Python and other executables other than ABC

- error.cpp – Contains code for error calculation in the network
- graph.py – Contains python code to generate file type.txt and type_node.txt from a given .bench file
- nonlinear_regression_package.py – Contains python code for performing non linear regression on Q-matrix to obtain the coefficients
- ML_multifile_v2.py – Contains python code for training purpose
- ML_Tester_v2.py – Contains python code for testing purpose

Supplementary files generated during the process

- dummy – Contains the script to run in ABC
- dummy.dump – Contains the console output of ABC
- final_error.txt – Contains the output error of the network
- final_result.txt – Contains the Q-matrix obtained after training (regression still needs to be done)
- gate_error.txt – Contains the gate error for nodes in network based on chosen hamming distance
- generated_result.txt – Contains the maximum hamming distances provided to ABC
- nodes.txt – Contains the total number of nodes in network and their maximum permissible hamming distances
- reg_coeff.txt – Contains the coefficients obtained after performing non-liner regression on Q-matrix in file final_result.txt
- type.txt and type_nodes.txt – This files are used to generate network graph for error calculation

Changes made in ABC

- File abc.c
    - In abc_commandMap function added code to read generated_result.txt to add maximum hamming distances to an array named bitErrors.
    - Passed bitErrors to function Abc_NtkMap.
- File abcMap.c
    - In Abc_NtkMap function, passed bitErrors to function map_mapping
- File mapperCore.c
    - Added function Map_CalculateNumberOfNodes which calculates number of nodes and their maximum permissible bit error and writes them to file nodes.txt
    - Added function writeGateError which calculated gate error for each node and writes them to file gate_error.txt
    - In map_mapping function, passed bitErrors to function Map_MappingTruths
- File mapperTruth.c
    - In map_mappingTruths function, passed bitError for a node to function Map_TruthsCut
    - In map_truthsCut function, passed number of leaves of a cut and bioterror for a node to function Map_SuperTableLookupC
- File mapper_Table.c

- Modified function Map_SuperTableLookupC as per our requirements
- Added function Map_MatchSuperGate which matches the exact and approximate supergates for the cut

Steps to run Q-ALS

1. Compile ABC.
2. Training:
    a. In Networks directory, add all the circuits on which training needs to be done.
    b. In Networks directory, edit "networks.txt" to add full path of all circuits on which training needs to be done.
    example: benchmarks/ISCAS85/c17.bench
    c. Open console and "abc\" as present working directory run ML_multifile_v2.py file using following command "python3 ML_multifile_v2.py"
    d. Wait till the whole process is completed.
3. Non-linear regression:
    a. Open console and "abc\" as present working directory run nonlinear_regression_package.py file using following command "python3 nonlinear_regression_package.py"
4. Testing:
    a. In Networks directory, add the circuit which needs to be tested.
    b. For example, we want to test c17.bench
    c. Open console and "abc\" as present working directory run ML_Tester_v2.py file using following command "python3 ML_Tester_v2.py benchmarks/ISCAS85/c17.bench"