Report

Name: 張硯博 **Student ID**: 113062645

1. Introduction

The programming assignment requires implementing a technology mapping for FPGA, aiming to minimize the number of lookup tables (LUTs) used, which is an NP-hard problem.

---

2. Approach and Complexity

2.1 Approach v1.0

I use a cut-enumeration framework, which generates every feasible cut for each gate in the circuit, then applies a scoring function to determine which cut is better.

$$f(K, v) = \otimes_{u \in input(v)}^{K} [u + f(K, u)]$$

The above equation from [2] gives the way to recusively calculate all the K feasible cut set of a gate v.

After obtaining the feasible cut set for a gate v, I use the formula below to calculate the cost(A_C) of a cut C, which attempts to estimate the number of LUTs required if this cut is used [1, 2, 3, 4]. Then, I choose the cut with the minimal cost for v, which represents the LUT rooted at v.

$$A_C = \sum_{i \in input(C)} [A_i / ref(i)] + 1$$

, where $ref(i)$ is the number of outgoing edge of gate i.

After the LUT for each gate is chosen based on the above approach, I start mapping the circuit into LUTs, beginning with the primary output gates. In the meantime, update $ref(i)$ to actual outgoing edge in mapping.

To summarize, my approach consists three steps:

1. generate all feasible cut for all gates.
2. Label each gate with an LUT which has minimun cost.
3. Map the circuit using the chosen LUTs and update $ref()$.
   (then go back to step 2)

The number of iterations for steps 2 and 3 is set to a constant.

2.2 Time complexity of Approach v1.0

Let N denote the number of gates in the circuit, MC the maximum number of cuts among all gate, and K the number of inputs for the LUT used in mapping. Assume that every gate in the circuit is 2-bounded.

The time complexity of steps 1, 2, and 3 is $O(N \cdot MC^2 \cdot K)$, $O(N \cdot MC \cdot K)$, and $O(N \cdot K)$, respectively.

It is clear that step 1 is the dominant step, as it involves a component that raises MC to the power of 2, which is problematic since MC is $O(N^K)$.

2.3 Approach v2.0

The execution time for Approach v1.0 on the spla circuit (which contains 7,454 gates) under $K = 8$ was approximately 12 minutes. This aligns with the high complexity discussed in Section 2.2. However, this execution time exceeds the assignment requirement of 10 minutes, indicating that Approach v1.0 may not be suitable for this particular constraint.

To address the time complexity issue, I applied cut pruning [2]. In simple terms, this technique limits the number of cuts retained for each gate to a constant, denoted as $\mid C \mid$. If the number of cuts exceeds $|C|$, the cut with the larger cost is discarded. This approach helps in reducing the overall computational complexity and brings the execution time closer to the assignment requirements.

To summerize, approch v2.0 revises step 1 of v1.0:

1. generate **at most |C|** feasible cut for all gates.
   The other steps remain unchanged.

It’s worth mentioning that the performance of the cut-pruning version is very close to that of the original version. Due to the significantly reduced time complexity, the algorithm can run more iterations of steps 2 and 3, which I set to 10. As a result, the overall performance of approach v2.0 is slightly better than v1.0.

2.4 Time complexity of Approach v2.0

The time complexity of step 1 becomes $O(N \cdot |C|^2 \cdot max(K, \log_2(|C|^2)))$, where $|C|$ is set to be 80 when $N <= 25000$, and $80 \cdot \sqrt{\frac{25000}{N}}$ when $N > 25000$ to meet the time constraint. Hope my estimation is not wrong 🤫. The $\log$ term is due to sorting the cuts by cost. For step 2 and 3, time complexity remain the same.

---

3. Experiment result

Circuit	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8
spla	4425	3152	2539	2084	1743	1398
alu4	1644	1186	969	810	687	575
apex4	1405	1028	863	750	667	597
cordic	492	364	274	235	213	189

***Table 1**: Comparison of number of LUTs for different circuits and values of K.*

Circuit	K = 3	K = 4	K = 5	K = 6	K = 7	K = 8
spla	128.941	397.763	2398.82	14411.7	35743.2	44773.1
alu4	35.7409	83.4478	228.527	837.63	2377.19	3335.59
apex4	33.6302	107.485	465.359	2603.24	6806	8374.41
cordic	12.7399	28.8603	145.519	934.414	2413.27	2824.14

***Table 2**: Comparison of the execution time (in ms) for different circuits and values of K.*

4. Conclusion

In this assignment, I implemented a heuristic algorithm aimed at minimizing the number of LUTs used in mapping. Since depth is not considered, the concept of this algorithm is straightforward: enumerate the feasible cuts and select the one with the minimal area. Accurate area estimation is crucial—the closer the estimated area is to the actual mapping result, the fewer LUTs will be used. After cut-pruning, performance is very close to the version that enumerates all cuts (at worst, only 2% less efficient) but with a significant speedup (approximately 16 times faster in the SPLA circuit with $K = 8$). Lastly, due to reduced computation time, the algorithm can perform more iterations within the time constraint, making the cut-pruning version superior to the full enumeration version.

5. References

[1] D. Chen and J. Cong, "DAOmap: a depth-optimal area optimization mapping algorithm for FPGA designs," IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004,

[2] Cong, Jason & Wu, Chang & Ding, Yuzheng. (1999). Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution.

[3] V. Manohararajah, S. D. Brown and Z. G. Vranesic, "Heuristics for Area Minimization in LUT-Based FPGA Technology Mapping," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[4] P. Wang et al., "EasyMap: Improving Technology Mapping via Exploration-Enhanced Heuristics and Adaptive Sequencing," 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD),