GEZEL Basic Syntax

```
// Simulation: fdlsim [-d] fileName.fdl numberOfCycles

dp dataPathName(in signalName1, signalName2: ns(12); out signalName3: ns(2)) {
   reg registerName1, registerName2: tc(12);
   sig signalName: nc(32);
               // Simulation directive: $trace(expr,"file.txt")
  always {
    <expressions>
    $display($hex, "Cycle=", $cycle, $bin, ": i=", registerName, ", o=", signalName);
  }            // Other simulation directives: $dec, $dp, $sfg
               //                         and: $finish
  sfg signalFlowGraphName {
     <expressions>
  }
}

fsm controlerName(dataPathName) {
  initial s0;
  state s1, s2;
  @s0 signalFlowGraphName      -> s1;
  @s1 (sfgName1, sfgName2)     -> s2;          // Simulation directive: (…,$trace)
  @s2 if (RegisterCondition1) then (sfgName)          -> s3;
      else if (RegisterCondition2)then (sfgName2)     -> s0;
           else sfgName        -> s0;
}

hardwired controlerName(dataPathName) { sfgName1; sfgName2 }

sequencer controlerName (dataPathName) { sfgName1; (sfgName2, sfgName3) }

dp dataPathCopy : dataPathName

dp datapathName {
  sig signalName1, signalName2 : ns(32);
  use dataPathName(signalName1, signalName2 );
  use dataPathCopy(signalName2 , signalName1);

}

system systemName {
 datapathName;
}
```

Operators:

a = expression	a < b	a * b	Binary number:
b ? c : d;	a > b	a # b	0b0000, 0b110011
b | c	a <= b	-a	
b ^ c	a >= b	b = (tc(3)) a;	Hexadecimal number:
b & c	a << b	a[n]	0x4f, 0xA4B
~b	a >> b	a[m:n]	
a == b	a + b	lookup a : ns(8) = {15, 0, 22, 12};	
a != b	a - b	a(2)	

The four rules:

- During any clock cycle, all datapath outputs are defined.
- During any clock cycle, no combinatorial loop between signals can exist.
- If an expression uses the value of a signal, then that signal must also appear at the left-hand side of an assignment.
- Neither registers, nor signals or datapath outputs can be assigned more than once during a clock cycle.