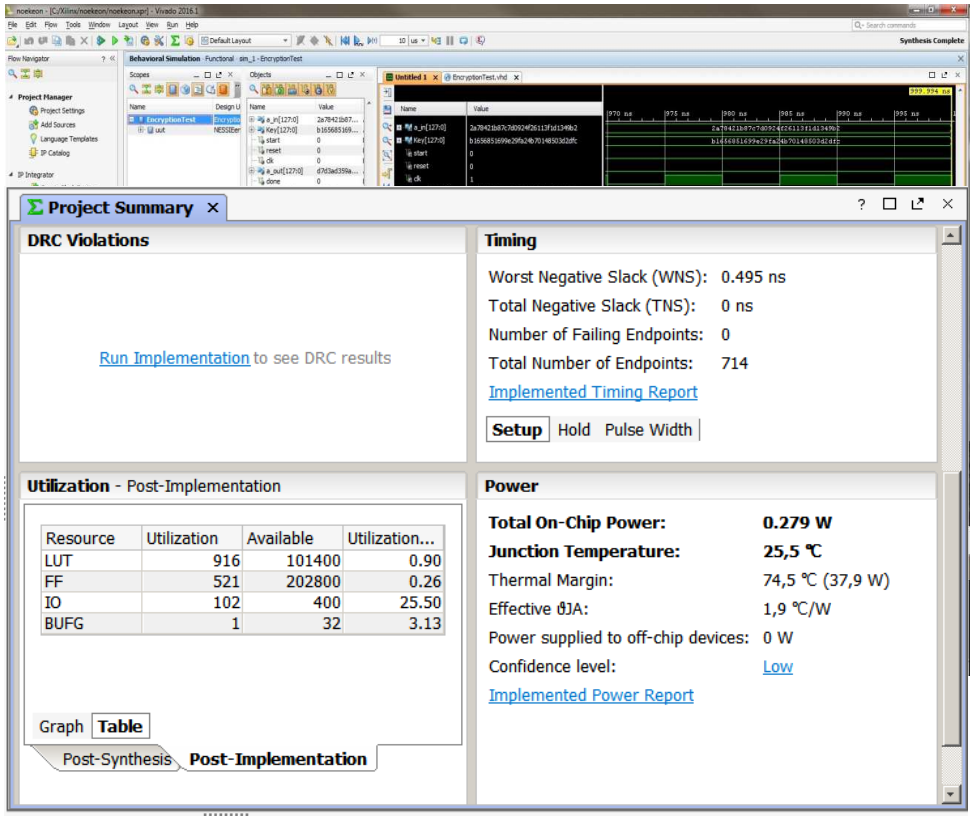
Tutorial Tasks

Simulating and getting results

technology
from seed

Making sure it does what it should:

- *Simulate it:*
  - *Behavioral*
  - *Post-synthesis*
  - *Post-Implementation*
- *Using a test bench*
  - *EncryptionTest.vhd*
    - *Includes the test vectors*



Getting it onto the device Implementation:

- *Synthesis*
  - *Translating it into logic gates*
- *Implementation*
  - *Actually placing it onto the device*
    - *LUTs*
    - *Registers*
    - *…*
- *Using the Wrapper module: NoekeonWrapper.vhd*

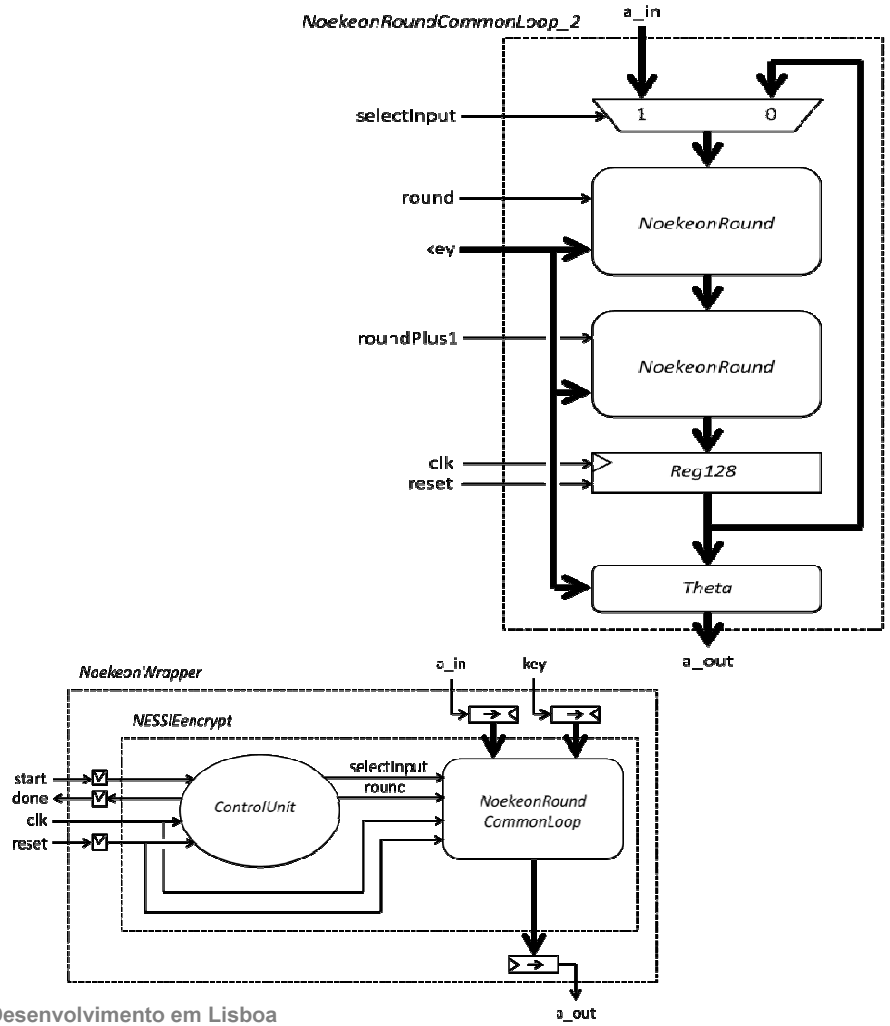
Tutorial Tasks

Unroll the loop

- Unroll the loop
  - Computing 2 rounds each clk cycle

- We now need to increment the round counter by 2 each cycle.
  - 0, 1
  - 2, 3
  - 4, 5
  - …

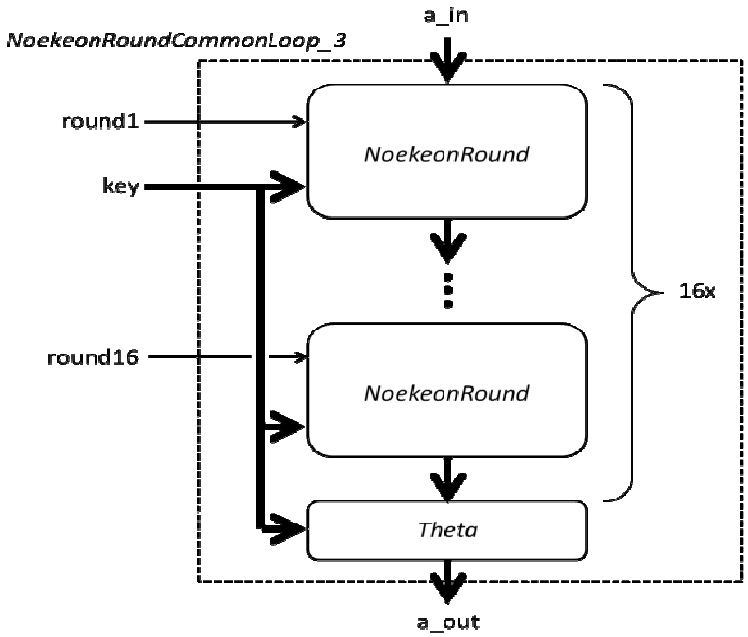
- How much faster do I get?
  - At what cost?
  - Can the clock run at the same frequency?



Tutorial Tasks

Fully unroll the loop

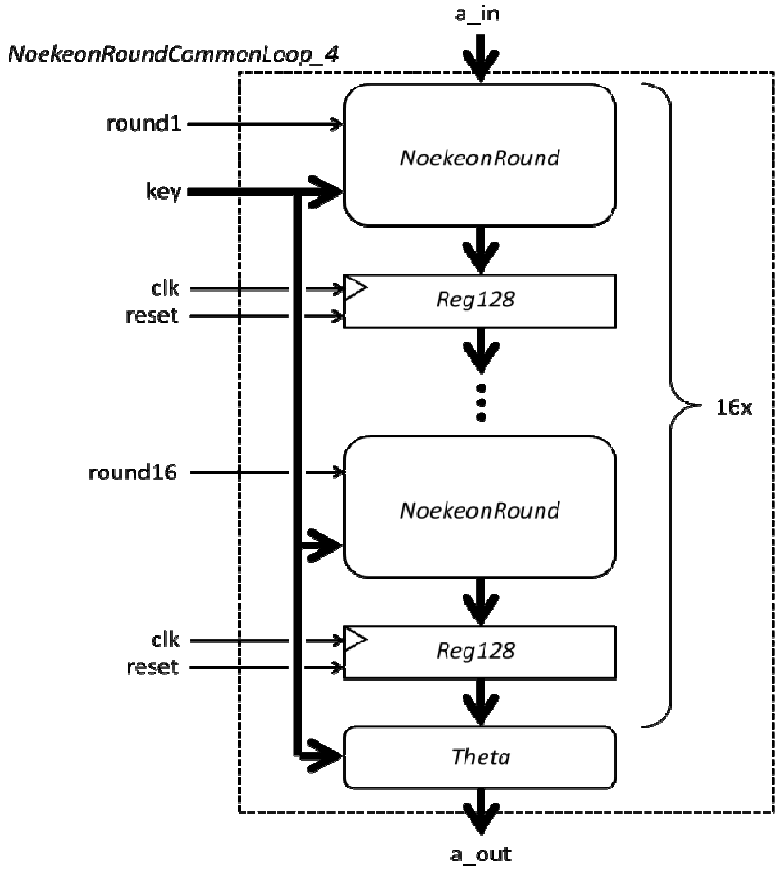
- *Fully unroll the loop*
  - Computing all rounds in a single clk cycle
  - I do not really need a control unit now!
  - How much faster do I get now?
    - Is this interesting?
    - What happens to the clock frequency?



Tutorial Tasks

Fully unroll the loop - add pipelining

- *Fully unroll the loop with pipeline*
  - Computing all rounds in 16 clk cycles
    - But each clk cycle I get a result
  - How much faster do I get now?
    - What is the resulting latency?
    - What happens to the clock frequency?
  - *Interesting or not?*

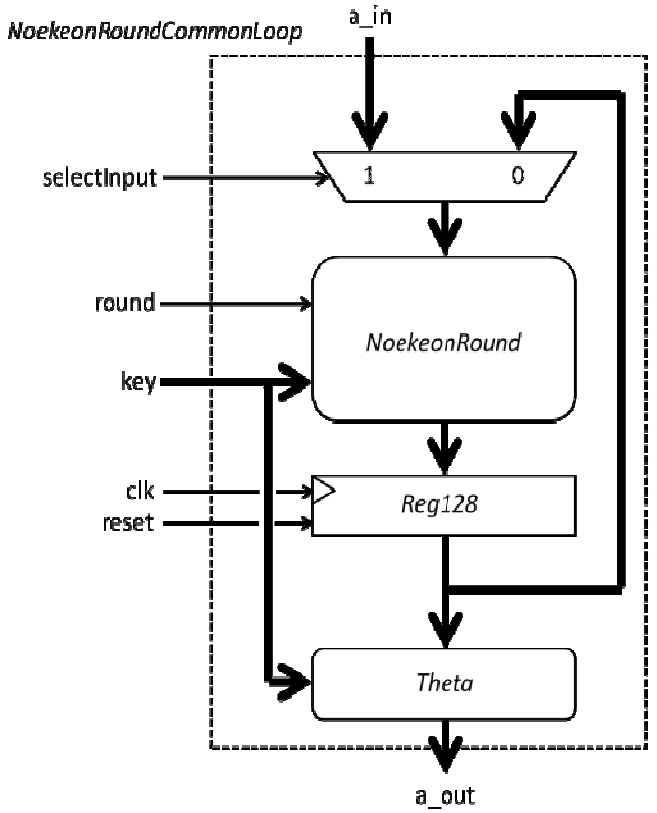


Tutorial Tasks

Implementing decryption and round constants

Implementing decryption

- What changes?
  - Same exact structure
  - Round constants are different
    - RC2 in decryption = RC1 in the inverse order for encryption
  - The input key changes
    - key_decryption = Theta(key_encryption);
- Can the same structure be used to compute encryption and decryption?



Compute the round constants using a Linear Feedback Shift Register (LFSR)

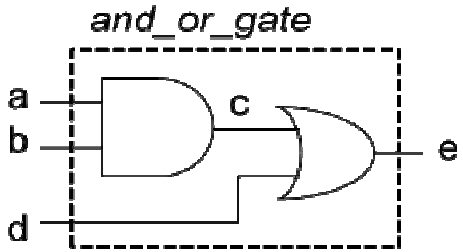
- RC[i+1] = RC[i]<<1;

Digesting VHDL

- Basic component
  - Define the component
  - Define an internal signal
  - Define its function

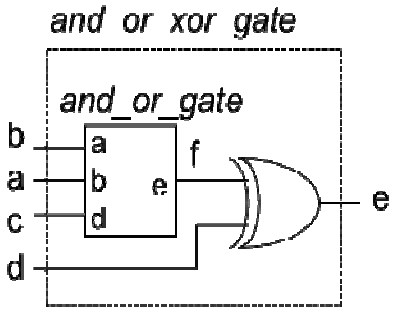
```
entity and_or_gate is
        port(       a, b, d: in bit;
                    e: out bit);
end and_or_gate;

architecture arch of and_or_gate is
        signal c: bit;
begin
        c <= a and b;
        e <= c or d;
end arch;
```



Digesting VHDL

- Basic component
  - Declare an existing component
  - Instantiate it



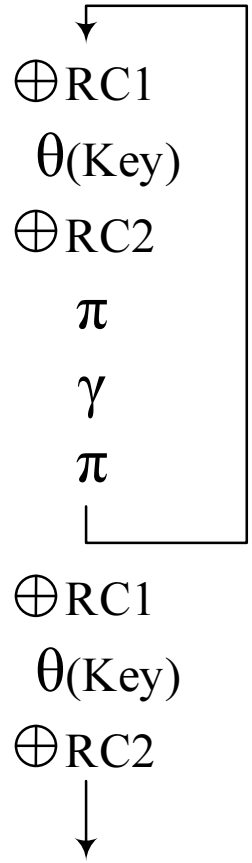

```
entity and_or_xor_gate is
      port(      a, b, c, d: in bit;
                 e: out bit);
end and_or_xor_gate;

architecture arch of and_or_xor_gate is
      component and_or_gate is
            port(      a, b, d: in bit;
                       e: out bit);
      end component;
      signal f: bit;
begin
      inst_and_or_gate: and_or_gate
            port map(      a => b,
                           b => a,
                           d => c,
                           e => f);
      e <= d xor f;
end arch;
```

Noekeon particularities

- For encryption:
  - RC2 = 0
  - RC1 = varies, different value per round
    - 801B366CD8AB4D9A2F5EBC63C697356AD4
- For decryption:
  - RC1= 0
  - RC2 = varies
    - D46A3597C663BC5E2F9A4DABD86C361B80

It's the inverse



- $key_{\_decryption} = \theta(key_{\_encryption})$;
- The rest is the same.

FPGA

The basics

- Configurable interconnection between configurable logic blocks.
- Main logic blocks are:
  - LUTs
  - Registers
  - Memory blocks
  - DSP
  - …