Backtracking Soduku Solver

Shabbar Vejlani
Design Verification Engineer
shabbarvejlani@gmail.com

***Abstract*—This paper describes a backtracking based soduku solver block from opencores.org[1]. It is based on implementation of an exact cover algorithm. As part of the NEILIT organized Capture the Bug Design Verification hackathon, the paper attempts to investigate the specification and come up with python based verification environment**

Keywords— Soduku solver, exact cover algorithm

I. Introduction (*Heading 1*)

Soduku is a class of exact cover problem, wherein the intent is to fill a 9x9 matrix with numbers from 1 to 9, with constraints such that:

1. each row consists of every number exactly one
2. each colum consists of every number exactly once
3. each 3x3 sub-block contains every number exactly once.

Exact cover algorithm[2] tries to iterate over every possible solution in a systematic way while ensuring that the constraints are met. If at any point the constraints are not met, it back tracks and proceeds with another possible solution. Eventually the alogrithm stops once the solution is reached.

II. Blocks

A. Sudoku Search

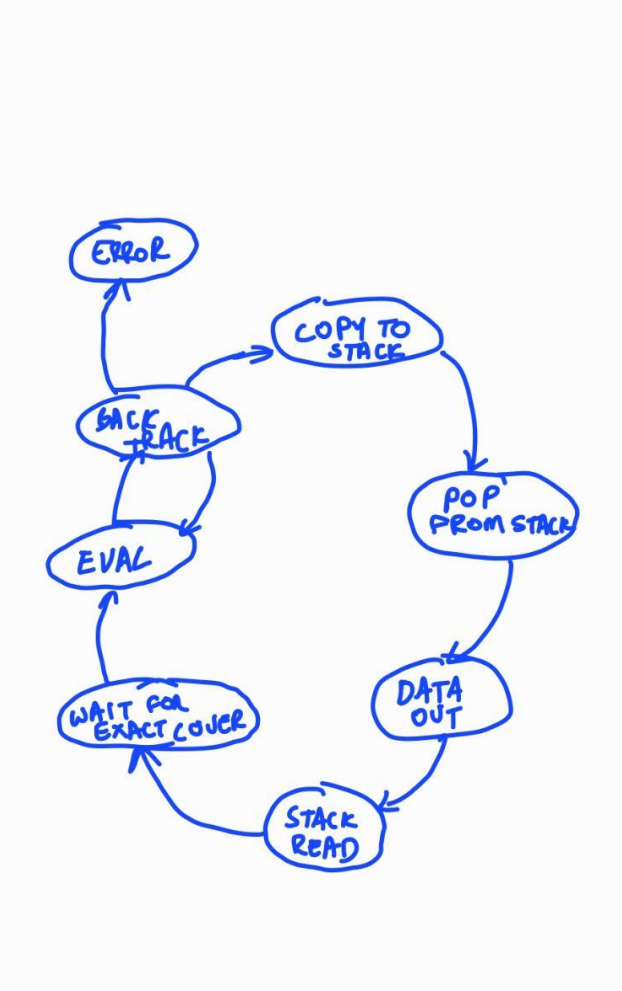
This is the top level block, which implements the FSM controller to sequence and iterate over the various steps of the exact cover algorithm. It generates the control signals for the Stack RAM and the sudoku cover block.

The top level interface of the module is as follows:

signal	input/output
clk	input
rst	input
start	input
inGrid	input
outGrid	output
done	output
error	output

B. Sudoku Search FSM

Note: The FSM inference may not be fully correct as it is based on initial skimming of the rtl code. As part of the verification during hackathon, this FSM will be updated as per the exact cover actual algo.



III. References

[1] https://opencores.org/ocsvn/sudoku/sudoku/trunk
[2] https://en.m.wikipedia.org/wiki/Exact_cover