SystemVerilog Coding Standard

# Rule 1-1: Declaration spaces

Declarations shall be made only in modules, interfaces and packages.

# Rule 1-2: Variable identifier name convention

The variable identifier name shall have a prefix, which specifies its data type and array size. The prefix is comprised of an identifier <type> specific to the data type followed by the array size <size>. In case of a multi-dimensional array, the identifier shall have an additional prefix ‘a’.

<type><size><identifier>

|  |  |  |
| --- | --- | --- |
| Data type | <type> | <size> |
| logic | ul | 1 for a logic variable and equal to the number of bits for a logic array. If the size of the array is parameterized, then ‘v’. |
| logic signed | sl | 1 for a logic variable and equal to the number of bits for a logic array. If the size of the array is parameterized, then ‘v’. |
| bit | sb | 1 for a bit variable and equal to the number of bits for a bit array. If the size of the array is parameterized, then ‘v’. |
| bit unsigned | ub | 1 for a bit variable and equal to the number of bits for a bit array. If the size of the array is parameterized, then ‘v’. |
| interface | i | N/A |
| enum | e | N/A |
| time | tm | N/A |

Examples:

logic ul1Enable;

logic [7:0] ul8UnsignedData;

logic [BITS-1:0] ulvUnsignedData;

logic signed [15:0] sl16SignedData;

logic signed [1:0][7:0] sl16SignedData;

logic signed [BITS-1:0] slvSignedData;

logic [23:0] aul24RamBuffer[0:76799];

logic [BITS-1:0] aulvRamBuffer[0:76799];

bit sb1Enable;

bit [7:0] sb8UnsignedData;

bit [BITS-1:0] sbvUnsignedData;

bit unsigned [15:0] ub16SignedData;

bit unsigned [1:0][7:0] ub16SignedData;

bit unsigned [BITS-1:0] ubvSignedData;

bit [23:0] asb24RamBuffer[0:76799];

bit [BITS-1:0] asbvRamBuffer[0:76799];

tIImageTransfer iImageTransfer;

teMacroBlockType eMacroBlockType;

time tmDuration;

# Rule 1-3: Port identifier name convention

The port identifier name shall follow the variable name convention [Rule 1-2] with a prefix, which specifies the port direction or that it is an interface port. The prefix is comprised of the character ‘p’ followed by the direction identifier <dir> to specify the direction of the port or that it is an interface:

p<dir><identifier>

|  |  |
| --- | --- |
| Direction/Interface | p<dir> |
| input | pi |
| output | po |
| inout | px |
| interface | pI |

Examples:

input logic piul1Clk;

output logic [7:0] poul8Status;

inout logic [15:0] pxul16DIO;

# Rule 1-4: Active-low signal identifiers

Active-low signal variables shall have the postfix ‘\_n’.

<identifier>\_n

Examples:

logic ul1FpgaReset\_n;

# Rule 1-5: Constant signal identifiers

Constant signal variables shall have the prefix ‘c’.

c<identifier>

Examples:

const logic [7:0] cul8ResetDelayCC = 256;

# Rule 2-1: Module declaration

Every module be shall defined in a separate file containing only the module.

# Rule 2-2: Module filename

The filename of the module shall be the module identifier with a type prefix ‘M\_’.

M\_<identifier>.sv

Example:

M\_CameraFpga.sv

# Rule 2-3: Module identifier type prefix

The module identifier shall follow the variable name conventions [Rule 1-2], [Rule 1-3], [Rule 1-4] and have a type prefix ‘tM’ and the endmodule keyword shall be followed by a comment containing the module identifier:

tM<identifier>

Example:

module tMCameraFpga (

input logic piul1FpgaClock,

input logic piul1FpgaReset\_n,

tITRDB\_D5M.driver pIImageSensor,

tIADV7123.driver pIDisplay

);

endmodule//tMCameraFpga

# Rule 2-4: Module port signal type [optional]

The module ports should be of unsigned logic data type.

Example:

module tMCounter (

input logic piul1FpgaClock,

input logic piul1FpgaReset\_n,

input logic piul1Enable,

output logic [7:0] poul8Count

);

endmodule//tMCounter

# Rule 3-1: Package declaration

Every package be shall defined in a separate file containing only the package.

# Rule 3-2: Package filename

The filename of the package shall be the package identifier with a type prefix ‘P\_’.

P\_<identifier>.sv

Example:

P\_ImageProcessing.sv

# Rule 3-3: Package identifier type prefix

The package identifier shall follow the variable name conventions [Rule 1-2], [Rule 1-3], [Rule 1-4] and have a type prefix ‘tP’ and the endpackage keyword shall be followed by a comment containing the package identifier:

tP<identifier>

Example:

package tPImageProcessing;

endpackage//tPImageProcessing

# Rule 4-1: Interface declaration

Every interface be shall defined in a separate file containing only the interface.

# Rule 4-2: Interface filename

The filename of the interface shall be the interface identifier with a type prefix ‘I\_’.

I\_<identifier>.sv

Example:

I\_DataBus.sv

# Rule 4-3: Interface identifier type prefix

The interface identifier shall follow the variable name conventions [Rule 1-2], [Rule 1-3], [Rule 1-4] and have a type prefix ‘tI’ and the endinterface keyword shall be followed by a comment containing the interface identifier:

tI<identifier>

Example:

interface tIDataBus;

endinterface//tIDataBus

# Rule 5-1: Top module ports identifier name convention

The top module shall have all ports names in CAPITAL matching the PCB signals names.

# Rule 5-2: Tristate buffer generation [optional]

The tristate buffers should be inferred from RTL logic.

# Rule 5-3: Tristate buffer instantiation

The tristate buffers shall only be instantiated in the top module.

# Rule 5-4: Bidirectional signals in modules

Modules shall only have unidirectional signals and any bidirectional signal shall be represented by two unidirectional signals. These signals shall have the same identifier prefix matching the bidirectional signal name followed by a ‘\_in’ for the input signal and an ‘\_out’ for the output signal.

<idetifier>\_in <identifier>\_out

Examples:

//inout wire scl

input wire scl\_in

output wire scl\_out

//inout wire sda

input wire sda\_in

output wire sda\_out