Embedded Systems Design

First Assignment

ESD - First Assignment

- First assignment included 4 parts:
  - 1.1) RTL Design
  - 1.2) TLM Design
  - 1.3) AMS Design
  - 1.4) Heterogeneous Platform (RTL + TLM + AMS)

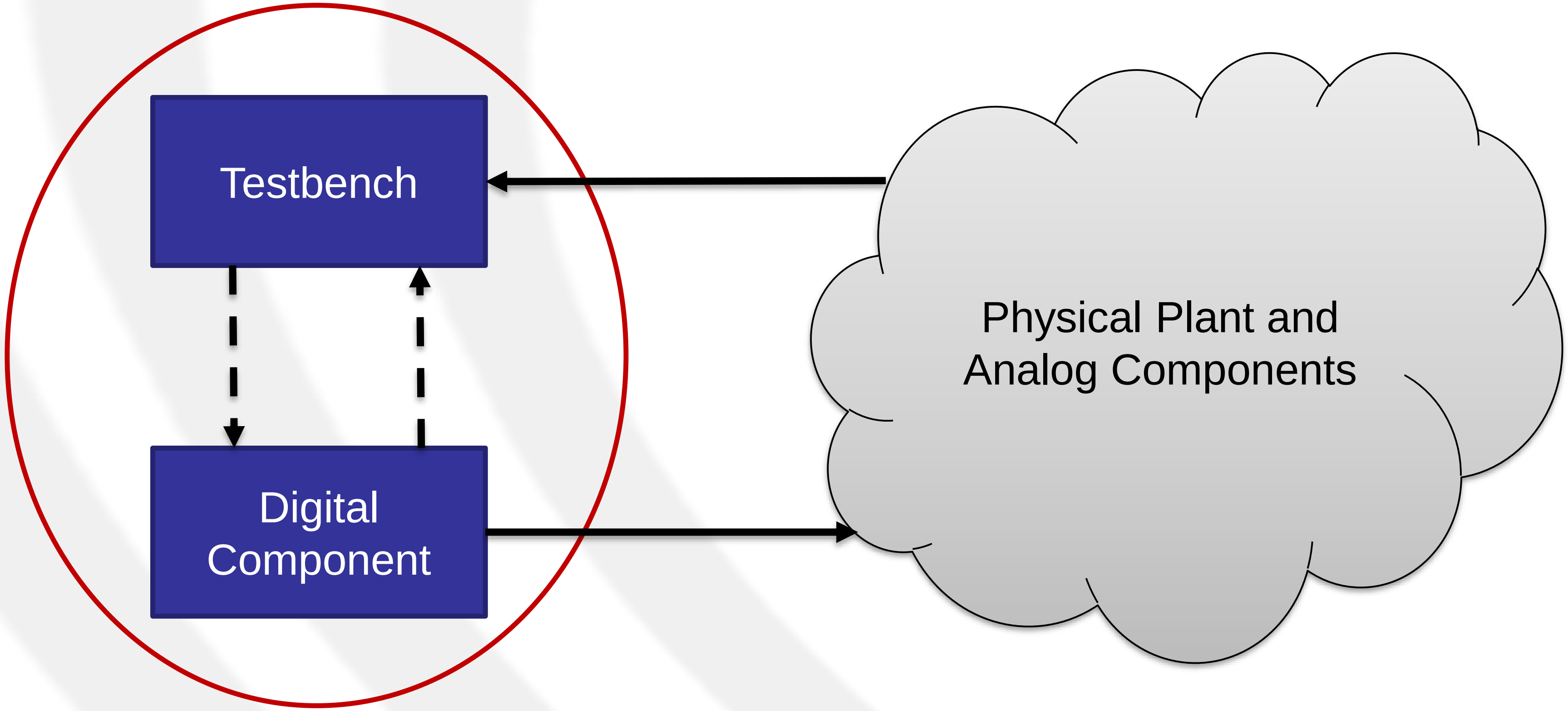
Delivery deadline:

Sunday, 30th December 23:59

Embedded Systems Design - Assignment

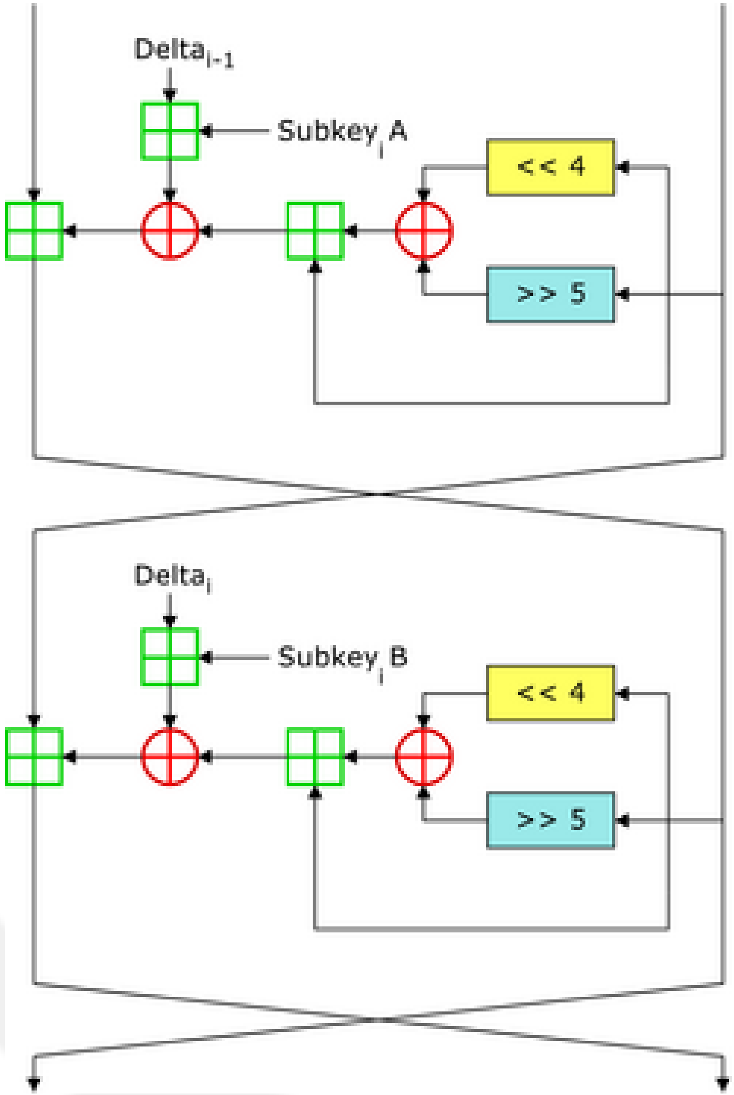
1.1 - RTL Modeling

1.1 - SystemC RTL Modeling



XTEA: eXtended Tiny Encryption Algorithm

- Simple block cipher
  - Characteristcs
    - Operates on 64-bit blocks
    - Uses a 128-bit key
    - 32 rounds made of two Feistel rounds each
      - Different multiples of a magic constant (delta(are used to prevent simple attacks based on the symmetry of the rounds
  - Wheeler and Needham, 1997
  - Stronger implementation of the TEA algorithm



C++ Implementation

INPUTS

```
void xtea (uint32_t word0, uint32_t word1, uint32_t key0,
      uint32_t key1, uint32_t key2, uint32_t key3,
      bool mode,
      uint32_t * result0, uint32_t * result1) {
```

OUTPUTS

```
 uint64_t sum;
 uint32_t i, delta, v0, v1, temp;
 v0 = word0;  v1 = word1;
 sum = *result0 = *result1 = 0;
```

ENCIPHER

```
 if(mode == 0) {
   // encipher
   delta=0x9e3779b9;
   for (i=0; i < 32; i++) {
   ... // Operations to determine the value of temp
     v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + temp);
  ...  // Operations to determine the value of temp
     v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + temp);
   }
 }
```

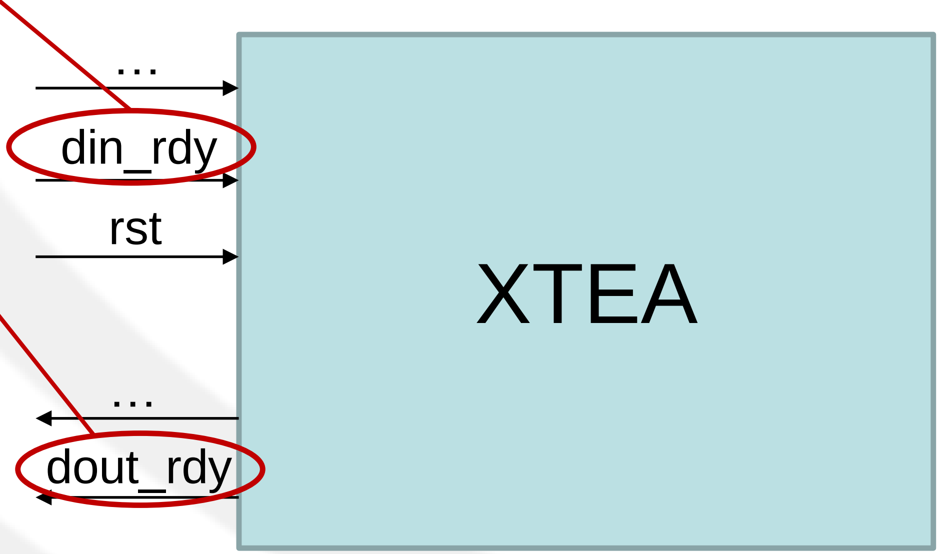
```
 else if (mode == 1) {
   // decipher
```

DECIPHER

```
   delta = 0x9e3779b9;
   sum = delta*32;
   for (i=0; i<32; i++) {
   …// Operations to determine the value of temp
     v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + temp);
    …// Operations to determine the value of temp
     v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + temp);
   }
 }
 *result0 = v0;
 *result1 = v1;
}
```

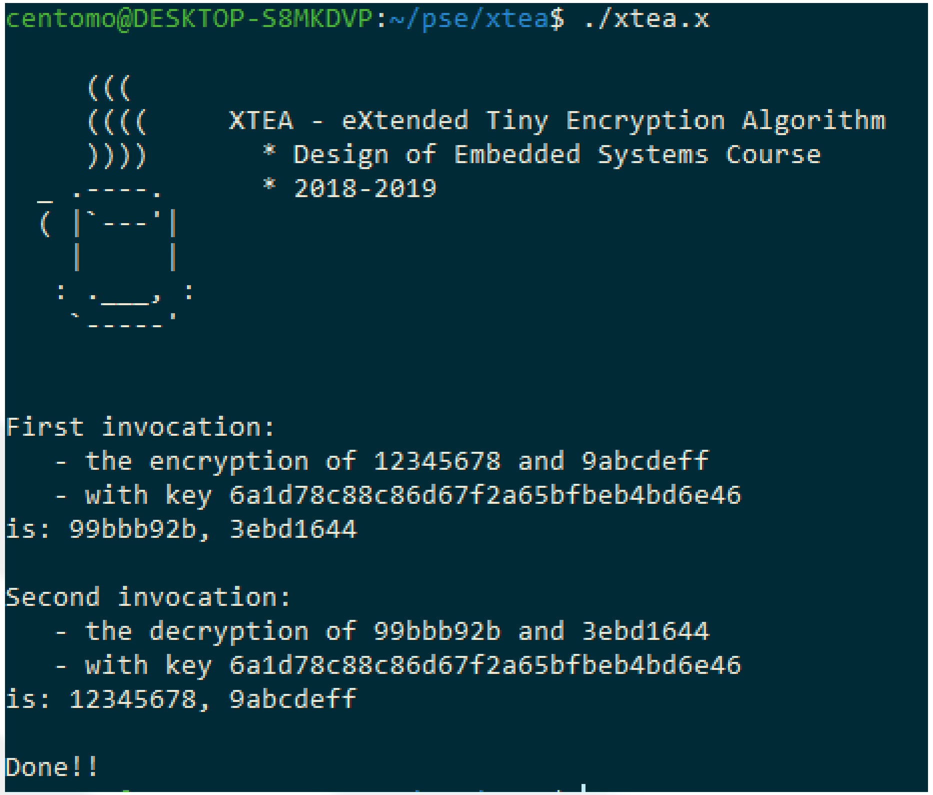
XTEA Design

- Implement the function of the previous slide
- Handshaking protocol
  - Ready flag
    - Inputs are ready, start computation
  - Done flag
    - Outputs are ready, testbench can read the results



XTEA Design (Cont.d)

- XTEA implementation
  - XTEA.cpp in moodle platform
  - Test different inputs/outputs on the algorithm to verify the correctness of your implementation
  - Example of launch on the assigned configurations:



```
centomo@DESKTOP-S8MKDVP:~/pse/xtea$ ./xtea.x

     (((
     ((((     XTEA - eXtended Tiny Encryption Algorithm
     ))))       * Design of Embedded Systems Course
  _ .-----.     * 2018-2019
  ( |`---'|
    |     |
   : .___, :
    `-----'


First invocation:
   - the encryption of 12345678 and 9abcdeff
   - with key 6a1d78c88c86d67f2a65bfbeb4bd6e46
is: 99bbb92b, 3ebd1644

Second invocation:
   - the decryption of 99bbb92b and 3ebd1644
   - with key 6a1d78c88c86d67f2a65bfbeb4bd6e46
is: 12345678, 9abcdeff

Done!!
```

1.1 - Final Report

- Implement the assigned design at RTL
- What the report should contains for this lesson:
  - Interface definition
  - FSMD and EFSM of the device behavior
    - For the FSMD schema, it is enough to show the connections between Controller and Datapath (not the internal implementation schema)
    - The EFSM in the report **has to be consistent** with the SystemC implementation and the FSMD schema!
      - Remember: define the EFSM and FSMD **before** implementing the design!
  - Organization of the processes
    - 2 or 3 processes, divide controller from datapath
    - Signals for internal communication
    - Describe process dependencies and sensitivity lists
  - Describe the main features of the design
    - How do you organize and use the internal signals? What about the latency?
  - Justify ALL your choices:
    if something is not on the specification, then **write it on the report**!

Embedded Systems Design - Assignment

1.2 - TLM Modeling

1.2 SystemC TLM Modeling

- Implement the XTEA Encipher/Decipher at TLM
  - UT/LT/AT4 coding styles
  - **Remember:** you are now modeling the functionality at a higher level of abstraction

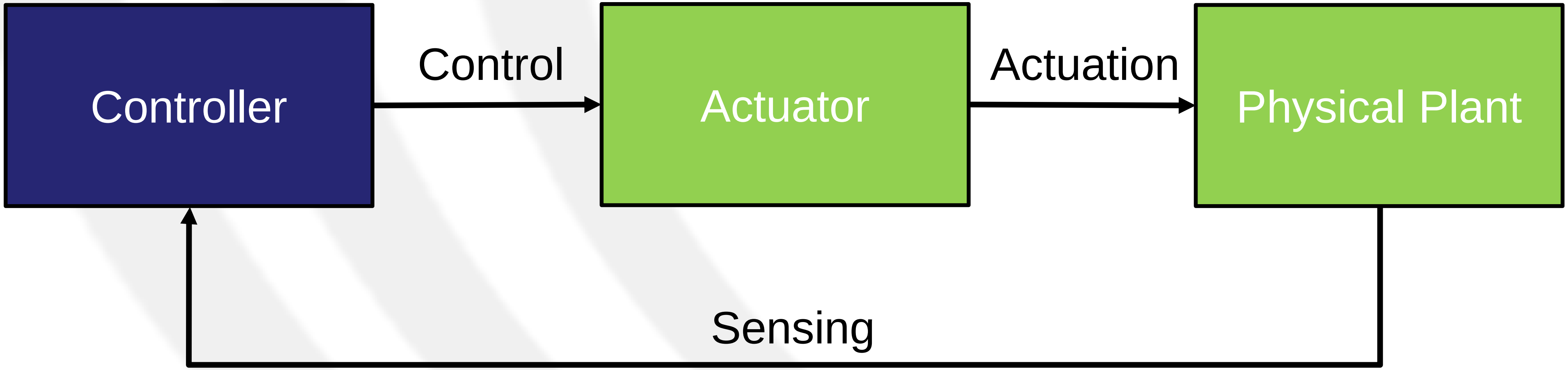
- Report:
  - **Briefly** explain each implementation
    - **How** do you implement the functionality?
    - **How** do you implement the protocol, and how it behaves?
  - Compare the simulation overhead…
    - …of the TLM implementations…
    - … and w.r.t. the RTL execution…
    - …and comment what you see!
      - Hint: This should let you understand why/when/where TLM is used

	UT	LT	AT4	RTL
Real time				
User time				
System time				

Embedded Systems Design - Assignment

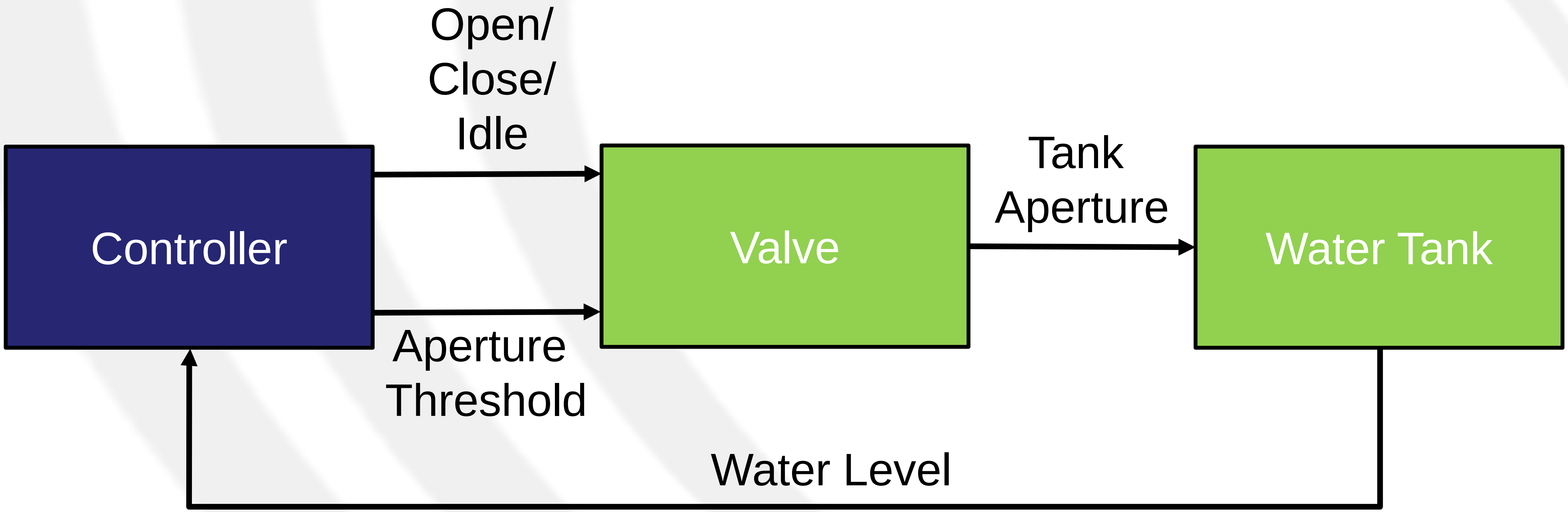
1.3 - AMS Modeling

1.3 - Modeling a Heterogenous system



Classic feed-back control System

1.3 - The WaterTank system



1.3 - Water Tank and Control

- **Requirement**: water level between 5 and 8.8
- Water Level x is a function of the valve aperture
  - $\dot{x} = 0.6 * a - 0.03 * x$
- **The controller is monitoring the water level**
  - If the water level is between 5 and 8.8:
    - Send the valve the command **IDLE**
  - If the water level is greater than 8.8:
    - 1) Modify every 5 seconds the valve aperture threshold t as follows:

      $$t = t * 0.7$$
    - 2) Send the valve the command **CLOSE**
  - If the water level is lower then 5:
    - 1) Modify every 5 seconds the valve aperture threshold t as follows:

      $$t = t * 1.1$$
    - 2) Send the valve the command **OPEN**

1.3 - Valve

- Input:
  - Commands
    - IDLE, OPEN, CLOSE
  - Aperture Threshold
    - Maximum value of the valve aperture (the miminum value **is 0)**
    - Initial Value: **0.7**

- Output:
  - Valve Aperture
    - Input for the Water Tank

- Behavior:
  - The valve aperture a is a function of time and command
    - If the command is IDLE: $\dot{a} = 0$
    - If the command is OPEN: $\dot{a} = 0.25$
    - If the command is CLOSE: $\dot{a} = -0.25$
  - The valve aperture is **always between 0 and the threshold!**

1.3 - Requirements

- Implement the Valve and the Water Tank using System-AMS
  - They must use two different modeling formalisms
  - They must be connected using SystemC-AMS interfaces
  - The controller can be implemented as you prefer
    - SystemC-AMS
    - SystemC-RTL
    - C++ function wrapped within a SystemC model
    - Etc…

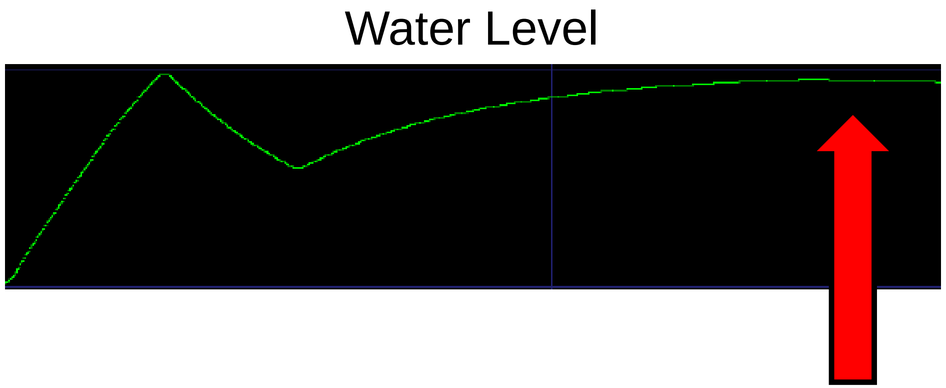
- The controller is slower than the sampling time of plant
  - Communication between controller and valve happens less often
  - 1 command from the controller, N samples of water tank and valve
    - Controller may ignore samples
    - Alternatively: Controller may use all the samples in smarter ways

1.3 - Final Report

- Justify the choices that you take:
  - Concerning the choice of the MoCs
  - Concerning the structure of the system
  - Describe the implementation
    - Modules, components, signals
    - Components interfaces
    - Control implementation
    - Etc…

- Check the execution:
  - Plot the system execution
    - Identify after how much time the system stabilizes
    - Gnuplot or tracing
      - Maybe easier using gnuplot!

You should obtain this behavior:

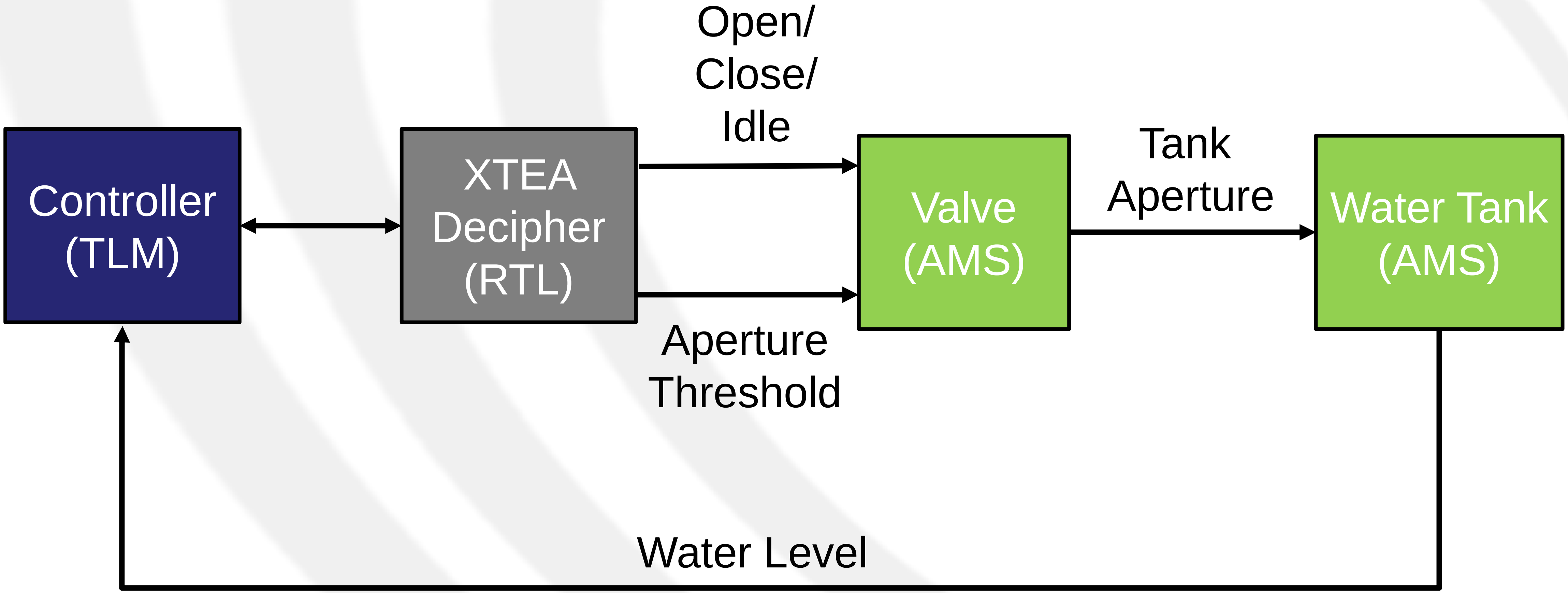


Water Level stabilization

Embedded Systems Design - Assignment

1.4 - Heterogeneous Platform

1.4 - The WaterTank system: Schematic



1.4 - Requirements

- RTL/TLM Transactors
  - Reuse the RTL implementation of the XTEA design
  - Reuse the TLM testbench for the XTEA design
  - Connect them using a transactor
- TLM/AMS transactors
  - Reuse the Tank and Valve design
  - Connect the Valve and the Tank to the Testbench of the multiplier
- The TLM Testbench implements the controller of your water tank system
- The Controller send cipher data to the RTL XTEA model
- The RTL XTEA decipher the data and send the results to the Valve

1.4 – Final Report

- Report:
  - Mapping between TLM payload and RTL ports
  - Translation between TLM functions and RTL sequence of operations
    - What do you associate with a read transaction?
    - What do you associate with a write transaction?
  - Description of the transaction behavior
    - For the AMS design: how do you propagate the values?
    - What processes are you using? Why?

Final Structure of the Assignment

- VR123456_Stefano_Centomo_01/
  - Report/
    - VR123456_Stefano_Centomo_01.pdf
  - Solutions/
    - HW_Subsystem/
      - RTL/
        - bin, include, Makefile, obj, src
      - UT/
        - bin, include, Makefile, obj, src
      - LT/
        - bin, include, Makefile, obj, src
      - AT4/
        - bin, include, Makefile, obj, src
    - Continous_Subsystem/
      - bin, include, MakeFile, obj, src
    - Heterogeneous_Platform/
      - bin, include, Makefile, obj, src

- **Everything within VR123456_Stefano_Centomo_01.tar.bz2**

Source Code delivery

- For every exercise create the following directories:
  - *src/* containing the source code files
  - *include/* containing the header files
  - *obj/* used to store the object files generated by the compilation
  - *bin/* to store the executable file

- If any parameter is required by the executable
  - Create a README file
  - Handle the parse line for the parameters and print the usage

- Makefile
  - **Use a Environment Variables called *SYSTEMC_HOME and SCAMS_HOME***

- ***IF IT DOES NOT COMPILE ON MY COMPUTER, THE ASSIGNMENT WILL BE DISCARDED AUTOMATICALLY!***
  - ***Same configuration of the laboratory Computers! Check your implementation on it!***

Assignments not properly delivered will not be corrected, and the report will not be considered!

Consegne che non rispettano le regole descritte sopra non verranno corrette e il report verrà considerato non consegnato!