

# Decomposition Methods for Circuit Sizing

SRC Project 1460.001

December 31, 2007

## 1 Introduction

Circuit sizing methods that use Geometric Programming (GP) are criticized for being too slow for practical applications. For large scale circuits linear programming approximations of the Geometric Program are used to convert the problem into a Linear Program. However, recent work in [3] shows that the scalability of GP based methods can be improved. In their paper, a million gate example was solved in less than an hour [3]. However, the need for faster implementations are always needed, to give designers the ability to consider multiple solutions under different scenarios.

In this report, we apply the decomposition techniques to the circuit sizing problem. Decomposition methods break a large problems into smaller ones, solving each smaller problem independently of the others. Because the smaller problems can be solved much faster than the larger ones, this can result in a significant improvement in runtime.

The runtime improvement in time relies on the fact that the time it takes to solve a problem of size  $x$  is more than linear. For example, suppose the solution time  $t(x)$  is related to the size of the problem by:

$$t(x) \approx x^\alpha$$

where  $\alpha$  is a constant. Now suppose the decomposition breaks the problem of size  $x$  into  $n$  smaller problems of size  $x/n$ . The relation for the solution time of the decomposed problem is then:

$$t_n(x) \approx nk\left(\frac{x}{n}\right)^\alpha$$

for some constant  $k > 1$ . The improvement in time can then be approximated as:

$$t(x) - t_n(x) \approx x^\alpha\left(1 - \frac{nk}{n^\alpha}\right)$$

Clearly, if  $\alpha \leq 1$  then there is no improvement. Thus rationale behind decomposition is that  $\alpha$  is large enough so that

$$1 - \frac{k}{n^{\alpha-1}} > 0$$

. This motivates for our application of decomposition to the circuit sizing problem.

The formulation of the circuit sizing problem we consider here is:

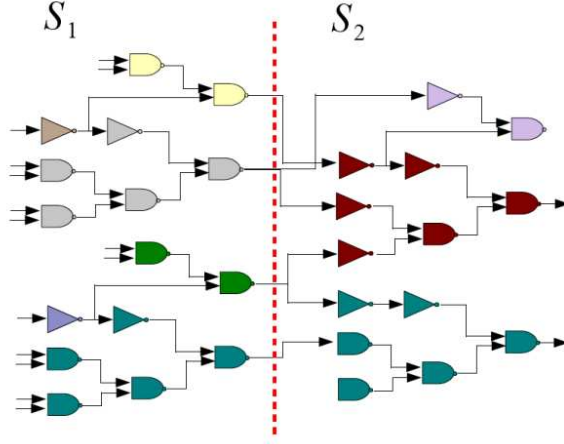


Figure 1: An example of a cut that splits the circuit into sets  $S_1$  and  $S_2$

$$\begin{aligned}
 & \text{minimize} && \text{Power}(x) \\
 & \text{subject to} && \text{Delay}(x) \leq T_{\max} \\
 & && 1 \leq x \leq x_{\max}
 \end{aligned} \tag{1}$$

where  $x$  denotes the gate scaling factors,  $T_{\max}$  is the maximum delay and  $x_{\max}$  is the maximum gate size. The function  $\text{Power}(x)$  is approximated using the linear expression:

$$\text{Power}(x) = \sum_{\forall i} p_{i,\text{nom}} x_i \tag{2}$$

where the coefficient  $p_{i,\text{nom}}$  describes the power usage for a nominal sized gate. The delay equations, however, are not as simple. To write  $\text{Delay}(x)$ , we employ a dynamic programming trick and create two auxiliary variables for each gate. The first variable  $D_i$ , is the time that is required for gate  $i$  to drive its fan-out gates:

$$D_i = \frac{\sum_{\forall j \in \text{FO}(i)} \alpha_j x_j}{\beta_i x_i} \tag{3}$$

Next, we create the variables  $T_i$ , which is the slowest arrival time from all of the fan-ins of  $i$ :

$$T_i = \max_{\forall j \in \text{FO}(i)} \{D_j + T_j\} \tag{4}$$

If  $i$  is a primary input of the circuit we let  $T_i = 0$ , and if  $i$  is a primary output of the circuit, we apply the constraint:

$$T_i \leq T_{\max} \tag{5}$$

## 2 Separating the Problem into Loosely-Coupled Subproblems

In problem (1) above, the objective is completely separable, which means that each of the variables can be considered independently of the next. This because it is linear in the variables and all the coefficients are positive. However, the delay constraint poses a difficult problem, because it couples different sets of variables (see (3)). This is because the delay of each gate depends on the ratio of the size of its fan-out gates to the current gate. However, this presents a methodology for creating loosely coupled subproblems. For example consider the cut in Fig. 1 of the graph that creates the partitions  $S_1$  and  $S_2$  with the property that there are no fan-ins of  $S_1$  in the set  $S_2$ . In other words, all the inputs of the set  $S_2$  come from  $S_1$ . If we fix the arrival times  $\{T_i\}$  for the gates in  $S_2$  that have inputs in  $S_1$ , and fix the total load of its outputs, then portion of the circuit in  $S_1$  can be optimized independently of the portion in  $S_2$ .

However this brings up the natural question: is the problem with the extra variables - the arrival times and the total loads - convex? Well the answer is yes, but there is a technicality. As with the original GP itself, the problem is not convex as it is written, but it is convex with a change of variables. Instead of using the actual gate scaling factors and the arrival times, the log of the variables are used:

$$x \rightarrow \log(x') \tag{6}$$

$$T \rightarrow \log(T') \tag{7}$$

This yields a convex formulation of the problem.

## 3 Clustering

An effective separation requires a way to cluster the gates in a circuit. In this report, we use the Maximum Fanout-Free Decomposition (MFFC).

The Maximum Fanout-Free Decomposition of a circuit has the property that each of the clusters only has one output, but the number inputs is unrestricted. This decomposition also has the property that the size of each cluster is maximal, thus each cluster is the largest set of gates that share the same output. Using this principle, the MFFC algorithm starts with a backwards topological sort, and runs through the circuit using a Breadth-First Search from a gate to its fan-ins:

*At iteration  $i$ , start with empty stack  $S$*

1. Take an gate  $n$  from the backwards topological sort that is not labeled, and label it as in cluster  $i$ . Set  $k = n$
2. Check each fan-in  $l$  of the gate  $k$  to see if all its fan-outs are labeled  $i$ . If so, label the gate  $i$  and push gate  $l$  onto the stack.
3. If the stack is empty, increment  $i$  and go to step (1). Otherwise, pop the next element from the stack and set  $k$  equal to the number of this gate.

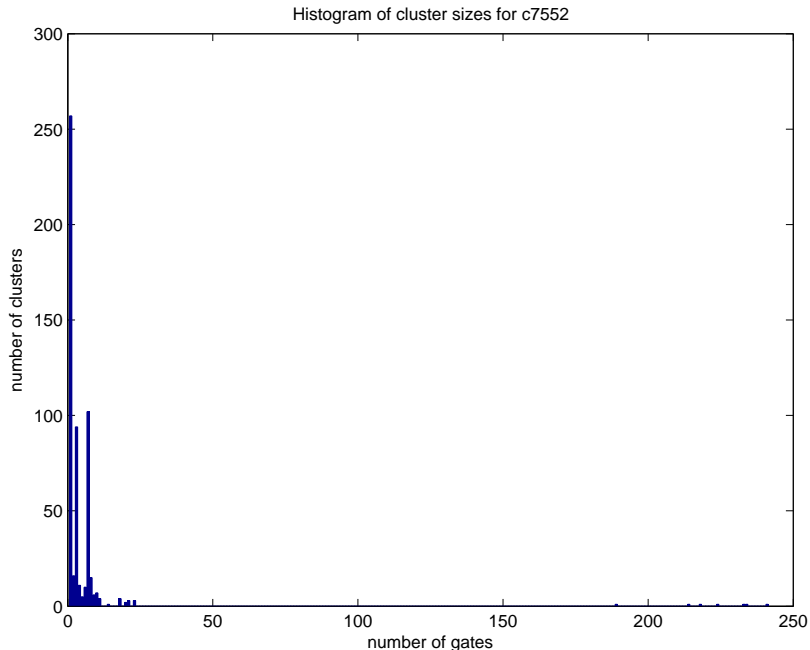


Figure 2: The histogram for the sizes of the clusters ISCAS’85 circuit c7552. Most of the clusters have small cardinality, but there are a few very large clusters.

The quality of the MFFC decomposition of a circuit depends on the specific structure of the circuit. However, in general, the sizes of the clusters vary highly, with many clusters having only one gate (see figure 2).

To reduce the occurrence of very large clusters, we may put a limit on the maximum size of an MFFC. This is very convenient, because large clusters may reduce the performance of our algorithm. In this report, we will limit the size of the clusters to maximize the performance. However, for the sake of simplicity, these size limited clusters will still be referred to as MFFCs.

## 4 A Primal-Dual Decomposition Method for circuit sizing

In this section, we describe our approach to primal-dual decomposition for the circuit sizing problem.

Let  $\{\mathcal{C}_i\}$  be a MFFC decomposition of the given circuit. We would like to decompose the problem (1) into a smaller problems of the form:

$$\begin{aligned}
 & \text{minimize} && \text{Power}_{\mathcal{C}_i}(x) \\
 & \text{subject to} && \text{Delay}_{\mathcal{C}_i}(x) \leq T_{\max} \\
 & && 1 \leq x_i \leq x_{\max} \quad \forall i \in \mathcal{C}_i
 \end{aligned} \tag{8}$$

Circuit	# of gates	# of MFFC's
c432	161	61
c499	203	60
c880	384	79
c1355	547	60
c1908	881	163
c2670	1194	168
c3540	1670	358
c5315	2308	387
c6288	2417	1459
c7552	3513	547

However, there is a problem. If we recall the discussion from section 2, the delay constraint  $\text{Delay}_{C_i}(x) \leq T_{\max}$  may couple gates from different clusters. This is because gates from other clusters set the arrival times for the inputs of the current cluster. This changes the amount of timing slack that can be used in this subproblem, so other clusters besides the current one need to be considered when solving problem (8).

The natural solution to this problem is to have a master problem that allocates times to a series of smaller circuit sizing problems of the form:

$$\begin{aligned}
& \text{minimize} && \text{Power}_{C_i}(x) \\
& \text{subject to} && \text{Delay}_{C_i}(x; t) \leq T_{\max} \\
& && 1 \leq x_i \leq x_{\max} \quad \forall i \in C_i
\end{aligned} \tag{9}$$

in the above, the vector  $t$  is a set of arrival times for the output of each cluster. Because each cluster  $C_i$  only has one output, we use the convention that  $t_i$  gives the arrival time for the output of cluster  $i$ . This yields a master level problem with as many variables as clusters.

There is another consideration to decompose the problem. This is because the timing is dependent on *both* the size of the driver gate *and* the total capacitive load of the gates that are being driven. This nuance is the last barrier that prevents the problem from being separated into loosely coupled subproblems.

To fix this problem, we can place a constraint on the total load for the output of each cluster. This has a simple form:

$$\sum_{j \in \text{FO}(C_i)} c_j x_j \leq C_{L,i}$$

which has the convex formulation (using a change of variables):

$$e^{-\log(C_{L,i})} \sum_{j \in \text{FO}(C_i)} c_j e^{x_j} \leq 1$$

However, this constraint is not yet separable, because the fan-out of  $C_i$  may contain different  $x_i$  from different clusters. To enforce this constraint while maintaining separability, we employ dual decomposition and use a Lagrange multiplier:

$$\lambda_i (e^{-\log(C_{L,i})} \sum_{j \in \text{FO}(C_i)} c_j e^{x_j} - 1)$$

where the  $\lambda_i$  are non-negative. This term is then added to the objective, creating the new objective

$$\text{Power}(x) + \sum_{\forall i} \lambda_i (e^{-\log(C_{L,i})} \sum_{j \in \text{FO}(\mathcal{C}_i)} c_j e^{x_j} - 1)$$

and the problem is solved by first finding  $t, C_L$  and  $x$  to minimize the objective, and then the  $\lambda$  are found to maximize the resulting minimum. This method is comes from the theory of duality, is equivalent to solving a dual problem. Splitting the variables into their respective sub-problems now gives us the loosely coupled subproblems:<sup>1</sup>

$$\begin{aligned} & \text{minimize} && \text{Power}_{\mathcal{C}_i}(x) + \sum_{j \in \text{CFO}} (\sum_{\forall k \in i_j} \lambda_{i_j(k)} e^{-\log(C_{L,i_j(k)})} c_j e^{x_j}) \\ & \text{subject to} && \text{Delay}_{\mathcal{C}_i}(x; t, C_L) \leq T_{\max} \\ & && 1 \leq x_i \leq x_{\max} \qquad \qquad \qquad \forall i \in \mathcal{C}_i \end{aligned} \tag{10}$$

If we let  $\psi_i(t, C_L, \lambda)$  represent the optimal value for the subproblem above, then the master level problem finds  $\lambda, t, C_L$  to solve:

$$\begin{aligned} & \text{maximize}_{\lambda} && \inf_{t, C_L} (\sum_{\forall i} \psi_i(t, C_L, \lambda) - \sum_{\forall k} \lambda_k) \\ & \text{subject to} && 0 \leq \lambda, C_L. \end{aligned} \tag{11}$$

To simplify notation, let

$$\Psi(\lambda) = \inf_{\forall i} (\sum_{\forall i} \psi_i(t, C_L, \lambda) - \sum_{\forall k} \lambda_k)$$

Then we can rewrite (11) as:

$$\begin{aligned} & \text{maximize}_{\lambda} && \Psi(\lambda) \\ & \text{subject to} && 0 \leq \lambda. \end{aligned} \tag{12}$$

## 5 Solving the Primal-Dual Decomposition

Solving the problem (11) requires the solution of three levels of problems:

1.  $\psi_i(t, C_L, \lambda)$
2.  $\Psi(\lambda) = \inf(\sum_{\forall i} \psi_i(t, C_L, \lambda) - \sum_{\forall k} \lambda_k)$
3. minimize  $\Psi(\lambda)$

Problem (1) is solved by using the general purpose solver Mosek [4]. Problem (2) is solved using the Analytic Center Cutting Plane method [1]. The gradients that are needed to solve (2) are collected from the solutions of (1) (see eqn (18)).

These optimization methods were selected for their stability and rate of convergence, making them well suited for testing the performance of the decomposition methods. In the future, however, these methods will be replaced with others that are more scalable.

---

<sup>1</sup>In the following equation, the set CFO consists of the gates that are driven by a different cluster, and  $i_j$  is the set of clusters that fan-in to the gate  $j$

## 6 Performance Analysis

The decomposition method that is submitted has a complexity of  $\mathcal{O}(n^{3.3})$ . However, there is a significant amount of improvements that can be made in this algorithm.

### 6.1 Models

In this subsection, we describe the models we use to estimate performance.

The time that is used to solve  $\psi_i(t, C_L, \lambda)$  is proportional to:

$$t_\psi(n) = t_{\min} + n^{\alpha_1}$$

where  $n_s$  is the number of gates in cluster  $i$ .

Let  $m$  be the size of the vectors  $t$  and  $C_L$ , and suppose that the size of each subproblem  $n_s$  is set to minimize the time per variable. Then, the time that is used to solve  $\Psi(\lambda)$  is a product of:

1.  $n_{sp} \cdot t_\psi(n_s)$
2.  $k_c m^{\alpha_c}$  for each centering operation
3.  $k_{nc} \log(2m) + c_{nc}$  centering operations

Finally, to solve the problem minimize  $\Psi(\lambda)$ , we need to solve  $\Psi(\lambda)$  approximately  $k_{nc} \log(m) + c_{nc}$  times. This is similar to the above.

In the attached implementation, the constants are approximately:

$$\begin{aligned} t_\psi(n_s) &= .31s \\ k_c &= 2e^{-8} \\ \alpha_c &= 2.8 \\ k_{nc} &= 141 \\ c_{nc} &= -282 \end{aligned}$$

The last relation we need for the performance analysis is to relate the size of the circuit to the number of clusters. However, this is by far the worst approximation, because the number of clusters is highly dependent on the structure of the circuit. As a rough approximation however, we fit the ISCAS '85 circuits using a linear relation. With an upper limit on the MFFC at 200, this puts the number of clusters at approximately 30% of the number of gates.

### 6.2 Estimated Performance

Using the models above, we can estimate how this algorithm will scale in proportion to the size of the circuit. Although the models are approximate, they give a good idea of how well the program will scale for large numbers of variables.

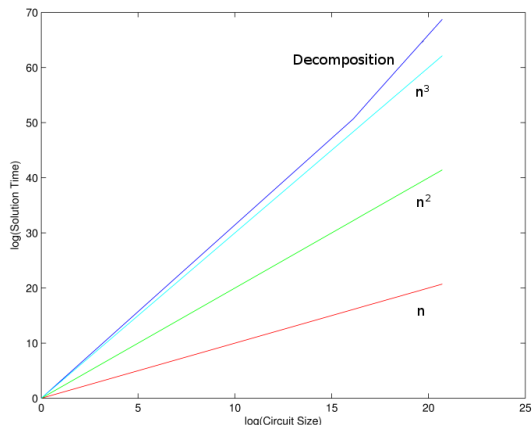


Figure 3: The performance for the decomposition algorithm is approximately  $n^{3.3}$

Figure 3 shows the performance of this method, which is estimated to grow as  $\mathcal{O}(n^{3.3})$ . This is acceptable for a general purpose algorithm, but is certainly not suitable for a large-scale implementation. This is similar to, but slightly better than the complexity for the general purpose solver (approx  $\mathcal{O}(n^{3.75})$ ).

The reason the complexity is high is due to the complexity of the analytic center cutting plane algorithm. In fact the complexity is roughly

$$n^{\alpha_c} (\log(n))^2$$

which means that if the complexity of the solver  $\alpha_c$  can be brought to close to 1, then we would have a very scalable algorithm.

Some ways to improve  $\alpha_c$  would be to use a gradient based descent methods, a conjugate gradient method or a quasi-newton method. These methods have low complexity and would bring  $\alpha_c$  between 1 and 2. Furthermore, routines that exploit sparsity can be used to improve the runtime of the underlying linear algebra.

## 7 Conclusion

Decomposition methods give an interesting way to break down the circuit sizing problem into smaller subproblems. It gives us a way to make a slight improvement of the performance of the solver over the general purpose solver. However, there is room for significant improvement. Specifically, methods that are better suited for large-scale optimization can be used to solve the subproblems, and the linear algebra routines can be improved to take advantage of sparsity. Further work will be done to improve the scalability of this method.

## 8 Appendix

### 8.1 Program Notes

The Decomposition Circuit Sizing program folder has the directories:

`src/` contains all the source code  
`mk/` contains the makefiles and compiled objects  
`bench/` contains the ISCAS '85 Benchmarks  
`test/` contains the compiled binary

This program requires the `cBLAS`, `cLAPACK` and `Mosek` libraries. Please set the locations in `mk/Makefile`.

To run the program, type:

```
./csize ../bench/cXXXX.bench mffc
```

where `XXXX` is the number of the benchmark circuit to be run. This will:

1. Find the minimum possible delay
2. Set the delay specification `t` be 105% of the minimum
3. Run the Decomposition Circuit Sizing algorithm with this delay specification
4. Solve the same problem using `Mosek`

As noted above, this algorithm is quite slow, so please be patient. The gate sizes for the decomposition method is stored in `mlcs_x`, and the `Mosek` solution is stored in `opt_x`. `mlcs.log` contains a log of what happened in the decomposition algorithm.

Alternatively, try:

```
./csize ../bench/cXXXX.bench qmffc
```

This will run the Decomposition Circuit Sizing algorithm with the delay specification set at 70% of the maximum delay.

### 8.2 Geometric Programming

The standard geometric program has the following convex form:

$$\begin{aligned} & \text{minimize} && \mathbf{lse}(A_0x + b_i) \\ & \text{subject to} && \mathbf{lse}(A_ix + b_i) \leq 0 \quad i = 1, \dots, m \end{aligned} \tag{13}$$

where  $\mathbf{lse}$  denotes the log-sum exp function:

$$\mathbf{lse}(x) = \log \left( \sum_{\forall i} e^{x_i} \right)$$

This can be rewritten using additional variables:

$$\begin{aligned} & \text{minimize} && \mathbf{lse}(y_0) \\ & \text{subject to} && \mathbf{lse}(y_i) \leq 0 \quad i = 1, \dots, m \\ & && y_i = A_i x + b_i \end{aligned} \quad (14)$$

The dual of this problem is (see [2]):

$$\begin{aligned} & \text{maximize} && b_0^T \nu_0 - \sum_{k=1}^{n_0} \nu_{0k} \log(\nu_{0k}) + \sum_{i=1}^m (b_i^T \nu_i - \sum_{k=1}^{n_i} \nu_{ik} \log(\nu_{ik}/1^T \nu_i)) \\ & \text{subject to} && \nu_i \geq 0, \quad i = 1, \dots, m \\ & && 1^T \nu_0 = 1 \\ & && \sum_{i=0}^m A_i^T \nu_i = 0. \end{aligned} \quad (15)$$

This gives us one dual variable for every monomial in the problem. To understand its connections with duality, we examine the Lagrangian of the problem (14):

$$L(x, y, \nu, \lambda) = \mathbf{lse}(y_0) + \nu_0^T (A_0 x + b_0 - y_0) + \sum (\lambda_i \mathbf{lse}(y_i) + \nu_i^T (A_i x + b_i - y_i)) \quad (16)$$

$$= \mathbf{lse}(y_0) - \nu_0^T y_0 + \nu_0^T (A_0 x + b_0) + \sum (\lambda_i \mathbf{lse}(y_i) - \nu_i^T y_i) + \sum \nu_i^T (A_i x + b_i) \quad (17)$$

Thus, the gradient of the lagrangian with respect to a coefficient  $b_i$  is its associated multiplier, e.g.

$$\nabla_{b_{i(j)}} L(x, y, \nu, \lambda) = \nu_{i(j)} \quad (18)$$

where  $\nu_{i(j)}$  denotes the  $j^{\text{th}}$  element of the vector  $\nu_i$ .

To derive the dual function, we observe that the conjugate of  $\mathbf{lse}(x)$  is:

$$\mathbf{lse}^*(\nu) = \begin{cases} \sum \nu_k \log \nu_k & \nu \geq 0, \quad 1^T \nu = 1 \\ \infty & \text{otherwise.} \end{cases}$$

we can now write the Lagrangian dual problem:

$$L(x, y, \nu, \lambda) = \inf_x \inf_y \mathbf{lse}(y_0) - \nu_0^T y_0 + \nu_0^T (A_0 x + b_0) + \sum (\lambda_i \mathbf{lse}(y_i) - \nu_i^T y_i) + \sum \nu_i^T (A_i x + b_i) \quad (19)$$

$$= \inf_x - \mathbf{ent}(\nu_0) + \nu_0^T (A_0 x + b_0) + \sum (-\lambda_i \mathbf{ent}(\nu_i/\lambda_i)) + \sum \nu_i^T (A_i x + b_i) \quad (20)$$

where  $\mathbf{ent}$  denotes the entropy function:

$$\mathbf{ent}(x) = \sum x_i \log(x_i).$$

This gives us the further condition that  $\nu_i^T A_i x = 0$ , which gives us (15).

## References

- [1] D. S. Atkinson and P. M. Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1):1–44, July 1995.
- [2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [3] S. Joshi and S. Boyd. An efficient method for large-scale gate sizing. Submitted for publication, available at <http://www.stanford.edu/boyd/gatesizing.html>.
- [4] MOSEK ApS. The MOSEK Optimization Tools Version 4.0. Available from <http://www.mosek.com>.