A Low Power Scheduler Using Game Theory

N. Ranganathan and Ashok K. Murugavel
Dept. of Computer Science and Engineering
Nanomaterials and Nanomanufacturing Research Center
University of South Florida
Tampa, Florida 33620

ABSTRACT
In this paper, we describe a new methodology based on game theory for minimizing the average power of a circuit during scheduling in behavioral synthesis. The problem of scheduling in data-path synthesis is formulated as an auction based non-cooperative finite game, for which solutions are developed based on the Nash equilibrium function. Each operation in the data-path is modeled as a player bidding for executing an operation in the given control cycle, with the estimated power consumption as the bid. Also, a combined scheduling and binding algorithm is developed using a similar approach in which the two tasks are modeled together such that the Nash equilibrium function needs to be applied only once to accomplish both the scheduling and binding tasks together. The combined algorithm yields further power reduction due to additional savings during binding. The proposed algorithms yield better power reduction than ILP-based methods with comparable run times and no increase in area overhead.

Categories and Subject Descriptors
B.5.1 [Design]: Data-path design

General Terms
Algorithms, Low Power Design, High-level Synthesis, Game Theory, Auction Theory

1. INTRODUCTION
The advent of portable and wireless computing has increased the need for low power high performance compute intensive VLSI circuits. Increased power consumption, reduces the battery life in portable systems as well as the reliability and increases the packaging costs [12]. The various steps of behavioral synthesis are: scheduling, allocation and binding [12]. Scheduling is the process of determining when a resource is to be executed, allocation is the process of determining the number of instances of a required resource, and binding is the process of attaching a resource to an operation that needs to be performed [13]. A detailed treatment on low power behavioral synthesis can be found in [2]. Numerous algorithms have been proposed in the literature for low power scheduling. In [15], a force directed low power scheduling algorithm based on heuristics is described. Minimizing the number of times the input operands to a functional unit change reduces the power consumed by the functional units, which is explored in [7]. In [9], a module assignment algorithm for pipelined datapath’s is formulated as a max-cost multi-commodity flow problem. A combined register and module assignment algorithm for low power data-path synthesis is presented in [18].

Power reduction through combined scheduling and binding has been investigated in the following works. A scheduling and binding algorithm that minimizes switching activity with simulated annealing is investigated in [1]. In [2], an iterative mechanism based on an initial solution with efficient pruning techniques is used for low power synthesis. A novel approach involving the use of constrained logic programming to minimize the switched capacitance, during scheduling and binding is described in [8]. A heuristics based algorithm under a resource constraint can be found in [13]. In [5], a stepwise approximation algorithm for combined scheduling and binding is described. In [3], a scheduling and binding algorithm, that reduces glitches through clock gating is investigated. Most recently, an integer linear programming (ILP) method for scheduling and a linear programming (LP) method for binding are given in [19].

In this work, we model the problems of low power scheduling and binding as auction based game theoretic problems and solve them using the Nash equilibrium function. The scheduling and binding algorithms achieve power reduction through the use of neighborhood operations, path balancing, functional unit sharing to reduce switched capacitance and glitch reduction. The proposed algorithms do not increase the area overhead, since there is no addition of modules or multiplexors. The application of economic models and game theory as well as the Nash equilibrium for power optimization for the scheduling task in high-level synthesis is being attempted for the first time in this work.

2. GAME THEORY
Game theory is a tool for analyzing the interaction of decision makers with conflicting objectives. Economists have used game theory as a tool to understand the action of economic agents. The basic building blocks of game theory are based on theories proposed by von Neumann in 1928 [10] and Nash in 1950 [11]. A game, in which the rules can be previously stated or agreed upon by the players for use in deducing common strategies, is called a cooperative game. A game in which such agreements cannot be made is called a non-cooperative game [11]. Non-cooperative games are played with fully rational players who know the complete details
of the game, including each others preferences and outcomes. One of the first works in applying game theory to VLSI CAD problems was attempted in [17], a 2-person zero sum game theoretic formulation has been used to solve the system-level power management policy problem. A game theoretic solution to the problem of binding is described in [4].

The salient features of game theory that aid in the formulation of the behavioral synthesis problem are: (i) Rationality: Each player is always selfishly trying to optimize its gain. (ii) Coalition: Coalition formation is significant when a subset of players have the same agenda in terms of strategies for optimization can pool in their resources during resource allocation. (iii) Competition: In game theory, multiple decision makers who control a specified set of system variables and seek to optimize their conflicting objectives. (iv) Equilibrium: A solution is in an equilibrium state when all the players’ objectives have been optimized with respect to one another.

The first step is to transform the behavioral synthesis problem into a performance model considering power, delay and area as resources. The second step is to transform the performance model into an economic optimization model based on auction theory, with a set of buyers and a seller, all selfishly trying to optimize their individual resources. The goal is to achieve a stable solution which is good for everyone including the seller and the buyers. The Nash equilibrium proposed in [11], is an elegant solution to achieve the stable point. Auction theory is the science of studying the selling of objects among different buyers, such that the seller and the buyers optimize their gain [6]. The cost that each buyer is willing to pay is called a bid. Among the different types of models available in auction theory, the first-bid sealed auction model is most suitable for modeling power optimization in behavioral synthesis. In the first bid sealed auction, the buyer places sealed bids to the seller that cannot be changed and the seller decides, to which buyer the seller is willing to sell his item based on the bids from the potential buyers.

A finite game can be generalized to consist of n players who can choose from a set of strategies S_i, where i = 1, ..., n, and a set of payoff functions p_i where, i = 1, ..., n : S_1 x ... x S_n → R, where, R is the set of all real numbers and a payoff value is assigned to each pair of strategies chosen by the players. The rationality or the equilibrium point is a set of strategies that maximizes the payoff of the player assuming that all other players strategies are fixed. The game is played until each players strategy is optimal with respect to the strategies of others. The following example is useful in understanding the concepts.

\[
\begin{array}{c|c|c|c}
& P1 & P2 \\
\hline
1 & 3 & -2 \\
2 & 1 & 0 \\
3 & 1 & -2 \\
\end{array}
\]

The game consists of two players, P1 and P2, and the matrix given above. The entries in the matrix are the possible outcomes of the game, corresponding to the selections made by the two players. The various possible alternatives for player P1 are the rows of the matrix and for player P2 are the columns of the matrix. The alternatives are the strategies that each of the players can choose from. In this game, for P2, column 1 is the best since at most he loses 1 and for P1, rows 2 and 3 are the best since at most he loses 2. These are the secure strategies of the two players. If P2 plays the game first then his strategy will be to choose column 1, and the unique strategy of P1 is row 3, with an outcome gain of 1 by P1. But, if P1 plays first, he can choose either row 2 or row 3. If P1 chooses row 2, then the best strategy for P2 is column 1, but if P1 chooses row 3, then the best strategy for P2 will be column 4. In this game, we could assume that the players do not make their decisions independently, and there is a predetermined ordering for the players. If P1 decides first and passes the choice of his row to P2 then P2 has an advantage over P1. If the best choice for P1 (i*), and the optimal response of P2 (j*), is given as:

\[
\alpha_{i^* j^*} = \max_j \alpha_{i j} = \hat{v} = \min_i \{\min_j \alpha_{ij}\}
\]

where, the min and max operations are specified in the order in which they are to be performed. The outcome of the game when P1 follows P2 is a gain of 1 by player P1. When P2 follows P1, the outcome of the game is a gain of 2 by P2. If the two players choose independent of each other which is normally the case, player P2 gains 2, which is the equilibrium solution for this game. Determining this equilibrium solution is complex for games with multiple players. Hence, we apply the rational Nash equilibrium to find such a solution.

Let S_i be the set of various possible strategies of player i, where, i = 1, ..., N, and p_i be the payoff for player i. Nash equilibrium can be stated as: a set of strategies (n_1 \in S_1, ..., (n_N \in S_N) such that p_{[n_1, ..., n_N]} \geq p_{[n_1, ..., n_i', n_N]} for all i, n_i \in S_i. In words, this translates to, a player’s payoff does not increase if any of the players unilaterally deviate from the Nash equilibrium strategy. The Nash equilibrium NE, defines the payoff function for all the players in the game. The result that we are interested in, is the global payoff \(\bar{p}\) of all the players in the game, or the average payoff of each player due to the combined strategies, \(NE = \{n_1, ..., n_N\}\). Mathematically, this can be given as:

\[
\bar{p} = \sum_{i=1}^{N} p_{[n_1, ..., n_N]} \quad (2)
\]

The Nash equilibrium for an N-player finite game is a N-tuple set of strategies \(\{n_1, n_2, ..., n_N\}\), given by N inequalities such that, no single player can gain by individually changing his strategy. The inputs needed to calculate the Nash equilibrium are: (i) the strategies available for each player, (ii) the number of players in the game, and (iii) the number of alternatives \(a_i\) that are available for each player i. The steps for calculating the Nash equilibrium are: (i) determine all possible outcomes for the given game were the number of possible outcomes for the given game is \((\sum_{i=1}^{N} a_i)^N\), where \(a_i\) denotes the number of alternatives for player i, (ii) determine the inequalities for each player i such that his departure from the Nash equilibrium will lead to no increase in his gain which is stated formally as (\(p_i^{\star} \leq p_{[n_1, ..., n_i', n_N]} \geq p_{[n_1, ..., n_i, n_N]}\)), and (iii) the outcome that satisfies all the inequalities defined in step (ii) forms the Nash equilibrium solution of the game.

An elegant proof for the existence of equilibrium points for a finite game is given in [11]. An interesting fact about the Nash equilibrium is that, there is a guaranteed existence of a solution if we allow \(n_i\) to be of mixed strategies [11].

3. SCHEDULING

We model the scheduling problem as an auction based first-bid sealed auction and then describe a game theoretic solution. The auctioning of items by a seller through bidding can be extended to the auctioning of operations in the DFG, \(G_f\), among the modules of the architecture, \(G_a\), as, with an operation being bought by only one module and the sale entity being power consumption. In the presence of multiple operations and modules, an equilibrium point needs to be achieved. An auction consists of a set of available resources \(M\), and a set of interested buyers \(O\), where \(M\) is not equal to \(O\) and a buyer can only buy a single item at a time. The buyer \((b \in O)\) may also have some preference for some resources over the others, and this can be specified in terms of a cost function.
The scheduling of the feasible resources under a given resource set of all available resources at the time instant and where, is power consumed by module for operation at control step is mathematically given as,
\[
CM\text{D}_{\text{arch}}[m, o] = D_{\text{perwi}} + D_{\text{comi}} + \sum_{k=0}^{\omega} (D_{\text{perwik}} + D_{\text{comik}})
\] (6)

Formally, we can look at the preference mapping as: A buyer has a preference over resource set of all available resources at the time instant and where, is power consumed by module for operation at control step is mathematically given as,
\[
CM\text{P}_{\text{arch}}[m, o] = P_{\text{peryi}} + P_{\text{comi}} + \sum_{k=0}^{\omega} (P_{\text{perwik}} + P_{\text{comik}})
\] (5)

where, is the power consumed by module for operation at level and, is power consumed by communication between the modules with being the destination at level and the last term is the average power consumed by the given module from level 0. The delay cost function of a module for operation at control step is mathematically given as,
\[
CM\text{D}_{\text{arch}}[m, o] = D_{\text{perwi}} + D_{\text{comi}} + \sum_{k=0}^{\omega} (D_{\text{perwik}} + D_{\text{comik}})
\] (6)

Algorithm 2: Algorithm to calculate the Nash equilibrium

where, is the delay of module for executing operation at level and, is communication delay between the modules at level and the modules to which it was connected in the previous cycles and the last term is the average delay for the given module from level 0. In our formulation, the two optimizing parameters, namely delay and power are considered as a single optimizing parameter. This helps to simplify the formulation and solve problem as a simple resource allocation game.

Consider an architecture with two adders and a multiplier for the data-flow graph given in Figure 1. From the example given in Figure 1-(i), operations o1, o2 can be scheduled in control cycle 1. We will discuss the game for control cycle 1 to help in understanding the problem. Figure 1-(ii) gives the power cost matrix for cycle 1. Let, a, c, e be the cost associated with scheduling operation of to the modules add1, add2, mult respectively and b, d, f be the cost associated with scheduling operation of to the modules add1, add2, mult respectively. Simply, is the cost of scheduling operation to module . The bidding strategy, is the bids by the operation for the modules add1, add2, mult. The Nash Equilibrium is the set of bids for the modules, such that no operation can gain by deviating unilaterally (i.e. operations changing their bids).
The complete set of module and operation pairs that are possible for a given cycle is referred to as MO. In the examples, if adder 1 is combined with operation 1 and adder 2 with operation 2, the MO will be \{(1,1),(2,2)\}. For the set MO, the total cost \(CO = \text{co}(1,1) + \text{co}(2,2)\). Another such combination is adder 1 with operation 2 and adder 2 with operation 1, the corresponding MO is \{(1,2),(2,1)\}. Now, for each control step, the problem is to find the minimum of all \(CO\)'s (\(CO_{\text{MIN}}\)).

\[
CO_{\text{MIN}} = \min_{i,k=1,2,3,\ldots,x} \{\text{co}(m_i,1) + \text{co}(m_k,2)\}
\]  

This work is based on the assumption that Nash equilibrium strategies exist for the scheduling of operations such that the total power for the set MO is minimized. In other words the competition drives the operations to choose the modules such that power minimization is achieved. Normally, in a bid, the bidder has a profit but in the case of the modules, they don't have a profit of their own and the bid is a value from the cost matrix. The bid by module \(i\) for operation 1 will be given as, \(b(1,1) = \text{co}(1,1)\). The generalized cost matrix \(CM\) of size \(x\times y\) is given as,

\[
\begin{bmatrix}
\text{co}(1,1) & \ldots & \text{co}(1,y_j) \\
\ldots & \ldots & \ldots \\
\text{co}(x_j,1) & \ldots & \text{co}(x_j,y_j)
\end{bmatrix}
\]

The set of module-operation pairs \(A \in S\) is given as \(A = \{m_1, a_1, \ldots, m_{y_j}, o_{y_j}\}\), where \(m_1, \ldots, m_{y_j} \in M_i\) are the modules, \(a_1, \ldots, o_{y_j} \in O_{ij}\) are the operations, and \(S\) is the set of all possible module-operation pairs. \(A\) is a feasible set of module-operation pairs, there could be several such possible \("A"\) sets within \(S\). The set of module-operation pairs \(A \in S\) is given as \(A = \{m_1, a_1, \ldots, m_{y_j}, o_{y_j}\}\), where \(m_1, \ldots, m_{y_j} \in M_i\) are the modules, \(a_1, \ldots, o_{y_j} \in O_{ij}\) are the operations, and \(S\) is the set of all possible module-operation pairs. \(A\) is a feasible set of module-operation pairs, there could be several such possible \("A"\) sets within \(S\). The module set corresponding to \(A\) is, \(MS_A = \{m_1, \ldots, m_{A}(y_j)\}\). The Nash equilibrium solution \(A^*\), represents the set of module-operation pairs satisfying the Nash equilibrium inequality. The total power (CO) due to all the modules in a given set \(A\) is,

\[
CO(A) = \sum_{\alpha=1}^{y_j} \text{co}(m_{A}(\alpha), \alpha)
\]  

The proposed algorithm is aimed at minimizing the above equation. Thus, the total power \(CO(A^*)\) corresponding to the power optimal set \(A^*\) is,

\[
CO(A^*) = CO_{\text{MIN}} = \min_{A \in S} CO(A)
\]  

Notations and Definitions

- \(G_f\) directed data-flow graph
- \(G_s\) scheduled data-flow graph
- \(O_{ij}(i)\) operations scheduled during control step \(i\)
- \(A\) architecture
- \(x_j\) number of modules of type \(j\) in the architecture
- \(y_j\) number of operations of type \(j\) in the architecture
- \(M_{ij}\) set of all modules of type \(j\) in control step \(i\)
- \(O_{ij}\) set of all operations of type \(j\) in control step \(i\)
- \(NE_{arch}(i)\) Nash equilibrium solution at control step \(i\)
- \(CG\) current data-flow graph
- \(CO\) operations in the top level of the levelized DFG
- \(co(m,o)\) power for executing operation \(o\) on module \(m\)
- \(CM_{arch}\) power cost matrix, \(CM_{arch} = [\text{co}(m,o)]\)
- \(CM_{arch}\) delay cost matrix, \(CM_{arch} = [\text{co}(m,o)]\)
- \(S\) set of all module-operation pairs available
- \(A\) \(A \in S\) is any feasible set of module-operation pairs
- \(MS_A\) module set of \(A\), \(MS_A = \{m\}\), where \(m = 1, \ldots, y_j\)
- \(CO\) total cost for a specific \(A \in S\)
- \(CO_{\text{MIN}}\) the minimum total cost of any \(A \in S\)

The inputs to the scheduling algorithm are the data-flow graph \(G_f\) and the architecture \(G_s\), and the output from the algorithm is the scheduled data-flow graph \(G_s\). A breadth first search of the data-flow graph is performed and the input nodes to the DFG form the nodes in the top level of the DFG. For each set of compatible operations in the top level nodes of the DFG, the payoff matrices \(CM_{arch}\) are compiled Nash equilibrium solution \(NE\) for the game is determined. The Nash equilibrium solution gives, the power optimal schedule for the given DFG at each control step. The payoff matrix equations 5 and 6 are used in Algorithm 3. The nodes that have been scheduled are removed from the data-flow graph. For the current data-flow graph the process is repeated. If all the nodes in the DFG are scheduled, the S-DFG is given as the union of all the Nash equilibrium solutions.

4. SCHEDULING AND BINDING

The algorithm for scheduling and binding can be applied in sequence to obtain a power optimal behavioral synthesis solution. However, this involves applying the Nash equilibrium twice. In this section, we propose a combined scheduling and binding algorithm that applies the Nash equilibrium only once and results in better power reduction than applying the separate algorithms in sequence. The most important step in the combined algorithm is the determination of the payoff matrix, such that the power optimization is maximized. The first-bid sealed auction discussed in the previous sections is used here and hence is not explained again. The algorithm to calculate the payoff matrix used in Algorithm 4 is given in Algorithm 5.

The cost function aims at reducing the delay and the switching activity of the circuit. The power consumption for a module operation pair is,

\[
P_{m} = P_{\text{power}} + P_{\text{comm}} + \sum_{k=0}^{i-1} (P_{\text{power}_k} + P_{\text{comm}_k})
\]  

where, \(P_{\text{power}_k}\) is the power consumed by the module \(m\) for the operation \(o\) in cycle \(i\), similarly \(P_{\text{comm}}\) is the power consumed due to the communication between the modules with \(m\) as the destination at cycle \(i\), the final term is the total average power of the module from level 0 to level \(i - 1\). For binding we identify neighborhood operations that can be grouped for the same module to reduce switching activity. If the power due to the neighborhood operations is \(P_{\text{op}}\), then the power cost equation is given as,

\[
CMF[m,o] = P_{m} - P_{\text{op}}
\]
The combined scheduling and binding problem is formulated as a resource allocation game with two inter-related parameters to be optimized among the N-players. The notations in Sections 3 are also applicable here.

The inputs to the algorithm are the data-flow graph $G_f$ and the architecture $G_a$, and the output of the algorithm is the binding matrix $B$. A breadth first search of the data-flow graph is performed and the input nodes to the DFG form the nodes in the top level of the DFG. For each set of compatible operations in the top level nodes of the DFG, the payoff matrix $CM$ is calculated. The payoff matrix is given in Algorithm 5. The Nash equilibrium for the game is calculated using Algorithm 2 to determine the optimal binding of the operations to the modules and also obtain the power optimal schedule. The register binding task determines the optimal binding for the variables to the registers in each control cycle. Each functional unit that was active in the current control cycle is determined if it was active in the previous control cycle. Then, the input variables to the functional unit in the current cycle are bound to the same registers as in the previous cycle. If the functional unit was inactive in the previous control cycle, then variables are assigned to registers that are equidistant from the functional unit in-terms of the interconnect length.

Algorithm 5: Payoff matrix calculator for the combined algorithm

5. RESULTS AND CONCLUSIONS

The experimental results on benchmark circuits for scheduling and their resource constraints are given in Table 1. The power values for the circuits and the individual modules were obtained by simulation with 100,000 input vectors using the Synopsys RTL power estimation tool "Power Arc". The input vectors are random, 16-bit, two’s complement integers. A library for all the functional units and the registers has been developed based on the TSMC 0.25μm technology. This library is used by the algorithms to provide a realistic comparison of the various algorithms. The proposed scheduling algorithm ($S_{CTR}$) is compared with an integer linear programming (ILP) based methodology ($S_{ILP}$) proposed in [19] and a latency based scheduler ($S_L$) [14]. It should be noted that the latency based scheduler is not optimized for power. Column 2 in Table 1 specifies the resource constraint for the algorithms as the # of adders (A) and the # of multipliers (M). The results for the combined scheduling and binding algorithm are tabulated in Table 2. The ILP-based methodology was obtained by applying in sequence the ILP-based scheduling algorithm and the LP-based binding algorithm proposed in [19]. The Nash equilibrium is computed using Gambit: Software tools for game theory [16]. The algorithm has a % power savings of about 11.8 % and 41.3 % on an average over the ILP-based methodology and the algorithm with a random binding and latency based schedule respectively.

Since, the Nash equilibrium algorithm is NP-complete, the proposed game theoretic algorithms are of exponential time complexity. However, it should be noted from the Tables 1 and 2, that the run-times are small, which is due to the fact that the time complexity of the algorithm is a function of the number of players in the game and the set of strategies for each player of the game. In our algorithm, we have restricted the number of players in each game to be less than 4, and there are multiple games for each control step in the behavioral synthesis process. Hence, the run-times are much smaller than one would expect if all the games in a single level are combined into a single game. The time complexity of the Nash equilibrium for an $N$ player game with $S$ strategies for each player is given as $O(N^S NS)$ [16]. Assuming this to be the
Table 1: Experimental results for scheduling

<table>
<thead>
<tr>
<th>Benchmark circuits</th>
<th>Resource constraint, A &amp; M</th>
<th>Sehwa scheduler [14], (S_2, T)</th>
<th>ILP approach [19], (S_{ILP}, T)</th>
<th>Game-theoretic approach (S_{GT}, T)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency cycles</td>
<td>Power mW</td>
<td>Latency cycles</td>
<td>Power mW</td>
</tr>
<tr>
<td>Diff. eqn</td>
<td>2.2</td>
<td>4.0</td>
<td>2.2</td>
<td>4.0</td>
</tr>
<tr>
<td>FIR</td>
<td>3.3</td>
<td>9.9</td>
<td>9</td>
<td>8.8</td>
</tr>
<tr>
<td>HRR</td>
<td>3.2</td>
<td>8.8</td>
<td>18.3</td>
<td>18.6</td>
</tr>
<tr>
<td>Lattice</td>
<td>3.3</td>
<td>8.4</td>
<td>2.11</td>
<td>9.1</td>
</tr>
<tr>
<td>EWF</td>
<td>2.2</td>
<td>16</td>
<td>386.6</td>
<td>17</td>
</tr>
<tr>
<td>WAVE</td>
<td>2.2</td>
<td>25</td>
<td>352.1</td>
<td>26</td>
</tr>
<tr>
<td>NC filter</td>
<td>2.2</td>
<td>27</td>
<td>307.2</td>
<td>27</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Experimental results for combined scheduling and binding

<table>
<thead>
<tr>
<th>Benchmark circuits</th>
<th>Latency scheduler &amp; ILP-based binding (S_{L+B}, T)</th>
<th>ILP-based scheduling &amp; ILP-based binding (S_{ILP}, T)</th>
<th>Latency scheduler &amp; Game-theoretic binding (S_{GT}, T)</th>
<th>Game-theoretic scheduling &amp; binding (S_{GT}, T)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Latency cycles</td>
<td>Power mW</td>
<td>% sav w.r.t. (1)</td>
<td>Runtime s</td>
</tr>
<tr>
<td>Diff. eqn</td>
<td>2.2</td>
<td>4.0</td>
<td>22.9</td>
<td>13.9</td>
</tr>
<tr>
<td>FIR</td>
<td>9</td>
<td>9.9</td>
<td>61.1</td>
<td>35.5</td>
</tr>
<tr>
<td>HRR</td>
<td>8</td>
<td>8</td>
<td>13.3</td>
<td>20.0</td>
</tr>
<tr>
<td>Lattice</td>
<td>8</td>
<td>11.0</td>
<td>66.3</td>
<td>41.3</td>
</tr>
<tr>
<td>EWF</td>
<td>16</td>
<td>317.2</td>
<td>204.9</td>
<td>35.6</td>
</tr>
<tr>
<td>WAVE</td>
<td>25</td>
<td>291.9</td>
<td>26</td>
<td>201.2</td>
</tr>
<tr>
<td>NC filter</td>
<td>27</td>
<td>438.1</td>
<td>324.3</td>
<td>26.8</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td>30.5</td>
<td></td>
</tr>
</tbody>
</table>

6. REFERENCES


