# HLS Contest


| | | |
|---|---|---|
| Students | Castagneri Dario | 277967 |
| | Guerriero Augusto Maria | 278018 |
| | Ras Antonio | 270259 |


**Thought process** We decided to limit the solution space exploration by predetermining a low number of resources, mapping to each one a “color” (a “color” corresponds to an allocated Functional Unit, and a set of nodes mapped to it with the LEFT_EDGE algorithm), then trying to find the optimal FU for each “color”.

**Explore_resource** It tries to find one feasible MLAC scheduling with the lowest possible amount of resources per operation. Explore_resources potentially iterates through all the possible combinations of ‘number of allocated resources’ for each resource; we decided to consider the first returned solution as “good enough”, as multiple tests confirmed that it is often the best solution or really close in terms of cost anyway.

**Explore_colors** given a predetermined set of colors, generated using the left_edge algorithm, tries to find the optimal mapping between color and Functional Unit.

Pruning was made considering the mobility of each node:

- If a color has at least one node with mobility 0, the explore_colors function will not iterate over the available FUs for that color. As the mobility of each node was calculated initially using the fastest resource for each node, we have concluded that if a color has at least one node with 0 mobility we cannot try to optimize area and power at the cost of increasing the delay, as a node with 0 mobility would not have time to complete its computation in time.
- For each color we computed what we improperly called “slack” as the sum of the delay of the fastest FU available for the operation related to that color and the minimum value of mobility among nodes inside that color. When iterating over the available FUs for a given color, we avoid to assign to that color a FU whose delay is superior to the “slack” value, as also in this case, at least one node inside that color would not be able to complete its execution before its output is due.

**MLAC_adhoc** We decided to create a MLAC scheduling optimized for the iteration over colors instead of operations, as each color is mapped to a potentially different FU, so it can be considered as a different operation from the scheduling perspective.

**Mobility** It returns a list with the nodes of each value and their value of mobility, defined as ALAP(Ni) - ASAP(Ni), with $\lambda$, the latency constraint passed as parameter to the script, passed as parameter to the ASAP.