**You are tasked with coding a functional model of a hardware circuit that will approximate trigonometric function(s).**

**Task I.Modeling:Octave/ MATLAB**

1. Quadratic Interpolator – familiarize yourself with the following publications:

<https://www.mdpi.com/2079-9292/5/2/17>

+ some Jean-Michel Muller positions…

(*see Figure 3. Quadratic interpolator block diagram*)

1. Read through and understand the attached script *quadratic.m* for generation of approximation coefficients of the following function:

**f(x) = sin(2x – pi/4) for x in range: [0,2)**

**Task II.Modeling:Cmodel (C++)**

1. Make yourself comfortable with the attached model of f(x) function approximation.   
     
   **y[x] = a\*x2 + b\*x + ca, b, c**= attached coefficient with (draft) accuracy **Δ < 5.25e-6**   
   x = 24 bit input (u1.23)   
   y = output  
   1. Modify the attached Makefile according to your needs.
   2. Be aware of use of **bit-accurate data types - *Algorithmic-C Datatypes***in the model. Download it form [here](https://github.com/hlslibs/ac_types) and specify the path in the Makefile.
   3. To enable testbench logging, please modify DBG\_\* defines.
   4. e (attached) pre-calculated **LUT** (draft) coefficient: **a, b, c**where M = 7 == 128 entries selected by x[23:17]  
      ( *your Task V. is to adjust the size of coefficient* ***a, b, c*** *to optimize the design area* )  
      
   5. be mindful of the algorithm’s suitability for **hardware implementation**   
      (ASIC or FPGA).
2. Verify your approximation functions against IEEE double-precision floating point calculations on a CPU or GPU.
   1. Check accuracy **Δ = fref(x) (@FP DP) – yimpl[x] < 5.25e-6**

**Task III. Design: RTL model (Verilog)**

1. Implement the above function **using SystemVerilog**.
2. Use Intel® Quartus® Prime Lite Edition Design Software, Version 20.1 (or 23.1):
   1. Intel® Quartus® Prime Lite Edition Design Software, Version 23.1<https://www.intel.com/content/www/us/en/software-kit/795188/intel-quartus-prime-lite-edition-design-software-version-23-1-for-windows.html>
   2. more info on FPGA Academy:  
      <https://fpgacademy.org/tools.html>

**Task IV. Simulation: Verilator + GTKWave**

1. **Simulate / verify your approximation functions**
   1. *Instruction how to install and run Verilator + GTKWave*
      1. Verilator link:   
         <https://verilator.org/guide/latest/>  
         <https://www.veripool.org/verilator/>

Simplified installation process is listed at the end of this document.

* + 1. GTKWave link:   
       <https://gtkwave.sourceforge.net>
  1. *Use attached Verilator testbench to compare C++ model against RTL implementation. Make sure it matches the specified accuracy. Feel free to modify the script according to your needs.*

**Task V. Coefficients optimisation / accuracy improvement**:

1. **Optimise the width of coefficients a, b, c keeping in mind the hardware implementation. Ensure the design accuracy meets the specified boundaries.**
2. Coefficient optimization  
   <https://www.mdpi.com/2079-9292/5/2/17>

**==============================================================================  
Prepare at least Task I. – IV.**Tasks V is optional.  
**Be prepared to explain your design process and key choices during the interview**.

You have 2+ week to complete this challenge from the moment you receive this email.

**Please email your submissions to me before interview:**[**zdzislaw.mrowka@intel.com**](mailto:zdzislaw.mrowka@intel.com)**.Show your work even if not fully complete.  
==============================================================================**

-------------------  
ZIP (drop #2s)



-------------------  
I. Octave / MATLAB - *Example script*-------------------

./matlab-model/

approx.m   
coeffs.txt   
plot\_err.m   
print\_coeffs.m   
run-approx.m

<https://octave.org/>  
octave --gui &

-------------------  
II. Modeling / Cmodel - *Example script*-------------------

./c++-model/

CoeffTable.hpp   
Makefile   
Quadra.cpp   
Quadra.hpp   
QuadraTb   
QuadraTb.cpp   
run-test.csh

<https://github.com/hlslibs/ac_types>

-------------------  
III. RTL Design – *Design sources*-------------------

./rtl-model

lut.vs   
quadra.vh   
quadra.vs   
quadra\_tb.vs   
quadra\_top.vs   
square.vs

-------------------  
IV. Verilator - *Testbench*-------------------

./verilator

run-all.csh   
run-binary.csh   
run-clean.csh   
run-gtkwave.csh   
run-verilator.csh   
sim\_main.cpp   
waves.gtkw

<https://verilator.org/guide/latest/>  
<https://www.veripool.org/verilator/>

-------------------  
V. Coefficients optimisation / accuracy improvement -------------------

// Further decreasing values results in an error 255

// Coefficients:

// a:

const int A\_I =  1;  // <optimize>

const int A\_F = 18;  // <optimize>

const int A\_W = A\_I + A\_F;

typedef ac\_int  < A\_W,       true>               a\_int\_t;

typedef ac\_fixed< A\_W,  A\_I, true, rdz, AC\_WRAP> a\_fxd\_t;

// b:

const int B\_I =  3;  // <optimize>

const int B\_F = 12;  // <optimize>

const int B\_W = B\_I + B\_F;

typedef ac\_int  < B\_W,       true>               b\_int\_t;

typedef ac\_fixed< B\_W,  B\_I, true, rdz, AC\_WRAP> b\_fxd\_t;

// c:

const int C\_I =  2;  // <optimize>

const int C\_F = 7;  // <optimize>

const int C\_W = C\_I + C\_F;

-------------------

--- Ubuntu  
-------------------  
**OS. Windows 10/11**

install **WSL 2** / **Ubuntu 22.04**go to **Windows Features** / enable **Windows Subsystem for Linux  
Manual installation steps by step**<https://learn.microsoft.com/en-us/windows/wsl/install-manual>   
*Download the Linux kernel update package*  
go to **Microsoft Store /** install **Ubuntu 22.04 LTS**

…

**OS. Ubuntu 20.04 / 22.04**

-------------------

--- Verilator / Install ---  
-------------------

# Prerequisites:  
sudo apt-get install git help2man perl python3 make

sudo apt-get install g++ *# Alternatively, clang*

sudo apt-get install libgz *# Non-Ubuntu (ignore if gives error)*

sudo apt-get install libfl2 *# Ubuntu only (ignore if gives error)*

sudo apt-get install libfl-dev *# Ubuntu only (ignore if gives error)*

sudo apt-get install zlibc zlib1g zlib1g-dev *# Ubuntu only (ignore if gives error)*

sudo apt-get install ccache *# If present at build, needed for run*

sudo apt-get install mold *# If present at build, needed for run*

sudo apt-get install libgoogle-perftools-dev numactl

sudo apt-get install autoconf flex bison

Use at least 10.0 version of gcc/g++ and C++17.  
  
git clone https://github.com/verilator/verilator *# Only first time*  
  
cd verilator

git pull *# Make sure we're up-to-date*

git tag *# See what versions exist*

*#git checkout master # Use development branch (e.g. recent bug fix)*

*#git checkout stable # Use most recent release*

*#git checkout v{version} # Switch to specified release version*

autoconf # Create ./configure script

./configure # Configure and create Makefile

make -j `nproc` # Build Verilator itself (if error, try just 'make')

sudo make install

For details, please visit [Verilator Install Guide](https://verilator.org/guide/latest/install.html)

-------------------

--- gtkwave / Install ---  
-------------------

sudo apt-get -y install gtkwave