"""

#### Addre Switch

prompt = f"""

create the chisel code for the Magma-si AdderSwitch solution is correct or not.

Question:

You are expert in Chisel,Adder Switch in which he work for adder which check the value and reduce the value of adder , i want to adder switch in chisel languge,

* - It has input valid pin name i\_valid in Bool.
* - It has input in vector 2 bit and 32 bit data bit name i\_data\_bus in unsign.
* - It has input encoder name i\_add\_en of 3 bit data in unsign.
* - It has input name i\_cmd in 3 bit data in unsign.
* - It has input selection pin in 2 data bit name i\_sel in unsign.
* - It has output of output adder in vector 2 bit and 32 bdata bit name o\_adder in unsign.
* - It has Output vn name o\_en of 32 bit data in unsign.
* - It has output vn valid pin name o\_vn\_valid in 2 bit data in unsign.

Solution:

* - The module AdderSwitch is defined with parameters DATA\_TYPE, NUM\_IN, SEL\_IN,and NUM\_OUT.
* -It declares input and output ports using the IO method, defining signals for input validity, input
* -It instantiates two internal modules: ReductionMux and SimpleAdder.
* -ReductionMux is instantiated with parameters W, NUM\_IN, and SEL\_IN, and its inputs (i\_data and i\_sel) are connected to the corresponding inputs of AdderSwitch.
* -SimpleAdder is instantiated to perform simple addition operations.
* -Registers r\_adder, r\_vn, and r\_vn\_valid are declared to store intermediate values.
* - r\_add\_en is a registered version of input i\_add\_en.
* - The when block is used to define behavior based on conditions (reset and input validity).
* -During reset (rst), registers and outputs are initialized to zero.
* -When input is valid (io.i\_valid), a switch statement is used based on io.i\_cmd.
* -For different command values, different operations are performed on the inputs and intermediate registers are updated accordingly.
* - Depending on the condition !r\_add\_en, outputs io.o\_adder are set either directly from the r\_adder registers or from the adder32 module outputs.
* -Final outputs io.o\_vn and io.o\_vn\_valid are assigned from the registers r\_vn and r\_vn\_valid.

"""

#### Edge Addre Switch

prompt = f"""

create the chisel code for the Magma-si Edge AdderSwitch solution is correct or not.

Question:

You are expert in Chisel,Edge Adder Switch in which he work for adder which check the value and reduce the value of edge adder , i want to edge adder switch in chisel languge,

* - It has input valid pin name i\_valid in Bool.
* - It has input in vector 2 bit and bit data bit name i\_data\_bus in unsign.
* - It has input encoder name i\_add\_en of 3 bit data in unsign.
* - It has input name i\_cmd in 3 bit data in unsign.
* - It has input selection pin in 2 data bit name i\_sel in unsign.
* - It has output of output adder in vector 2 bit and 32 bdata bit name o\_adder in unsign.
* - It has Output vn name o\_en of 32 bit data in unsign.
* - It has output vn valid pin name o\_vn\_valid in 2 bit data in unsign.

Solution:

* The module EdgeAdderSwitch is defined with parameters DATA\_TYPE, NUM\_IN, SEL\_IN, and NUM\_OUT.
* It declares input and output ports using the IO method, defining signals for input validity, input data, adder enable signal, command, selection, and outputs for result and validity.
* It instantiates two internal modules: ReductionMux and SimpleAdder.
* ReductionMux is instantiated with parameters W, NUM\_IN, and SEL\_IN, and its inputs (i\_data and i\_sel) are connected to the corresponding inputs of EdgeAdderSwitch.
* SimpleAdder is instantiated to perform simple addition operations.
* Registers r\_valid, r\_adder, r\_vn, and r\_vn\_valid are declared to store intermediate values.
* r\_add\_en is a registered version of input i\_add\_en.
* The when block is used to define behavior based on conditions (reset and input validity).
* During reset (rst), registers and outputs are initialized to zero.
* When input is valid (r\_valid), a switch statement is used based on io.i\_cmd.
* For different command values, different operations are performed on the inputs and intermediate registers are updated accordingly.
* Depending on the condition !r\_add\_en, outputs io.o\_adder are set either directly from the r\_adder register or from the output of adder32 module.
* Final outputs io.o\_vn and io.o\_vn\_valid are assigned from the registers r\_vn and r\_vn\_valid.

"""

#### Simple Adder

prompt = f"""

create the chisel code for the Magma-si Simple Adder solution is correct or not.

Question:

You are expert in Chisel,Simple Adder Switch in which he add two value to give one , i want to simple adder switch in chisel languge,

* - It has input for input 1 name A in 32 width unsign.
* - It has input for input 2 name B in 32 width unsign.
* - It has output name O of 32 bit data in unsign.

Solution:

* The module SimpleAdder is defined as a subclass of Module.
* It declares input and output ports using the IO method, defining signals for two input operands A and B, and an output signal O to store the result of the addition.
* The module's behavior is straightforward. It takes the two input operands A and B and performs addition on them.
* The result of the addition (io.A + io.B) is assigned to the output signal O.

"""

#### Simple Multiplier

prompt = f"""

create the chisel code for the Magma-si Simple Multiplier solution is correct or not.

Question:

You are expert in Chisel,Simple Multiplier in which he multipoly two value to give one , i want to simple multiplayer in chisel languge,

* - It has input for input 1 name A in 16 width unsign.
* - It has input for input 2 name B in 16 width unsign.
* - It has output name O of 16 bit data in unsign.

Solution:

* The module SimpleMultiplier is defined as a subclass of Module.
* It declares input and output ports using the IO method, defining signals for two input operands A and B, and an output signal O to store the result of the multiplication.
* The module's behavior is straightforward. It takes the two input operands A and B and performs multiplication on them.
* The result of the multiplication (io.A \* io.B) is assigned to the output signal O.

"""

#### Buffer Multiplication

prompt = f"""

create the chisel code for the Magma-si Buffer Multiplication solution is correct or not.

Question:

You are expert in Chisel,Buffer Multiplication in which he multiply two buffer vector of length 4 to give one output vector , i want to buffer multiplayer in chisel languge,

* - It has input in vector 4 bit and 32 width data bit name buffer1 in unsign.
* - It has input in vector 4 bit and 32 width data bit name buffer2 in unsign.
* - It has output in vector 4 bit and 32 width data bit name out in unsign.

Solution:

* The module buffer\_multiplication is defined as a subclass of Module. It takes an implicit parameter config of type MagmasiConfig, which presumably contains configuration parameters like the number of processing elements (NUM\_PES) and the data type width (DATA\_TYPE).
* It declares input and output ports using the IO method, defining signals for two input vectors buffer1 and buffer2, each with config.NUM\_PES elements, and an output vector out with the same number of elements.
* The module defines a recursive function multiply that performs element-wise multiplication of two vectors.
* The multiply function takes three parameters: vec1, vec2, and index. vec1 and vec2 are the input vectors to be multiplied, and index keeps track of the current element being processed.
* If index is greater than or equal to config.NUM\_PES, indicating that all elements have been processed, the function returns a vector filled with zeros.
* Otherwise, it computes the product of the elements at the current index in vec1 and vec2. Then, it recursively calls itself with the incremented index and updates the result vector accordingly.
* Finally, the output io.out is assigned the result of calling the multiply function with the input vectors io.buffer1 and io.buffer2.

"""

#### Ivncontrol4

prompt = f"""

create the chisel code for the Magma-si Ivncontrol solution is correct or not.

Question:

You are expert in Chisel,IvnControl in which he control the ivn which is flow from adder and edge adder switch and give ivn value to fan control , i want to ivn control in chisel languge,

* - It has input in 2d vector 4 bit row and 4 bit colunms and 32 width data bit name stationary\_matrix in unsign.
* - It has output in vector 4 bit and 5 width data bit name o\_vn in unsign.

Solution:

* The module ivncontrol4 is defined as a subclass of Module. It takes an implicit parameter Config of type MagmasiConfig.
* It declares input and output ports using the IO method. Inputs include a 2D vector Stationary\_matrix, and outputs include a vector o\_vn.
* input\_valid: A register initialized to false.B, indicating the validity of input data.
* counter: A register initialized to zero, used for delay.
* i\_vn: A register representing the output vector, initialized with random values.
* rowcount: A register representing the count of non-zero elements in each row of the input matrix.
* i and j: Registers to keep track of the current row and column indices.
* rowlength and matlength: Variables storing the length of rows and the length of the input matrix, respectively.
* valid and valid1: Signals indicating the validity of certain conditions.
* mat: A register representing the input matrix.
* count: A register representing the count of non-zero elements in each row of the input matrix.
* The module seems to initialize some registers and signals.
* It checks for the condition counter === 6.U and sets input\_valid to true when met.
* It reads elements from io.Stationary\_matrix and updates mat.
* It counts the non-zero elements in each row of the matrix and stores the count in count.
* It updates i and j to iterate through the rows and columns of the matrix.
* It sets valid to true when i and j reach their maximum values.
* Based on the counts stored in rowcount, it sets values for i\_vn.
* The output o\_vn is assigned the value of i\_vn.
* There are several dontTouch statements and debug prints present in the code for debugging purposes.
* There are some commented-out code blocks that may have been used for testing or debugging.