High Level Synthesis Scheduling with Evolutionary Programming

Shilpa Gundagi[1], Sangamesh[2]

[1]*Asst Professor, Department of Electronics & Communication, SMSMPITR*
Akluj-413118, Maharashtra, India
ssgundagi@gmail.com

[2]Department of Electronics & Communication, Dr Ambedkar Institute of Technology
Bangalore, Karnataka, India
sangameshvs88@gmail.com

Abstract: ***Time as a constraint finding the optimal solution of scheduling in High level synthesis using Evolutionary Programming. Because of pressure of designing high performance chip may be in terms of the cost or speed scheduling plays very important role, without proper scheduling it is nearly impossible to meet the desired objective with the current challenge and the complexity of the circuit there is a very important requirement of automated tool which could deliver the optimal solution. In this regard rather than applying the conventional method it is always better to apply the algorithms which are nature inspired. Hence genetic algorithms opted for this purpose.***

Keywords: ASAP, ALAP, Time constraints, ILP, Evolutionary Algorithms.

1. Introduction

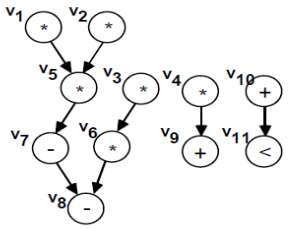
Scheduling is an inherent part of high-level synthesis and it is very necessary to satisfy the various objectives associated with chip designing. So whenever there is a chip design process, scheduling is one of the integral part of the design and optimal scheduling for that required design has to be carried out.

Time-constrained scheduling is important for designs targeted towards applications in a real-time system. For example, in many digital signal processing (DSP) systems, the sampling rate of the input data stream dictates the maximum time allowed for carrying out a DSP algorithm on the present data sample before the next sample arrives. Since the sampling rate is fixed, the main objective is to minimize the cost of the hard-ware. Given the control step length, the sampling rate can be expressed in terms of the number of control steps that are required for executing a DSP algorithm.

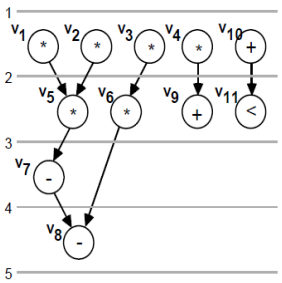
Given a data flow graph, the scheduling problem is to assign each operation in the DFG to a control step under certain constraints. Any assignment that is feasible under these constraints is a valid schedule. Out of possible valid schedules the goal is to find one that optimizes a given objective function. An immediate objective function for any scheduling algorithm is the length of the schedule or the latency. In addition, depending on the specific context in which the scheduler is used, other components are incorporated into the objective function. Objectives such as power and register usage have been incorporated into scheduling algorithms.

The control and data flow graphs depict the inherent parallelism in a design, based on which, each node could be assigned a range of control steps. Most of the scheduling algorithms require the earliest and the latest bounds that define the range of control steps for each node in the CDFG.

Two simple schemes that are widely used to determine these bounds are called the As Soon As Possible (ASAP) and the As Late as Possible (ALAP) algorithms. The ASAP algorithm begins with scheduling the initial nodes, i.e. nodes without any predecessors, in the first time step, and assigns the time steps in increasing order as it proceeds downwards. The ALAP algorithm is analogous to the ASAP scheme, except that the operations here are intentionally postponed to the latest possible control step. The algorithm begins at the bottom of the CDFG, i.e., with nodes that have no successors, and proceeds upwards to nodes that have no predecessors. ASAP is to schedule an operation in a clock cycle as soon as all its predecessor operations are scheduled. ALAP is to schedule an operation in a clock cycle, if all its successor operations are scheduled.



A DFG Example



ASAP Scheduled DFG

Related Work

Early in behavioral synthesis focused on constructive approaches as in force directed scheduling [4], [5]. These approaches being greedy are vulnerable to local minima. Transformational approaches on the other hand, start with an initial schedule and apply transformations to improve the initial solution. However, the quality of the solutions largely depends on the transformations used and the heuristics used to select between applicable transformations [3].

Mathematical approaches formulate the synthesis problem using Integer Linear Programming approach or some other mathematical optimization tool such as game theory. A technique for simultaneously scheduling and binding a DFG using Integer Linear Programming (ILP) is described in [7]. The precedence and time constraints are built into the ILP formulation. The cost function is evaluated based on signal statistics of the inputs obtained through a onetime simulation. The technique suffers from the drawback that ILP methods do not scale well for large circuits and may have to be combined with some heuristics. A game-theoretic approach for power optimization of a scheduled DFG is described in [8]. The functional units are modeled as bidders for the operations in the DFG with Power Consumption as the cost. The algorithm does not scale well for larger number of functional units since the complexity increases exponentially as the number of players (bidders).

2. Integer Linear Programming (Ilp) Formulation of Scheduling as a Constraint Optimization

In integer linear programming ILP is used to formulate the feasible scheduling problem. To illustrate the ILP formulations for scheduling, the data flow graph in Fig. is used in the following. The start time of every operation is bound by the result of ASAP and ALAP scheduling. Let *Nm*, *Na*, *Ns* and *Nc* be the number of multipliers, adders, subtractions and comparators, respectively.

The problem of the scheduling transforms as problem of constraint optimization where there is an objective function which will take care of the number of required resources along with the types of the resources, so that the cost of the solution could be minimum. Along with that depends upon three different types of constraints.

- Given by the ASAP & ALAP.
- Given by the required resources in a cycle.
- To satisfy successor and predecessor constraint.

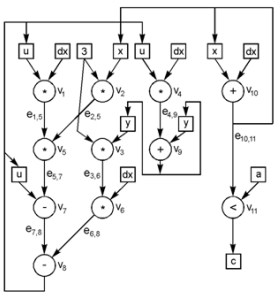


Figure. DFG for ILP Algorithm

Minimize:

$Cm*Nm+Ca*Na+Cs*Ns+Cc*Nc$

Subject to: All operations must start only once i.e. constraints from ASAP & ALAP.

$$X_{1,1}=1,$$
$$X_{2,1}=1,$$
$$X_{3,1}+X_{3,2}=1,$$
$$X_{4,1}+X_{4,2}+X_{4,3}=1,$$
$$X_{5,2}=1,$$
$$X_{6,2}+X_{6,3}=1,$$
$$X_{7,3}=1,$$
$$X_{8,4}=1,$$
$$X_{9,2}+X_{9,3}+X_{9,4}=1,$$
$$X_{10,1}+X_{10,2}+X_{10,3}=1,$$
$$X_{11,2}+X_{11,3}+X_{11,4}=1.$$

Then by resource constraints i.e. constraints with respect to requirement of the resources in any cycle must be less than or equal to the resource number in the cost function,

$$X_{1,1}+X_{2,1}+X_{3,1}+X_{4,1}\leq Nm,$$
$$X_{3,2}+X_{4,2}+X_{5,2}+X_{6,2}\leq Nm,$$
$$X_{4,3}+X_{6,3}\leq Nm,$$
$$X_{7,3}\leq Ns,$$
$$X_{8,4}\leq Ns,$$
$$X_{10,1}\leq Na,$$
$$X_{9,2}+X_{10,2}\leq Na,$$
$$X_{9,3}+X_{10,3}\leq Na,$$
$$X_{9,4}\leq Na,$$
$$X_{11,2}\leq Nc,$$
$$X_{11,3}\leq Nc,$$
$$X_{11,4}\leq Nc.$$

Finally, data dependence relations i.e. this will have predecessors and successors constraints in any case there must be the minimum difference of one cycle between the execution of the successor and predecessor modules, represented by data flow graph, are,

$$X_{3,1}+2X_{3,2}-2X_{6,2}-3X_{6,3}\leq -1,$$
$$X_{4,1}+2X_{4,2}+3X_{4,3}-2X_{9,2}-3X_{9,3}-4X_{9,4}\leq -1,$$
$$X_{10,1}+2X_{10,2}+3X_{10,3}-2X_{11,2}-3X_{11,3}-4X_{11,4}\leq -1,$$

Any solution which doesn't satisfy the constraints is called unfeasible solution; any solution which satisfies the constraints is called feasible solution.

All possible feasible solution create one space i.e. called feasible space, any solution from the feasible space which satisfy the objective in optimal manner that is going to consider as final solution.

3. Evolutionary Programming

Evolutionary Programming is one of the most important research areas which uses ideas and gets inspiration from natural evolution and adaptation. The main stream of algorithm for evolutionary Programming is Genetic Algorithms (GA's). In nature, evolution is mostly determined by natural selection or different individuals competing for resources in the environment. Those individuals that are better are more likely to survive and propagate their genetic material. The encoding for genetic information (genome) is done in a way that admits asexual reproduction which results in offspring that are genetically identical to the parent. Sexual reproduction allows some exchange and re-ordering of chromosomes, producing

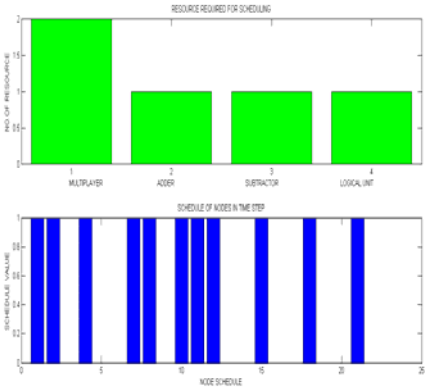
offspring that contain a combination of information from each parent. This is the recombination operation, which is often referred to as crossover because of the way strands of chromosomes cross over during the exchange. The diversity in the population is achieved by mutation. Evolutionary algorithms are ubiquitous nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems. In many cases the mathematical function, which describes the problem is not known and the values at certain parameters are obtained from simulations.

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved. Genetic algorithm is the part of evolutionary computation Genetic algorithms provides an alternative to traditional optimization techniques by using directed random searches to locate optimal solutions in complex landscapes.

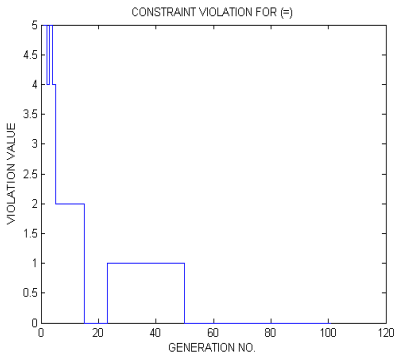
3.1 Working Process

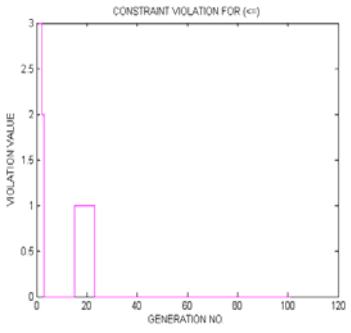
1) Initially one random population defined (population contains number of solutions).
2) Depends upon the objective the fitness function defined, it means A mathematical function which define the quality of solution on quantity manner. This will help to make a comparative decision among the solution who are good and who are bad.
3) From the population the two parents (solutions) randomly selected, genetic operator called Crossover applied; in cross over process we fuse the genetic characteristics of parents, so objective of the cross over is to produce offspring solution from the parent solution.
4) To maintain the Diversity another operator called Mutation applied, as in the nature each and every entity having some kind of the individuality or uniqueness, this uniqueness come in random manner same way, in the GA's mutation provide some kind of arbitrary changing in the offspring solution to maintain diversity.
5) The process will continue until the size of the solution is not equal to the parent population size.
6) Fitness of parents and off springs can be defined using the fitness function.
7) To create the next generation selection operator applied, selection means that will be the part of next generation among the parent and offspring population.
8) The best way to define selection is tournament selection, in this for each solution number of opponents pick up randomly, comparison made with respect to fitness value higher fitness score 1, lower fitness score 0, in the result each solution has one score value sort these score value from minimum to maximum, right half score value is taken and the corresponding solution will be the part of next generation.
9) The whole process will continue until terminating criteria will not satisfy.
10) Termination criteria: It can define in terms of iteration or generation number.
11) Self termination: In this case solution itself determine the time of termination this facility can achieve, if there is no continuous improvement in the fitness value of the best solution for N continuous generation.
12) Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (offspring) are selected according to their fitness; the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.



4. Result





5. Implementation Details

ASAP is to schedule an operation in a clock cycle as soon as all its predecessor operations are scheduled. The ALAP assignment is performed as the latest possible control step ALAP is to schedule an operation in a clock cycle, if all its successor operations are scheduled.

6. Conclusion

Scheduling is very important step in high level synthesis, in order to get very good performance as with given time we use ASAP and ALAP algorithms these gives the start time and end time for scheduling the given DFG. By using this we define a programming method called Integer Linear Programming (ILP) with respect to three types of constraints.

1. Given by the ASAP & ALAP,
2. Given by the required resources in a cycle.
3. To satisfy successor and predecessor constraint.

To handle this, nature inspired algorithms namely GA has been applied. Fundamental advantage with this approach is, it generates the global solution even with major complexities of the problem with the scheduling.

References

[1] http://www.obitko.com/tutorials/genetic-algorithms/index.php
[2] Abraham, A., Evolutionary Computation, In: Handbook for Measurement, Systems Design, Peter Sydenham and Richard Thorn (Eds.), John Wiley and Sons Ltd., London, ISBN 0-470-02143-8, pp. 920–931, 2005.
[3] Vyas Krishnan and Srinivas Katkoori, "A Genetic Algorithm for the Design Space Exploration of Datapaths during High-Level Synthesis", IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, June 2006.
[4] Sabih. H. Gerez, Algorithms for VLSI Design Automation, John Wileyand Sons, June 2000
[5] K.Oyarna. And I<.Clhinori, "An LSI fiinct,ion syiit, hesis system with a decision supporting system for its specification ," *DA Syniposiunz '94,* pp.31- 36, 1994. (in Japanese)
[6] X. Tang, T. Jiang, A. Jones and P. Banerjee, "Behavioral Synthesis of Data Dominated Circuits for Minimal Energy Implementation" in Proceedings of VLSI 2005
[7] Ashok. K. Murugavel and Nagarajan Ranganathan, "A Game Theoretic Approach For Power Optimization