SystemC Polynomial

MICROELECTRONIC
SYSTEMS DESIGN
RESEARCH GROUP

FPGAHS Lab - SystemC Polynomial
Dr.-Ing. Christian De Schryver

Goals

- Build a new SystemC module from scratch.
- Implement a mathematical algorithm in SystemC.
- Use this module to create a hierarchical module.
- Implement a sequential module in SystemC.
- Develop the test bench for a sequential module.

Setup

A template code is provided on the GitHub system in the repository:

```
polynomial.systemc
```

Please clone this repository to a working directory. You will find template code for this task and a Makefile there.

Part a: Polynomial Evaluation

Change into the subdirectory `polynom`. The folder contains the following files:

- `poly.h` contains the polynomial evaluation module.
- `poly.cpp` contains implementation details for the module in `poly.h`.
- `stim_polynom.h` generates the stimuli for the test run.
- `mon_polynom.h` reads the stimuli and the output of the module and displays the results.
- `main.cpp` specifies the executable program that combines all modules to a complete simulation.
- `Makefile` is a pre-defined config file for the make command that holds the settings for building this project.
- `polynomial.systemc.pro` is the project file for QT Creator. Open it with `qtcreator polynomial.systemc.pro &`.

Task Description

1) Provide a fully functional module in `poly.h` and `poly.cpp` that calculates the value of a polynomial $f(x) = \sum_{i=0}^{n}(\mathrm{a_i} * x^i)$ with a fixed degree and fixed coefficients a_i and variable x-coordinate. Use the C-type `double` for arithmetic calculations.
    a. Create a function in the file `poly.cpp` that implements the evaluation of the polynomial.

b. Create the module in `poly.h` that uses this function.

c. Integrate the module into the main function in `main.cpp`.

2) Build and run the project with QT Creator or using the command line and `make`. The correct output of your simulation should look like this:

```
        SystemC 2.3.1-Accellera --- Dec  1 2014 20:17:56
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED

Correct result for X = -5
Correct result for X = -4.9
Correct result for X = -4.8
[…]
Correct result for X = -0.3
Correct result for X = -0.2
Correct result for X = -0.1
Correct result for X = 0
Correct result for X = 0.1
Correct result for X = 0.2
[…]
Correct result for X = 4.6
Correct result for X = 4.7
Correct result for X = 4.8
Correct result for X = 4.9

Info: /OSCI/SystemC: Simulation stopped by user.
```

Part b: Polynomial Integral

The files for this part are located in the top folder of the repository. The folder contains the following files:

- `polyInt.h` contains the polynomial integral module.
- `polyInt.cpp` contains implementation details for the module in `polyint.h` if necessary.
- `polyInt_tb.h` defines the testbench module.
- `polyInt_tb.cpp` contains the implementation of the testbench.
- `main.cpp` specifies the executable program that combines all modules to a complete simulation.
- `Makefile` is a pre-defined config file for the make command that holds the settings for building this project.
- `systemc.polynomial.pro`is the project file for QT Creator. Open it with `qtcreator systemc.polynomial.pro&`.

Task Description

1) Provide a fully functional module in `polyint.h` and `polyint.cpp` that numerically approximates the definite integral of a polynomial. You are free to choose an algorithm of your choice.
   a. Specify a simple interface and communication protocol that includes control signals for transferring data between your module and the testbench.
   b. Implement a sequential module using the module from Part a as a submodule.

2) Create a Testbench that verifies the correctness of the module you implemented in 1).
   a. Implement an algorithm that reliably calculates the correct result.
   b. Use the specified protocol to create test stimuli for the module.
   c. Retrieve the results from the module and compare them to the correct result.
   d. Write the results of the test to the command-line.

3) Build and run the project with QT Creator or using the command line and `make`. Analyse the output manually and make sure the relative error is $<10^{-2}$. The output could look like this:

```
        SystemC 2.3.1-Accellera --- Dec  1 2014 20:17:56
        Copyright (c) 1996-2014 by all Contributors,
        ALL RIGHTS RESERVED

0 s Starting Test

0 s Input set: -3 to 0
 Calculated Result: 9.0135
 Correct Result: 9
 Relative Error: 0.0015005
 after 10020 ns

10020 ns Input set: -2 to 1
 Calculated Result: 3.0075
 Correct Result: 3
 Relative Error: 0.0025015
 after 20050 ns

20050 ns Input set: -1 to 2
 Calculated Result: 3.0075
 Correct Result: 3
 Relative Error: 0.0025015
 after 30080 ns

30090 ns Input set: 0 to -3
 Calculated Result: -9.0135
 Correct Result: -9
 Relative Error: 0.0015005
 after 40100 ns

40110 ns Input set: -1 to -4
 Calculated Result: -21.0225
 Correct Result: -21
 Relative Error: 0.00107164
 after 50120 ns

50130 ns Input set: -2 to -5
 Calculated Result: -39.0315
 Correct Result: -39
 Relative Error: 0.000807808
 after 60140 ns

Info: /OSCI/SystemC: Simulation stopped by user.
```

Questions

- What does the structure of a complex SystemC module look like?
- What are the advantages using SystemC in comparison to VHDL?
- What is the difference between combinational and sequential modules?
- How are sequential designs described in SystemC?
- What is needed to transfer data between sequential modules?