## A. Subset of SystemVerilog Syntax

This section presents the subset of SystemVerilog syntax used in this paper, in Backus–Naur form. The nonterminals in this subset correspond to one or more patterns in the transformation rules in Figure 4. Table 4 describes the primary nonterminals and provides a mapping between them and the patterns used in Figure 4.

| Nonterminal | Description | Patterns |
|---|---|---|
| ⟨*expression*⟩ | an expression | *e* |
| ⟨*fragment*⟩ | a single assignment | `Assignment` |
| ⟨*call*⟩ | a function call | `FunctionCall` |
| ⟨*task-call*⟩ | a task call statement | `TaskCall` |
| ⟨*block*⟩ | a sequence of statements | `Block` |
| ⟨*if-else*⟩ | a single branch conditional block or a complete conditional block | `If` `IfElse` |
| ⟨*while*⟩ | a while block | `While` |
| ⟨*always*⟩ | an always block | `Always` |
| ⟨*fork-join*⟩ | a fork-join block or a join-none block or a join-any block | `ForkJoin` `JoinNone` `JoinAny` |
| ⟨*module-init*⟩ | the instantiation of a module | `ModuleInit` |
| ⟨*module*⟩ | the declaration of a module | `Module` |

Table 4: SystemVerilog syntax and pattern correspondence

```
⟨module⟩         ::= ⟨module-header⟩
                     ⟨declaration-or-init⟩*
                     ⟨module-item⟩ 'endmodule'

⟨module-header⟩ ::= 'module' ⟨symbol⟩ ';'
                  | 'module' ⟨symbol⟩ '(' ⟨params⟩ ')' ';'

⟨module-item⟩   ::= ⟨always⟩
                  | ⟨initial⟩
                  | ⟨statement⟩
                  | ''

⟨always⟩         ::= 'always' ⟨block⟩

⟨initial⟩        ::= 'initial' ⟨block⟩

⟨interface⟩      ::= 'interface' ⟨symbol⟩ ';'
                     ⟨declaration⟩* ⟨task⟩*
                     'endinterface'

⟨task⟩           ::= ⟨task-header⟩ ⟨statement⟩* 'endtask'

⟨task-header⟩    ::= 'task' ⟨symbol⟩ '()' ';'
                   | 'task' ⟨symbol⟩ '(' ⟨params⟩ ')' ';'

⟨block⟩          ::= 'begin' ⟨statement⟩* 'end'

⟨fork-join⟩      ::= 'fork' ⟨statement⟩* 'join'
                   | 'fork' ⟨statement⟩* 'join_any'
                   | 'fork' ⟨statement⟩* 'join_none'

⟨statement⟩      ::= ⟨wait⟩
                   | ⟨assignment⟩
                   | ⟨if-else⟩
                   | ⟨fork-join⟩
                   | ⟨task-call⟩
                   | ⟨declaration⟩
                   | ⟨block⟩
                   | ⟨command⟩ ';'

⟨wait⟩           ::= 'wait' '(' ⟨expression⟩ ')' ';'

⟨call⟩           ::= ⟨symbol⟩ ⟨args⟩

⟨task-call⟩      ::= ⟨symbol⟩ '.' ⟨symbol⟩ ⟨args⟩ ';'

⟨if-else⟩        ::= 'if' '(' ⟨expression⟩ ')' ⟨block⟩
                   | 'if' '(' ⟨expression⟩ ')' ⟨block⟩ 'else'
                     ⟨block⟩

⟨while⟩          ::= 'while' '(' ⟨expression⟩ ')' ⟨block⟩

⟨declaration-or-init⟩ ::= ⟨declaration⟩
                   | ⟨interface-init⟩
                   | ⟨module-init⟩

⟨declaration⟩    ::= ⟨type⟩ ⟨assignment⟩ | ⟨type⟩ ⟨symbol⟩ ';'

⟨type⟩           ::= ⟨primitive-type⟩ | ⟨symbol⟩

⟨primitive-type⟩ ::= 'logic'
                   | 'logic' '[' ⟨unsigned⟩ ':' '0' ']'

⟨type-def⟩       ::= 'typedef' ⟨primitive-type⟩ ⟨symbol⟩ ';'

⟨enum-def⟩       ::= 'typedef' 'enum'
                    '{' ⟨symbol⟩ (',' ⟨symbol⟩)* '}'
                     ⟨symbol⟩ ';'

⟨assignment⟩     ::= ⟨assign-fragments⟩ ';'

⟨command⟩        ::= '$' ⟨symbol⟩ ⟨args⟩

⟨assign-fragments⟩ ::= ⟨fragment⟩ (',' ⟨fragment⟩)*

⟨fragment⟩       ::= ⟨symbol⟩ '=' ⟨expression⟩

⟨interface-init⟩ ::= ⟨symbol⟩ ⟨symbol⟩ '()' ';'

⟨module-init⟩    ::= ⟨symbol⟩ ⟨symbol⟩ ⟨args⟩ ';'

⟨args⟩           ::= '()' | '(' ⟨symbol⟩ (',' ⟨symbol⟩)* ')'
```

```
⟨params⟩      ::= ⟨param⟩ ( ‘,’ ⟨param⟩ )*

⟨param⟩       ::= ‘interface’ ⟨symbol⟩
                |  ⟨direction⟩ ⟨type⟩ ⟨symbol⟩

⟨expression⟩  ::= ⟨call⟩
                |  ⟨unary-op⟩ ⟨primary⟩
                |  ⟨primary⟩ ⟨binary-op⟩ ⟨primary⟩
                |  ‘(’ ⟨expression⟩ ‘)’
                |  ⟨command⟩

⟨primary⟩     ::= ‘z’ | ⟨unsigned⟩ | ⟨symbol⟩ | ⟨expression⟩

⟨direction⟩   ::= ‘input’ | ‘output’ | ‘’

⟨symbol⟩      ::= [a-zA-Z_] [a-zA-Z0-9_]*

⟨unsigned⟩    ::= [0-9]+

⟨unary-op⟩    ::= ‘+’ | ‘-’ | ‘~’

⟨binary-op⟩   ::= ‘+’ | ‘-’ | ‘*’ | ‘/’ | ‘%’ | ‘&’ | ‘|’ | ‘^’ | ‘&&’
                  | ‘||’ | ‘>’ | ‘>=’ | ‘<’ | ‘<=’ | ‘==’ | ‘!=’
```