INTRODUCTION

Since during the development of the code we found two different algorithms and after comparing them using the same inputs we figured out that both fails in few particular situations, while the other one succeed in them, we decided to combine the two algorithms in order to reach the highest percentage of success.
In particular, we run the second algorithm only if the first one is not able to apply any modification to the original schedule in order to increase the computational time only when it is needed.
In both the algorithms we have used the following cost function to compare the new schedule with the previous one:

$$\frac{new\_area}{old\_area} * \frac{new\_power}{old\_power}$$

where for each functional unit the power is computed multiplying its attribute power with its attribute delay.

1st ALGORITHM

The idea of this algorithm is to first search for possible nodes whose functional unit could be substituted and change them only if we estimate to reduce the total cost function (considering the total area and total power). The new schedule is derived by running again the list scheduling minimum resources algorithm in order to find the best schedule in terms of resources giving the new node's characteristic (delay and functional unit used).
The code is structured in this way: after running the first sharing algorithm, we pass through each type of functional unit used in the schedule. Foreach color belonging to that type we estimate the possible substitutions (nodes whose functional unit can be changed with a new one) considering all the alternative functional unit that perform the same operation. For each alternative we compute the possible gain that could be reached in terms of power and area and the best one is chosen. If there is a valid alternative, a new list scheduling minimum resources algorithm is run and after it the new cost is compared with the previous one. If the new cost is lower the schedule is updated.

2nd ALGORITHM

The idea here is to look at each node in the schedule and try to find the best alternative functional unit that can be mapped to it and to other possible nodes that are not concurrent with him.
Foreach node we try to substitute its functional unit with the alternative ones that are feasible with the slack of the node. After this we update the start time and the slack of every node in the local subgraph rooted at the current node. Then we map the same alternative for all possible nodes in the schedule such that they can share the same hardware resource of the current one, always updating the start time and the slack of the subgraph. At the end we compute the cost of the obtained new schedule and we compare it with the previous one. If it is lower, we update the schedule otherwise we drop it.

RETURN FORMAT

The function returns a list composed of three sub-lists: the first one is the schedule where each element inside it is composed by the node id and its start time, the second one is composed by the couples node - functional unit, and the third one is the list containing the amount of functional units allocated after the optimization.