Cluster-wise State Encoding

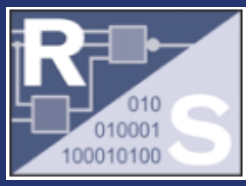
Mini Task Report

Florian Prott and Julian Käuser
Supervisor: Dennis Wolf M.Sc.

Start: 05/11/2017 | Submission: 06/14/2017

Institute for Computer Engineering | Computer Systems Group | Prof. Dr.-Ing. Christian Hochberger

TECHNISCHE
UNIVERSITÄT
DARMSTADT



1 Introduction

1.1 Cluster Search

In order to encode the states cluster-wise, a good partition into clusters must be found. In the following, the criteria for well-formed clusters concerning the previously presented encoding strategy are described.

Available LUT Inputs

In any case, each cluster must fit into one slice respectively lookup table. Therefore, all inputs and necessary signals for the state transitions inside the cluster and from other clusters must be fed to the LUT inputs. The amount of signals is calculated as shown in eqn. . One input is occupied for the cluster's own active bit; the states inside the cluster are encoded binarily, therefore ld(number of states) bits are needed for the states. From the external inputs, each bit which is not a shared dc (here called external inputs that care) in any of the contained transitions must be fed into the LUT inputs. At last, each other cluster's input which is associated with any of the contained states must be considered as one input. All of these conditions must be met simultaneously.

$$n_{ExternalInputsThatCare} + \tag{1.1}$$
$$n_{incomingInterClusterTransitions} + \tag{1.2}$$
$$log_2(n_{statesInCluster}) + \tag{1.3}$$
$$1 \tag{1.4}$$
$$<= N_{LUT-inputs} \tag{1.5}$$

Since the clustering problem is a combinatorial one, especially for large state machines, no ideal clusters can be searched. A heuristical approach is necessary. In this project, the application of a simulated annealing has been chosen because of its well-known effectiveness in hardware synthesis and the problem-independent implementation. The problem reduces to finding a suitable fitness function and mutation scheme. Due to the timed scope of this project, no further tuning of the other parameters like starting and end temperature could be included. Both fitness function and the mutation scheme are described in the next chapter.

2 Simulated Annealing

Simulated Annealing Algorithm

Fitness Function

3 Evaluation

In this chapter, a look on the output files and the quality of the cluster-wise state encoding is taken.

BLIF Format as Output

The BLIF format for finite state machines as described in [**?**] has some advantages which qualify it over a HDL representation as output format. Among these are the the good parseability, which is especially useful in small projects, and its independence of a specific platform. Additionally, the input files describing state machines in the scope of this project are .kiss filesm which are a subspecification of the BLIF format. Therefore, only the circuit mapping has to be written to the output file, and no conversion from BLIF to a HDL description of the functional behaviour has to be pursued. The .kiss format does not allow a more detailed circuit description than the mapping of latches. This must be done in a detailed .blif file. In our case, this disadvantage could be neglected, so that BLIF was the output format of choice.

Achievements

The presented method is capable of parsing .kiss2 files describing the behaviour of finite state machines, finding clusters of the included states according to an optimization towards their mapping to FPGA lookup tables and assigning codes to these found clusters. The results are written back into a .blif file containing the logical description and the codes for each cluster and state. Clusters are determined so that the logic associated with each of them fits into one lookup table. In the scope of this project, some parameter restrictions had to be assumed. The amount of inputs for available LUTs was assumed as 8, which is a possible input count in modern FPGA series. For thenumber of external inputs of the finite state machine, the soft limit of 4 is used. This value has been chosen so that the limitations for one cluster can be ensured; nevertheless, larger input counts may work depending on the amount of significant (i.e. non-don't-care) bits in the input vectors.

Results

The resulting logic circuits implementing the assigned codes have been fed into the ABC logic synthesis tool. As parameters, an 8-input LUT .lib description and `fpga -v; print_stats` have been used.

4 Bibliography