Financial Technology with a Processing Unit

QueenField

1. INTRODUCTION

A Processing Unit (PU) is an electronic system within a computer that carries out instructions of a program by performing the basic arithmetic, logic, controlling, and I/O operations specified by instructions. Instruction-level parallelism is a measure of how many instructions in a computer can be executed simultaneously. The PU is contained on a single Metal Oxide Semiconductor (MOS) Integrated Circuit (IC).

An Automation Financial Method (AFM) is the technology and innovation that aims to compete with Traditional Financial Methods in the delivery of financial services. It is an emerging industry that uses technology to improve activities in finance. AFM is the new applications, processes, products, or business models in the financial services industry, composed of complementary financial services and provided as an end-to-end process via the Internet.

1.0. DO-254

1.0.1. HARDWARE PLANNING PROCESS

1.0.1.1. Plan for Hardware Aspects of Certification

1.0.1.2. Hardware Design Plan

1.0.1.3. Hardware Validation Plan

1.0.1.4. Hardware Verification Plan

1.0.1.5. Hardware Configuration Management Plan

1.0.1.6. Hardware Process Assurance Plan

1.0.2. HARDWARE DESIGN PROCESS

1.0.2.1 Requirements Capture Process

1.0.2.2 Conceptual Design Process

1.0.2.3 Detailed Design Process

1.0.2.4 Implementation Process

1.0.2.5 Production Transition Process

1.0.3. VALIDATION AND VERIFICATION PROCESS

1.0.3.1 Validation Process

1.0.3.2 Verification Process

1.0.4. CONFIGURATION MANAGEMENT PROCESS

1.0.5. PROCESS ASSURANCE

1.0.6. CERTIFICATION LIAISON PROCESS

1.0.7. HARDWARE DESIGN LIFECYCLE DATA

1.0.7.1 Certification Authority

1.0.7.2 Certification Reviews

1.0.7.3 Scheduling of Reviews

1.0.8. ADDITIONAL CONSIDERATIONS

1.0.8.1 Previously Developed Hardware

1.0.8.2 Commercial Components Usage

1.1. Model

1.1.1. MatLab Language

1.1.1.1. Structural UML diagrams

1.1.1.1.1. Class diagram

1.1.1.1.2. Component diagram

1.1.1.1.3. Composite diagram

1.1.1.1.4. Deployment diagram

1.1.1.1.5. Object diagram

1.1.1.1.6. Package diagram

1.1.1.1.7. Profile diagram

1.1.1.2. Behavioral UML diagrams

1.1.1.2.1. Activity diagram

1.1.1.2.2. Communication diagram

1.1.1.2.3. Interaction diagram

1.1.1.2.4. Sequence diagram

1.1.1.2.5. State diagram

1.1.1.2.6. Timing diagram

1.1.1.2.7. Use case diagram

1.1.2. Rust Language

1.1.2.1. Structural UML diagrams

1.1.2.1.1. Class diagram

1.1.2.1.2. Component diagram

1.1.2.1.3. Composite diagram

1.1.2.1.4. Deployment diagram

1.1.2.1.5. Object diagram

1.1.2.1.6. Package diagram

1.1.2.1.7. Profile diagram

1.1.2.2. Behavioral UML diagrams

1.1.2.2.1. Activity diagram

1.1.2.2.2. Communication diagram

1.1.2.2.3. Interaction diagram

1.1.2.2.4. Sequence diagram

1.1.2.2.5. State diagram

1.1.2.2.6. Timing diagram

1.1.2.2.7. Use case diagram

1.2. Design

1.2.1. VHDL

1.2.2. Verilog

1.3. Verification

1.3.1. OSVVM-VHDL

1.3.1.1. OSVVM Checker

1.3.1.2. OSVVM Stimulus

1.3.1.3. OSVVM Testbench

1.3.2. UVM-Verilog

1.3.2.1. UVM Agent

1.3.2.2. UVM Driver

1.3.2.3. UVM Enviroment

1.3.2.4. UVM Monitor

1.3.2.5. UVM Scoreboard

1.3.2.6. UVM Sequence

1.3.2.7. UVM Sequencer

1.3.2.8. UVM Subscriber

1.3.2.9. UVM Test

1.3.2.10. UVM Testbench

1.3.2.11. UVM Transaction

2. PROJECTS

2.1. INSTRUCTION CACHE

2.1.1 Instruction INPUTS/OUTPUTS AMBA4 AXI-Lite Bus

2.1.1.1. Signals of the Read and Write Address channels

Write Port	Read Port	Size	Direction	Description
AWID	ARID	AXI_ID_WIDTH	Output	Address ID, to identify multiple streams
AWADDR	ARADDR	AXI_ADDR_WIDTH	Output	Address of the first beat of the burst
AWLEN	ARLEN	8	Output	Number of beats inside the burst
AWSIZE	ARSIZE	3	Output	Size of each beat
AWBURST	ARBURST	2	Output	Type of the burst
AWLOCK	ARLOCK	1	Output	Lock type, to provide atomic operations
AWCACHE	ARCACHE	4	Output	Memory type, progress through the system
AWPROT	ARPROT	3	Output	Protection type
AWQOS	ARQOS	4	Output	Quality of Service of the transaction
AWREGION	ARREGION	4	Output	Region identifier, physical to logical
AWUSER	ARUSER	AXI_USER_WIDTH	Output	User-defined data
AWVALID	ARVALID	1	Output	xVALID handshake signal
AWREADY	ARREADY	1	Input	xREADY handshake signal

2.1.1.2. Signals of the Read and Write Data channels

Write Port	Read Port	Size	Direction	Description
WID	RID	AXI_ID_WIDTH	Output	Data ID, to identify multiple streams
WDATA	RDATA	AXI_DATA_WIDTH	Output	Read/Write data
--	RRESP	2	Output	Read response, current RDATA status
WSTRB	--	AXI_STRB_WIDTH	Output	Byte strobe, WDATA signal
WLAST	RLAST	1	Output	Last beat identifier
WUSER	RUSER	AXI_USER_WIDTH	Output	User-defined data
WVALID	RVALID	1	Output	xVALID handshake signal
WREADY	RREADY	1	Input	xREADY handshake signal

2.1.1.3. Signals of the Write Response channel

Write Port	Size	Direction	Description
BID	AXI_ID_WIDTH	Input	Write response ID, to identify multiple streams
BRESP	2	Input	Write response, to specify the burst status
BUSER	AXI_USER_WIDTH	Input	User-defined data
BVALID	1	Input	xVALID handshake signal
BREADY	1	Output	xREADY handshake signal

2.1.2. Instruction INPUTS/OUTPUTS AMBA3 AHB-Lite Bus

Port	Size	Direction	Description

Port	Size	Direction	Description
HRESETn	1	Input	Asynchronous Active Low Reset
HCLK	1	Input	System Clock Input
IHSEL	1	Output	Instruction Bus Select
IHADDR	PLEN	Output	Instruction Address Bus
IHRDATA	XLEN	Input	Instruction Read Data Bus
IHWDATA	XLEN	Output	Instruction Write Data Bus
IHWRITE	1	Output	Instruction Write Select
IHSIZE	3	Output	Instruction Transfer Size
IHBURST	3	Output	Instruction Transfer Burst Size
IHPROT	4	Output	Instruction Transfer Protection Level
IHTRANS	2	Output	Instruction Transfer Type
IHMASTLOCK	1	Output	Instruction Transfer Master Lock
IHREADY	1	Input	Instruction Slave Ready Indicator
IHRESP	1	Input	Instruction Transfer Response

2.1.3. Instruction INPUTS/OUTPUTS Wishbone Bus

Port	Size	Direction	Description
rst	1	Input	Synchronous Active High Reset
clk	1	Input	System Clock Input
iadr	AW	Input	Instruction Address Bus
idati	DW	Input	Instruction Input Bus
idato	DW	Output	Instruction Output Bus
isel	DW/8	Input	Byte Select Signals
iwe	1	Input	Write Enable Input
istb	1	Input	Strobe Signal/Core Select Input
icyc	1	Input	Valid Bus Cycle Input
iack	1	Output	Bus Cycle Acknowledge Output
ierr	1	Output	Bus Cycle Error Output
iint	1	Output	Interrupt Signal Output

2.2. DATA CACHE

2.2.1. Data INPUTS/OUTPUTS AMBA4 AXI-Lite Bus

2.2.1.1. Signals of the Read and Write Address channels

Write Port	Read Port	Size	Direction	Description
AWID	ARID	AXI_ID_WIDTH	Output	Address ID, to identify multiple streams
AWADDR	ARADDR	AXI_ADDR_WIDTH	Output	Address of the first beat of the burst
AWLEN	ARLEN	8	Output	Number of beats inside the burst
AWSIZE	ARSIZE	3	Output	Size of each beat
AWBURST	ARBURST	2	Output	Type of the burst
AWLOCK	ARLOCK	1	Output	Lock type, to provide atomic operations

Write Port	Read Port	Size	Direction	Description
AWCACHE	ARCACHE	4	Output	Memory type, progress through the system
AWPROT	ARPROT	3	Output	Protection type
AWQOS	ARQOS	4	Output	Quality of Service of the transaction
AWREGION	ARREGION	4	Output	Region identifier, physical to logical
AWUSER	ARUSER	AXI_USER_WIDTH	Output	User-defined data
AWVALID	ARVALID	1	Output	xVALID handshake signal
AWREADY	ARREADY	1	Input	xREADY handshake signal

2.2.1.2. Signals of the Read and Write Data channels

Write Port	Read Port	Size	Direction	Description
WID	RID	AXI_ID_WIDTH	Output	Data ID, to identify multiple streams
WDATA	RDATA	AXI_DATA_WIDTH	Output	Read/Write data
--	RRESP	2	Output	Read response, current RDATA status
WSTRB	--	AXI_STRB_WIDTH	Output	Byte strobe, WDATA signal
WLAST	RLAST	1	Output	Last beat identifier
WUSER	RUSER	AXI_USER_WIDTH	Output	User-defined data
WVALID	RVALID	1	Output	xVALID handshake signal
WREADY	RREADY	1	Input	xREADY handshake signal

2.2.1.3. Signals of the Write Response channel

Write Port	Size	Direction	Description
BID	AXI_ID_WIDTH	Input	Write response ID, to identify multiple streams
BRESP	2	Input	Write response, to specify the burst status
BUSER	AXI_USER_WIDTH	Input	User-defined data
BVALID	1	Input	xVALID handshake signal
BREADY	1	Output	xREADY handshake signal

2.2.2. Data INPUTS/OUTPUTS AMBA3 AHB-Lite Bus

Port	Size	Direction	Description
HRESETn	1	Input	Asynchronous Active Low Reset
HCLK	1	Input	System Clock Input
DHSEL	1	Output	Data Bus Select
DHADDR	PLEN	Output	Data Address Bus
DHRDATA	XLEN	Input	Data Read Data Bus
DHWDATA	XLEN	Output	Data Write Data Bus
DHWRITE	1	Output	Data Write Select
DHSIZE	3	Output	Data Transfer Size
DHBURST	3	Output	Data Transfer Burst Size
DHPROT	4	Output	Data Transfer Protection Level
DHTRANS	2	Output	Data Transfer Type
DHMASTLOCK	1	Output	Data Transfer Master Lock
DHREADY	1	Input	Data Slave Ready Indicator
DHRESP	1	Input	Data Transfer Response

2.2.3. Data INPUTS/OUTPUTS Wishbone Bus

Port	Size	Direction	Description
rst	1	Input	Synchronous Active High Reset
clk	1	Input	System Clock Input
dadr	AW	Input	Data Address Bus
ddati	DW	Input	Data Input Bus
ddato	DW	Output	Data Output Bus
dsel	DW/8	Input	Byte Select Signals
dwe	1	Input	Write Enable Input
dstb	1	Input	Strobe Signal/Core Select Input
dcyc	1	Input	Valid Bus Cycle Input
dack	1	Output	Bus Cycle Acknowledge Output
derr	1	Output	Bus Cycle Error Output
dint	1	Output	Interrupt Signal Output

2.3. FUNCTIONALITY

```
class classes {
   private:
      int number_pu;

   public:
      void method_0();  // method 0
      void method_1();  // method 1
      void method_2();  // method 2
      void method_3();  // method 3
};
```

2.3.1. Philosophers T-DNC/NTM-PU

```
class philosophers : private classes {
   private:
      int number_p_pu;

   public:
      void method_p0();  // method 0
      void method_p1();  // method 1
      void method_p2();  // method 2
      void method_p3();  // method 3
};
```

2.3.1.1. PU-NTM

2.3.2. Soldiers T-DNC/NTM-PU

```
class soldiers : private classes {
   private:
      int number_s_pu;
```

```
   public:
      void method_s0();  // method 0
      void method_s1();  // method 1
      void method_s2();  // method 2
      void method_s3();  // method 3
};
```

2.3.2.1. PU-NTM

2.3.3. Workers T-DNC/NTM-PU

```
class workers : private classes {
   private:
      int number_w_pu;

   public:
      void method_w0();  // method 0
      void method_w1();  // method 1
      void method_w2();  // method 2
      void method_w3();  // method 3
};
```

2.3.3.1. PU-NTM

3. WORKFLOW

1. System Level (SystemC/SystemVerilog)

The System Level abstraction of a system only looks at its biggest building blocks like processing un

2. Behavioral & Register Transfer Level (VHDL/Verilog)

At the Behavioural Level abstraction a language aimed at hardware description such as Verilog or VHDL

A design in Register Transfer Level representation is usually stored using HDLs like Verilog and VHDL

3. Logical Gate

At the Logical Gate Level the design is represented by a netlist that uses only cells from a small nu

4. Physical Gate

On the Physical Gate Level only gates are used that are physically available on the target architectu

5. Switch Level

A Switch Level representation of a circuit is a netlist utilizing single transistors as cells. Switch

3.1. FRONT-END OPEN SOURCE TOOLS

3.1.1. Modeling System Level of Hardware

*A System Description Language Editor is a computer tool that allows to generate software code. A Sys

System Description Language Editor

type:

git clone https://github.com/emacs-mirror/emacs

3.1.2. Simulating System Level of Hardware

*A System Description Language Simulator (translator) is a computer program that translates computer

SystemVerilog System Description Language Simulator

type:

git clone http://git.veripool.org/git/verilator

cd verilator autoconf ./configure make sudo make install

cd sim/verilog/regression/wb/vtor source SIMULATE-IT

cd sim/verilog/regression/ahb3/vtor source SIMULATE-IT

cd sim/verilog/regression/axi4/vtor source SIMULATE-IT

3.1.3. Verifying System Level of Hardware

*A UVM standard improves interoperability and reduces the cost of repurchasing and rewriting IP for e

SystemVerilog System Description Language Verifier

type:

git clone https://github.com/QueenField/UVM

3.1.4. Describing Register Transfer Level of Hardware

*A Hardware Description Language Editor is any editor that allows to generate hardware code. Hardware

Hardware Description Language Editor

type:

git clone https://github.com/emacs-mirror/emacs

3.1.5. Simulating Register Transfer Level of Hardware

*A Hardware Description Language Simulator uses mathematical models to replicate the behavior of an a

VHDL Hardware Description Language Simulator

type:

git clone https://github.com/ghdl/ghdl

cd ghdl ./configure –prefix=/usr/local make sudo make install

cd sim/vhdl/regression/wb/ghdl source SIMULATE-IT

cd sim/vhdl/regression/ahb3/ghdl source SIMULATE-IT

cd sim/vhdl/regression/axi4/ghdl source SIMULATE-IT

Verilog Hardware Description Language Simulator

type:

git clone https://github.com/steveicarus/iverilog

cd iverilog sh autoconf.sh ./configure make sudo make install

cd sim/verilog/regression/wb/iverilog source SIMULATE-IT

cd sim/verilog/regression/ahb3/iverilog source SIMULATE-IT

cd sim/verilog/regression/axi4/iverilog source SIMULATE-IT

3.1.6. Synthesizing Register Transfer Level of Hardware

*A Hardware Description Language Synthesizer turns a RTL implementation into a Logical Gate Level imp

Verilog Hardware Description Language Synthesizer

type:

git clone https://github.com/YosysHQ/yosys

cd yosys make sudo make install

cd synthesis/yosys source SYNTHESIZE-IT

3.1.7. Optimizing Register Transfer Level of Hardware

*A Hardware Description Language Optimizer finds an equivalent representation of the specified logic

Verilog Hardware Description Language Optimizer

type:

git clone https://github.com/YosysHQ/yosys

cd yosys make sudo make install

cd synthesis/yosys source SYNTHESIZE-IT

3.1.8. Verifying Register Transfer Level of Hardware

*A Hardware Description Language Verifier proves or disproves the correctness of intended algorithms

Verilog Hardware Description Language Verifier

type:

git clone https://github.com/YosysHQ/SymbiYosys

3.2. BACK-END OPEN SOURCE TOOLS

Library

type:

sudo apt update sudo apt upgrade

sudo apt install bison cmake flex freeglut3-dev libcairo2-dev libgsl-dev
libncurses-dev libx11-dev m4 python-tk python3-tk swig tcl tcl-dev tk-dev tcsh

Back-End Workflow Qflow

type:

git clone https://github.com/RTimothyEdwards/qflow

cd qflow ./configure make sudo make install

mkdir qflow cd qflow

3.2.1. Planning Switch Level of Hardware

*A Floor-Planner of an Integrated Circuit (IC) is a schematic representation of tentative placement o

Floor-Planner

type:

git clone https://github.com/RTimothyEdwards/magic

cd magic ./configure make sudo make install

3.2.2. Placing Switch Level of Hardware

*A Standard Cell Placer takes a given synthesized circuit netlist together with a technology library

Standard Cell Placer

type:

git clone https://github.com/rubund/graywolf

cd graywolf mkdir build cd build cmake .. make sudo make install

3.2.3. Timing Switch Level of Hardware

*A Standard Cell Timing-Analizer is a simulation method of computing the expected timing of a digital

Standard Cell Timing-Analizer

type:

git clone https://github.com/The-OpenROAD-Project/OpenSTA

cd OpenSTA mkdir build cd build cmake .. make sudo make install

3.2.4. Routing Switch Level of Hardware

*A Standard Cell Router takes pre-existing polygons consisting of pins on cells, and pre-existing wir

Standard Cell Router

type:

git clone https://github.com/RTimothyEdwards/qrouter

cd qrouter ./configure make sudo make install

3.2.5. Simulating Switch Level of Hardware

*A Standard Cell Simulator treats transistors as ideal switches. Extracted capacitance and lumped res

Standard Cell Simulator

type:

git clone https://github.com/RTimothyEdwards/irsim

cd irsim ./configure make sudo make install

3.2.6. Verifying Switch Level of Hardware LVS

*A Standard Cell Verifier compares netlists, a process known as LVS (Layout vs. Schematic). This step

Standard Cell Verifier

type:

git clone https://github.com/RTimothyEdwards/netgen

cd netgen ./configure make sudo make install

cd synthesis/qflow source FLOW-IT

3.2.7. Checking Switch Level of Hardware DRC

*A Standard Cell Checker is a geometric constraint imposed on Printed Circuit Board (PCB) and Integra

Standard Cell Checker

type:

git clone https://github.com/RTimothyEdwards/magic

cd magic ./configure make sudo make install

3.2.8. Printing Switch Level of Hardware GDS

*A Standard Cell Editor allows to print a set of standard cells. The standard cell methodology is an

Standard Cell Editor

type:

git clone https://github.com/RTimothyEdwards/magic

cd magic ./configure make sudo make install

4. CONCLUSION

4.1. OPEN SOURCE TOOLS FOR WINDOWS USERS!

4.1.0. Install Prerequisites

1. Settings → Apps → Apps & features → Related settings, Programs and
Features → Turn Windows features on or off → Windows Subsystem for
Linux

2. Microsoft Store → INSTALL UBUNTU

Library
type:

sudo apt update sudo apt upgrade

sudo apt install bison cmake flex freeglut3-dev libcairo2-dev libgsl-dev
libncurses-dev libx11-dev m4 python-tk python3-tk swig tcl tcl-dev tk-dev tcsh

4.1.1. Front-End

type:

sudo apt install verilator sudo apt install iverilog sudo apt install ghdl

cd /mnt/c/../sim/verilog/regression/wb/iverilog source SIMULATE-IT

sudo apt install yosys

cd /mnt/c/../synthesis/yosys source SYNTHESIZE-IT

4.1.2. Back-End

type:

mkdir qflow cd qflow

git clone https://github.com/RTimothyEdwards/magic git clone https://github.com/rubund/graywolf git clone https://github.com/The-OpenROAD-Project/OpenSTA git clone https://github.com/RTimothyEdwards/qrouter git clone https://github.com/RTimothyEdwards/irsim git clone https://github.com/RTimothyEdwards/netgen git clone https://github.com/RTimothyEdwards/qflow

cd /mnt/c/../synthesis/qflow source FLOW-IT

4.2. OPEN SOURCE SYNTHESIZER FOR WINDOWS USERS!

4.2.0. Install Prerequisites

4.2.0.1. For WINDOWS Users!

1. Settings → Apps → Apps & features → Related settings, Programs and Features → Turn Windows feature

2. Microsoft Store → INSTALL UBUNTU

4.2.0.2. For WINDOWS and LINUX Users

sudo apt update sudo apt upgrade

sudo apt -y install build-essential clang bison flex
libreadline-dev gawk tcl-dev libffi-dev git make gnat
graphviz xdot pkg-config python3 libboost-system-dev
libboost-python-dev libboost-filesystem-dev zlib1g-dev

4.2.1. Install GHDL

git clone https://github.com/ghdl/ghdl

cd ghdl ./configure –prefix=/usr/local make sudo make install

4.2.2. Install Yosys

git clone https://github.com/YosysHQ/yosys

cd yosys make sudo make install

4.2.3. Install Synthesizer Plugin

git clone https://github.com/ghdl/ghdl-yosys-plugin cd ghdl-yosys-plugin make GHDL=/usr/local sudo yosys-config –exec mkdir -p –datdir/plugins sudo yosys-config –exec cp “ghdl.so” –datdir/plugins/ghdl.so “‘