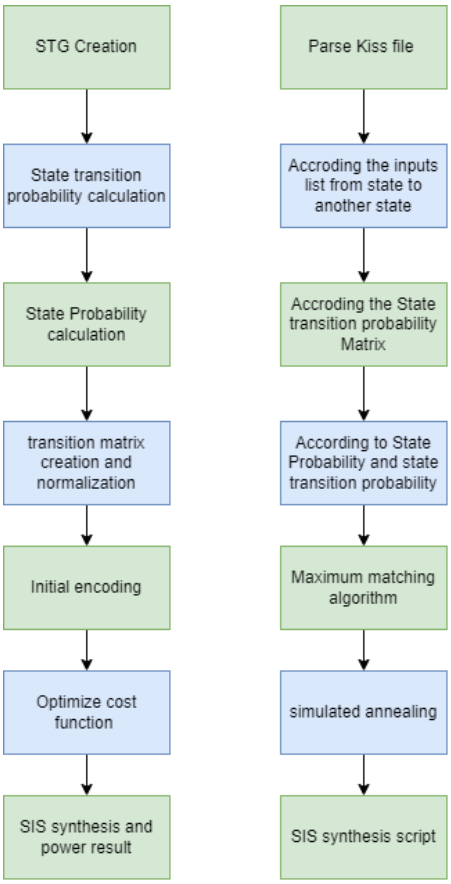
Advanced Logic Synthesis HW3 Report

Jie-Hong Liu | 劉杰閎
jiehong0914@gmail.com
College of Semiconductor Research, National Tsing Hua University
Hsinchu, Taiwan

Design Flow:

My design flow starts from input the kiss file, I would calculate state transition probability from the input list. After calculate state transition probability, we can use Linear Programming tool (GLPK) to calculate State probability with the constraint $\sum Prob(S_i) = 1$, where i starts from state 0 to n (the number of states). After that, we can get our transition matrix by multiply the state probability to state transition probability matrix. It indicates the truly probability for the states. Normalization is convenience for us to observe each weight and calculate the cost.

I use maximum matching from LEDA to return the edges with higher weights, it means there is more chance from FSM to perform this transition, then I gave the highest weight edge with closely encoding as their initial encode. After that, I use a Simulated Annealing to optimize my cost function, which is defined below. Finally, I will use SIS script to synthesis my FSM and get the power result by its estimation.



Cost Function:

$$cost = \sum_{allpairs(s,t)} w(s,t) * hanminDist(enc(s), enc(t))$$

I use hamming Distance to estimate the distance between two states. I will use Simulated Annealing to further improve my result.

Power Results:

My power is estimated by 20 MHz clock and VDD = 5v. (using Zero delay model).

	Initial Encoding		SA optimization		Result	
Benchmark(states)	Power(uW)	Cost	Power(uW)	Cost	Power Reduction(%)	Cost Reduction (%)
Beecount (4)	331.8	125	342.7	88	3.285111513	-29.6
S27(5)	438.3	164	377.1	148	-13.96303901	-9.756097561
Bbtas(6)	264	26	246.4	20	-6.666666667	-23.07692308
Bbara(7)	449.9	342	389.4	283	-13.44743276	-17.25146199
Dk14(7)	1276.3	728	1168.7	576	-8.43061976	-20.87912088
Bbsse(13)	1040.5	4.92E+07	1114	3.98E+07	7.063911581	-19.00081384
Cse(16)	2066.1	1126	2042.2	1070	-1.156768791	-4.973357016
Ex1(18)	3062.6	363	2944.6	318	-3.852935414	-12.39669421
Dk16(27)	2791.7	2455	2615.1	1792	-6.325894616	-27.00610998
Snad(32)	6263.9	1191	6211.3	739	-0.839732435	-37.95130143
AVG					-4.433406637	-20.189188

Table 1. Power result after design flow.

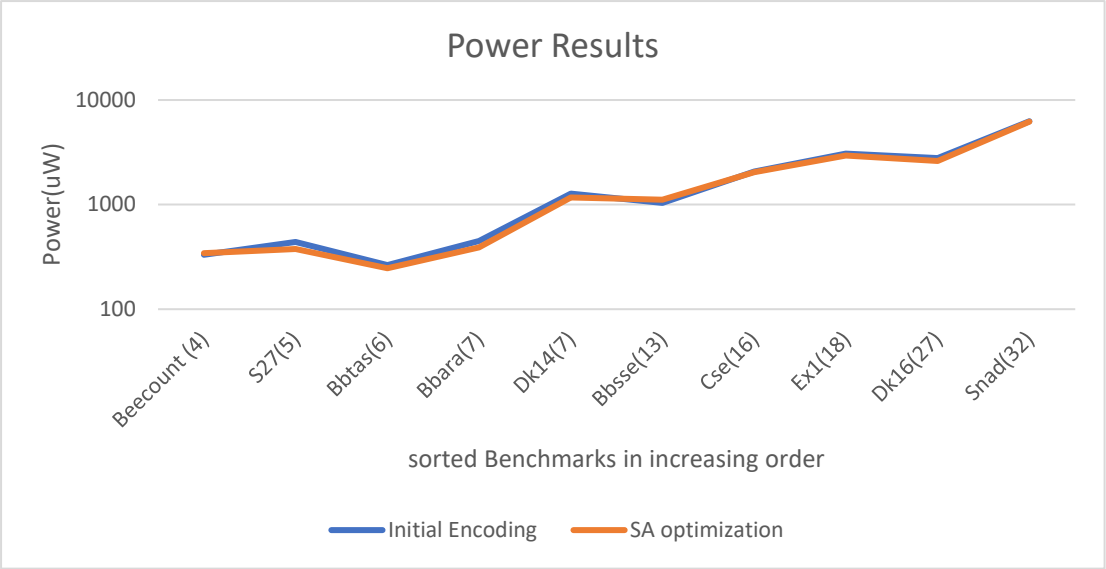


Fig1. Power result of each benchmark

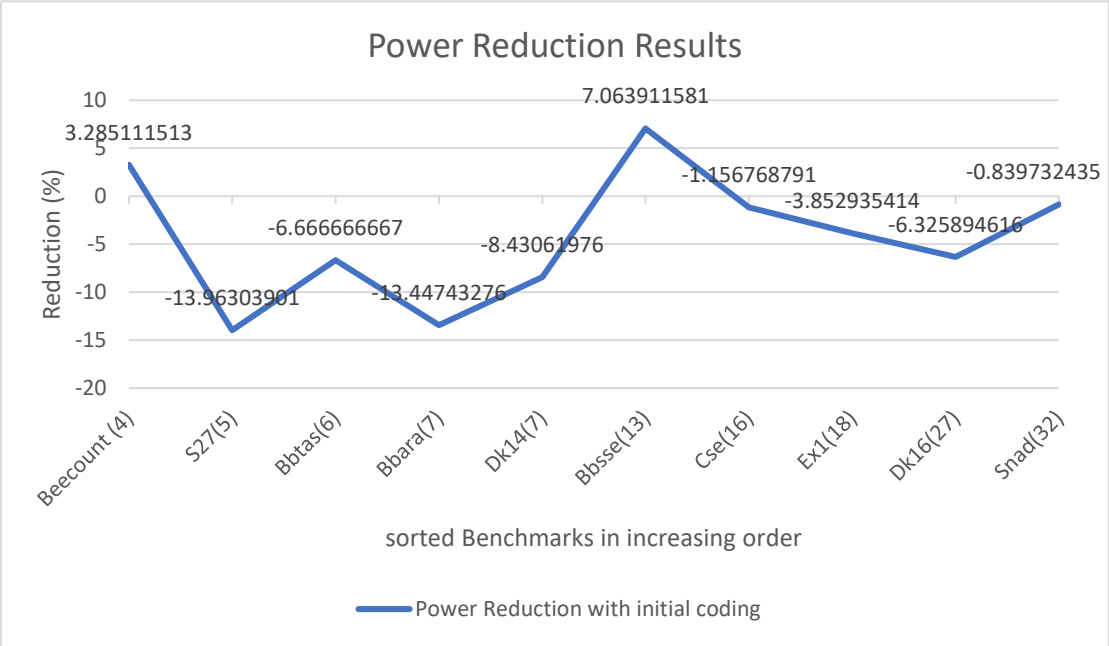


Fig2. Power reduction result of each benchmark

As the Table1 result, we can find that although our cost always has reduction due to Simulated Annealing, but in some case such as “beecount” or “bbsse”. In these two benchmarks, as the cost decrease, the power increase. Fig1 is the power result of each benchmark. And Fig2 is the power reduction results of each benchmark. The benchmark with positive power reduction shows that the cost function is not detail modeling enough for power dissipation, so that the effort we did on SA, can not promise the improvement on power reduction.

By this reason, I try to use SIS tool as my cost model. Table 2 is the result of adopting SIS power result as my cost function. Look back on “beecount” or “bbsse” benchmark, they didn’t increase as above case, since the cost function is same as our power estimation tool. Fig 3 is the power result of each benchmark. As the Fig 3 shows, the result always lower than initial encoding after SA optimization. And Fig 4 is the power reduction results of each benchmark. Note that S27 always has highest power reduction in both two experiments, it means that the initial encoding in this benchmark have a lot of space to improve.

	Initial Encoding	SA optimization	Result	
Benchmark(states)	Power(uW) = cost	Power(uW) = cost	Time (s)	Power Reduction(%)
Beecount (4)	331.8	303.5	82.545	-8.529234479
S27(5)	438.3	238.2	85.573	-45.65366188
Bbtas(6)	264	208.6	77.323	-20.98484848
Bbara(7)	449.9	378.7	77.280	-15.82573905
Dk14(7)	1276.3	860.4	103.261	-32.58638251
Bbsse(13)	1040.5	890.3	114.837	-14.43536761
Cse(16)	2066.1	1676.8	202.088	-18.8422632
Ex1(18)	3126.3	2163.9	380.88	-30.7839938
Dk16(27)	2791.7	2026.3	158.385	-27.41698607
Snad(32)	6694.2	4785.2	613.809	-28.517223
AVG				-24.3576

Table 2. Power result after adopting SIS model.

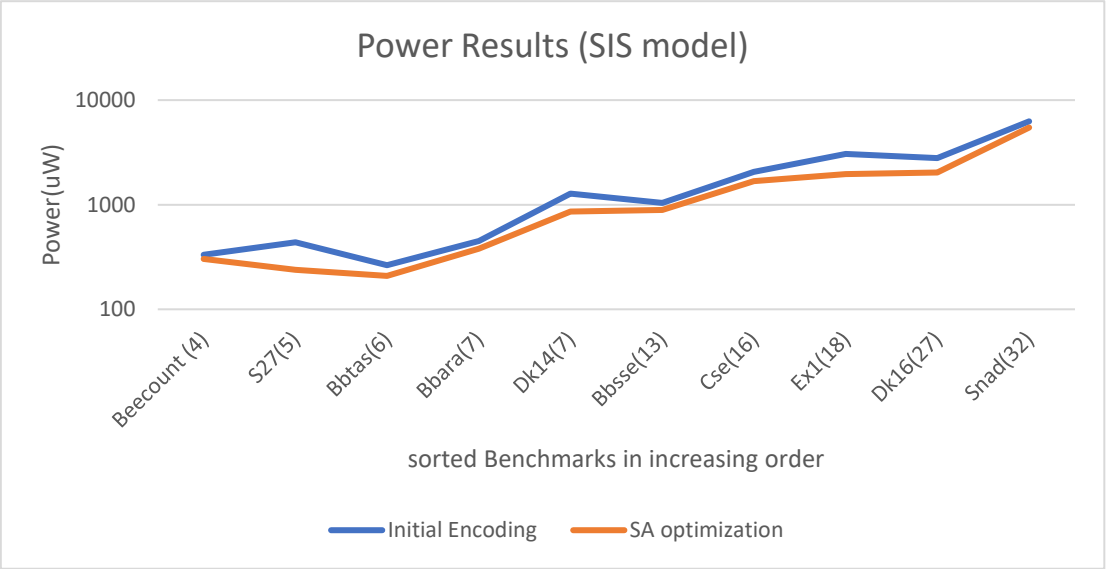


Fig3. Power result of each benchmark

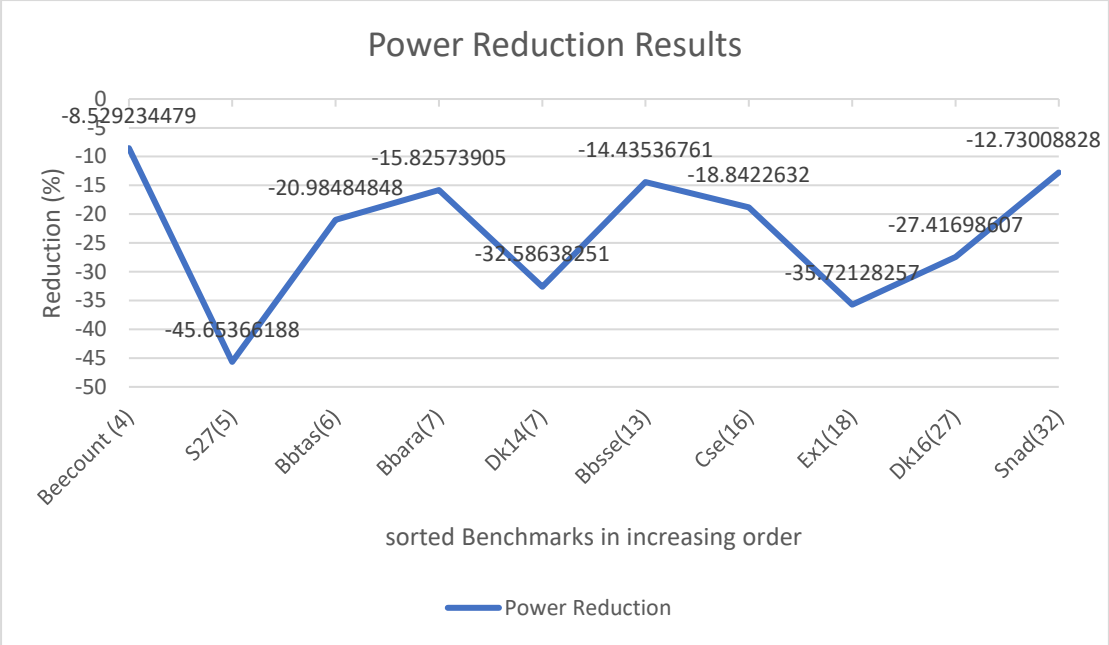


Fig 4. Power reduction result of each benchmark

Since we found that there is a command in SIS called “state assign”, this command can assign state to STG by its optimization. By Table 3, we can see in spite of “Beecount” benchmark (the smallest states), there is always have power reduction on each benchmark. As Fig 5 and Fig 6, my SA optimization result is always smaller or equal than SIS optimization result, it means my design flow is good enough to perform low power state assignment.

	SIS optimization	SA optimization	Result	
Benchmark(states)	Power(uW)	Power(uW) = cost	Time (s)	Power Reduction(%)
Beecount (4)	303.5	303.5	82.545	0
S27(5)	292.1	**238.2**	85.573	-18.45258473
Bbtas(6)	224.0	**208.6**	77.323	-6.875
Bbara(7)	388.2	**378.7**	77.280	-2.447192169
Dk14(7)	1023.2	**860.4**	103.261	-15.91086787
Bbsse(13)	1092.9	**890.3**	114.837	-18.53783512
Cse(16)	1906.2	**1676.8**	202.088	-12.03441402
Ex1(18)	2727.9	**2163.9**	380.88	-20.67524469
Dk16(27)	2433.0	**2026.3**	158.385	-16.71598849
Snad(32)	7142.8	**4785.2**	613.809	-33.00666405
AVG				-14.46557911

Table 3. Final power result compares with SIS optimization result

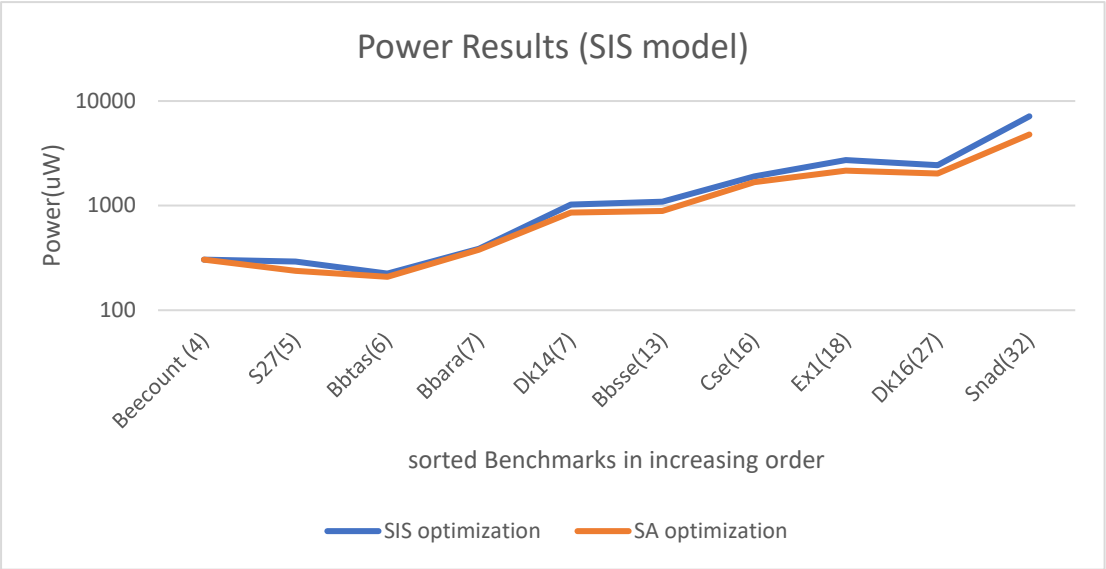


Fig 5. Power result of each benchmark

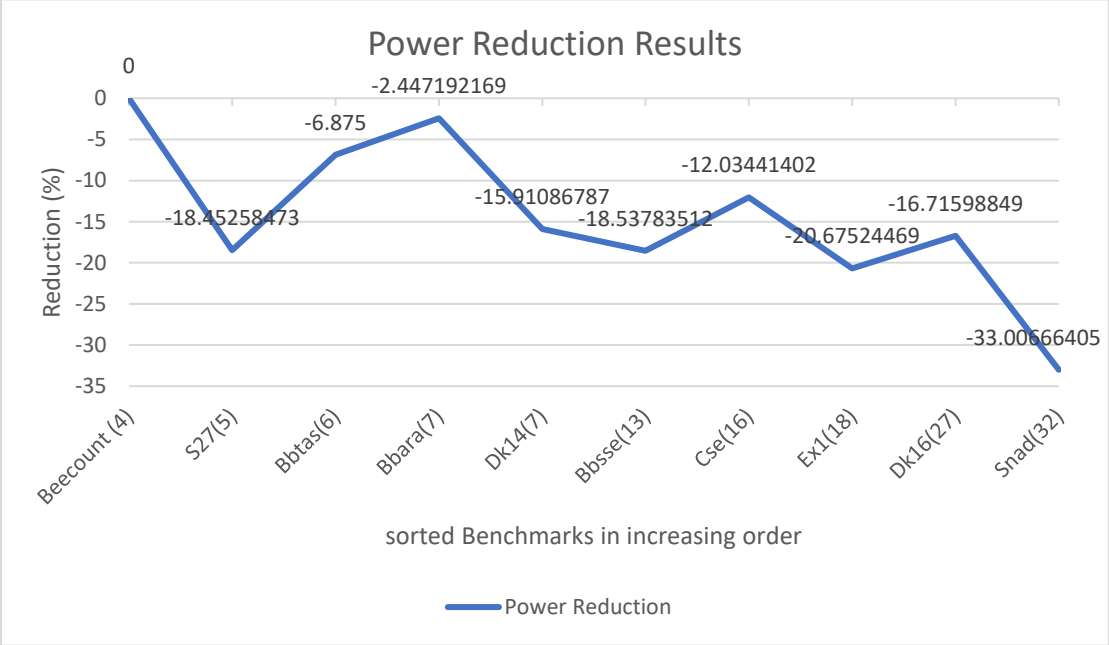


Fig 6. Power reduction result of each benchmark