Ethan Chen
Wenchuan Weng
CS M213A
Fall 2011
HW #4

The code is implemented in 3 parts. The first is contained in sdfg.cpp/.h.
This code contains representation and parsing of the SDFG input. Nodes and edges are represented using structs. We store both of these structures in a list indexed by node ID (assigned based on line order) and edge ID. Nodes and edges contain pointers to their children and parent. This allows quick lookup and traversal of the SDFG structure.
The next part of the code is contained in scheduler.cpp/.h. This part of the code is responsible for determining schedulability and creating a schedule. This is done by creating a matrix of the nodes in the SDFG (excluding I/O) where the each row corresponds to a producer-consumer node pair, with coefficients equal to the amount of input/output they produce/consume. This lets us represent the SDFG as a system of equations. By reducing the matrix to reduced row echelon form, we can create ratios and schedule. This basically corresponds to solving the system of equations represented by $Ax = 0$. The result, x, tells us how many times that specific node must be executed in a single period of execution, while the length of the period is equal to the sum of all values in the result x. The scheduler then uses a simple list scheduler to produce a valid schedule, placing it in the Schedule struct, also defined in scheduler.h. Note that using integer division in the RREF procedure means that our implementation will fail to schedule certain graphs where integer division results in a fraction, as it is rounded to 0. The list scheduler determines schedulability at each time slice by checking whether the internal count of items in the edge FIFO buffers are sufficient, and pushing the corresponding number of output items into the internal edge FIFO item counts if the node is schedulable at time slice t.
The final part is the simulator which is contained in the simulator.cpp/.h files. The simulator is responsible for reading input.txt, simulating the SDFG, and writing the output created to output.txt. It performs this simulation using the same push/pull mechanism described in the scheduler to push and pull node computation results to/from the edge FIFO buffers. The FIFO buffers are finite sized for simplicity of implementation, although they should be of infinite size. The scheduler processes samples by running the static schedule set up by the scheduler. It will run until it reaches a point where a node becomes unable to pull an operand from it's incoming edge FIFO buffer. At each node of the schedule, there is a check to see whether a node outputs to the global output node, and whether it takes input from the global input node. Should it do so, it will write/retrieve data to the large 4096 entry circular buffers.

Currently, the greatest shortcoming is the limited scheduler facility, due to the integer division-based RREF algorithm it relies on.
Additionally, the SDFG representation assumes edges provide a single input and single output, so multi-output edges are not supported.