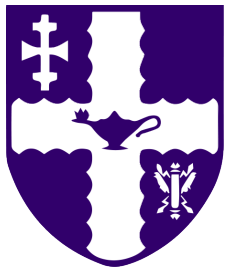


Loughborough University

20WSC054 – Electronic System Design with FPGAs Coursework

Luciano Ost

Deadline and basic information

- Coursework mark
  - **40%** of total module marks

- You are asked to submit **a ZIP file** including:
  - a report in **PDF**
  - the Quartus project including your scripts, VHDL and C codes
  - **Note**: organize your submission, e.g., software folder, RTL folder, waveforms, etc…

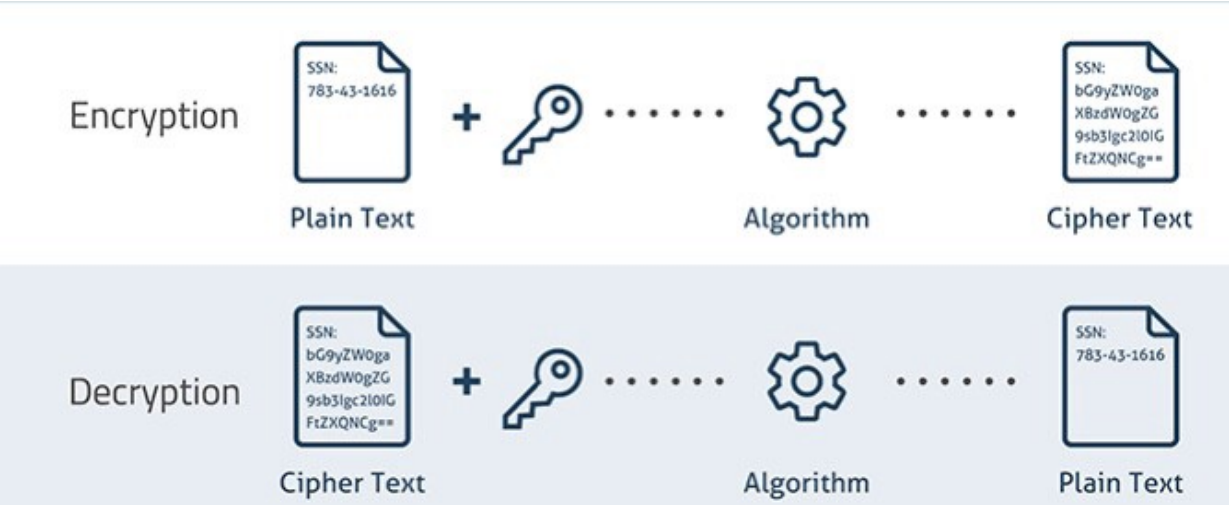
- How/where to submit?
  - you must upload your ZIP file on LEARN

- **Note**: It is your responsibility to submit a working file. If I can't read it, you don't get a grade.
  - So, the "corrupt file" trick will not work out.

Embedded Cryptography

Loughborough University

- Cryptography algorithms play a key-role in daily practices (e.g., e-payment, data exchange) in several sectors of society, including financial, healthcare and government institutions.

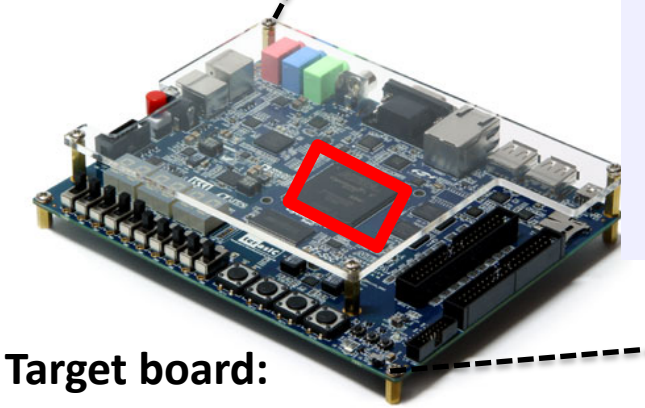


- The implementation of cryptography algorithms in low-level hardware designs presents a unique set of constraints (e.g., hardware and computation resources) and additional performance metrics (e.g., power consumption) to be optimised when compared to software solutions.

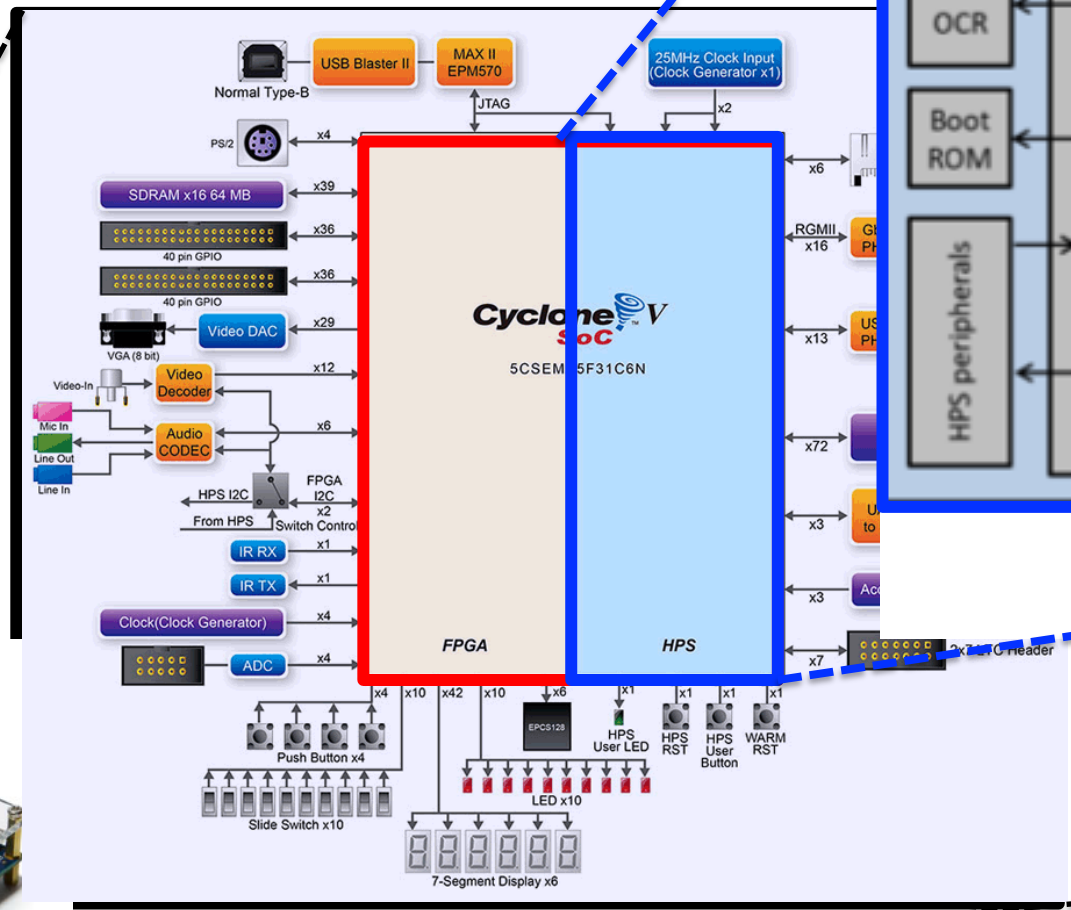
This assignment is designed to help you develop further your embedded and VHDL programming as well as writing skills.

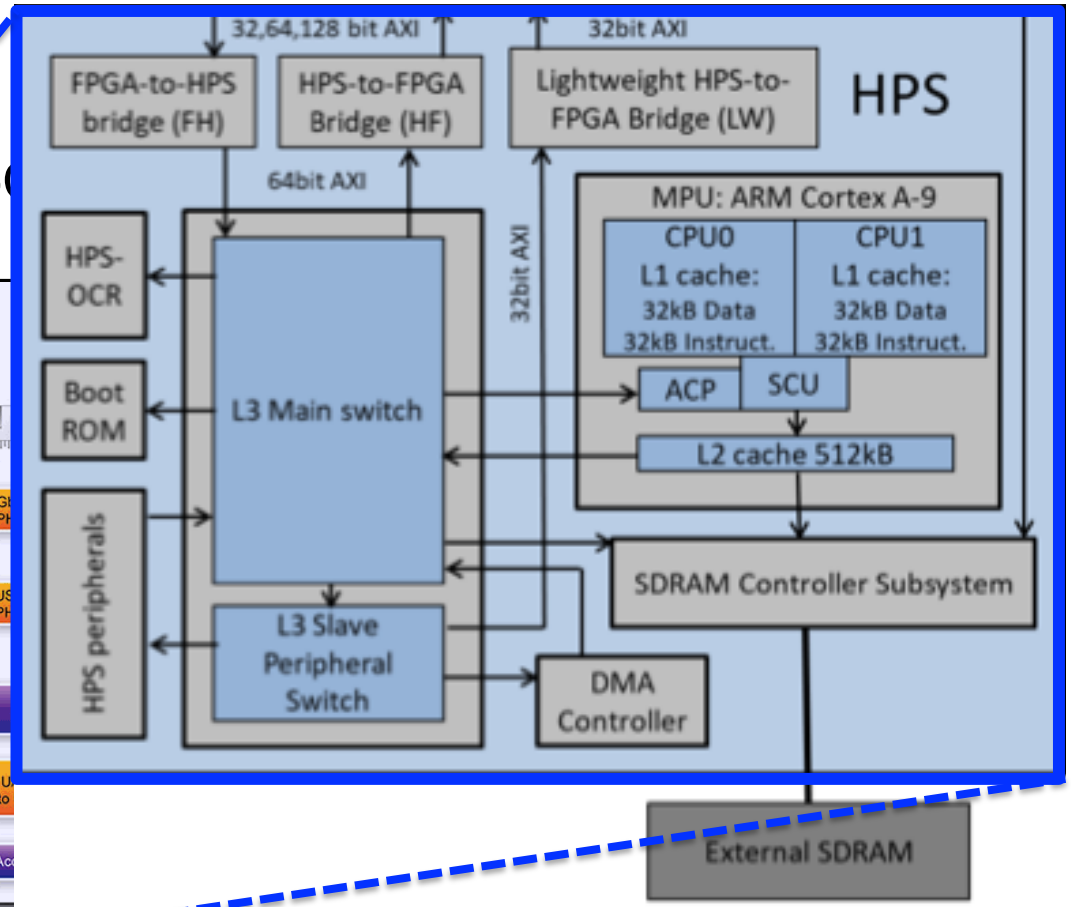
Goal

- Develop a hybrid cryptography system using the DE1-So



Target board:
DE1-SoC





Hybrid cryptography system

Loughborough University

Decryption module

32,64,128 bit AXI
32bit AXI
FPGA-to-HPS bridge (FH)
HPS-to-FPGA Bridge (HF)
Lightweight HPS-to-FPGA Bridge (LW)
HPS
64bit AXI
MPU: ARM Cortex A-9
CPU0 L1 cache: 32kB Data 32kB Instruct.
CPU1 L1 cache: 32kB Data 32kB Instruct.
ACP
SCU
L2 cache 512kB
HPS-OCR
Boot ROM
L3 Main switch
32bit AXI
HPS peripherals
L3 Slave Peripheral Switch
SDRAM Controller Subsystem
DMA Controller
External SDRAM

Full application

encryption function

Application (e.g., Fibonacci)

Typical

Application
OS
Drivers
Hardware

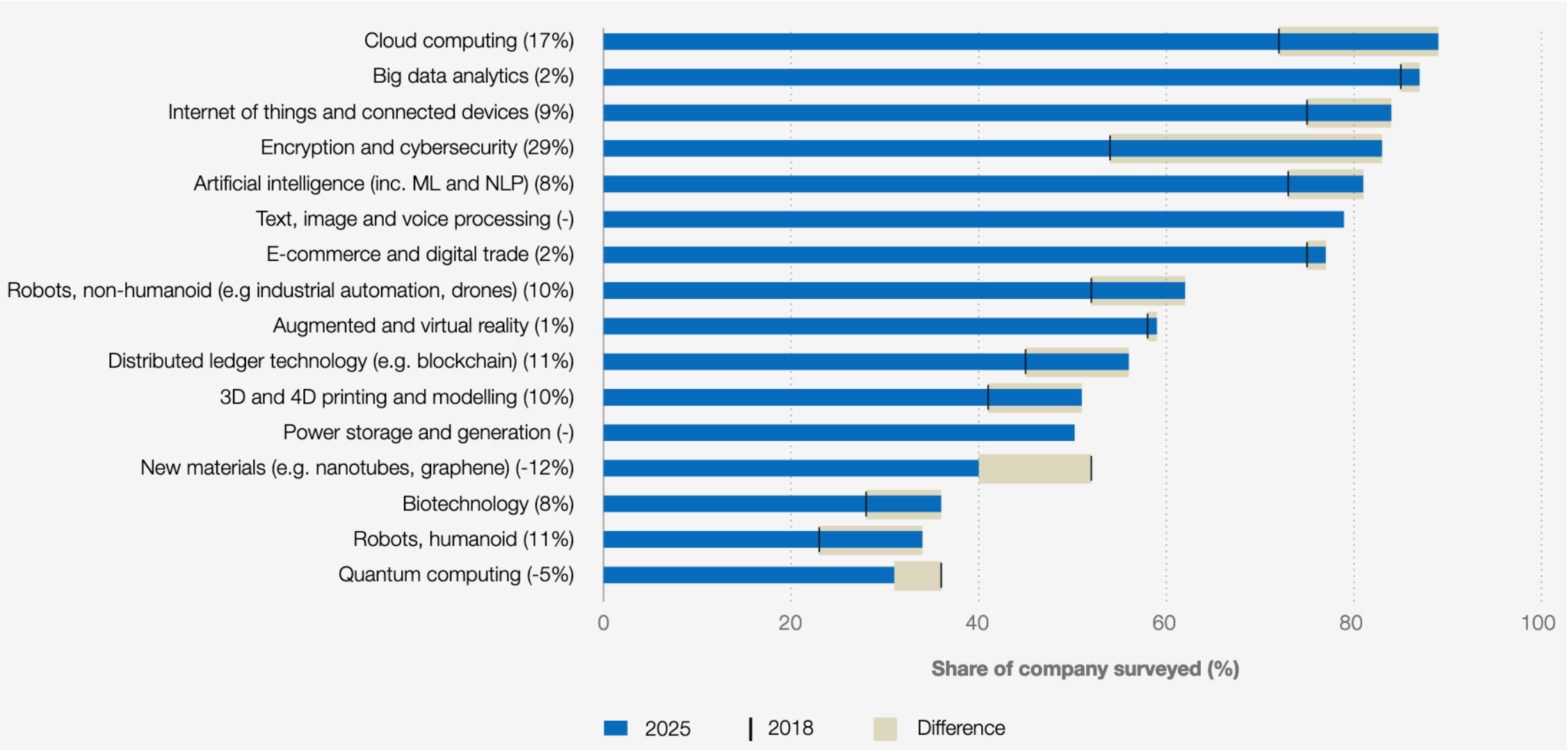
Bare-Metal

Application
HWLibs
Hardware

Software stack

Loughborough University

Technologies likely to be adopted by 2025 (by share of companies surveyed)



Source: Future of Jobs Survey 2020, World Economic Forum.

eXtended Tiny Encryption Algorithm (XTEA)

Loughborough University

- Basic description:
  - XTEA is a symmetric block cypher consisting of exclusive-or, addition, and shift operations. Due to its simplicity, it can be used in resource-constrained environments. Furthermore, XTEA is considered a Feistel cypher, i.e., encryption and decryption operate similarly, and both iterate a round function a fixed number of times.

- XTEA - extra material:

https://ieeexplore.ieee.org/document/8102419

https://link.springer.com/content/pdf/10.1007%2F978-3-540-89754-5_28.pdf

https://www.semanticscholar.org/paper/Hardware-Implementation-of-XTEA-Aumack-Koontz/5a008a529ccb8e76755a012131ea1365629c6223

https://dl.acm.org/doi/abs/10.1145/2435264.2435326

https://m.scirp.org/papers/79197

- Pseudo code for XTEA encryption and decryption,
- System design using Nios II processor,
- etc.

eXtended Tiny Encryption Algorithm (XTEA)

Loughborough University

- Reference C code implementation **[1].**

```c
#include <stdint.h>
#include <stdio.h>

void xtea_enc(void *dest, const void *v, const void *k) {
    uint8_t i;
    uint32_t v0=((uint32_t*)v)[0], v1=((uint32_t*)v)[1];
    uint32_t sum=0, delta=0x9E3779B9;
    for(i=0; i<32; i++) {
        v0 += ((v1 << 4 ^ v1 >> 5) + v1) ^ (sum + ((uint32_t*)k)[sum & 3]);
        sum += delta;
        v1 += ((v0 << 4 ^ v0 >> 5) + v0) ^ (sum + ((uint32_t*)k)[sum>>11 & 3]);
    }
    ((uint32_t*)dest)[0]=v0; ((uint32_t*)dest)[1]=v1;
}

void xtea_dec(void *dest, const void *v, const void *k) {
    uint8_t i;
    uint32_t v0=((uint32_t*)v)[0], v1=((uint32_t*)v)[1];
    uint32_t sum=0xC6EF3720, delta=0x9E3779B9;
    for(i=0; i<32; i++) {
        v1 -= ((v0 << 4 ^ v0 >> 5) + v0) ^ (sum + ((uint32_t*)k)[sum>>11 & 3]);
        sum -= delta;
        v0 -= ((v1 << 4 ^ v1 >> 5) + v1) ^ (sum + ((uint32_t*)k)[sum & 3]);
    }
    ((uint32_t*)dest)[0]=v0; ((uint32_t*)dest)[1]=v1;
}
```

```c
void main()
{
    uint32_t input[2];
    uint32_t key[4];
    uint32_t output[2];

    input[1] = 0x89646238;
    input[0] = 0x12300325;

    key[3] = 0xABCDEFAB;
    key[2] = 0xCDEFABCD;
    key[1] = 0xEFABCDEF;
    key[0] = 0xABCDEFAB;

    xtea_enc((void*)output, (void*)input, (void*)key);

    printf("%08x\n", output[1]);
    printf("%08x\n", output[0]);
}
```

[1] R. M. Needham and D. J. Wheeler, “Tea extensions,” University of Cambridge, Cambridge, UK, 1997.

eXtended Tiny Encryption Algorithm (XTEA)

Loughborough University

- Possible hardware implementation

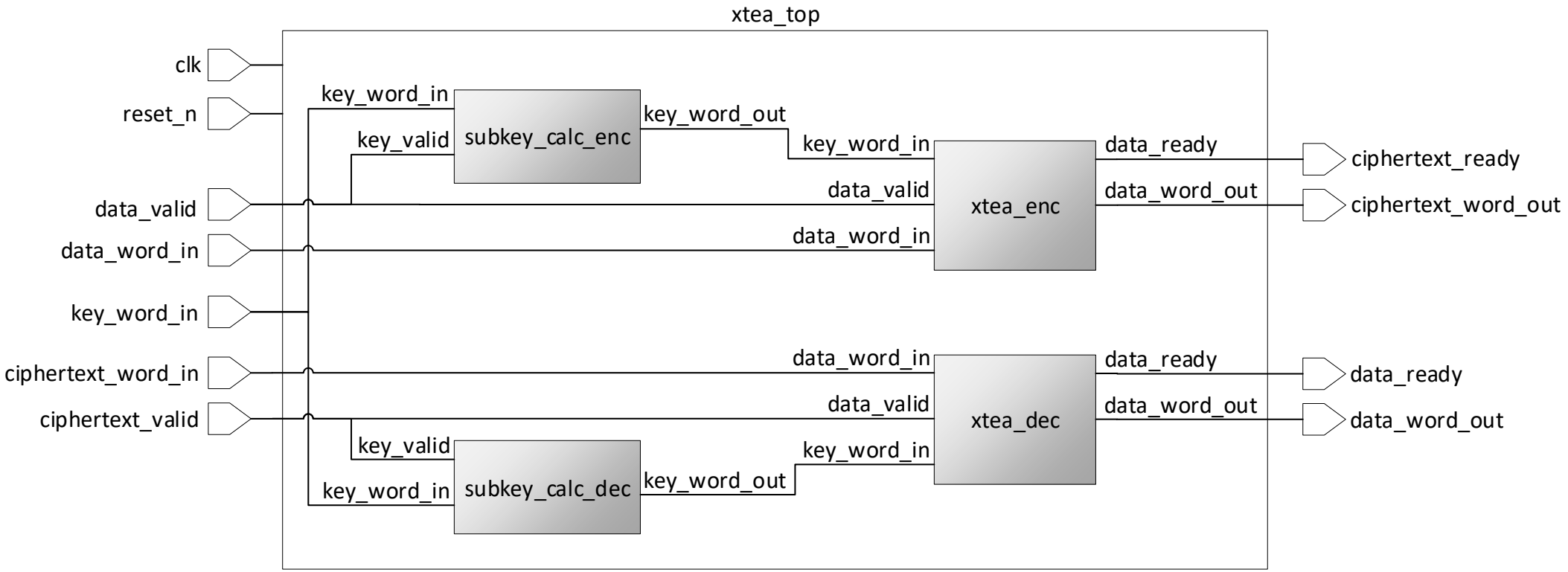
Whilst the original definition of XTEA does not specify any part of the calculation as the subkey, it can be described as the portion of the calculation that only relies on the key and not the input data. In this case, looking at lines 7-9 of the reference C implementation **[1].**

```
7   y+= (z<<4 ^ z>>5) + z ^ sum + k[sum&3],
8   sum+=DELTA,
9   z+= (y<<4 ^ y>>5) + y ^ sum + k[sum>>11 &3] ;
```

The sum + k[sum&3] and sum + k[sum>>11 &3] calculations can be considered the subkey calculations, and thus can be calculated independently of the encryption and decryption process. In addition to this, the value of the *sum* variable, which is updated every round, is only used when performing these subkey calculations. This means the subkey values for every round can be independently calculated, since *DELTA*, the value used to modify the *sum* variable, is a predefined value. This is used to reduce the calculation time by pre-calculating the subkey values one clock cycle before they are needed by the encryption/decryption process. (example of partial solution explanation)

eXtended Tiny Encryption Algorithm (XTEA)

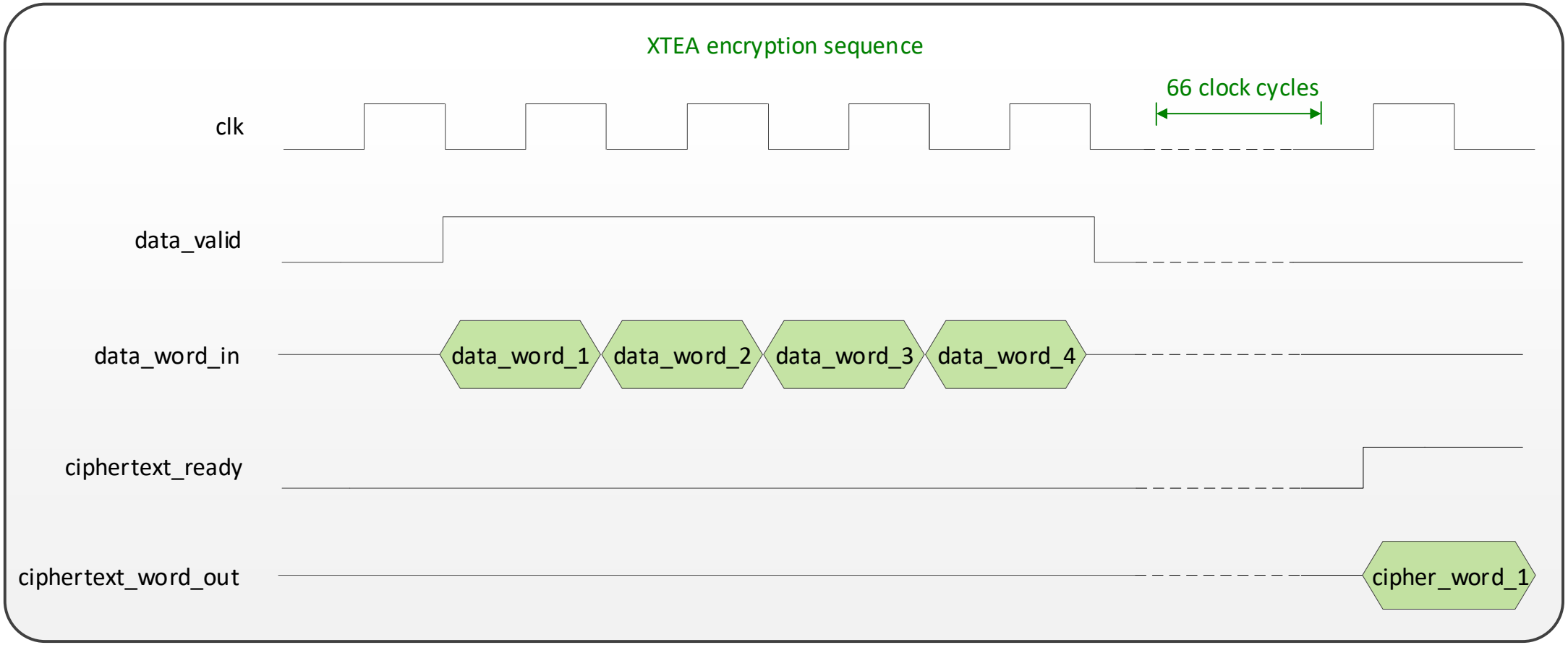
- Possible hardware implementation



Internal layout of duplex XTEA top-level, including inputs/outputs, internal blocks and their connections

eXtended Tiny Encryption Algorithm (XTEA)

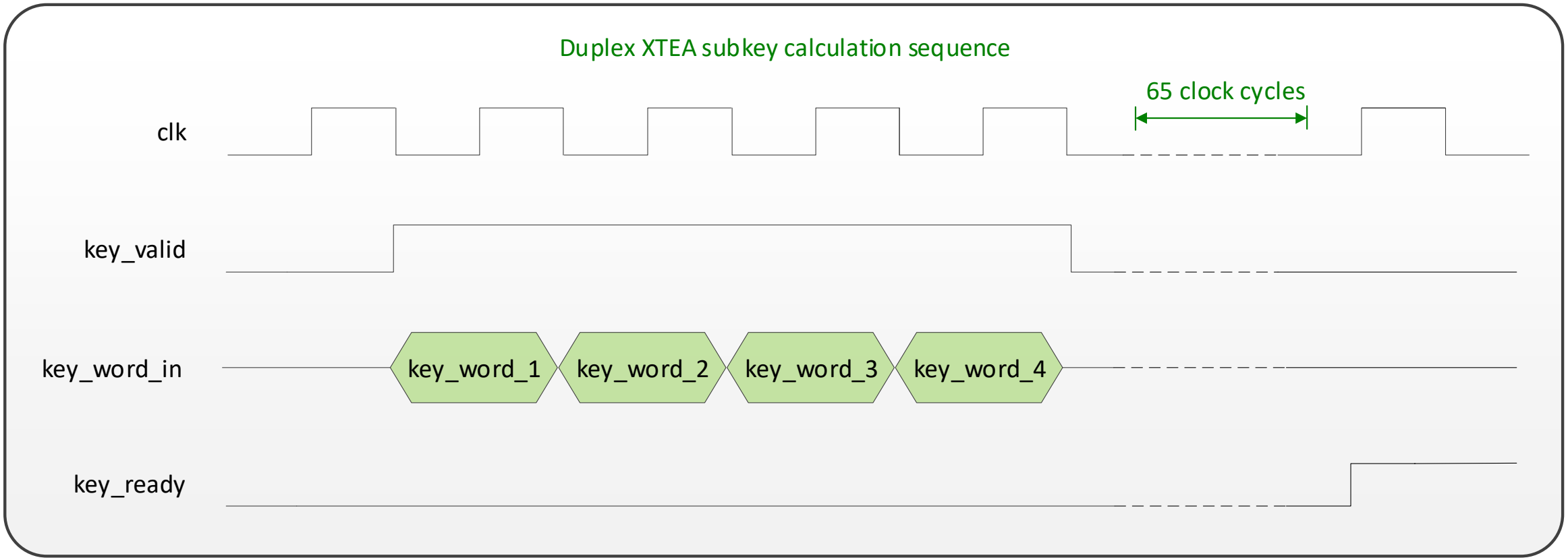
- Possible hardware implementation



eXtended Tiny Encryption Algorithm (XTEA)

Loughborough University

- Possible hardware implementation



Note: simplex XTEA implementations will be accepted

Description

Loughborough University

- To accomplish the coursework goal, the following specific objectives should be fulfilled:
  - understand the underlying problem / target system;
  - understand the XTEA reference implementation described in C;
  - come up with a reasonable design solution;
  - write the VHDL code that implements your solution;
  - create a testbench to validate your cryptography solution;
  - collect and explain the resulting waveform;
  - Write a report.

- What I will be looking at:
  - amount of information
  - quality of discussion
  - quality of you code and solution
  - application of discussed theoretical topics and practical development
  - references
  - document structure and presentation

Hybrid cryptography system requirements:

Loughborough University

- Your must consider the following functionalities:
  1. You shall execute one or more application(s) in the Cortex-A9. **Note**: **both bare-metal and Linux-based solutions will be accepted** but, in my opinion, Linux is better vs bare-metal. The overhead of using Linux is negligible for most applications, mainly when considering processors with reasonably computational capacity, which is the case. Further, Linux is highly used in several domains, opportunity to learn, etc. Furthermore, the use of Linux might facilitate your development as well, but yes, you should consider the learning curve. An interesting paper related to above is below:

  https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01245-optimize-real-time-performance-with-altera-socs.pdf

  2. The output of your application(s) must be encrypted in **software** and decrypted in **hardware** with your XTEA solution;
  3. The decrypted application output must be printed through the UART or Ethernet interface;
  4. You shall validate your XTEA solution against the reference C code. **Note**: your solution can be a simplex or a duplex XTEA implementation.
  5. You must use the provided testbench to validate the RTL implementation of the XTEA algorithm.
  6. You shall present FPGA resource utilization and implementation results;
  7. You must demonstrate the whole cryptography system operation. Feel feel to create one or more case studies.
  8. Design decisions must be justified in the report.

  Optional:

  1. You may provide individual validations for each module or subsystem, e.g., application + encryption / (app+encry) + A9
  2. You may compare the full RTL XTEA solution against the hybrid one. The comparison could, for instance, consider performance of both approaches.

Report format

Loughborough University

- IEEE or the LU formats are recommended
- You should refer to the Report Writing "Handbook" for Students:
  https://www.lboro.ac.uk/services/library/students/learningsupport/topics/reportwriting/

- Possible structure:
  - Introduction
  - Hybrid cryptography system development
    - I would split the content in subsections, which can be organized according to the system's modules
    - remember to justify your implementation decisions as well as the limitations of your design (if any of course)
    - Feel free to use figures/diagrams etc
  - Hybrid cryptography system validation
    - I would describe individual validations for each module or subsystem
    - Full system validation
  - Conclusion

Report format: possible introduction structure

Loughborough University

- Introduction
  - Goal and objectives
  - Structure of the report
- Goal
  - The aim of this report is…
  - which are the objectives?
  - well, thing about it
- Structure of the report
  - Chapter 2 describes ….
  - Chapter 3…
- Remember
  - Feel free to include figures, tables, equations, or whatever you believe you can use to pass the message…

Marking criteria

Loughborough University

Software: (10%)				software + hardware (HPS) : (25%)				
compilation and execute correctly (2%)	allow to set different applications (2%)	encryption + app (4%)	comments (2%)	Linux or bare-metal (2%)	correct execution (8%)	UART / debug support (5%)	performance (10%)	division of workload between cores? (XXX%)

HPS + XTEA : (45%)					
Code quality and comments (5%)	Code functionality / problem solution (10%)	Allows change in parameters (multiple key/data pairs, etc) (3%)	area and performance evaluation (15%)	validation: simulation + hardare (12%)	extras (XXXX%)

Note: The above marking criteria is a preliminary version, and it might suffer a few modifications ☺