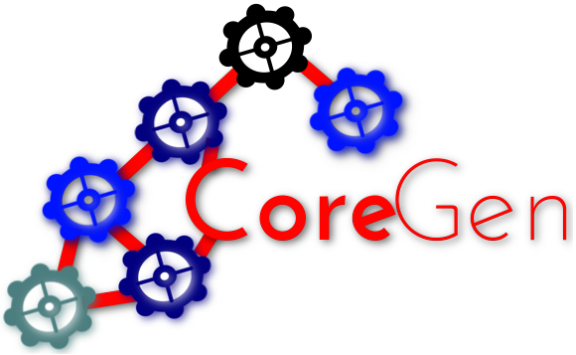

CoreGen Intermediate Representation Language Specification

Tactical Computing Laboratories, LLC

Date: April 28, 2020
Revision: 1.4.5
Authors: Tactical Computing Labs
contact@tactcomplabs.com

Contents

List of Figures

Listings

List of Tables

Revision History

Revision	Date	Author(s)	Description
1.0	09.12.2018	JLeidel	Initial public release
1.1	11.21.2018	JLeidel	Adding ThreadUnits to Core nodes and TUSReg register attributes
1.2	11.21.2018	JLeidel	Adding subregister pseduoregister sub-node structure for setting up register aliases to specific bit regions of parent registers
1.3	12.20.2018	JLeidel	Adding clarifying text to indicate the permissible characters utilized in IR node names
1.4	01.21.2019	JLeidel	Adding RTL Type fields for overloaded RTL models; Adding Syntax mnemonic fields for instructions
1.4.1	04.24.2019	JLeidel	Adding PC attribute to register node attributes
1.4.2	07.03.2019	JLeidel	Adding Override keyword to all candidate hardware nodes: Section 2.3
1.4.3	08.21.2019	JLeidel	Adding syntax mnemonics to pseudo instructions
1.4.4	08.27.2019	JLeidel	Adding notes syntax to each candidate IR node
1.4.5	04.28.2020	JLeidel	Adding LineSize cache line size parameter and memory ordering parameters; Adding read and write ports to register classes; Adding DataPath nodes; Adding RegIsDestination keyword to instruction formats

1 Overview

The CoreGen Intermediate Representation (IR) specification is utilized as the core mechanism by which to translate a hardware design to various levels of hardware design language and simulation infrastructure. The CoreGen IR has the following goals:

- **Support Rapid Design Workflows**: Traditional design techniques and tools require long design and prototyping cycles, often with expensive external tools. The CoreGen IR is design to provide rapid design and prototyping workflows such that hardware and software artifacts can be generated in minutes rather than days.
- **Preserve Hardware Dependencies**: The CoreGen IR is designed to preserve the dependency between individual hardware modules and their constituent submodules. These dependencies are preserved in a manner similar to traditional compiler technology, e.g. a *directed acyclic graph* (DAG). These dependencies are subsequently utilized to analyze certain inherent properties of the DAG in order to verify, optimized and document the design.
- **Support Modular Design Principles/Reuse**: One of the principle issues that plague previous design workflows and paradigms is the lack of potential reuse. The CoreGen IR provides modular reuse of individual design elements as well as the ability to use design templates to rapidly build similar, but specialized design elements. Further, all individual hardware modules can be overridden using external design elements in order to further construct specialized designs.
- **Support High-Level Design Verification**: One of the primary goals of the CoreGen infrastructure is the ability to perform high performance, low latency design verification without the need to perform low-level synthesis and simulation. This capability is provided by the inherent ability to express complex designs within the CoreGen IR.
- **Support Multiple Artifact Generation**: The final goal of the CoreGen IR is to drive secondary and, potentially, external tooling. These tools provide the ability to generate multiple levels of hardware design language for the complementary architecture as well as the potential to generate software artifacts such as compiler tool chains from the design description.

The CoreGen IR is constructed using the standard *YAML* [1] format. This is a human readable text structure that preserves both descriptions of individual hardware modules and references between adjacent modules (the DAG from above). YAML is formatted using blocks of *Collections*. Collections describe an individual type of node (hardware module) in the CoreGen IR. For example, the notion of a register file is encapsulated in a single collection. Each collection may contain a set of *sequences*. These sequences are individual instances of the appropriate module type are denoted with a name followed by a colon (NAME:). For example, your design may include multiple different register files, each represented as a sequence. Each sequence will include some number of elements, each denoted with a name followed by a dash (NAME-). These sequence instances may also include a hierarchical set of sequences with scalar values. The named elements and their associated scalar values are marked using the name and colon designator (NAME:). These sub-sequences and associated scalars represent the various design parameters for the target instance of the target node type. In our example, a register file may contain some number of registers. Each register has a name, index value, bit width and other such parameters. Each specific node type includes unique, recognizable scalar parameters (detailed in the sections below). The scalar values assigned to each parameter may include text, boolean values (`true`, `false`), integers and floating point values.

The YAML formatting paradigm uses indentation (spaces, *NOT* tabs) in order to define hierarchies and inheritance. For example, in our register node description, you'll find that there is a top-level `Register` designator with multiple sequences of registers, each marked with a top-level `RegName` to denote an individual register. We see an example of this in Listing 1.

```
Registers:
  - RegName: TEST46.0.reg
    Width: 64
    Index: 0
    PseudoName: pseudoreg.0
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
  - RegName: TEST46.1.reg
    Width: 64
    Index: 1
    PseudoName: pseudoreg.1
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
```

Listing 1: Example IR YAML Indentation

In addition to the basic ability to describe a design in the CoreGen IR YAML, the IR format also has the ability to describe the inherent connectivity and dependence between hardware modules. The naming conventions utilized by each individual instance of a node can be utilized to link modules together in order to preserve module dependence or physical hardware connectivity. For example, a register class utilized to describe inputs for a given instruction may have dependencies on individual registers. Likewise, an instruction format may require register classes as inputs. An example of describing this hierarchy is show in Listing 2.

```
Registers:
  - RegName: TEST46.0.reg
    Width: 64
    Index: 0
    PseudoName: pseudoreg.0
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
  - RegName: TEST46.1.reg
    Width: 64
    Index: 1
    PseudoName: pseudoreg.1
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
```

```
RegClasses:
  - RegisterClassName: TEST46.regclass
    Registers:
      - TEST46.0.reg
      - TEST46.1.reg
InstFormats:
  - InstFormatName: TEST46.if
    ISA: TEST46.isa
    FormatWidth: 32
    Fields:
      - FieldName: opcode
        FieldType: CGInstCode
        FieldWidth: 8
        StartBit: 0
        EndBit: 7
        MandatoryField: true
      - FieldName: RB
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 8
        EndBit: 15
        MandatoryField: false
        RegClass: TEST46.regclass
      - FieldName: RA
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 16
        EndBit: 23
        MandatoryField: false
        RegClass: TEST46.regclass
      - FieldName: RT
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 24
        EndBit: 31
        MandatoryField: false
        RegClass: TEST46.csrregclass
```

Listing 2: Example IR Node Linkage

The CoreGen IR unique IR node types to represent each individual style of hardware module. We briefly list the distinct node types as follows (each of with its own set of unique parameters):

- **SoC**: A *System on Chip*, or SoC node encapsulates a full SoC design. It is effectively a container for larger designs.
- **Core**: A core node contains the necessary arithmetic and register logic to implement a single core.
- **Instruction Format**: An instruction format node contains the necessary logic to define an instruction encoding.
- **Instruction**: An instruction is a representation of a single operation to take place within a core. This may include arithmetic, memory or other styles of operations.
- **Pseudo Instruction**: Pseudo instructions are templated versions of individual instructions, each

with a unique name. These are generally utilized to make an instruction set more expressive without expanding the encoding space.

- **Register Class**: Register class nodes are containers for some number of individual registers that are encapsulated into a hardware register file.
- **Register**: Register nodes are utilized to represent individual hardware registers
- **Instruction Set Architecture**: An instruction set architecture node is utilized to encapsulate multiple instructions and their constituent encodings.
- **Cache**: Cache nodes are utilized to represent an individual layer in a caching hierarchy (for example, an instruction cache or L1 data cache).
- **Encoding**: Encoding nodes are utilized to contain the encodings for individual fields of various structures (such as instruction formats)
- **Communication Link**: Communication nodes are utilized to represent physical links between individual modules.
- **Scratchpad**: Scratchpad nodes are utilized to represent special cases of embedded memories that are contained within an SoC.
- **Memory Controller**: Memory controller nodes are utilized to represent controller modules that connect to external memories.
- **Virtual to Physical Translation Unit**: Virtual to physical translation nodes are utilized to represent the virtual to physical translation mechanisms present within an SoC.
- **DataPath**: Data path nodes define data paths within a core's pipeline.
- **Extension**: Extensions are special node types in CoreGen. Extensions have the ability to extend existing modules or module hierarchies that are considered to be self contained. These nodes are utilized to extend existing designs in a modular fashion within breaking the internal dependencies present in already proven designs.
- **Plugin**: Plugin nodes are special node types in CoreGen. Plugins can be utilized to represent modules that do not follow the predefined node types already defined in CoreGen. Plugins can have arbitrary set of parameters that are defined by the plugin architecture. These parameters can be utilized to steer custom code generation mechanisms for HDL and other artifacts. Plugins can also be utilized to override any existing CoreGen node in a templated fashion.

Each of the aforementioned node types follows are predefined hierarchy and dependence profile. We represent these dependencies in Figure 1. The dependencies are maintained via the standard CoreGen IR formatting. As a result, dependencies are naturally expressed in the IR.

The two exceptions to this rule are *Extensions* and *Plugins*. These can effectively fall anywhere within the IR hierarchy. Further, both extensions and plugins can contain dependent nodes within themselves in order to express additional information. We will further see how these function in Sections 2.20 and 2.21.

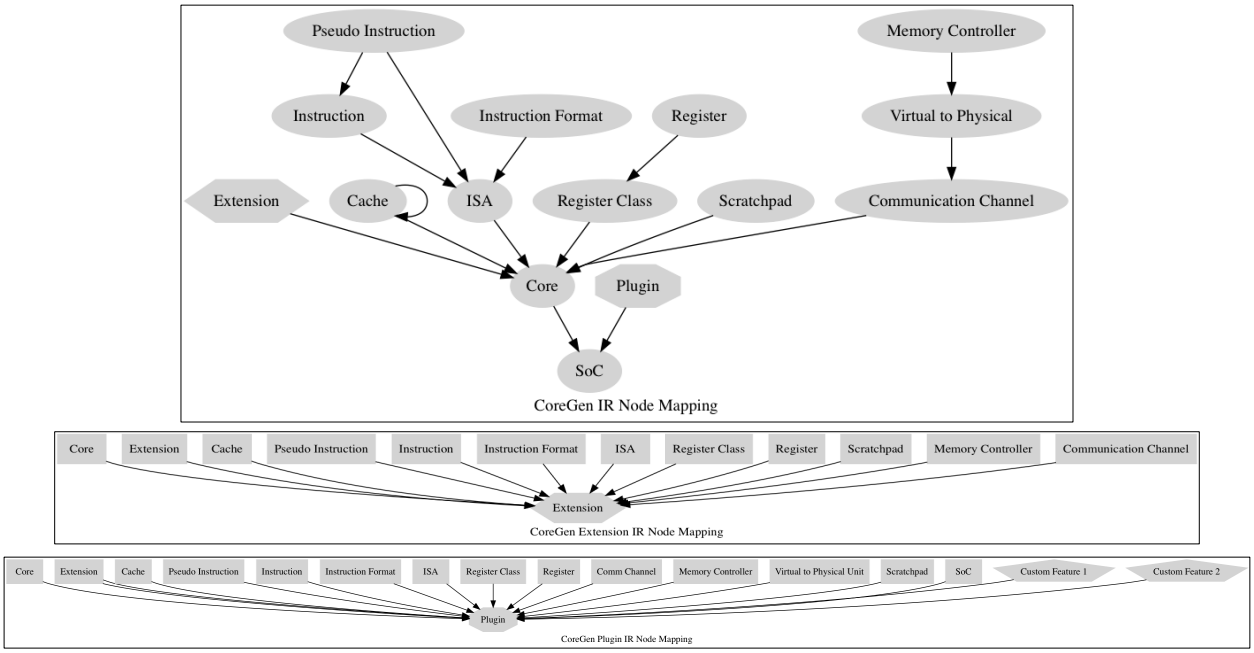

Figure 1: CoreGen IR Node Hierarchy

1.1 Required Packages and Manipulating IR

Given the textual nature of the CoreGen intermediate representation, there are no external packages required to read and/or write the IR. However, we *highly* recommend that you utilize the integrated CoreGen tooling in order to maintain the internal dependencies and verify the syntactical correctness of the IR. Further, parsing the YAML using non-CoreGen tools may result in unresolved internal dependency links due to incorrect parsing order. Regardless of the format of the IR text, the CoreGen tools will parse the file in the correct nodal order.

2 CoreGen IR Nodes

In this section, we outline how each individual node type is constructed, how they link to other nodes and the syntactical definitions utilized therein. Each section describes the generic format of the node. The sequences and sequence elements will include scalar values in one of the following types (Table 2):

Table 2: CoreGen IR Scalar Types

Type	Description
`String`	Strings are utilized to represent names and references to names. In some instances, Strings are used to represent special object types.
`Bool`	Bool values are utilized to represent pure booleans. They can be either `true` or `false`.
`Integer`	Integer values are utilized to represent indices and basic parameter values. Where appropriate, we indicate the perceived types.
`Special`	Special objects a textual keywords used to trigger specific architectural features. These are represented as: {`Value1,Value2,...`}.

2.1 Node Names and Linkage

Each IR node in the CoreGen IR format has a unique name that represents a unique hardware object. Names must be unique across all IR nodes. We require unique names as the respective node names are utilized to link nodes in a dependent manner across the design. For example, a `core` node may have a dependent `cache` node. We provide an example node dependency using these names in Listing 3. Notice how we define an instruction node (`TEST54.inst0`) and a pseudo instruction node (`TEST54.pinst0`) that depends upon the core instruction by name.

Given this use of textual names in the IR, we must define a valid naming convention that is preserved in the IR. The valid node name rules are listed as follows:

- Names must be non null (*length greater than 0*)
- Names must begin with an alpha character (*[a-z,A-Z]*)
- Names must contain only alphanumeric characters and periods (*[a-z,A-Z,0-9,.]*)
- Names are case sensitive, but can include any combination of upper and lower case characters

Example valid and invalid node names are listed in Table 3.

Table 3: CoreGen IR Valid and Invalid Node Names

Valid	Invalid
nodename, NodeName, nodeName0, Node.Name.9	9Name, Node&Name, %NodeName

```
Insts:
  - Inst: TEST54.inst0
    ISA: TEST54.isa
    InstFormat: TEST54.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 1
PseudoInsts:
  - PseudoInst: TEST54.pinst0
    ISA: TEST54.isa
    Inst: TEST54.inst0
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 0
```

Listing 3: Node Naming and Linkage

2.2 User Defined RTL

For each of the forthcoming node types, users have the ability to override the default CoreGen code generation facilities in order to include custom RTL. Currently, The CoreGen IR specification permits users to utilize this feature in two ways. First, users have the ability to define inline RTL. This RTL can currently be written in either Chisel (Chisel 3) or Verilog. Inline RTL must follow the naming conventions and port definitions supported by the CoreGen code generation facilities (NEED REFERENCE). An example of defining inline RTL for a single instruction in Chisel3 is shown in Listing 4. Notice how the inline RTL source code is quoted. For each inline RTL sample provided with the CoreGen node, you must also specify the `RTLType`. Types can be one of `Chisel`, `Verilog` and `Unknown`.

```
Insts:
  - Inst: TEST69.inst0
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 1
    RTL: "when( RA === 0x05){
            RT = RA << RB
          } .otherwise {
            RT = RA >> RB
          }"
     RTLType: Chisel
```

Listing 4: Sample Inline RTL

The second path by which users may override a default node using RTL is by utilizing an external RTL file. By utilizing the `RTLFile` parameter with any node, the user may specify an external RTL file that is managed outside of CoreGen and its constituent tools. Take note that the external RTL file is relative to the project path utilized by the CoreGen tools. An example of doing so is shown in Listing 5.

```
Insts:
  - Inst: TEST69.inst0
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 1
    RTLFile: Sample.v
    RTLType: Verilog
```

Listing 5: Sample External RTL File

2.3 Node Override

In addition to providing the ability to override each node with user-defined RTL, the CoreGen IR infrastructure also provides the ability to override an entire node's construction with user-defined logic. This mechanism is analogous to utilizing templates in modern object-oriented programming models. Users have the ability to utilize *plugins* to override the default behavior and/or define non-standard features for a target IR node. This feature can also be utilized to integrate pre-existing hardware IP into new designs. This functionality can be performed on all nodes except plugins, extensions, encoding and pseudo instructions.

An example of utilizing this feature is provided in Listing 6. The first step in this process is to define a plugin node that will be utilized to override the default functionality of a standard node. In this case, we utilize a pre-existing `CacheLayerPlugin` and define a `CachePlugin` node. This theoretical plugin node instantiates an N-level cache with pre-existing IP where the number of cache levels is defined by a custom feature (`Levels`). With this plugin node defined, we can now define a base cache node defined as the `L0.Cache`. Within this cache node, we use the `Override` keyword and attach the plugin node definition that we previously defined (`CachePlugin`). Note that you **must** utilize the plugin node name (`Plugin`), not the actual name of the plugin implementation (`PluginName`). The CoreGen IR supports multiple instantiations of the same plugin implementation under different names. This permits users to re-use existing IP and define unique features for each instance of the plugin. For more information regarding the definition and construction of the actual `Plugin` nodes, see Section 2.21.

```
1  Plugins:
2    - Plugin: CachePlugin
3      PluginName: CacheLayerPlugin
4      MajorVersion: 1
5      MinorVersion: 0
6      PatchVersion: 0
7      Features:
8      - FeatureName: Levels
9        FeatureType: unsigned
10       FeatureValue: 3
11 Caches:
12   - Cache: L0.Cache
13     Sets: 1024
14     Ways: 8
15     Override: CachePlugin
```

Listing 6: Sample Node Override

2.4 Node Notes

Each node has an additional attribute that permits users to add specific *notes* or comments for future reference. These `Notes` fields do not affect the overall functionality, stability or connectivity of the constituent design. Rather, these attributes may be utilized, queried or processed by downstream tools. For example, the CoreGen *SpecDoc* pass utilizes these notes to output specific information for constituent design elements. An example of doing so is shown in Listing 7.

```
1 Insts:
2   - Inst: TEST69.inst0
3     ISA: TEST69.isa
4     InstFormat: TEST69.if
5     Encodings:
6       - EncodingField: opcode
7         EncodingWidth: 8
8         EncodingValue: 1
9     Notes: This instruction performs carry-less addition.
```

Listing 7: Sample Node Notes

2.5 Projects

CoreGen IR files may provide an optional project definition block. This block of IR does not contain any dependencies with other IR nodes. Rather, it is designed to provide tools with additional information regarding the location and target function of the IR contained therein. Project nodes are utilized to define project names, paths, project types and Chisel output versions. Currently, we support four disparate project types as show in Table 4.

Table 4: CoreGen Project Types

Parameter	Description
`soc`	System on chip projects are full designs with deep hierarchies of nodes
`module`	Module projects are designed to be very compact hierarchies of nodes potentially with custom RTL
`extension`	Extension projects encapsulate a single block of extension IR nodes to be utilized in other projects
`unknown`	All other projects fall into the *unknown* category

We see an example block of project node IR in Listing 8.

```
1 ProjectInfo:
2   - ProjectName: String
3     ProjectRoot: String
4     ProjectType: Special
5     ChiselMajorVersion: Integer
6     ChiselMinorVersion: Integer
```

Listing 8: Project Node Definition

2.6 Registers

Register nodes define a single instance of a register in hardware. Registers may also have named aliases, referred to as a `PseudoName`. Further, subregisters (`SubRegs`) can be defined whereby a unique pseudoname is utilized to access a specific portion of the parent register. For example, if we define a parent register *FOO* with a width of 64 bits, we may also define a `SubReg` *FOO.BAR* to alias bits `[31-0]` (the lower half of the register).

Additional parameters are described in Table 5. We provide a sample register IR block in Listing 9.

Table 5: CoreGen Register Node Parameters

Parameter	Description
`Width`	Defines the width (in bits) of the register. (Int)
`Index`	Defines the index value of the register. (Int)
`PseudoName`	[Optional] Defines a register pseudo name (String)
`IsFixedValue`	Sets a parameter indicating that the register has a read-only, fixed value. (Bool).
`IsSIMD`	Sets a parameter indicating that the register is a SIMD register. The register width must encapsulate the entire SIMD block. (Bool)
`RWReg`	Sets a parameter indicating that the register is a read-write register (Bool)
`ROReg`	Sets a parameter indicating that the register is a read-write register Cannot be `true` if RWReg is `true`. (Bool)
`CSRReg`	Sets a parameter indicating that the register is a control status register. This can be read-only or read-write. (Bool)
`AMSReg`	Sets a parameter indicating that the register is an arithmetic machine state register. The hardware code generator will integrate these into the pipeline. (Bool)
`TUSReg`	Sets a parameter indicating that the register is shared across all thread units within the core. Either `Shared` or `TUSReg` can be set, but not both. (Bool)
`PCReg`	Sets a parameter indicating that the register is designated as a program counter. This attribute may only exist in a single register per instruction set. (Bool)
`Shared`	Sets a parameter indicating that the register is a shared register. The hardware generator will create **ONE** instance of the register for all cores. (Bool)
SubRegs	
`SubReg`	Sets the unique subregister name. The name must be unique across all registers within the register class (String)
`StartBit`	Sets the starting bit position of the subregister in the parent register. The start bit must be less than the end bit and a minimum of one bit. (Int)
`EndBit`	Sets the ending bit position of the subregister in the parent register. The end bit must be greater than the start bit and a minimum of one bit (Int)

```
Registers:
  - RegName: String
    Width: Integer
    Index: Integer
    PseudoName: String
    IsFixedValue: Bool
    IsSIMD: Bool
    RWReg: Bool
    ROReg: Bool
    CSRReg: Bool
    AMSReg: Bool
    TUSReg: Bool
    PCReg: Bool
    Shared: Bool
    SubRegs:
      - SubReg: String
        StartBit: Integer
        EndBit: Integer
      - SubReg: String
        StartBit: Integer
        EndBit: Integer
  - RegName: String
    Width: Integer
    Index: Integer
    PseudoName: String
    IsFixedValue: Bool
    IsSIMD: Bool
    RWReg: Bool
    ROReg: Bool
    CSRReg: Bool
    AMSReg: Bool
    TUSReg: Bool
    Shared: Bool
   ...more registers
```

Listing 9: Register Node Definition

2.7 Register Classes

Register class nodes define an instance of a register file. The register file will contain some number of registers, each of which may have specific attributes. Indexing into the register file is defined by the registers contained therein and *does not* need to be monotonically increasing. Despite the ability to registers within the same register file that have disparate attributes, the HDL code generation mechanisms in CoreGen will select the largest bit width register in order to determine the space requirements of the register file. As a result, if the design requires several different sizes of registers, it is advantageous to separate them into distinct register files for the purpose of optimizing the eventual RTL.

Register classes are defined by two parameters: the register class name and the registers that exist within the register class. The registers defined in the register class are references to register defined in the register node section (See Section 2.6). The register naming convention must match that of the defined registers. Register classes may also define the number of read ports and write ports for the associated register block. These are optional values that default to two (2) read ports and (1) write ports when not specified. Note that when the respective register class contains at least a single register defined to be writable, then the associated register class must include at least one write port. All register classes must include at least one read port. We provide a sample register class IR block in Listing 10.

```
1  RegClasses:
2    - RegisterClassName: String
3      Registers:
4        - String
5        - String
6        - String
7    - RegisterClassName: String
8      ReadPorts: Unsigned Integer
9      WritePorts: Unsigned Integer
10     Registers:
11       - String
12       - String
13       - String
14   ...more reg classes
```

Listing 10: Register Class Node Definition

2.8 Instruction Sets

Instruction set nodes are effectively container nodes. They are designated as a container for multiple instructions and instruction formats. As a result, the only parameter utilized to identify an individual instruction set is its respective name. Designs may have any number of instruction sets. We provide an example instruction set IR block in Listing 11.

```
1 ISAs:
2   - ISAName: String
3   - ISAName: String
```

Listing 11: Instruction Set Node Definition

2.9 Instruction Formats

The instruction format nodes define the formatting information, including field types, field widths and required information for each instruction format in **little endian**. The instruction format nodes do not define the actual instructions, only the encoding formats. Instruction formats can contain any number of instruction fields. The minimum field width is a single bit. The maximum field width is equivalent to the entire width of the instruction format (`FormatWidth`).

Each width must be designated as one of three potential field types. For register fields, you **must** also specify the target register class that is associated with the field (`RegClass`). Note that the width of the field should encapsulate the maximum potential index for the target register class. For instruction code fields, the width of the field will define the number of encodings.

Finally, instruction encodings are not required to contain contiguous fields. You may opt to leave undefined space between fields. The width of the encoding must match the largest ending bit field. For example, if your instruction format has the last field defined to end on bit 31 (`EndBit`), then your entire encoding must be at least 32 bits.

Table 6: CoreGen Instruction Format Node Parameters

Parameter	**Description**
`ISA`	Defines the instruction set architecture container. (String)
`FormatWidth`	Defines the width (in bits) of the register. (Int)
Fields	
`FieldName`	Defines the name of the respective field. (String)
`FieldType`	Defines the field type. This must be one of the following. (Special) `CGInstReg` : Defines a register field. Requires `RegClass` be defined. `CGInstCode` : Defines an instruction code (opcode, function code, etc). `CGInstImm` : Defines a field for an immediate value.
`FieldWidth`	Defines the width of the instruction field. (Integer)
`StartBit`	Defines the starting bit position of the field. (Integer)
`EndBit`	Defines the end bit position of the field. (Integer)
`MandatoryField`	Determines whether the field **must** be defined by the implementing instruction. (Bool)
`RegIsDestination`	Determines if the register field is a target register. (Bool)
`RegClass`	Sets the target register class parameter for `CGInstReg` fields. (String)

We provide an example Instruction Format IR block in Listing 12.

```
1  InstFormats:
2    - InstFormatName: String
3      ISA: String
4      FormatWidth: Integer
5      Fields:
6        - FieldName: String
7          FieldType: {CGInstReg, CGInstCode, CGInstImm}
8          FieldWidth: Integer
9          StartBit: Integer
10         EndBit: Integer
11         MandatoryField: Bool
12       - FieldName: String
13         FieldType: {CGInstReg, CGInstCode, CGInstImm}
14         FieldWidth: Integer
```

```
15          StartBit: Integer
16          EndBit: Integer
17          MandatoryField: Bool
18          RegClass: String
19          RegIsDestination: Bool
20        - FieldName: String
21          FieldType: {CGInstReg, CGInstCode, CGInstImm}
22          FieldWidth: Integer
23          StartBit: Integer
24          EndBit: Integer
25          MandatoryField: Bool
26          RegClass: String
27        - FieldName: String
28          FieldType: {CGInstReg, CGInstCode, CGInstImm}
29          FieldWidth: Integer
30          StartBit: Integer
31          EndBit: Integer
32          MandatoryField: Bool
33          RegClass: String
```

Listing 12: Instruction Format Node Definition

2.10 Instructions

Instruction nodes represent individual instances of single instructions. Each instruction node is attached to an instruction set reference as well as a specific instruction format. In addition to the aforementioned references, each instruction has the ability to include any necessary encoding information as well. A single instruction can define multiple encoding blocks, each with specific encoding information that is mapped back to an individual named field in the instruction format. For example, if the instruction format has a field called *opcode* that has the `MandatoryField` flag set to `true`, the instruction node may have an `Encoding` definition with the respective value for that instruction.

Note that it is up to the user to avoid collisions in the encodings across multiple instructions. The CoreGen tools may provide IR passes that check for these idiosyncracies, but the IR does not inherently prohibit collisions. Further, encoding definitions require that the user specify the encoding width (`EncodingWidth`). For opcode fields, this should be set to the same width as defined in the instruction encoding (`FieldWidth`). However, for fields such as immediate values, setting this width to something less than the full field width with automatically truncate the value to what is specified. For example, if the instruction encoding field for the immediate is 16 bits, but you define an instruction with a 12 bit immediate, the value in the immediate field of that instruction will be truncated to 12 bits.

Each instruction nodes can also drive the syntax by which the compiler's assembly code is generated and parsed. The `Syntax` field contains a string value in a specific format that defines the instruction mnemonics and associated syntax that is utilized to construct the associated assembler and disassembler. This syntax is constructed in a specific manner such that the CoreGen infrastructure has the ability to match the respective mnemonic to the correct instruction format field. The syntax must include the instruction name (which may differ from the instruction node name) and some number arguments. The argument names must match the respective register class or immediate field names of the respective instruction format. Register class arguments are delineated with *%* characters and immediate arguments are delineated with *$* characters. User may also intersperse commas and parenthesis throughout the syntax at will.

For example, consider an instruction `ADD` whose instruction format has three register class fields: `RT`, `RA` and `RB`. We may specify the instruction syntax as follows: `ADD %RT, %RA, %RB`. We may also have an additional instruction, `LOAD`, whose instruction format contains two register class fields (`RT`, `RA`) and a single immediate field (`IMM`). Our instruction syntax may then be formatted as: `LOAD %RT, $IMM(%RA)`. Note the use of the preceding characters for the register class and immediate field designators.

Finally, instructions have the ability to include inline StoneCutter definitions for each instruction (NEED REFERENCE). These inline StoneCutter instruction definitions are compiled consolidated into larger StoneCutter source files using the CoreGen infrastructure and subsequently compiled into Chisel HDL. Inline StoneCutter language is designated using `Impl` attributes within an individual instruction definition.

We provide an example instruction node in Listing 13.

```
Insts:
  - Inst: String
    ISA: String
    InstFormat: String
    Syntax: String
    Encodings:
      - EncodingField: String
        EncodingWidth: Integer
        EncodingValue: Integer
  - Inst: String
    ISA: String
    InstFormat: String
    Syntax: String
    Encodings:
      - EncodingField: String
        EncodingWidth: Integer
        EncodingValue: Integer
    Impl: String
```

Listing 13: Instruction Node Definition

2.11 Pseudo Instructions

Pseudo instruction nodes are special types of instruction nodes. They do not represent actual instructions. Rather, pseudo instructions contain special encoding directions for previously defined instructions and/or new instruction names. For example, in traditional RISC architectures register moves are implemented as unsigned addition instructions whereby the immediate values are set to zero. An example of this is noted as follows:

```
#  register R0 = zero
add r10, r11, r0  # R10 = R11+R0
mov r10, r11      # Pseudo instruction for 'add r10, r11, r0'
```

Pseudo instructions require a unique name, the associated ISA and the target instruction that you seek to alias. You must also define a set of encoding parameters that are utilized to incorporate specific encodings for specific fields. For example, if you seek to define a pseudo instruction whereby an individual register file is always set to the same register index, utilize a pseudo instruction with an encoding field. Further, much like the original instruction definition, pseudo instructions permit the user to define assembly mnemonics that are echoed in the compiler implementation. The assembly mnemonics follow the syntax rules defined in Section 2.10. An example set of pseudo instructions is provided in Listing 14.

```
1  PseudoInsts:
2    - PseudoInst: String
3      ISA: String
4      Inst: String
5      Encodings:
6        - EncodingField: String
7          EncodingWidth: Integer
8          EncodingValue: Integer
9      Syntax: String
10   - PseudoInst: String
11     ISA: String
12     Inst: String
13     Encodings:
14       - EncodingField: String
15         EncodingWidth: Integer
16         EncodingValue: Integer
17     Syntax: String
```

Listing 14: Pseudo Instruction Node Definition

2.12 Caches

Cache nodes represent a single level of a traditional cache hierarchy. Regardless of whether the target use is designated as an instruction cache, data cache or otherwise, the cache node is defined in the same manner. The target use is determined by how the cache is connected to the core (or cores). Cache nodes are required to have at least two parameters: `Sets` and `Ways`. These represent the number of cache sets and the number of *ways* are represented in the cache. By default, caches are defined to utilize a standard, 64-byte cache line size. Users may override this by specifying an optional parameter, `LineSize` with the size of each cache line in bytes.

Caches can also be configured in hierarchies. Each level in the cache must be defined as a cache node. To define a cache hierarchy, start with the cache layer closest to the core (usually defined as the *L1* layer) and add a `SubLevel` to the cache. The sublevel defines a connection to the next layer in the cache hierarchy. Nodes may exist as parents (sublevel to a lower level cache) and as a parent (contain a sublevel). You may define any number of caching layers providing that the connectivity is present between layers. An example caching hierarchy is providing in Listing 15.

```
1  Caches:
2    - Cache: String
3      Sets: Integer
4      Ways: Integer
5    - Cache: String
6      Sets: Integer
7      Ways: Integer
8      SubLevel: String
9    - Cache: String
10     Sets: Integer
11     Ways: Integer
12     LineSize: Integer
```

Listing 15: Cache Node Definition

2.13 Scratchpad Memories

Scratchpad nodes are utilized to represent special case memory nodes that reside on die (as opposed to external memories such as DRAM). Further, these memory nodes participate in the normal addressing mechanisms, unlike cache hierarchies which are implied as a part of the memory pipeline.

Scratchpad nodes can be unique for individual cores or attached to multiple cores or other nodes. Each scratchpad memory node must include the size of the scratchpad (`MemSize`), the number of request ports (`RqstPorts`), the number of response ports (`RspPorts`) and the starting address (`StartAddr`). The total address space provided by the individual scratchpad is *StartAddr + MemSize*. Note that individual scratchpads may have the same address space providing that they are not connected to the same core. Further, multiple scratchpads can be connected to individual cores providing that the address spaces do not collide. An example scratchpad node definition is provided in Listing 16.

```
1  Scratchpads:
2    - Scratchpad: String
3      MemSize: Integer
4      RqstPorts: Integer
5      RspPorts: Integer
6      StartAddr: Integer
7    - Scratchpad: String
8      MemSize: Integer
9      RqstPorts: Integer
10     RspPorts: Integer
11     StartAddr: Integer
```

Listing 16: Scratchpad Node Definition

2.14 Virtual to Physical Translation Units

The virtual to physical (*VTP*) nodes include the ability to perform virtual to physical memory address translation in conjunction with the backend memory controller and scratchpad memory nodes. Currently, the VTP nodes require a single parameter, the name of the VTP node. Note that it may be inherently dangerous to have multiple VTP nodes in a single design without sufficient separation. For example, a heterogeneous design that contains multiple, tightly integrated cores may still have a single VTP node. However, a design that includes a loosely connected accelerator (with its own constituent memory controller) may perform its own virtual to physical translation. An example of the VTP node syntax is shown in Listing 17.

```
1 VTPControllers:
2   - VTP: String
```

Listing 17: VTP Node Definition

2.15 Memory Controllers

Memory controller nodes are designed to receive physical addresses from upstream devices. The memory controller subsequently transfer the requests to the external memory devices. Many designs may warrant overriding these nodes with external RTL IP given design-specific external memory requirements. Otherwise, memory controller nodes require a unique node name as well as the number of upstream memory ports. Currently, the number of ports (`Ports`) should be an even number as these are divided evenly between request ports and response ports. In addition to the number of ports, memory controllers may optionally specify the memory ordering for the respective controller. The `MemoryOrder` parameter takes a string argument in one of the following types: `Weak`, `TSO` or `Strong`. The `Weak` parameter (which is also the default if not specified) instructs the memory controller to function in pure weak ordering for loads and stores. The `TSO` parameter instructs the memory controller to function in *total-store-ordering* mode. Finally, the `Strong` parameter enforces *strong* memory ordering.

An example memory controller IR block is shown in Listing 18.

```
1 MemoryControllers:
2   - MemoryController: String
3     Ports: Integer
4   - MemoryController: String
5     Ports: Integer
6     MemoryOrder: {Weak,TSO,Strong}
```

Listing 18: Memory Controller Node Definition

2.16 Communication Channels

Communication nodes provide connectivity between disparate nodes that would otherwise have no implied connectivity. For example, if you connect a cache node to a core, there is an implied link via the memory pipeline. However, if you connect a set of cores together (via a bus or other topology), there is no implication of how they may communicate. Communication nodes are utilized to connect various nodes or groups of nodes together into a singular topology. Nodes may include connectivity between multiple communication nodes in different networks. For example, a core may include connectivity to four adjacent cores in order to form a on-chip mesh interconnect.

Communication nodes require four main parameters as noted in Table 7. We show an example Communication node in Listing 19.

Table 7: CoreGen Comm Node Parameters

Parameter	Description
`Comm`	Defines the unique name of the communication node. (String)
`Type`	Defines the communication node type. This must be one of the following. (Special) `P2P` : Defines a peer-to-peer communication channel for two node endpoints. `Bus` : Defines a bus communication channel with min 2 endpoints and no maximum. `NOC` : Defines a network on chip communication channel with min 2 endpoints and no maximum. `Unknown` : User-defined communication channel that requires user-provided RTL.
`Endpoints`	This contains a sequenced list of endpoint nodes that are connected to the communication channel.

```
1 Comms:
2   - Comm: String
3     Type: {P2P,Bus,NOC,Unknown}
4     Width: Integer
5     Endpoints:
6       - String
7       - String
8       - String
```

Listing 19: Communication Node Definition

2.17 Cores

Core nodes define a single instance of a traditional "core". This includes arithmetic facilities, register files and any required encoding data (instruction formats, register indices, etc). Each core node must include an associated instruction set definition (`ISA`), any required register classes for the target ISA (`RegisterClasses`) and optionally, one or more cache units. While caching units are optional dependent nodes for a core, we highly recommend that cores have at least an instruction cache facility. Otherwise, the instruction fetch performance of the core will be very poor.

In addition to the aforementioned base nodes, cores can be augmented with one or more extensions. These extensions may include additional, registers, register classes or other non-standard nodes beyond what is defined in the base core infrastructure. Further, the cores may define the style of data path utilized to implement the respective ISA.

Core nodes may also define the degree of symmetric multithreading (SMT) by specifying the number of `ThreadUnits`. This optional keyword permits users to specify the number of unique thread units for the respective core. The hardware code generator will output unique register files for each thread unit unless a register has been designated as *thread unit shared* (`TUSReg`) or `Shared`. See Section 2.6 for descriptions of these register attributes. The code generator will not, however, output unique `Extensions` for each thread unit. A single instance of a shared register will be generated and shared across the cores. If not specified, the number of `ThreadUnits` is assumed to be *1*.

We see an example of a basic and extended core node in Listing 20.

```
1  - Core: String
2      Cache: String
3      ISA: String
4      Datapath: String
5      ThreadUnits: 2
6      RegisterClasses:
7        - RegClass: String
8        - RegClass: String
9    - Core: String
10     Cache: String
11     ISA: String
12     RegisterClasses:
13       - RegClass: String
14       - RegClass: String
15     Extensions:
16       - Extension: String
```

Listing 20: Core Node Definition

2.18 Data Paths

CoreGen DataPath nodes define the structure of the data path within a core node's ISA implementation. This is effectively the pipeline model for the respective core. The data path may have one or more pipelines defined which requires a name and an associated style. Currently, we only support *In-Order* pipeline styles.

```
1 DataPaths:
2   - Pipeline: String
3     Style: String
4   - Pipeline: String
5     Style: String
```

Listing 21: Data Path Node Definition

2.19 SoCs

SoC (or *System on Chip*) nodes are top-level, container nodes that are utilized to tie multiple cores and their dependent nodes together into a single package. Generally speaking, there should be only a single SoC node in a design. It is possible to have multiple SoC nodes in a given IR file, but the CoreGen pass infrastructure will likely flag this as being potentially erroneous. For each SoC node, you must define some number of cores that are attached. An example of this is shown in Listing 22.

```
1 Socs:
2   - Soc: String
3     Cores:
4       - Core: String
5       - Core: String
```

Listing 22: SoC Node Definition

2.20 Extensions

Extensions are unique container node types in CoreGen IR that represent nodes (or groups of nodes) that don't inherently fall into one of the other standard categories. Generally speaking, the node infrastructure within a given extension is not modified for a given design (although it can be modified if desired). An extension may fall into one of the following types:

- **Templates** : Template extensions are utilized to construct larger designs using basic building blocks of nodes. For example, a template extension may include a single core, L1 cache and communication link. This template extension could then be replicated multiple times within a design to build larger, multi-core SoC's.
- **Modules** : Module extensions are utilized to construct highly optimized hardware modules. Module extensions may include other node types, but often include user-provided RTL. Module extensions are analogous to "black box" IP that is re-used, but not modified.
- **Comms** : Communication extensions are utilized to construct special cases of templated extensions. These extensions are designed to construct more expressive communication infrastructures. For example, one may utilize a communication extension to construct a switching block for a custom network on chip architecture.
- **Unknown** : Unknown extensions are extensions that do not fall into the aforementioned types.

Given the containerized nature of extensions, one may include multiple types and hierarchies of other nodes within the extension. The extension may be very simple (single nodes) or arbitrarily complex. Extensions may include one or more of the following node types:

- Registers
- Register Classes
- ISAs
- Instruction Formats
- Instructions
- Pseudo Instructions
- Caches
- Cores
- Scratchpads
- Memory Controllers
- Communication Channels
- Extensions

We provide an example Extension IR block in Listing 23.

```
Extensions:
  - Extension: String
    Type: {Template, Module, Comm, Unknown}
    Registers:
      - RegName: String
        Width: Integer
        Index: Integer
        IsFixedValue: Bool
        IsSIMD: Bool
        RWReg: Bool
        ROReg: Bool
        CSRReg: Bool
        AMSReg: Bool
        Shared: Bool
    RegClasses:
      - RegisterClassName: String
        Registers:
          - String
    ISAs:
      - ISAName: String
    RTLFile: String
```

Listing 23: Extension Node Definition

2.21 Plugins

The final and most complex style of CoreGen IR node is the plugin node. These nodes contain both basic node data within the IR as well as the potential to trigger user-defined backend code generation mechanisms. Plugins are implemented as shared libraries that may contain one or more of the following:

- **Custom Node Features** : Plugin nodes have the ability to define an arbitrary set of features that can be modified by the user. Unlike *extensions* (Section 2.20) that may only include existing CoreGen node types as dependencies, plugins may include custom parameters. These parameters are defined in terms of their rudimentary types. We will describe this typing system below.

- **Custom Code Generation Mechanisms** : Given that each plugin is encapsulated in a shared library, the custom parameters defined above can be utilized to drive custom code generation mechanisms. This can be done in two ways. First, for existing RTL in the form of Chisel or Verilog, the custom parameter values can be utilized to drive custom CoreGen code generation output for the target plugin. Second, custom language backends can be utilized to output plugin-generated code in whatever form is necessary. However, be aware that any custom code generation facilities must adhere to the basic CoreGen naming conventions.

- **Custom Optimizations** : In addition to custom code generation mechanisms, plugins may also include custom optimizations. This includes plugin-specific interpretations of parameter values, dynamically generated child node dependencies and internal peephole optimizations.

For each plugin node defined in the IR, there are several required parameters that must be defined. First, each plugin must have a unique node name (`Plugin`). As is the case with other nodes, this parameter uniquely identifies the node with the dependence graph. Next, each plugin node must define its associated plugin library name (`PluginName`). This is the name of the plugin library utilized to implement the plugin. The `PluginName` *does not* contain any library prefix or file extensions. For example, if the target library was named *libSample.so*, the `PluginName` parameter would be *Sample*. Finally, the plugin node must initialize the major, minor and patch version numbers for the target plugin. In this manner, multiple versions of the same plugin can be installed on the development system and the CoreGen infrastructure will choose the correct version for the design. We see a full list of potential plugin parameters in Table 8.

Table 8: CoreGen Plugin Node Parameters

Parameter	Description
`Plugin`	Defines the target plugin unique *node* name. (String)
`PluginName`	Defines the name of the plugin library to load (no file extension). (String)
`MajorVersion`	Major version of the plugin to load. (Integer)
`MinorVersion`	Minor version of the plugin to load. (Integer)
`PatchVersion`	Patch version of the plugin to load. (Integer)
Features	
`FeatureName`	Name of the plugin-specific feature. (String)
`FeatureType`	Rudimentary type of the target feature (see list below). (Special)
`FeatureValue`	Value associated with the plugin-specific feature. (Special)

As mentioned above, each of the individual plugin nodes may include plugin-specific feature sets. Each feature must be defined and recognized by the plugin. The features are designated by the `FeatureName` and `FeatureType`. The permissible feature types are outlined in Table 9. For each `FeatureType`, you must also provide a `FeatureValue` parameter that defines the value of the respective feature in the respective data type. For example, if a feature is defined as a `Bool` type, then the value must be one of `true` or `false`.

In addition to the custom feature sets provided by plugin-specific logic, plugin nodes may also contain dependent nodes in a similar manner as extensions. These nodes can be dynamically generated by the actual

Table 9: CoreGen Plugin Node Feature Types

FeatureType	Description
`Unsigned`	unsigned 32 bit integer (C++ *unsigned*)
`Uint32t`	unsigned 32 bit integer (C *uint32_t*)
`Int32t`	signed 32 bit integer (C *int32_t*)
`Uint64t`	unsigned 64 bit integer (C *uint64_t*)
`Int64t`	signed 64 bit integer (C *uint64_t*)
`Float`	IEEE single precision floating point (C *float*)
`Double`	IEEE double precision floating point (C *double*)
`String`	String value (C++ *std::string*)
`Bool`	Boolean value (`true` or `false`)

plugin or explicitly added to the plugin dependency graph. Plugins may define dependent nodes using the following node types:

- Registers
- Register Classes
- ISAs
- Instruction Formats
- Instructions
- Pseudo Instructions
- Caches
- Cores
- Scratchpads
- Memory Controllers
- Communication Channels
- Extensions
- SoC's

We provide a full example IR block for a plugin node in Listing 24

```
1  Plugins:
2    - Plugin: String
3        PluginName: String
4        MajorVersion: Integer
5        MinorVersion: Integer
6        PatchVersion: Integer
7        Features:
8          - FeatureName: String
9            FeatureType: Special
10           FeatureValue: Special
11         - FeatureName: String
```

```
          FeatureType: Special
          FeatureValue: Special
  - Plugin: String
      PluginName: String
      MajorVersion: Integer
      MinorVersion: Integer
      PatchVersion: Integer
      Features:
        - FeatureName: String
          FeatureType: Special
          FeatureValue: Special
        - FeatureName: String
          FeatureType: Special
          FeatureValue: Special
     Registers:
      - RegName: String
        Width: Integer
        Index: Integer
        IsFixedValue: Bool
        IsSIMD: Bool
        RWReg: Bool
        ROReg: Bool
        CSRReg: Bool
        AMSReg: Bool
        Shared: Bool
    RegClasses:
      - RegisterClassName: String
        Registers:
          - String
    ISAs:
      - ISAName: String
```

Listing 24: Plugin Node Definition

3 Appendix A: Sample IR

```
ProjectInfo:
  - ProjectName: TEST69
    ProjectRoot: ./
    ProjectType: unknown
    ChiselMajorVersion: 3
    ChiselMinorVersion: 0
Registers:
  - RegName: TEST69.0.reg
    Width: 64
    Index: 0
    PseudoName: pseudoreg.0
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.1.reg
    Width: 64
    Index: 1
    PseudoName: pseudoreg.1
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.2.reg
    Width: 64
    Index: 2
    PseudoName: pseudoreg.2
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.3.reg
    Width: 64
    Index: 3
    PseudoName: pseudoreg.3
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.4.reg
```

```
    Width: 64
    Index: 4
    PseudoName: pseudoreg.4
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.5.reg
    Width: 64
    Index: 5
    PseudoName: pseudoreg.5
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.6.reg
    Width: 64
    Index: 6
    PseudoName: pseudoreg.6
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.7.reg
    Width: 64
    Index: 7
    PseudoName: pseudoreg.7
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.8.reg
    Width: 64
    Index: 8
    PseudoName: pseudoreg.8
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
```

```
  - RegName: TEST69.9.reg
    Width: 64
    Index: 9
    PseudoName: pseudoreg.9
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.10.reg
    Width: 64
    Index: 10
    PseudoName: pseudoreg.10
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.11.reg
    Width: 64
    Index: 11
    PseudoName: pseudoreg.11
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.12.reg
    Width: 64
    Index: 12
    PseudoName: pseudoreg.12
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.13.reg
    Width: 64
    Index: 13
    PseudoName: pseudoreg.13
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
```

```
    Shared: false
  - RegName: TEST69.14.reg
    Width: 64
    Index: 14
    PseudoName: pseudoreg.14
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.15.reg
    Width: 64
    Index: 15
    PseudoName: pseudoreg.15
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.16.reg
    Width: 64
    Index: 16
    PseudoName: pseudoreg.16
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.17.reg
    Width: 64
    Index: 17
    PseudoName: pseudoreg.17
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.18.reg
    Width: 64
    Index: 18
    PseudoName: pseudoreg.18
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
```

```
    AMSReg: false
    Shared: false
  - RegName: TEST69.19.reg
    Width: 64
    Index: 19
    PseudoName: pseudoreg.19
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.20.reg
    Width: 64
    Index: 20
    PseudoName: pseudoreg.20
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.21.reg
    Width: 64
    Index: 21
    PseudoName: pseudoreg.21
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.22.reg
    Width: 64
    Index: 22
    PseudoName: pseudoreg.22
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.23.reg
    Width: 64
    Index: 23
    PseudoName: pseudoreg.23
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
```

```
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.24.reg
    Width: 64
    Index: 24
    PseudoName: pseudoreg.24
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.25.reg
    Width: 64
    Index: 25
    PseudoName: pseudoreg.25
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.26.reg
    Width: 64
    Index: 26
    PseudoName: pseudoreg.26
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.27.reg
    Width: 64
    Index: 27
    PseudoName: pseudoreg.27
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.28.reg
    Width: 64
    Index: 28
    PseudoName: pseudoreg.28
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
```

```
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.29.reg
    Width: 64
    Index: 29
    PseudoName: pseudoreg.29
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.30.reg
    Width: 64
    Index: 30
    PseudoName: pseudoreg.30
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.31.reg
    Width: 64
    Index: 31
    PseudoName: pseudoreg.31
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.32.reg
    Width: 64
    Index: 32
    PseudoName: pseudoreg.32
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.33.reg
    Width: 64
    Index: 33
    PseudoName: pseudoreg.33
    IsFixedValue: false
    IsSIMD: false
```

```
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.34.reg
    Width: 64
    Index: 34
    PseudoName: pseudoreg.34
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.35.reg
    Width: 64
    Index: 35
    PseudoName: pseudoreg.35
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.36.reg
    Width: 64
    Index: 36
    PseudoName: pseudoreg.36
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.37.reg
    Width: 64
    Index: 37
    PseudoName: pseudoreg.37
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.38.reg
    Width: 64
    Index: 38
    PseudoName: pseudoreg.38
    IsFixedValue: false
```

```
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.39.reg
    Width: 64
    Index: 39
    PseudoName: pseudoreg.39
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.40.reg
    Width: 64
    Index: 40
    PseudoName: pseudoreg.40
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.41.reg
    Width: 64
    Index: 41
    PseudoName: pseudoreg.41
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.42.reg
    Width: 64
    Index: 42
    PseudoName: pseudoreg.42
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.43.reg
    Width: 64
    Index: 43
    PseudoName: pseudoreg.43
```

```
485       IsFixedValue: false
486       IsSIMD: false
487       RWReg: true
488       ROReg: false
489       CSRReg: true
490       AMSReg: false
491       Shared: false
492     - RegName: TEST69.44.reg
493       Width: 64
494       Index: 44
495       PseudoName: pseudoreg.44
496       IsFixedValue: false
497       IsSIMD: false
498       RWReg: true
499       ROReg: false
500       CSRReg: true
501       AMSReg: false
502       Shared: false
503     - RegName: TEST69.45.reg
504       Width: 64
505       Index: 45
506       PseudoName: pseudoreg.45
507       IsFixedValue: false
508       IsSIMD: false
509       RWReg: true
510       ROReg: false
511       CSRReg: true
512       AMSReg: false
513       Shared: false
514     - RegName: TEST69.46.reg
515       Width: 64
516       Index: 46
517       PseudoName: pseudoreg.46
518       IsFixedValue: false
519       IsSIMD: false
520       RWReg: true
521       ROReg: false
522       CSRReg: true
523       AMSReg: false
524       Shared: false
525     - RegName: TEST69.47.reg
526       Width: 64
527       Index: 47
528       PseudoName: pseudoreg.47
529       IsFixedValue: false
530       IsSIMD: false
531       RWReg: true
532       ROReg: false
533       CSRReg: true
534       AMSReg: false
535       Shared: false
536     - RegName: TEST69.48.reg
537       Width: 64
538       Index: 48
```

```
    PseudoName: pseudoreg.48
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.49.reg
    Width: 64
    Index: 49
    PseudoName: pseudoreg.49
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.50.reg
    Width: 64
    Index: 50
    PseudoName: pseudoreg.50
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.51.reg
    Width: 64
    Index: 51
    PseudoName: pseudoreg.51
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.52.reg
    Width: 64
    Index: 52
    PseudoName: pseudoreg.52
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.53.reg
    Width: 64
```

```
    Index: 53
    PseudoName: pseudoreg.53
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.54.reg
    Width: 64
    Index: 54
    PseudoName: pseudoreg.54
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.55.reg
    Width: 64
    Index: 55
    PseudoName: pseudoreg.55
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.56.reg
    Width: 64
    Index: 56
    PseudoName: pseudoreg.56
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.57.reg
    Width: 64
    Index: 57
    PseudoName: pseudoreg.57
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.58.reg
```

```
    Width: 64
    Index: 58
    PseudoName: pseudoreg.58
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.59.reg
    Width: 64
    Index: 59
    PseudoName: pseudoreg.59
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.60.reg
    Width: 64
    Index: 60
    PseudoName: pseudoreg.60
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.61.reg
    Width: 64
    Index: 61
    PseudoName: pseudoreg.61
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.62.reg
    Width: 64
    Index: 62
    PseudoName: pseudoreg.62
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
```

```
  - RegName: TEST69.63.reg
    Width: 64
    Index: 63
    PseudoName: pseudoreg.63
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.64.reg
    Width: 64
    Index: 64
    PseudoName: pseudoreg.64
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.65.reg
    Width: 64
    Index: 65
    PseudoName: pseudoreg.65
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.66.reg
    Width: 64
    Index: 66
    PseudoName: pseudoreg.66
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.67.reg
    Width: 64
    Index: 67
    PseudoName: pseudoreg.67
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
```

```
    Shared: false
  - RegName: TEST69.68.reg
    Width: 64
    Index: 68
    PseudoName: pseudoreg.68
    IsFixedValue: false
    IsSIMD: false
    RWReg: true
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.0.csr
    Width: 64
    Index: 0
    PseudoName: csr.0
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.1.csr
    Width: 64
    Index: 1
    PseudoName: csr.1
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.2.csr
    Width: 64
    Index: 2
    PseudoName: csr.2
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.3.csr
    Width: 64
    Index: 3
    PseudoName: csr.3
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
```

```
    AMSReg: false
    Shared: false
  - RegName: TEST69.4.csr
    Width: 64
    Index: 4
    PseudoName: csr.4
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.5.csr
    Width: 64
    Index: 5
    PseudoName: csr.5
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.6.csr
    Width: 64
    Index: 6
    PseudoName: csr.6
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.7.csr
    Width: 64
    Index: 7
    PseudoName: csr.7
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.8.csr
    Width: 64
    Index: 8
    PseudoName: csr.8
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
```

```
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.9.csr
    Width: 64
    Index: 9
    PseudoName: csr.9
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.10.csr
    Width: 64
    Index: 10
    PseudoName: csr.10
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.11.csr
    Width: 64
    Index: 11
    PseudoName: csr.11
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.12.csr
    Width: 64
    Index: 12
    PseudoName: csr.12
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.13.csr
    Width: 64
    Index: 13
    PseudoName: csr.13
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
```

```
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.14.csr
    Width: 64
    Index: 14
    PseudoName: csr.14
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.15.csr
    Width: 64
    Index: 15
    PseudoName: csr.15
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.16.csr
    Width: 64
    Index: 16
    PseudoName: csr.16
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.17.csr
    Width: 64
    Index: 17
    PseudoName: csr.17
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.18.csr
    Width: 64
    Index: 18
    PseudoName: csr.18
    IsFixedValue: false
    IsSIMD: false
```

```
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.19.csr
    Width: 64
    Index: 19
    PseudoName: csr.19
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.20.csr
    Width: 64
    Index: 20
    PseudoName: csr.20
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.21.csr
    Width: 64
    Index: 21
    PseudoName: csr.21
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.22.csr
    Width: 64
    Index: 22
    PseudoName: csr.22
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.23.csr
    Width: 64
    Index: 23
    PseudoName: csr.23
    IsFixedValue: false
```

```
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.24.csr
    Width: 64
    Index: 24
    PseudoName: csr.24
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.25.csr
    Width: 64
    Index: 25
    PseudoName: csr.25
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.26.csr
    Width: 64
    Index: 26
    PseudoName: csr.26
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.27.csr
    Width: 64
    Index: 27
    PseudoName: csr.27
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.28.csr
    Width: 64
    Index: 28
    PseudoName: csr.28
```

```
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.29.csr
    Width: 64
    Index: 29
    PseudoName: csr.29
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.30.csr
    Width: 64
    Index: 30
    PseudoName: csr.30
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.31.csr
    Width: 64
    Index: 31
    PseudoName: csr.31
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.32.csr
    Width: 64
    Index: 32
    PseudoName: csr.32
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.33.csr
    Width: 64
    Index: 33
```

```
    PseudoName: csr.33
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.34.csr
    Width: 64
    Index: 34
    PseudoName: csr.34
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.35.csr
    Width: 64
    Index: 35
    PseudoName: csr.35
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.36.csr
    Width: 64
    Index: 36
    PseudoName: csr.36
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.37.csr
    Width: 64
    Index: 37
    PseudoName: csr.37
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.38.csr
    Width: 64
```

```
    Index: 38
    PseudoName: csr.38
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.39.csr
    Width: 64
    Index: 39
    PseudoName: csr.39
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.40.csr
    Width: 64
    Index: 40
    PseudoName: csr.40
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.41.csr
    Width: 64
    Index: 41
    PseudoName: csr.41
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.42.csr
    Width: 64
    Index: 42
    PseudoName: csr.42
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.43.csr
```

```
    Width: 64
    Index: 43
    PseudoName: csr.43
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.44.csr
    Width: 64
    Index: 44
    PseudoName: csr.44
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.45.csr
    Width: 64
    Index: 45
    PseudoName: csr.45
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.46.csr
    Width: 64
    Index: 46
    PseudoName: csr.46
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.47.csr
    Width: 64
    Index: 47
    PseudoName: csr.47
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
```

```
  - RegName: TEST69.48.csr
    Width: 64
    Index: 48
    PseudoName: csr.48
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.49.csr
    Width: 64
    Index: 49
    PseudoName: csr.49
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.50.csr
    Width: 64
    Index: 50
    PseudoName: csr.50
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.51.csr
    Width: 64
    Index: 51
    PseudoName: csr.51
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.52.csr
    Width: 64
    Index: 52
    PseudoName: csr.52
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
```

```
    Shared: false
  - RegName: TEST69.53.csr
    Width: 64
    Index: 53
    PseudoName: csr.53
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.54.csr
    Width: 64
    Index: 54
    PseudoName: csr.54
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.55.csr
    Width: 64
    Index: 55
    PseudoName: csr.55
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.56.csr
    Width: 64
    Index: 56
    PseudoName: csr.56
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.57.csr
    Width: 64
    Index: 57
    PseudoName: csr.57
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
```

```
    AMSReg: false
    Shared: false
  - RegName: TEST69.58.csr
    Width: 64
    Index: 58
    PseudoName: csr.58
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.59.csr
    Width: 64
    Index: 59
    PseudoName: csr.59
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.60.csr
    Width: 64
    Index: 60
    PseudoName: csr.60
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.61.csr
    Width: 64
    Index: 61
    PseudoName: csr.61
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.62.csr
    Width: 64
    Index: 62
    PseudoName: csr.62
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
```

```
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.63.csr
    Width: 64
    Index: 63
    PseudoName: csr.63
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.64.csr
    Width: 64
    Index: 64
    PseudoName: csr.64
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.65.csr
    Width: 64
    Index: 65
    PseudoName: csr.65
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.66.csr
    Width: 64
    Index: 66
    PseudoName: csr.66
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.67.csr
    Width: 64
    Index: 67
    PseudoName: csr.67
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
```

```
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
  - RegName: TEST69.68.csr
    Width: 64
    Index: 68
    PseudoName: csr.68
    IsFixedValue: false
    IsSIMD: false
    RWReg: false
    ROReg: false
    CSRReg: true
    AMSReg: false
    Shared: false
RegClasses:
  - RegisterClassName: TEST69.regclass
    Registers:
      - TEST69.0.reg
      - TEST69.1.reg
      - TEST69.2.reg
      - TEST69.3.reg
      - TEST69.4.reg
      - TEST69.5.reg
      - TEST69.6.reg
      - TEST69.7.reg
      - TEST69.8.reg
      - TEST69.9.reg
      - TEST69.10.reg
      - TEST69.11.reg
      - TEST69.12.reg
      - TEST69.13.reg
      - TEST69.14.reg
      - TEST69.15.reg
      - TEST69.16.reg
      - TEST69.17.reg
      - TEST69.18.reg
      - TEST69.19.reg
      - TEST69.20.reg
      - TEST69.21.reg
      - TEST69.22.reg
      - TEST69.23.reg
      - TEST69.24.reg
      - TEST69.25.reg
      - TEST69.26.reg
      - TEST69.27.reg
      - TEST69.28.reg
      - TEST69.29.reg
      - TEST69.30.reg
      - TEST69.31.reg
      - TEST69.32.reg
      - TEST69.33.reg
      - TEST69.34.reg
      - TEST69.35.reg
```

```
      - TEST69.36.reg
      - TEST69.37.reg
      - TEST69.38.reg
      - TEST69.39.reg
      - TEST69.40.reg
      - TEST69.41.reg
      - TEST69.42.reg
      - TEST69.43.reg
      - TEST69.44.reg
      - TEST69.45.reg
      - TEST69.46.reg
      - TEST69.47.reg
      - TEST69.48.reg
      - TEST69.49.reg
      - TEST69.50.reg
      - TEST69.51.reg
      - TEST69.52.reg
      - TEST69.53.reg
      - TEST69.54.reg
      - TEST69.55.reg
      - TEST69.56.reg
      - TEST69.57.reg
      - TEST69.58.reg
      - TEST69.59.reg
      - TEST69.60.reg
      - TEST69.61.reg
      - TEST69.62.reg
      - TEST69.63.reg
      - TEST69.64.reg
      - TEST69.65.reg
      - TEST69.66.reg
      - TEST69.67.reg
      - TEST69.68.reg
  - RegisterClassName: TEST69.csrregclass
    Registers:
      - TEST69.0.csr
      - TEST69.1.csr
      - TEST69.2.csr
      - TEST69.3.csr
      - TEST69.4.csr
      - TEST69.5.csr
      - TEST69.6.csr
      - TEST69.7.csr
      - TEST69.8.csr
      - TEST69.9.csr
      - TEST69.10.csr
      - TEST69.11.csr
      - TEST69.12.csr
      - TEST69.13.csr
      - TEST69.14.csr
      - TEST69.15.csr
      - TEST69.16.csr
      - TEST69.17.csr
      - TEST69.18.csr
```

```
      - TEST69.19.csr
      - TEST69.20.csr
      - TEST69.21.csr
      - TEST69.22.csr
      - TEST69.23.csr
      - TEST69.24.csr
      - TEST69.25.csr
      - TEST69.26.csr
      - TEST69.27.csr
      - TEST69.28.csr
      - TEST69.29.csr
      - TEST69.30.csr
      - TEST69.31.csr
      - TEST69.32.csr
      - TEST69.33.csr
      - TEST69.34.csr
      - TEST69.35.csr
      - TEST69.36.csr
      - TEST69.37.csr
      - TEST69.38.csr
      - TEST69.39.csr
      - TEST69.40.csr
      - TEST69.41.csr
      - TEST69.42.csr
      - TEST69.43.csr
      - TEST69.44.csr
      - TEST69.45.csr
      - TEST69.46.csr
      - TEST69.47.csr
      - TEST69.48.csr
      - TEST69.49.csr
      - TEST69.50.csr
      - TEST69.51.csr
      - TEST69.52.csr
      - TEST69.53.csr
      - TEST69.54.csr
      - TEST69.55.csr
      - TEST69.56.csr
      - TEST69.57.csr
      - TEST69.58.csr
      - TEST69.59.csr
      - TEST69.60.csr
      - TEST69.61.csr
      - TEST69.62.csr
      - TEST69.63.csr
      - TEST69.64.csr
      - TEST69.65.csr
      - TEST69.66.csr
      - TEST69.67.csr
      - TEST69.68.csr
ISAs:
  - ISAName: TEST69.isa
InstFormats:
  - InstFormatName: TEST69.if
```

```
    ISA: TEST69.isa
    FormatWidth: 32
    Fields:
      - FieldName: opcode
        FieldType: CGInstCode
        FieldWidth: 8
        StartBit: 0
        EndBit: 7
        MandatoryField: true
      - FieldName: RB
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 8
        EndBit: 15
        MandatoryField: false
        RegClass: TEST69.regclass
      - FieldName: RA
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 16
        EndBit: 23
        MandatoryField: false
        RegClass: TEST69.regclass
      - FieldName: RT
        FieldType: CGInstReg
        FieldWidth: 8
        StartBit: 24
        EndBit: 31
        MandatoryField: false
        RegClass: TEST69.csrregclass
Insts:
  - Inst: TEST69.inst0
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 1
  - Inst: TEST69.inst1
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 2
  - Inst: TEST69.inst2
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 3
  - Inst: TEST69.inst3
    ISA: TEST69.isa
```

```
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 4
  - Inst: TEST69.inst4
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 5
  - Inst: TEST69.inst5
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 6
  - Inst: TEST69.inst6
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 7
  - Inst: TEST69.inst7
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 8
  - Inst: TEST69.inst8
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 9
  - Inst: TEST69.inst9
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 10
  - Inst: TEST69.inst10
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 11
```

```
  - Inst: TEST69.inst11
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 12
  - Inst: TEST69.inst12
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 13
  - Inst: TEST69.inst13
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 14
  - Inst: TEST69.inst14
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 15
  - Inst: TEST69.inst15
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 16
  - Inst: TEST69.inst16
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 17
  - Inst: TEST69.inst17
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 18
  - Inst: TEST69.inst18
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
```

```
        EncodingWidth: 8
        EncodingValue: 19
  - Inst: TEST69.inst19
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 20
  - Inst: TEST69.inst20
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 21
  - Inst: TEST69.inst21
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 22
  - Inst: TEST69.inst22
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 23
  - Inst: TEST69.inst23
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 24
  - Inst: TEST69.inst24
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 25
  - Inst: TEST69.inst25
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 26
  - Inst: TEST69.inst26
    ISA: TEST69.isa
    InstFormat: TEST69.if
```

```
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 27
  - Inst: TEST69.inst27
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 28
  - Inst: TEST69.inst28
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 29
  - Inst: TEST69.inst29
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 30
  - Inst: TEST69.inst30
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 31
  - Inst: TEST69.inst31
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 32
  - Inst: TEST69.inst32
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 33
  - Inst: TEST69.inst33
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 34
  - Inst: TEST69.inst34
```

```
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 35
  - Inst: TEST69.inst35
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 36
  - Inst: TEST69.inst36
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 37
  - Inst: TEST69.inst37
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 38
  - Inst: TEST69.inst38
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 39
  - Inst: TEST69.inst39
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 40
  - Inst: TEST69.inst40
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 41
  - Inst: TEST69.inst41
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
```

```
1997          EncodingValue: 42
1998    - Inst: TEST69.inst42
1999      ISA: TEST69.isa
2000      InstFormat: TEST69.if
2001      Encodings:
2002        - EncodingField: opcode
2003          EncodingWidth: 8
2004          EncodingValue: 43
2005    - Inst: TEST69.inst43
2006      ISA: TEST69.isa
2007      InstFormat: TEST69.if
2008      Encodings:
2009        - EncodingField: opcode
2010          EncodingWidth: 8
2011          EncodingValue: 44
2012    - Inst: TEST69.inst44
2013      ISA: TEST69.isa
2014      InstFormat: TEST69.if
2015      Encodings:
2016        - EncodingField: opcode
2017          EncodingWidth: 8
2018          EncodingValue: 45
2019    - Inst: TEST69.inst45
2020      ISA: TEST69.isa
2021      InstFormat: TEST69.if
2022      Encodings:
2023        - EncodingField: opcode
2024          EncodingWidth: 8
2025          EncodingValue: 46
2026    - Inst: TEST69.inst46
2027      ISA: TEST69.isa
2028      InstFormat: TEST69.if
2029      Encodings:
2030        - EncodingField: opcode
2031          EncodingWidth: 8
2032          EncodingValue: 47
2033    - Inst: TEST69.inst47
2034      ISA: TEST69.isa
2035      InstFormat: TEST69.if
2036      Encodings:
2037        - EncodingField: opcode
2038          EncodingWidth: 8
2039          EncodingValue: 48
2040    - Inst: TEST69.inst48
2041      ISA: TEST69.isa
2042      InstFormat: TEST69.if
2043      Encodings:
2044        - EncodingField: opcode
2045          EncodingWidth: 8
2046          EncodingValue: 49
2047    - Inst: TEST69.inst49
2048      ISA: TEST69.isa
2049      InstFormat: TEST69.if
2050      Encodings:
```

```
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 50
  - Inst: TEST69.inst50
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 51
  - Inst: TEST69.inst51
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 52
  - Inst: TEST69.inst52
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 53
  - Inst: TEST69.inst53
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 54
  - Inst: TEST69.inst54
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 55
  - Inst: TEST69.inst55
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 56
  - Inst: TEST69.inst56
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 57
  - Inst: TEST69.inst57
    ISA: TEST69.isa
```

```
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 58
  - Inst: TEST69.inst58
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 59
  - Inst: TEST69.inst59
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 60
  - Inst: TEST69.inst60
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 61
  - Inst: TEST69.inst61
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 62
  - Inst: TEST69.inst62
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 63
  - Inst: TEST69.inst63
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 64
  - Inst: TEST69.inst64
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 65
```

```
  - Inst: TEST69.inst65
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 66
  - Inst: TEST69.inst66
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 67
  - Inst: TEST69.inst67
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 68
  - Inst: TEST69.inst68
    ISA: TEST69.isa
    InstFormat: TEST69.if
    Encodings:
      - EncodingField: opcode
        EncodingWidth: 8
        EncodingValue: 69
PseudoInsts:
  - PseudoInst: TEST69.pinst0
    ISA: TEST69.isa
    Inst: TEST69.inst0
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 0
  - PseudoInst: TEST69.pinst1
    ISA: TEST69.isa
    Inst: TEST69.inst1
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 1
  - PseudoInst: TEST69.pinst2
    ISA: TEST69.isa
    Inst: TEST69.inst2
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 2
  - PseudoInst: TEST69.pinst3
    ISA: TEST69.isa
    Inst: TEST69.inst3
    Encodings:
```

```
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 3
  - PseudoInst: TEST69.pinst4
    ISA: TEST69.isa
    Inst: TEST69.inst4
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 4
  - PseudoInst: TEST69.pinst5
    ISA: TEST69.isa
    Inst: TEST69.inst5
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 5
  - PseudoInst: TEST69.pinst6
    ISA: TEST69.isa
    Inst: TEST69.inst6
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 6
  - PseudoInst: TEST69.pinst7
    ISA: TEST69.isa
    Inst: TEST69.inst7
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 7
  - PseudoInst: TEST69.pinst8
    ISA: TEST69.isa
    Inst: TEST69.inst8
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 8
  - PseudoInst: TEST69.pinst9
    ISA: TEST69.isa
    Inst: TEST69.inst9
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 9
  - PseudoInst: TEST69.pinst10
    ISA: TEST69.isa
    Inst: TEST69.inst10
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 10
  - PseudoInst: TEST69.pinst11
    ISA: TEST69.isa
```

```
    Inst: TEST69.inst11
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 11
  - PseudoInst: TEST69.pinst12
    ISA: TEST69.isa
    Inst: TEST69.inst12
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 12
  - PseudoInst: TEST69.pinst13
    ISA: TEST69.isa
    Inst: TEST69.inst13
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 13
  - PseudoInst: TEST69.pinst14
    ISA: TEST69.isa
    Inst: TEST69.inst14
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 14
  - PseudoInst: TEST69.pinst15
    ISA: TEST69.isa
    Inst: TEST69.inst15
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 15
  - PseudoInst: TEST69.pinst16
    ISA: TEST69.isa
    Inst: TEST69.inst16
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 16
  - PseudoInst: TEST69.pinst17
    ISA: TEST69.isa
    Inst: TEST69.inst17
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 17
  - PseudoInst: TEST69.pinst18
    ISA: TEST69.isa
    Inst: TEST69.inst18
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 18
```

```
  - PseudoInst: TEST69.pinst19
    ISA: TEST69.isa
    Inst: TEST69.inst19
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 19
  - PseudoInst: TEST69.pinst20
    ISA: TEST69.isa
    Inst: TEST69.inst20
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 20
  - PseudoInst: TEST69.pinst21
    ISA: TEST69.isa
    Inst: TEST69.inst21
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 21
  - PseudoInst: TEST69.pinst22
    ISA: TEST69.isa
    Inst: TEST69.inst22
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 22
  - PseudoInst: TEST69.pinst23
    ISA: TEST69.isa
    Inst: TEST69.inst23
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 23
  - PseudoInst: TEST69.pinst24
    ISA: TEST69.isa
    Inst: TEST69.inst24
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 24
  - PseudoInst: TEST69.pinst25
    ISA: TEST69.isa
    Inst: TEST69.inst25
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 25
  - PseudoInst: TEST69.pinst26
    ISA: TEST69.isa
    Inst: TEST69.inst26
    Encodings:
      - EncodingField: RA
```

```
        EncodingWidth: 8
        EncodingValue: 26
  - PseudoInst: TEST69.pinst27
    ISA: TEST69.isa
    Inst: TEST69.inst27
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 27
  - PseudoInst: TEST69.pinst28
    ISA: TEST69.isa
    Inst: TEST69.inst28
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 28
  - PseudoInst: TEST69.pinst29
    ISA: TEST69.isa
    Inst: TEST69.inst29
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 29
  - PseudoInst: TEST69.pinst30
    ISA: TEST69.isa
    Inst: TEST69.inst30
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 30
  - PseudoInst: TEST69.pinst31
    ISA: TEST69.isa
    Inst: TEST69.inst31
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 31
  - PseudoInst: TEST69.pinst32
    ISA: TEST69.isa
    Inst: TEST69.inst32
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 32
  - PseudoInst: TEST69.pinst33
    ISA: TEST69.isa
    Inst: TEST69.inst33
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 33
  - PseudoInst: TEST69.pinst34
    ISA: TEST69.isa
    Inst: TEST69.inst34
```

```
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 34
  - PseudoInst: TEST69.pinst35
    ISA: TEST69.isa
    Inst: TEST69.inst35
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 35
  - PseudoInst: TEST69.pinst36
    ISA: TEST69.isa
    Inst: TEST69.inst36
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 36
  - PseudoInst: TEST69.pinst37
    ISA: TEST69.isa
    Inst: TEST69.inst37
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 37
  - PseudoInst: TEST69.pinst38
    ISA: TEST69.isa
    Inst: TEST69.inst38
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 38
  - PseudoInst: TEST69.pinst39
    ISA: TEST69.isa
    Inst: TEST69.inst39
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 39
  - PseudoInst: TEST69.pinst40
    ISA: TEST69.isa
    Inst: TEST69.inst40
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 40
  - PseudoInst: TEST69.pinst41
    ISA: TEST69.isa
    Inst: TEST69.inst41
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 41
  - PseudoInst: TEST69.pinst42
```

```
    ISA: TEST69.isa
    Inst: TEST69.inst42
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 42
  - PseudoInst: TEST69.pinst43
    ISA: TEST69.isa
    Inst: TEST69.inst43
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 43
  - PseudoInst: TEST69.pinst44
    ISA: TEST69.isa
    Inst: TEST69.inst44
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 44
  - PseudoInst: TEST69.pinst45
    ISA: TEST69.isa
    Inst: TEST69.inst45
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 45
  - PseudoInst: TEST69.pinst46
    ISA: TEST69.isa
    Inst: TEST69.inst46
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 46
  - PseudoInst: TEST69.pinst47
    ISA: TEST69.isa
    Inst: TEST69.inst47
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 47
  - PseudoInst: TEST69.pinst48
    ISA: TEST69.isa
    Inst: TEST69.inst48
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 48
  - PseudoInst: TEST69.pinst49
    ISA: TEST69.isa
    Inst: TEST69.inst49
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
```

```
2537          EncodingValue: 49
2538    - PseudoInst: TEST69.pinst50
2539      ISA: TEST69.isa
2540      Inst: TEST69.inst50
2541      Encodings:
2542        - EncodingField: RA
2543          EncodingWidth: 8
2544          EncodingValue: 50
2545    - PseudoInst: TEST69.pinst51
2546      ISA: TEST69.isa
2547      Inst: TEST69.inst51
2548      Encodings:
2549        - EncodingField: RA
2550          EncodingWidth: 8
2551          EncodingValue: 51
2552    - PseudoInst: TEST69.pinst52
2553      ISA: TEST69.isa
2554      Inst: TEST69.inst52
2555      Encodings:
2556        - EncodingField: RA
2557          EncodingWidth: 8
2558          EncodingValue: 52
2559    - PseudoInst: TEST69.pinst53
2560      ISA: TEST69.isa
2561      Inst: TEST69.inst53
2562      Encodings:
2563        - EncodingField: RA
2564          EncodingWidth: 8
2565          EncodingValue: 53
2566    - PseudoInst: TEST69.pinst54
2567      ISA: TEST69.isa
2568      Inst: TEST69.inst54
2569      Encodings:
2570        - EncodingField: RA
2571          EncodingWidth: 8
2572          EncodingValue: 54
2573    - PseudoInst: TEST69.pinst55
2574      ISA: TEST69.isa
2575      Inst: TEST69.inst55
2576      Encodings:
2577        - EncodingField: RA
2578          EncodingWidth: 8
2579          EncodingValue: 55
2580    - PseudoInst: TEST69.pinst56
2581      ISA: TEST69.isa
2582      Inst: TEST69.inst56
2583      Encodings:
2584        - EncodingField: RA
2585          EncodingWidth: 8
2586          EncodingValue: 56
2587    - PseudoInst: TEST69.pinst57
2588      ISA: TEST69.isa
2589      Inst: TEST69.inst57
2590      Encodings:
```

```
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 57
  - PseudoInst: TEST69.pinst58
    ISA: TEST69.isa
    Inst: TEST69.inst58
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 58
  - PseudoInst: TEST69.pinst59
    ISA: TEST69.isa
    Inst: TEST69.inst59
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 59
  - PseudoInst: TEST69.pinst60
    ISA: TEST69.isa
    Inst: TEST69.inst60
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 60
  - PseudoInst: TEST69.pinst61
    ISA: TEST69.isa
    Inst: TEST69.inst61
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 61
  - PseudoInst: TEST69.pinst62
    ISA: TEST69.isa
    Inst: TEST69.inst62
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 62
  - PseudoInst: TEST69.pinst63
    ISA: TEST69.isa
    Inst: TEST69.inst63
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 63
  - PseudoInst: TEST69.pinst64
    ISA: TEST69.isa
    Inst: TEST69.inst64
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 64
  - PseudoInst: TEST69.pinst65
    ISA: TEST69.isa
```

```
    Inst: TEST69.inst65
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 65
  - PseudoInst: TEST69.pinst66
    ISA: TEST69.isa
    Inst: TEST69.inst66
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 66
  - PseudoInst: TEST69.pinst67
    ISA: TEST69.isa
    Inst: TEST69.inst67
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 67
  - PseudoInst: TEST69.pinst68
    ISA: TEST69.isa
    Inst: TEST69.inst68
    Encodings:
      - EncodingField: RA
        EncodingWidth: 8
        EncodingValue: 68
Caches:
  - Cache: TEST69.L2.cache
    Sets: 2
    Ways: 8
  - Cache: TEST69.core0.L1.cache
    Sets: 1
    Ways: 1
    SubLevel: TEST69.L2.cache
  - Cache: TEST69.L2.cache
    Sets: 2
    Ways: 8
  - Cache: TEST69.core1.L1.cache
    Sets: 1
    Ways: 1
    SubLevel: TEST69.L2.cache
  - Cache: TEST69.L2.cache
    Sets: 2
    Ways: 8
  - Cache: TEST69.core2.L1.cache
    Sets: 1
    Ways: 1
    SubLevel: TEST69.L2.cache
  - Cache: TEST69.L2.cache
    Sets: 2
    Ways: 8
  - Cache: TEST69.core3.L1.cache
    Sets: 1
    Ways: 1
```

```
    SubLevel: TEST69.L2.cache
  - Cache: TEST69.L2.cache
    Sets: 2
    Ways: 8
Comms:
  - Comm: TEST69.comm
    Type: NOC
    Width: 64
    Endpoints:
      - TEST69.core0
      - TEST69.core1
      - TEST69.core2
      - TEST69.core3
      - TEST69.L2.cache
      - TEST69.vtp
      - TEST69.0.mctrl
      - TEST69.1.mctrl
      - TEST69.2.mctrl
      - TEST69.3.mctrl
Scratchpads:
  - Scratchpad: TEST69.0.spad
    MemSize: 1024
    RqstPorts: 1
    RspPorts: 1
    StartAddr: 34359739392
  - Scratchpad: TEST69.1.spad
    MemSize: 2048
    RqstPorts: 1
    RspPorts: 1
    StartAddr: 34359740416
  - Scratchpad: TEST69.2.spad
    MemSize: 3072
    RqstPorts: 1
    RspPorts: 1
    StartAddr: 34359741440
  - Scratchpad: TEST69.3.spad
    MemSize: 4096
    RqstPorts: 1
    RspPorts: 1
    StartAddr: 34359742464
MemoryControllers:
  - MemoryController: TEST69.0.mctrl
    Ports: 2
  - MemoryController: TEST69.1.mctrl
    Ports: 2
  - MemoryController: TEST69.2.mctrl
    Ports: 2
  - MemoryController: TEST69.3.mctrl
    Ports: 2
VTPControllers:
  - VTP: TEST69.vtp
Extensions:
  - Extension: TEST69.ext0
    Type: unknown
```

```
    Registers:
      - RegName: TEST69.ext0.reg
        Width: 64
        Index: 0
        IsFixedValue: false
        IsSIMD: false
        RWReg: false
        ROReg: false
        CSRReg: true
        AMSReg: false
        Shared: false
    RegClasses:
      - RegisterClassName: TEST69.ext0.regclass
        Registers:
          - TEST69.ext0.reg
    ISAs:
      - ISAName: TEST69.ext0.isa
    RTLFile: TEST69.ext0.v
  - Extension: TEST69.ext1
    Type: unknown
    Registers:
      - RegName: TEST69.ext1.reg
        Width: 64
        Index: 0
        IsFixedValue: false
        IsSIMD: false
        RWReg: false
        ROReg: false
        CSRReg: true
        AMSReg: false
        Shared: false
    RegClasses:
      - RegisterClassName: TEST69.ext1.regclass
        Registers:
          - TEST69.ext1.reg
    ISAs:
      - ISAName: TEST69.ext1.isa
    RTLFile: TEST69.ext1.v
  - Extension: TEST69.ext2
    Type: unknown
    Registers:
      - RegName: TEST69.ext2.reg
        Width: 64
        Index: 0
        IsFixedValue: false
        IsSIMD: false
        RWReg: false
        ROReg: false
        CSRReg: true
        AMSReg: false
        Shared: false
    RegClasses:
      - RegisterClassName: TEST69.ext2.regclass
        Registers:
```

```
2807            - TEST69.ext2.reg
2808      ISAs:
2809        - ISAName: TEST69.ext2.isa
2810      RTLFile: TEST69.ext2.v
2811    - Extension: TEST69.ext3
2812      Type: unknown
2813      Registers:
2814        - RegName: TEST69.ext3.reg
2815          Width: 64
2816          Index: 0
2817          IsFixedValue: false
2818          IsSIMD: false
2819          RWReg: false
2820          ROReg: false
2821          CSRReg: true
2822          AMSReg: false
2823          Shared: false
2824      RegClasses:
2825        - RegisterClassName: TEST69.ext3.regclass
2826          Registers:
2827            - TEST69.ext3.reg
2828      ISAs:
2829        - ISAName: TEST69.ext3.isa
2830      RTLFile: TEST69.ext3.v
2831  Cores:
2832    - Core: TEST69.core0
2833      Cache: TEST69.core0.L1.cache
2834      ISA: TEST69.isa
2835      RegisterClasses:
2836        - RegClass: TEST69.regclass
2837        - RegClass: TEST69.csrregclass
2838      Extensions:
2839        - Extension: TEST69.ext0
2840    - Core: TEST69.core1
2841      Cache: TEST69.core1.L1.cache
2842      ISA: TEST69.isa
2843      RegisterClasses:
2844        - RegClass: TEST69.regclass
2845        - RegClass: TEST69.csrregclass
2846      Extensions:
2847        - Extension: TEST69.ext1
2848    - Core: TEST69.core2
2849      Cache: TEST69.core2.L1.cache
2850      ISA: TEST69.isa
2851      RegisterClasses:
2852        - RegClass: TEST69.regclass
2853        - RegClass: TEST69.csrregclass
2854      Extensions:
2855        - Extension: TEST69.ext2
2856    - Core: TEST69.core3
2857      Cache: TEST69.core3.L1.cache
2858      ISA: TEST69.isa
2859      RegisterClasses:
2860        - RegClass: TEST69.regclass
```

```
      - RegClass: TEST69.csrregclass
    Extensions:
      - Extension: TEST69.ext3
Socs:
  - Soc: TEST69.soc
    Cores:
      - Core: TEST69.core0
      - Core: TEST69.core1
      - Core: TEST69.core2
      - Core: TEST69.core3
```

Listing 25: Sample IR File

References

[1] O. Ben-Kiki, C. Evans, and I. dotNet, "Yaml ain't markup language (yaml) version 1.2," 2009. [Online]. Available: http://yaml.org/spec/1.2/spec.html