CS 613200 Advanced Logic Synthesis
Final Project (2023, Spring)

Due Date: Jun.21st, 2023

TOPIC

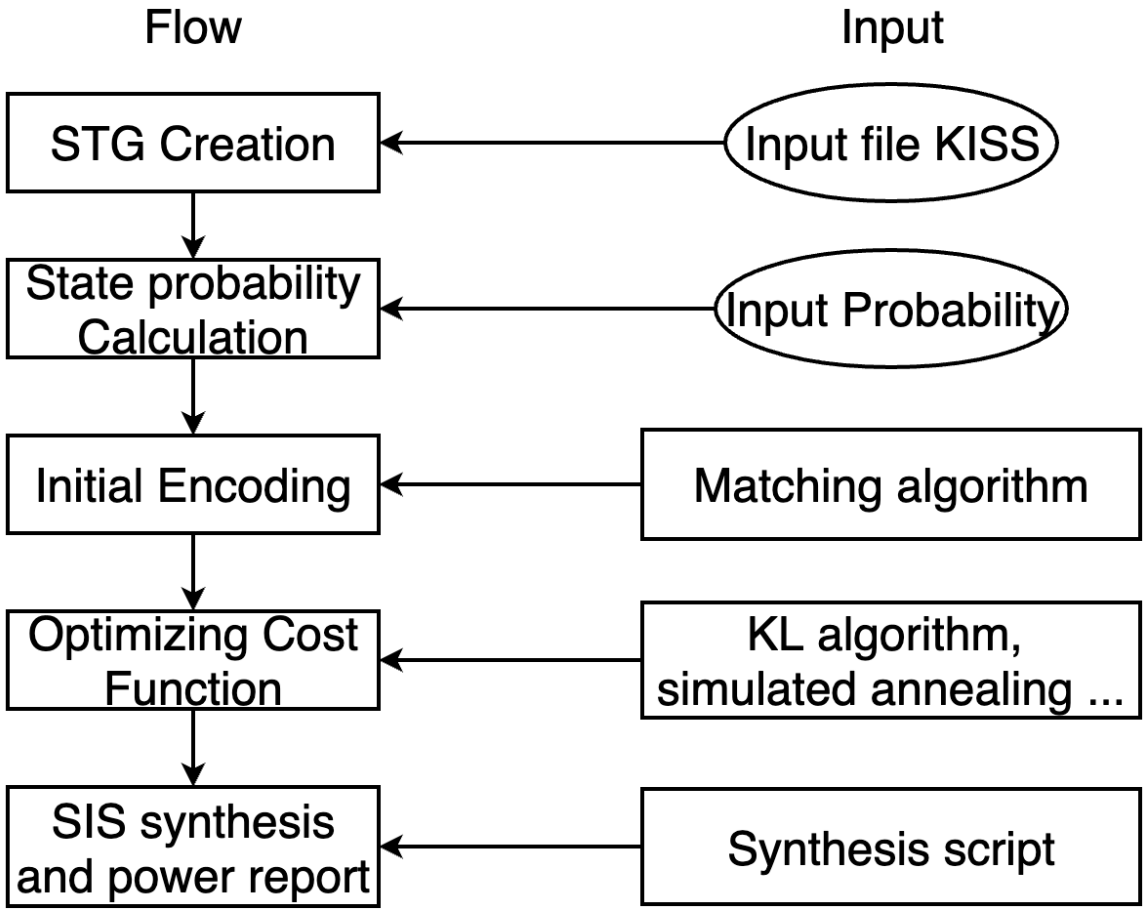
FSM State Assignment for Low Power Dissipation.

INTRODUCTION

The target of this project is to minimize power consumption of a finite-state machine (FSM) in multilevel logic implementations. The state coding will impact switching activity of sequential and combinational circuit of a FSM because of different number of gates switched during state transitions. Students are required to develop an automatic design flow to encode a given FSM for power optimization. Finally, the encoded FSM will be synthesized and power analyzed by SIS.

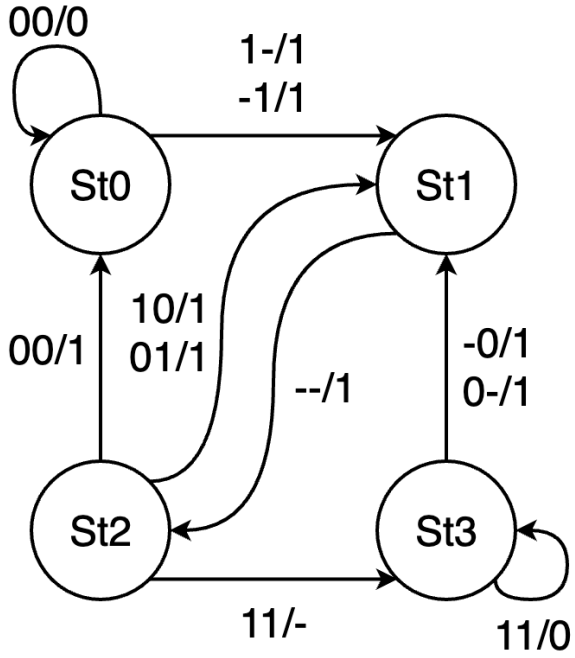
PROBLEM DESCRIPTION

Design flow:



STG Creation:

State Transition Graph (STG) is a representation of a FSM. The input file, KISS, describes the translation between each state. A STG of a FSM is shown in Fig. 1(a). The corresponding input file is shown in Fig. 1(b).

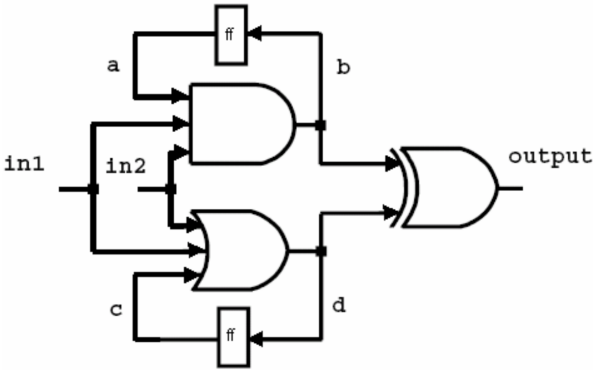



```
.model myfsm    #name of top module
.strat_kiss     #to embed a kiss file in a blif file
.i 2
.o 1
.s 4
.p 11
.r st2
00 st0 st0 0
-1 st0 st1 1
1- st0 st1 1
-- st1 st1 1
00 st2 st0 1
10 st2 st1 1
01 st2 st1 1
11 st2 st3 0
-0 st3 st1 1
01 st3 st1 1
11 st3 st3 0
.end_kiss       #kiss finish here
.code st0 00    #st0 is assigned to 00
.code st1 01    #st1 is assigned to 01
.code st2 10    #st2 is assigned to 10
.code st3 11    #st3 is assigned to 11
.end            #end of blif file
```

(a) State Transition Graph. (b) Corresponding KISS[1]

Fig. 1

For example, the column "**00 st0 st0 0**" means that in the current state **st0** if the input is **00**, the next state will be **st0** and the output will be **0**. The detailed description of KISS is shown in [8]. Figure 2 shows an example of state coding and the corresponding logical circuit.




```
.code st0 00
.code st1 01
.code st2 10
.code st3 11
```

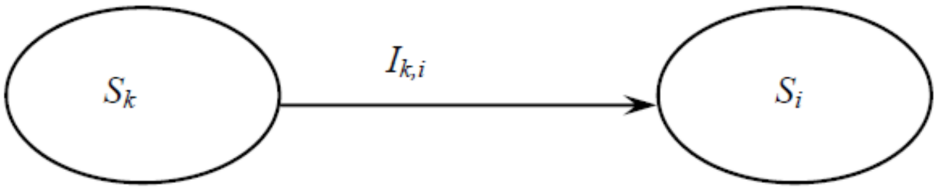
(a) Corresponding Logical Circuit. (b) State Assignment

Fig. 2

[1] Note that "-- st1 st1 1" is wrong, the correct is "-- st1 st2 1"

STG Creation [3][4][5]:

By assuming the switching probabilities of all input signals are 0.5. The state probability can be calculated by the state transition model [3] and linear programming tool (*GLPK*) [5],



$$\begin{cases} \Pr ob(S_1) + \Pr ob(S_2) + ... + \Pr ob(S_m) = 1, \\ \Pr ob(S_i) = \sum_{S_k \in PS(S_i)} \Pr ob(S_k) \cdot \Pr ob(I_{k,i}), \text{i} = 1 \text{ to m}. \end{cases}$$

where $PS(S_i)$ is the set of immediately previous states of S_i ; Prob(S_i) is the state probability of stae S_i, and Prob($I_{k,j}$) is the probability of input pattern $I_{k,i}$. The state and state transition probability will be used as a part of cost function in your algorithm. Here have some references that you can use to solve matrix computation [4].

Initial Encoding of FSM [6][7]:

To find an optimized state assignment of a FSM for low power consumption is a NP-hard problem. By exhaustive search to find the solution is computationally expensive. Lots of work is approximate method by a cost function to estimate power consumption before the FSM synthesized. A good initial solution will reduce the computation time of searching. You are required to use an existing tool (*LEDA*) [7] to implement matching algorithm for an initial solution.

For a given FSM, if there are n states in STG, encoding bit length is between $log_2 n$ to n.

Optimizing cost function [3]:

Because of the uneven distribution of state transitions in FSM, state assignment that states with high transitions are given state codes of short

distance will reduce the number of signal's switching and hence the dynamic power consumption. The objective is to minimize,

$$\sum_{\substack{allpairs \\ s,t}} w(s,t) \times dist(enc(s), enc(t))$$

where *w(s,t)* is the power cost function of the transition between state *s* and state *t* under the encoding, *enc(s)* and *enc(t)*. In this project, we only consider dynamic power consumption. Please refer to [5] and [6] for more detail modeling about cost functions. You can perform simulation annealing algorithm or other techniques for power optimization in this phase.

SIS synthesis and power report [8]:

After optimization, an encoded FSM is written out with a specified format, BLIF. Figure 3 shows an output example of the FSM shown in Figure 1.

```
.model myfsm    #name of top module
.strat_kiss     #to embed a kiss file in a blif file
.i 2
.o 1
.s 4
.p 11
.r st2
00 st0 st0 0
-1 st0 st1 1
1- st0 st1 1
-- st1 st1 1
00 st2 st0 1
10 st2 st1 1
01 st2 st1 1
11 st2 st3 0
-0 st3 st1 1
01 st3 st1 1
11 st3 st3 0
.end_kiss       #kiss finish here
.code st0 00    #st0 is assigned to 00
.code st1 01    #st1 is assigned to 01
.code st2 10    #st2 is assigned to 10
.code st3 11    #st3 is assigned to 11
.end            #end of blif file
```

Fig. 3[2]

[2] Same as corner mark 1.

A script “opt_map_power.scr” is provided for power report in SIS. There are 3 phases in the script. The first phase is logic independent optimization by SIS default script “*script*”. The second phase is to perform technology mapping by library “synch.genlib”. The final phase is power report by SIS internal command “power_estimate”. Here is an example of power report.

```
Combinational power estimation, with Zero delay model.
Network: jedi_output, Power = 877.6 uW assuming 20 MHz clock and Vdd =
5V
```

The frequency and Vdd setting is fixed in 20Mhz and 5V respectively. To use the script, the example is as follow

```
$> sis

sis> read_blif my_fsm_outputfile #read in the encoded fsm

sis> source opt_map_power.scr #run script
```

In SIS, the power dissipated in a circuit due to switching activity is calculated by

$$P=0.5\times Vdd^2\times\sum(p\times C)\times f$$

where

p_i = expected number of transitions of node i in one clock cycle

C_i = capacitive load of node I

f = 20Mhz

Vdd = 5V

The expected number of transitions of each node per clock cycle is calculated through symbolic simulation, based on the static probabilities of the primary inputs (by default prob_one = prob_zero = 0.5). The capacitive load of a node is obtained by summing the gate capacitances of its fanout nodes and adding some internal drain capacitance. Gate capacitances are multiple of a minimum sized transistor (0.01pF), admitting transistor sizing based on the number of inputs to the node (up to a value max_input_sizing, default 4). Drain capacitances are calculated from the number of transistors this node has (multiple of 0.005pF) and this number can be obtained either from a factored form or sum of products.

Requirements

- One page report to summarize your design flow, cost function and result
- Field Demo.

Grading

- Program Functionality Design Flow
- Cost Function
- Power Result

References

[1] Benini, L., De Micheli, G., "*State assignment for low power dissipation*" IEEE Journal of Solid-State Circuits, Volume 30, Issue 3, March 1995, Pages: 258- 268.

[2] Chi-Ying Tsui, Pedram, M. Despain, A.M., "*Low-power state assignment targeting two- and multilevel logic implementations*", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems Volume 17, Dec. 1998 Page(s):1281 – 1291.

[3] ALS Chapter7: Low Power Design, pp.12-16.

[4] Gauss Elimination

[5] GNU Linear Programming Kit (GLPK)

[6] ALS Chapter6: Finite State Machine

[7] Library of Efficient Data types and Algorithms (LEDA)

[8] SIS: A System for Sequential Circuit Synthesis