FPGA Implementation of BDD-based Dynamic Symbolic Controllers

Yue Li
May 12,2017
Period: March 13,2017-May 12,2017
Supervisor: Mahmoud Khaled

Abstract

The goal of this internship is to implement and improve the function of BDD2FPGA tool, which converts BDD (Binary decision diagram) file to VHDL file. There are various objectives in my internship tasks. Firstly, the workflow of SCOTS tool needed to be understood. The SCOTS tool has been implemented by former designers Matthias Rungger. Then BDD2FPGA tool is improved and implemented, which now can also serve for dynamic symbolic controllers BDD file. Furthermore a VHDL template is include as an input of BDD2FPGA. For debugging the tools, a BDD file of dynamic symbolic controller is generated personally, while the SCOTS tool could not synthesis dynamic symbolic controller. Finally a VHDL file is successfully automatically generated by BDD2FPGA tool.

During the nine weeks' research internship several improper functions left were fixed and some were modified. Basically BDD2FPGA is significantly improved, is suitable also for dynamic symbolic controller.

I. Introduction

SCOTS tool is an open-source tool, which synthesis symbolic controller for control system. The input of SCOTS tool is the specification of system, while the output is synthesis result saved as BDD file. It is implemented in C++, based on one of algorithm using BDD data structure. BDD data structure has advantages than other data structure. The BDD file requires smaller memory size, because it gives a compact representation for many practical functions as well as fast manipulation algorithms. It is widely used in CAD-tools nowadays. As well, it is very convenient for formal verification, while comparing of two functions is done by comparing two pointers to the root nodes. The BDD based implementation use the CUDD library by Fabio Somenzi.

BDD2FPGA tool is the main focus of this internship. This tool can read BDD file and then convert it to VHDL file, which can be implemented on FPGA. The input of this tool is symbolic controller synthesized result in BDD file as well a VHDL template, while the output of this tool is VHDL file. In the further part of report it will be discussed in detail.

II. Objectives

a. To understand the workflow of SCOTS as well as to use it synthesis symbolic controller for a control system based in small vehicle traction example.

b. To learn difference between various type of symbolic controllers: static symbolic controller and dynamic symbolic controller.

c. To implement BDD2FPGA tool in “header-only” style in C++.

d. To write a VHDL template for BDD2FPGA tool.

e. To generate a BDD file to debugging implemented tools.

III. Internship Contribution

Based on the basic work described in the chapter before, latest version of BDD2FPGA was applied, some functions were revised. Furthermore, some new useful functions are added. Details are interpreted in this chapter.

3.1 Understanding SCOTS tool

On the first period of the internship, the main task is to figure out the workflow of SCOTS. Using some example to understand it.

The workflow is showed as follow in Fig.1,

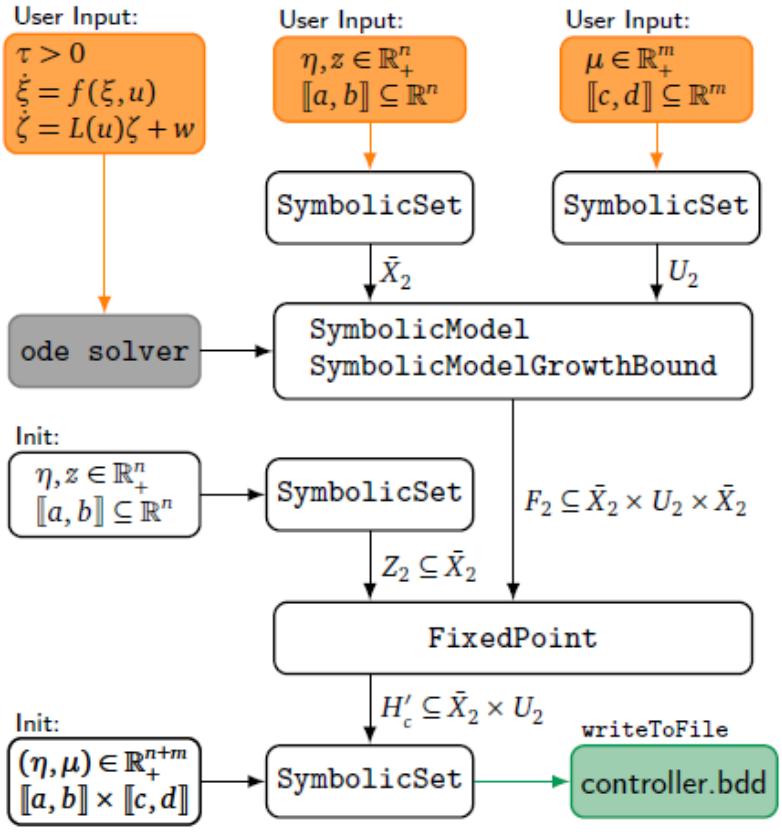

Fig1. The workflow of SCOTS

The synthesis result using SCOTS tool follows in Fig.2,

3.2 Static symbolic controller and dynamic symbolic controller.

The static symbolic controller controls the object of control system, which maintain a physical variable in the presence of disturbances. The input of a static controller is the state of system. The output of a static controller is the action for controlling system. The structure of static symbolic controller is showed in Fig.3,

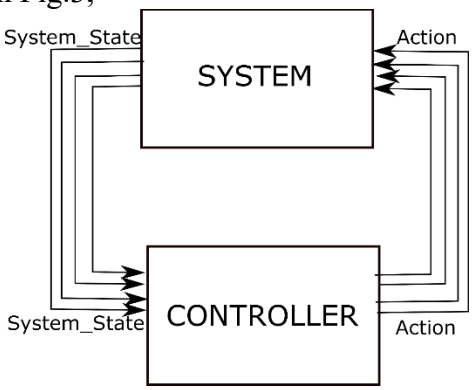

Fig.3

The dynamic symbolic controller controls the system, which a physical variable is required to follow and track. There are additional memory or register file to store the controller state. The input of a dynamic symbolic controller are the state of system and the current state of controller. The output of a dynamic symbolic controller are the action for controlling system and next state of controller. Besides there are additional memory for storing controller state, at the rising clock edge the next controller state can be shifted to the current controller state. The structure of dynamic symbolic controller is showed in Fig.4,

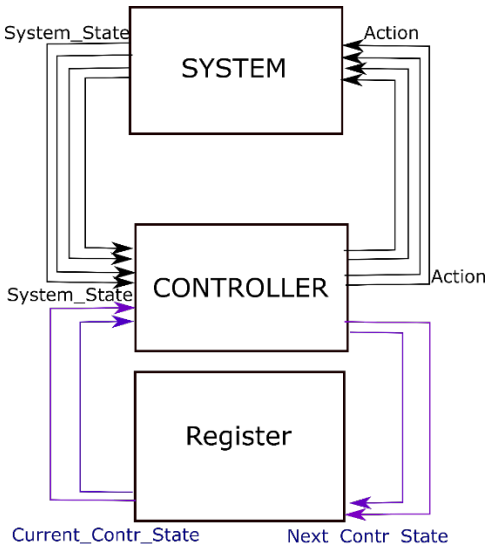

Fig.4

3.3 The implementation of BDD2FPGA

The whole BDD2FGA project is quite big and contains many classes. The most important and mainly modified classes are listed as follows:

- BDDReader
- BddDeterminizer
- BddToHDLdynamic
- BddDecomposer
- BddToString

The workflow is described as follow in Fig.5,

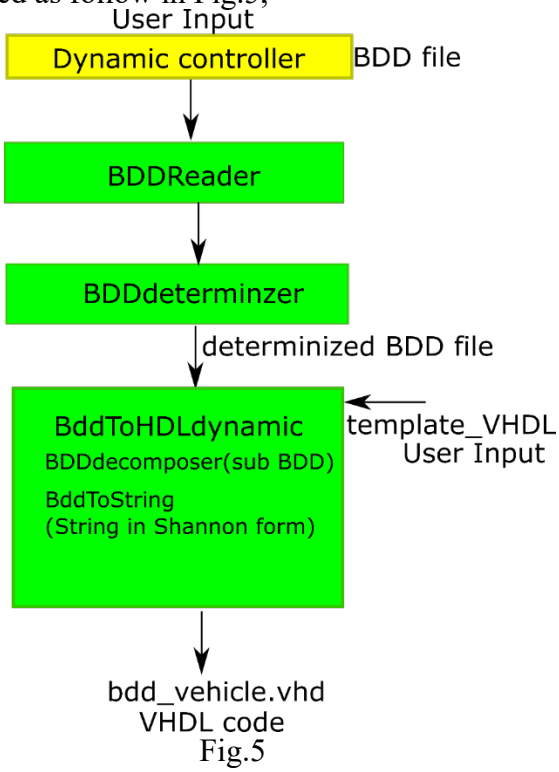

Fig.5

1. Use the class BDDReader to read input BDD file
2. Use the class BddDeterminizer to ensure unicity, bijection
3. Use the class BddToHDLdynamic to convert BDD file to VHDL code
4. In BddToHDLdynamic class

- the class BddDecomposer is used to decompose BDD file to sub BDD, aimed to be convenient converting BDD file as string in Shannon form
- the class BddToString is used to convert BDD file as string in Shannon form

3.4 VHDL template

The VHDL template is severed as the input of the BDD2FPGA tools.

The VHDL template contains 3 parts. Input and output entity declaration, architecture function implementation.

```
library ieee;
use ieee.std_logic_1164.all;

entity template_vhdl is
port(
clk: in std_logic;
res: in std_logic;
res_state: in std_logic;
#$ENTITY_INPUT_PORTS$#
#$ENTITY_OUTPUT_PORTS$#
);end template_vhdl;

architecture template_vhdl_behavioral of template_vhdl_entity is

signal current_state, next_state: std_logic_vector(#$noBddvarState$# downto 0);
process(clk,res)
begin
if res = '1' then
current_state <=res_state;

else if clk'event and clk='1' then
current_state <= next_state;
end if;
end if;

begin
#$BEHAVIORAL_OUTPUT_FUNCTIONS$#
end template_vhdl_behavioral ;
```

3.5 BDD file of dynamic symbolic controller.

For the purpose to debug BDD2FPGA, the CUDD library and some class in BDD2FPGA are used to generate BDD file for dynamic symbolic controller, while SCOTS can't synthesis for dynamic controller.

The main piece of code is as follow:

The obtained BDD file in Fig.

IV. An system example for debugging BDD2FPGA

The specification of the system: Infinitely often visit s1 and s4.

The state diagram for this example system is showed in Fig.6, there are 4 states in the system:S0, S1, S2, S3. There are 2 action in set of action: a,b.

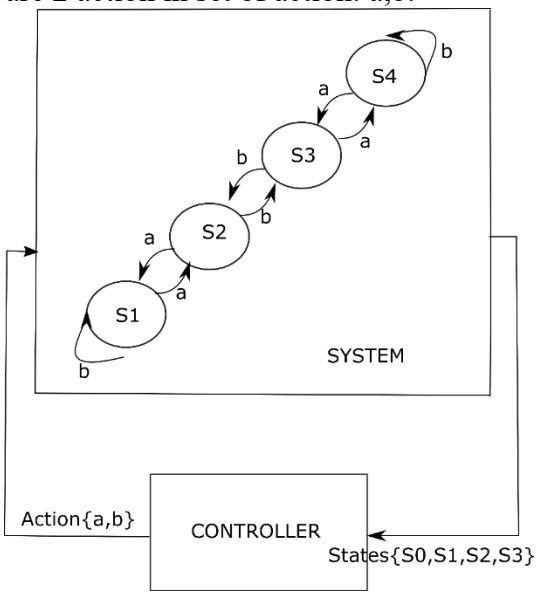

Fig.6

After synthesis personally, the dynamic controller structure is like follows, Fig.7:

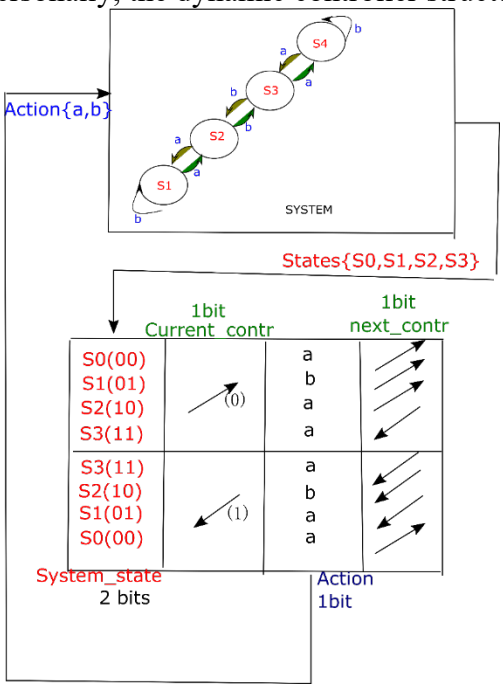

Fig.7

This system is in the 1 dimension, the system state can be represented by 2 bits. Controller state as well as action can be symbolled by 1 bit. The truth table of this dynamic symbolic controller can be expressed by Fig.7.

The result is a generated VHDL file, can be seen as following in Fig.8,

```
entity template_vhdl is
port(
clk: in std_logic;
res: in std_logic;
res_state: in std_logic;
x0: in std_logic;
x1: in std_logic;
f0: out std_logic
);
end template_vhdl;

architecture template_vhdl_behavioral of template_vhdl_entity is
    signal current_state, next_state: std_logic_vector(0 downto 0);
          process(clk,res)
          begin
          if res = '1' then
          current_state <=res_state;

        else if clk'event and clk='1' then
        current_state <= next_state;
        end if;
        end if;

            begin
          f0 <= not((x0 and ((x1) or (not((x2))))) or (not(x0) and (((x2)) or (not(x1)))));
f1 <= ((x0 and ((x1) or (((x2))))) or (not(x0) and ((x1 and ((x2))))));


            end template_vhdl_behavioral;
```

Fig.8

V. Conclusion and future work.

The work during the 9 weeks' research internship is based on the existing SCOTS and BDD2FPGA project. After the internship several improper functions left were adjusted and some were modified. After that the BDD2FPGA function is more powerful. Besides, a VHDL template has been written, serves as the input of the BDD2FPGA tool. Furthermore, a BDD file are generate by hand. Basically the tool is significantly improved, runs more stable and provides more functions now. During this internship I personally got a better understanding of what SCOTS tool and BDD2FPGA does and how it is executed. Besides, I practiced my programming skill with C++, which I think very meaningful.

In the future, the function of BDD2FPGA with a bigger BDD file could be examined and handled. Except this, the function of SCOTS, which synthesizing dynamic controller should be added, so that the design can be completed.

VI. Acknowledgment

At this time I would like to thank Institute of to offer me such a good opportunity to work on this project. Also I want to thank Mahmoud Khaled for the support and guidance during the whole period of time.