

HDL Generator Documentation

Overview

AUTHOR

Wyatt Gronnemose

Contents

1	Project Introduction	1
1.1	Purpose	1
1.2	System Architecture	1
2	Language Structure	1
2.1	State Machine Syntax	2
2.1.1	Inputs	2
2.1.2	Pseudocode	2
2.2	System Syntax	3

List of Figures

1.1	System Architecture	1
2.1	Moore Machine	2

1 Project Introduction

1.1 Purpose

The purpose of this project is to be able to easily describe a Moore state machine and turn it into a hardware description language (HDL). The HDL can then be used with FPGA's.

1.2 System Architecture

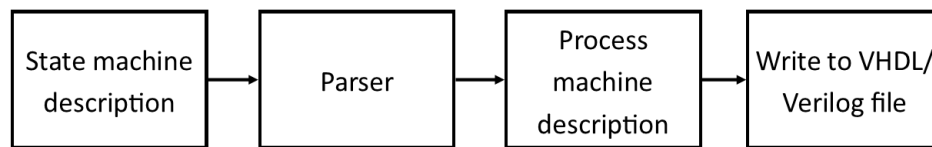


Figure 1.1: System Architecture

2 Language Structure

This section defines how you express a state machine or system of state machines. A Moore machine can be defined using a set of six items. These are:

- Finite set of states (S)
- Initial state (S_0)
- Finite input alphabet (Σ)
- Finite output alphabet (Λ)
- Transition function ($T : S \times \Sigma \rightarrow S$)
- Output function ($G : S \rightarrow \Lambda$)

Graphically this is shown below in Figure [2.1](#).

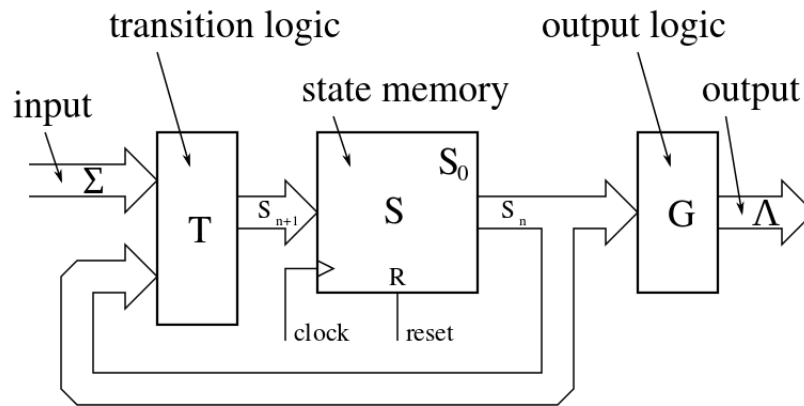


Figure 2.1: Moore Machine

Taken from: <https://commons.wikimedia.org/wiki/File:Moore-Automat-en.svg>

2.1 State Machine Syntax

The structure of a state machine is shown in the code below. It describes the six parts of a Moore machine described in Section 2 Language Structure.

2.1.1 Inputs

There are three input types. These are bit, vector and integer.

2.1.2 Pseudocode

```
Machine <NAME> {
  inputs {
    <TYPE> <NAME>;
    ...
    <TYPE> <NAME>;
  }

  outputs {
    <TYPE> <NAME> = <DEFAULT>;
    ...
    <TYPE> <NAME> = <DEFAULT>;
  }

  states {
    names = <Comma separated list of names>;
    launch = <One name from above>;
    <async or sync> reset <signal name>;
    clock <Name of clock signal for HDL>;
  }
}
```

```

TransitionLogic {
  <StateName1>:
    if <CONDITION>:
      nextState = <STATE>;
    else if <CONDITION>:
      nextState = <STATE>;
    ...
    else:
      nextState = <STATE>;
  <StateName2>:
    if <CONDITION>:
      nextState = <STATE>;
    else if <CONDITION>:
      nextState = <STATE>;
    ...
    else:
      nextState = <STATE>;
  ...
}

OutputLogic {
  <StateName1>:
    <OutputName> = <VALUE>;
    ...
  <StateName2>:
    <OutputName> = <VALUE>;
    ...
  ...
}
}

```

2.2 System Syntax

The purpose of a system is to be able to connect multiple state machines or systems together. The structure is shown in the code below.

```

System <NAME>
{
  // First define inputs/outputs
  In {
    <TYPE> <NAME>;
    ...
    <TYPE> <NAME>;
  }

  Out {
    <TYPE> <NAME>;
    ...
    <TYPE> <NAME>;
  }

  // List the systems/machines that this system is comprised of

```

```

System <NAME> <localname>;
...
System <NAME> <localname>;
Machine <NAME> <localname>;
...
Machine <NAME> <localname>;

// Next define the interconnections between systems/machines
// Only constraint is that all system outputs need to be driven
<localname>.input.<inputname> = <localname>.output.<outputname>;
...
<localname>.input.<inputname> = <localname>.output.<outputname>;

<Name of System>.output.<outputname> = <localname>.output.<outputname>;
...
<Name of System>.output.<outputname> = <localname>.output.<outputname>;
}

```