FPGA Architecture & CAD Final Project

Multi-level Topology-Driven Partitioning

Team Member's SID/Name (Contribution %):
109062556 李濬安 (50%, Data Structure, I/O, Partitioning)
110062526 小山翼 (50%, Algorithm Construction, Propagation, Report)

Ⅰ. How to compile and execute the program.

- To compile the program, navigate to the "./src" directory and execute the following command. The executable file named "topart" will be generated in the "./bin" directory.

Command: $ make

- The program can be executed using one of the following command in either the "./src" or "./bin" directories.

Command 1: $ ../bin/topart <input file> <output file>

Command 2: $ ./topart <input file> <output file>

e.g.: $ ./topart ../input/B1.txt ../output/output1.txt

Ⅱ. Overall Flow

In this work, we implement a multi-level topology-driven partitioning on FPGA. Fig.1 shows the overall flow, which starts with the propagation phase followed by the partitioning phase. We do not implement the coarsening stage [1] as we think the case is small and there is no need. To efficiently compute unions and inter-sections, we make use of **bitset**, a data structure from the C++ Standard Template Library (STL), in our code. The details of our algorithms for candidate FPGA propagation and topology-driven partitioning are presented in Sections Ⅲ and Ⅳ, respectively.

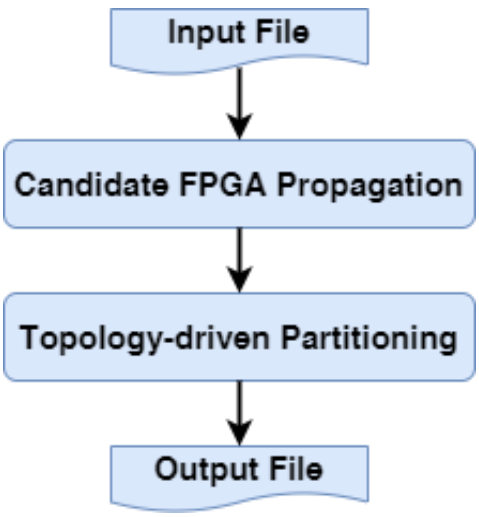


Fig. 1: Overall Flow

Ⅲ. Propagation

Algorithm 1 shows our proposed propagation method.

Bit Sets Initialization (lines 1-2):

- Initialize the bit sets that store the FPGA candidates that can be reached from each FPGA (line 2) and the FPGA candidates that are available for each circuit node (line 3).

Calculate reachable FPGAs at distance 1 from each FPGA (lines 3-5):

- Examples are shown in Fig.1.

Calculate Remaining Distances (lines 6-11):

- The distance for each FPGA can be calculated by taking the union of the bit set from the previous distance and the bit sets of each FPGA that is reachable at a distance of 1.
- Ex: distance 2 from F0 in Fig.2
  - 1001 (F0, dis: 1) ∪ 1111 (F3, dis: 1) = 1111 (F0, dis: 2)

FPGA Propagation for Movable Nodes (lines 12-19):

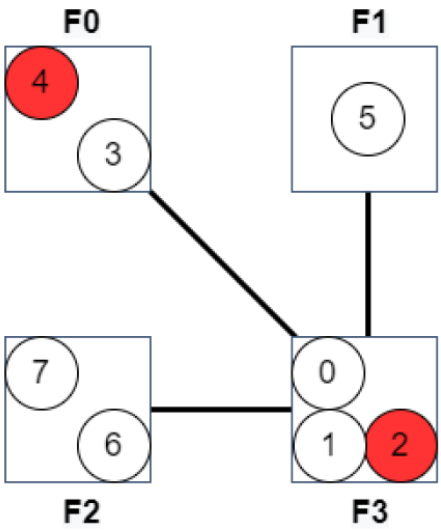
- **For each fixed node's FPGA and each distance, repeat the following steps:**
  1. find the current nodes' neighbor
  2. perform the intersection between the bit set for the fixed node's FPGA at distance i and the bit set for the current node's neighbors
  3. replace the current nodes with the current neighbor nodes

Algorithm 1: Candidate FPGA Propagation

Input: FPGA Graph $\hat{G}(\hat{E}, \hat{V})$,
Circuit Graph $G(E, V)$,
Fixed node-FPGA pair queue Q

Output: Candidate FPGA bit set $Cddt(v_i)$ for each node v_i

1. initialize $FPGA_{candidate}$ with $bitset(|\hat{V}|, 0)$
2. initialize $Cddt(v_i), \forall v_i \in V$ with $bitset(|\hat{V}|, 1)$
3. **foreach** FPGA connection channel **do**
4. $set(id_1, id_2, FPGA_{candidate}(id_1,1), 1)$
5. $set(id_1, id_2, FPGA_{candidate}(id_2,1), 1)$
6. **foreach** FPGA node $\hat{v_i} \in \hat{V}$ **do**
7. $dist \leftarrow 1$
8. **while** $bitcount(FPGA_{candidate}(\hat{v_i})) < |\hat{V}|$ **do**
9. **foreach** $j \in bit(FPGA_{candidate}(\hat{v_i}, dist))$ **do**
10. $Union(FPGA_{candidate}(\hat{v_i}, dist), FPGA_{candidate}(\hat{v_j}, 1))$
11. $dist \leftarrow dist + 1$
12. **while** $Q.size() > 0$ **do**
13. $(v_i, \hat{v_i}) \leftarrow Q.pop()$
14. $Circuit_{nodes} \leftarrow \{v_i\}$
15. **for** $x = 1, 2, \ldots, maxDist(FPGA_{candidate}(\hat{v_i})) - 1$ **do**
16. $Circuit_{neighbors} \leftarrow FindNeighbors(Circuit_{nodes})$
17. **foreach** $nNode \in Circuit_{neighbors}$ **do**
18. $Intersection(Cddt(nNode), FPGA_{candidate}\hat{v_i}, 1)$
19. $Circuit_{nodes} \leftarrow Circuit_{neighbors}$



FPGA: Bit Set for dis. 1

F0: 1001

F1: 0101

F2: 0011

F3: 1111

Fig. 2: Examples of bit set settings.

Ⅳ. Partitioning

Algorithm 2 (on the next page) roughly shows our proposed partition method.

Neighbor Node Placement for each Fixed Node (lines 1-11):

- Find neighbor nodes, and sort them in increasing order of possible FPGA candidates (lines 2-3).
- **Node Placement & Candidate Update (lines 5-11):**
  - Place into the FPGA where the fixed node is located if FPGA capacity does not exceed the given value (lines 6-7).
  - Otherwise, place it into one of the neighboring FPGAs (lines 9-11).
  - Update the candidates for the movable circuit nodes (lines 7&11).

Remaining Node Placement (lines 12-24):

- Identify the nodes with the least possible candidates (line 13).
- **Candidate Restrictions (lines 15-22):**
  1. Allow the node to be placed only in the union of its unplaced neighbor nodes' possible FPGA candidates (lines 15-17).
  2. Allow the node to be placed only in the union of its already placed neighbor nodes' FPGA candidates. (lines 18-22).
     - Hope it can help to decrease external degrees.
     - Cancel it if this restriction removes all possible candidates for the current node (lines 21-22).
- **Node Placement & Candidate Update (lines 23-24):**
  - Place the node into one of the possible candidates after performing the previous two restrictions (line 23).
  - Update the candidates for the movable circuit nodes (line 24).

Ⅴ. Results

Benchmark							External Degrees	Runtime (s)
Case	# FPGAs	# channels	FPGA capacity	# nodes	# fixed nodes	# nets		
B1	8	11	6	26	25	5	41	0.001
B2	8	11	30	200	100	5	225	0.005
B3	43	214	30	1000	500	43	699	0.043
B4	43	214	60	2000	1000	43	1833	0.073
B5	43	214	281	10000	6666	129	15517	0.438
B6	43	214	1297	50000	33333	129	92745	2.739
B7	43	214	2634	100000	66666	129	188721	6.738
B8	43	214	5594	200000	133333	129	382054	17.51

Algorithm 2: Topology-driven Partitioning

Input: Bit Set $Dis(\hat{v}_i, j)$ at each distance j for every FPGA $\hat{v}_i \in \hat{V}$,
Candidate FPGA Bit Set $Cddt(v_i)$ for each circuit node $v_i \in V$,
Fixed node-FPGA pair set F
Output: Partition solution P

1. **foreach** $fixed\ node\text{-}FPGA\ pair\ (v_i, \hat{v}_i) \in F$ **do**
2. $NeighborNodes \leftarrow FindNeighbor(v_i)$
3. $Sort\_increasing(NeighborNodes)$
4. **foreach** $nNode \in NeighborNodes$ **do**
5. **if** $FPGA_{capacity}\hat{v}_i < given\ FPGA\ capacity$ **then**
6. $FPGA\_placement(nNode, \hat{v}_i)$
7. $UpdateCddt(FindNeighbor(nNode), Dis(\hat{v}_i, 1))$
8. **else**
9. $FPGA_{neighbor} \leftarrow FindNeighborFPGA(\hat{v}_i)$
10. $FPGA\_placement(nNode, FPGA_{neighbor})$
11. $UpdateCddt(FindNeighbor(nNode), Dis(FPGA_{neighbor}, 1))$
12. **while** $MovableNode(V) > 0$ **do**
13. $LeastCddtNode \leftarrow FindLeastCddt(MovableNode(V))$
14. **foreach** $mNode \in LeastCddtNode$ **do**
15. $UnplacedNeighbor \leftarrow FindNeighbor_{unplaced}(mNode)$
16. $Cddt_{UnplacedNeighbor} \leftarrow Union(UnplacedNeighbor)$
17. $Cddt_{Update}(mNode) \leftarrow Intersection(Cddt(mNode), Cddt_{UnplacedNeighbor})$
18. $PlacedNeighbor \leftarrow FindNeighbor_{placed}(mNode)$
19. $Cddt_{PlacedNeighbor} \leftarrow Union(PlacedNeighbor)$
20. $Cddt_{final}(mNode) \leftarrow Intersection(Cddt_{update}(mNode), Cddt_{PlacedNeighbor})$
21. **if** $bitcount(Cddt_{final}(mNode)) = 0$ **then**
22. $Cddt_{final}(mNode) \leftarrow Cddt_{update}(mNode)$
23. $FPGA\_placement(mNode, Cddt_{final}(mNode))$
24. $UpdateCddt(FindNeighbor(mNode), Dis(FindFPGA(mNode), 1))$

VI. Conclusion

In this work, we implemented a multi-level topology-driven partitioning on FPGA. To improve our performance, especially in terms of runtime, we made extensive use of bitset, a data structure from the C++ STL, and imposed some restrictions on candidate selection to reduce computational complexity. The results suggest that it is possible to further optimize the trade-off between external degrees and runtime.

VII. Learned Knowledge & Skills

小山翼**:** I learned how to collaborate on a coding project with my team members using the techniques of GitHub. I was also introduced to the concept of constructing my own namespace, which was new to me. I enjoyed the process of developing the complete algorithms for this project.

李濬安**:** During this project, we had the opportunity to collaborate frequently and I was able to learn some coding style and FPGA algorithms. The process of creating the entire algorithms for this work was satisfying and interesting to me.

VIII. Reference

[1] Dan Zheng, Xinshi Zang, and Martin D.F.Wong, “TopoPart: a Multi-level Topology-Driven Partitioning Framework for Multi-FPGA Systems,” IEEE/ACM ICCAD, 2021.