phoeniX Processor Modules

Address Generator

Responsible for generating target address of BRANCH, JUMP and LOAD/STORE instructions

Signal	Width	Direction	Description
opcode	[6 : 0]	input	Opcode field of instruction
rs1	[31 : 0]	input	Source register 1
PC	[31 : 0]	input	Instruction's Program Counter
immediate	[31 : 0]	input	Immediate field of instruction
address	[31 : 0]	output	Target Address of instruction

Fetch Unit

Logic required for fetching instructions from memory and setting value of the program counter

Signal	Width	Direction	Description
enable	1	input	Enable signal to control fetch action
PC	[31 : 0]	input	Instruction's Program Counter
jump_branch_address	[31 : 0]	input	Instruction's Program Counter
jump_branch_enable	1	input	Instruction's Program Counter
next_PC	[31 : 0]	output	Value to latch on PC register
memory_interface_enable	1	output	Enabling data transfer with memory
memory_interface_state	1	output	Memory interface state (READ/WRITE)
memory_interface_address	[31 : 0]	output	Memory interface address
memory_interface_frame_mask	[3 : 0]	output	Memory interface frame mask (byte select)

Hazard Forward Unit

Module responsible for detecting true data dependency detection and forwarding logic to reduce stalls

Signal	Width	Direction	Description
source_index	[4 : 0]	input	Index of data source requiring forwarding
destination_index_1	[4 : 0]	input	Index of the first data option for forwarding
destination_index_2	[4 : 0]	input	Index of the second data option for forwarding

Signal	Width	Direction	Description
destination_index_3	[4 : 0]	input	Index of the third data option for forwarding
data_1	[31 : 0]	input	Value of the first data option for forwarding
data_2	[31 : 0]	input	Value of the second data option for forwarding
data_3	[31 : 0]	input	Value of the third data option for forwarding
enable_1	1	input	Validity of the first data option for forwarding
enable_2	1	input	Validity of the second data option for forwarding
enable_3	1	input	Validity of the third data option for forwarding
forward_enable	1	output	Enable signal for controlling forwarding action
forward_data	[31 : 0]	output	Data to be forwarded to the source

Immediate Generator

Parser of a fetched instruction for generating immediate values according to the instruction's type

Signal	Width	Direction	Description
instruction	[31 : 0]	input	Instruction to be parsed for immediate generation
instruction_type	[2 : 0]	input	Type of the instruction used for different immediate variants
immediate	[31 : 0]	output	Immediate value generated from the instruction

Instruction Decoder

Responsible for decomposing an instruction to separate fields

Signal	Width	Direction	Description
instruction	[31 : 0]	input	Instruction to be parsed for generating different fields
opcode	[6 : 0]	output	Opcode of the instruction
funct3	[2 : 0]	output	Field of funct3 indicating primary function of the instruction's family
funct7	[6 : 0]	output	Field of funct7 indicating secondary function of the instruction's family
funct12	[11 : 0]	output	Field of funct12 indicating tertiary function of special instructions
read_index_1	[4 : 0]	output	Address of the first source register
read_index_2	[4 : 0]	output	Address of the second source register

Signal	Width	Direction	Description
write_index	[4 : 0]	output	Address of the destination register
csr_index	[11 : 0]	output	Address of the control satus register (source/destination)
instruction_type	[2 : 0]	output	Type of the instruction inferred
read_enable_1	1	output	Correctness of reading first source register from register file
read_enable_2	1	output	Correctness of reading second source register from register file
write_enable	1	output	Correctness of writing destination register to register file
read_enable_csr	1	output	Correctness of reading control status register from CSR register file
write_enable_csr	1	output	Correctness of writing control status register to CSR register file

Jump Branch Unit

Condition checker and decision maker for individual branch types and jump instruction

Signal	Width	Direction	Description
opcode	[6 : 0]	input	Opcode of the instruction
funct3	[2 : 0]	input	Instruction family's primary function
instruction_type	[2 : 0]	input	Type of the instruction begin processed
rs1	[31 : 0]	input	Source register 1
rs2	[31 : 0]	input	Source register 2
jump_branch_enable	1	output	Indicator whether jump or branch should be taken

Load Store Unit

Module responsible for Load and Store operations for aligned addresses and wordsize management

Signal	Width	Direction	Description
opcode	[6 : 0]	input	Opcode of the instruction
funct3	[2 : 0]	input	Instruction family's primary function
address	[31 : 0]	input	Target address for Load/Store operations
store_data	[31 : 0]	input	Value to be written in memory in case of Store operations
load_data	[31 : 0]	output	Value loaded from memory

Signal	Width	Direction	Description
memory_interface_enable	1	output	Signal to enable memory interface for processor and memory interactions
memory_interface_state	1	output	Signal to set the direction of the processor and memory interactions
memory_interface_address	[31 : 0]	output	Address sent to the memory during interactionns
memory_interface_frame_mask	[3 : 0]	output	Frame mask for selection of relevant bytes of the memory frame as seen by the core
memory_interface_data	[31 : 0]	inout	Data bits transferring during the processor and memory interactions

Register File

A parametrized register file suitable for general purpose and control-status registers benefiting from 2 read ports and 1 write port

Signal	Width	Direction	Description
CLK	1	input	Clock signal for synchronization of register's flip-flops
reset	1	input	Reset signal to set all regsiter values to zero
read_enable_1	1	input	Enable signal for first read port of the file
read_enable_2	1	input	Enable signal for second read port of the file
write_enable	1	input	Enable signal for the write port of the file
read_index_1	[4 : 0]	input	Index passed to first read port of the file
read_index_2	[4 : 0]	input	Index passed to second read port of the file
write_index	[4 : 0]	input	Index passed to the write port of the file
write_data	[31 : 0]	input	Data value to be written on a register in the file
read_data_1	[31 : 0]	output	Data value read from the first port of the file
read_data_2	[31 : 0]	output	Data value read from the second port of the file

phoeniX Execution Engine

Platform's execution engine which is designed for an RV32IEM core, has three main modules: `Arithmetic Logic Unit`, `Multiplier Unit` and `Divider Unit`. These three modules are designed in a novel way which gives designers the ability to **change or add executer circuits** in the function units, without any need to change the control logic in the modules. Obviously there are some guidelines for this kind of transformations and changes like this for the platform, but in the end it is very simple to do and needs lowest requirements

and changes. For each execution unit, there is one special purpose register defined named as `alucsr`, `mulcsr` and `divcsr`. These Control Status Registers (CSRs) are designed in a way to provide phoeniX's special features for approximate computing. You can see the structure of the mentioned registers in the following table:

CSR [31 : 16]	CSR [15 : 12]	CSR [11 : 8]	CSR [7 : 3]	CSR [2 : 1]	CSR [0]
Error Control	Truncation Control	Custom 2	Custom 1	Circuit Select	Enable Approximation

Each entry in these registers is dedicated to a specific feature, aiming to assist users and designers in demonstrating advancements in their respective fields of study and work. The ultimate objective of these features, which contributes to the overall goal of this project, is to enhance user accessibility in programming and provide designers with greater flexibility in implementations, and achieving this target with the integration of **precise circuits' accuracy and the efficiency** and potential **energy savings offered by approximate circuits**.

These three registers are mapped as `0x800 (alucsr)`, `0x801(mulcsr)` and `0x802 (divcsr)` in platform's register file. Here is a sample RISC-V assembly code included to show the programming convention of the phoeniX using an approximate arithmetic circuit which also the error level is configurable in the circuit.

```
factorial:  add     a4,     a0,     zero
            add     a2,     a0,     a1
            add     a3,     a0,     zero
            addi    x31,    x0,     -127
            csrrw   x0,     0x801,  x31     # set value to the mulcsr
loop:       mul     a4,     a3,     a4
            addi    a3,     a3,     1
            blt     a3,     a2,     loop
            add     a0,     a4,     zero
            csrrw   x0,     0x801,  x0      # reset value of the mulcsr
            ret
```

Arithmetic Logic Unit

Arithemtic logic unit with support for `I_TYPE` and `R_TYPE` instructions plus `U_TYPE` and return address calculations for `J-TYPE`. ALU in phoeniX includes 2 adders:

- Error Configurable Very Fast Approximate Carry Select Adder (phoeniX original design)
- Accurate Kogge Stone Adder (Very Fast)

Signal	Width	Direction	Description
opcode	[6 : 0]	input	Opcode field of instruction
funct3	[2 : 0]	input	Instruction family's first function
funct7	[6 : 0]	input	Instruction family's second function
control_status_register	[31 : 0]	input	ALU's special CSR (for circuit select feature and error configuration)

Signal	Width	Direction	Description
PC	[31 : 0]	input	Instruction's Program Counter
rs1	[31 : 0]	input	Source register 1
rs2	[31 : 0]	input	Source register 2
immediate	[31 : 0]	input	Immediate field of instruction
alu_output	[31 : 0]	output	Result of alu operations on selected operands

Accuracy comparison for the 8-bit, with 4 bit error control version of Error Configurable Carry Select Adder:

Error Level	NMED	MRED	ER
0	0.427318	1.142415	54.6875
1	0.434198	1.160022	46.875
2	0.438784	1.171529	50
3	0.440313	1.173151	37.5
4	0.412794	1.101233	50
5	0.415851	1.108248	40.625
6	0.40362	1.073216	42.1875
7	0.391389	1.036706	25
8	0.217099	0.590805	46.875
9	0.220157	0.598222	37.5
10	0.207926	0.565063	40.625
11	0.195695	0.530081	25
12	0.110078	0.302379	37.5
13	0.097847	0.268345	25
14	0.048924	0.135	25
15	0	0	0

Multiplier Unit

Multiplier unit with integrated approximate circuit:

- Low-Power, High-Speed, Area-Efficient and Error Configurable Approximate Multiplier (phoeniX original design)

Signal	Width	Direction	Description
clk	1	input	Clock signal for sequential logic based deisgned multipliers
opcode	[6 : 0]	input	Opcode field of instruction
funct3	[2 : 0]	input	Instruction family's first function
funct7	[6 : 0]	input	Instruction family's second function
control_status_register	[31 : 0]	input	MUL's special CSR (for circuit select feature and error configuration)
PC	[31 : 0]	input	Instruction's Program Counter
rs1	[31 : 0]	input	Source register 1
rs2	[31 : 0]	input	Source register 2
multiplier_unit_busy	1	output	Multiplier unit's busy signaling for correct timing and stalling in pipeline
multiplier_unit_output	[31 : 0]	output	Result of multiplier unit's operations on selected operands

NMED, MRED and ER for 8-bit version of the purposed approximate dynamic error configurable multiplier:

Error Level	NMED	MRED	ER
7’1b mask	0.25	0.85	36.16
6’1b mask	0.25	0.85	36.16
5’1b mask	0.26	0.97	39.73
4’1b mask	0.28	1.33	44.33
3’1b mask	0.34	2.11	50.33
2’1b mask	0.48	3.59	57.02
1’1b mask	0.76	5.89	61.8
0’1b mask	1.25	8.94	65.07

Divider Unit

Divider unit with integrated approximate circuit:

- 32-bit non-restoring divider with an 8-bit error-configuration

Signal	Width	Direction	Description
clk	1	input	Clock signal for sequential logic based deisgned dividers
opcode	[6 : 0]	input	Opcode field of instruction
funct3	[2 : 0]	input	Instruction family's first function
funct7	[6 : 0]	input	Instruction family's second function
control_status_register	[31 : 0]	input	DIV's special CSR (for circuit select feature and error configuration)
PC	[31 : 0]	input	Instruction's Program Counter
rs1	[31 : 0]	input	Source register 1
rs2	[31 : 0]	input	Source register 2
divider_unit_busy	1	output	Divider unit's busy signaling for correct timing and stalling in pipeline
divider_unit_output	[31 : 0]	output	Result of divider unit's operations on selected operands