International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 9, September- 2014

Effect of Critical Path Minimization on Total Wire Length for Unequal Sized Recursive Min-Cut Placement

Kalimidi Yugandhar
Research scholar, Department of Electrical Engineering,
Sri Venkateswara University College of Engineering
Tirupati, Andhra Pradesh, India

G. V. Marutheswar
Professor, Department of Electrical Engineering
Sri Venkateswara University College of Engineering
Tirupati, Andhra Pradesh, India

***Abstract*—In this paper, the effect of critical path minimization on wire length is examined for a new method of min-cut partitioning. Traditional min-cut placers use equal sized partitioning for VLSI placement. They use either bisection or quadrisecton to divide the circuit. This paper gives an introduction to a new method of partitioning called unequal sized partitioning. In this method of unequal sized min-cut partitioning the circuit is recursively divided into unequal sized partitions. Firstly an introduction to unequal sized recursive partitioning is presented. Then the net weights on the critical path are varied for MCNC benchmark circuits for different partition ratios ranging from 0.1 to 0.9 and the placement results are examined. The results prove that each circuit has optimal wire length at different partition ratio for a different critical path weighting. Finally this paper suggests the need for unequal sized partitioning which improves the wire length significantly as compared to the conventional equal sized partitioning.**

Keywords— critical path, net weighting, min-cut algorithm, placement, unequal sized partitioning

I. INTRODUCTION

Min-cut partitioning is an important constructive placement algorithm which is used extensively for global placement of VLSI circuits. In this algorithm, the circuit is divided recursively into sub-circuits with minimum interconnections at each level of partitioning. The modules allotted to sub-circuits during min-cut partitioning entirely depend on the weights of modules and nets of the circuit. The weighting of constituent elements of the circuit can change the placement results considerably.

The designers choose the weights of elements according to the optimisation at hand. For example, during the global placement since the main objective of placement is wire length minimization, the longer nets are given higher weights. Since the placer give high preference to higher weighted elements first, those nets with higher net lengths are given higher net weights. For some of the circuits, meeting the timing constraints is much more important than minimizing any other placement objectives. The process of placement for such kind of circuits is to be done entirely on different basis and the placement process is totally driven by the timing constraints. This timing-driven placement can be done both at global and detailed placement stages.

This Timing-driven placement algorithm can be broadly classified into two groups namely *net-based* and *path-based*. The net-based approach gives higher preference to nets on the critical paths. Critical path usually refers to the path which is longer among all the paths of the circuit. A longer path causes the input signal to reach the output after a longer time. In other words, the delay on the critical path is higher which slows down the circuit performance or the time taken to perform a computation for the circuit increases. So the designers try to minimize the delay by reducing the length of the critical path. The length of the critical path can be decreased during the min-cut partitioning by providing higher weights to elements on the critical path. In other words, minimizing the wire length on critical path implicitly minimizes the critical path delay. *Net weighting*[*1*][*2*][*3*][*4*] and *net constraints* [*5*][*6*][*7*][*8*][*9*][*10*] are the basic techniques used for net-based optimizations. The path-based approach works on the paths [*11*][*12*][*13*][*14*] using a mathematical formulation.

The conventional min-cut placers such as Dragon [*15*], FengShui [*16*], NTUPlace2 [*17*] and Capo [*18*] use this min-cut algorithm. They divide the circuit into two sub-circuits or four sub-circuits [*19*] of equal size at each partitioning level.. All of this placement tools ignore the unequal sized partitioning and assume that the equal sized min-cut partitioning give an optimal partitioning at all times for all the circuits. This paper employs a new method of partitioning called unequal sized partitioning to solve VLSI placement problem [*20*]. Since the problem of VSLI placement problem is NP complete [*21*], it cannot be denied that unequal sized partitioning may provide better optimal solution. The objective of this work is to examine the effect of critical path minimization for unequal sized recursive partitioning.

This paper is organised as follows. In section 2, a new partitioning approach called unequal sized partitioning is introduced with illustrations. Different possible implementations of unequal sized partitioning are discussed in detail in this section. In section 3, the hypergraph implementation of circuit is discussed in detail in addition to the benchmark circuits, software and operating system used to carry out the experiment. In section 4, the experimental results

are reported which show a significant improvement of placement results with unequal sized partitioning. And finally the last section concludes the work emphasizing the necessity of unequal sized partitioning as an alternative approach to equal sized partitioning.

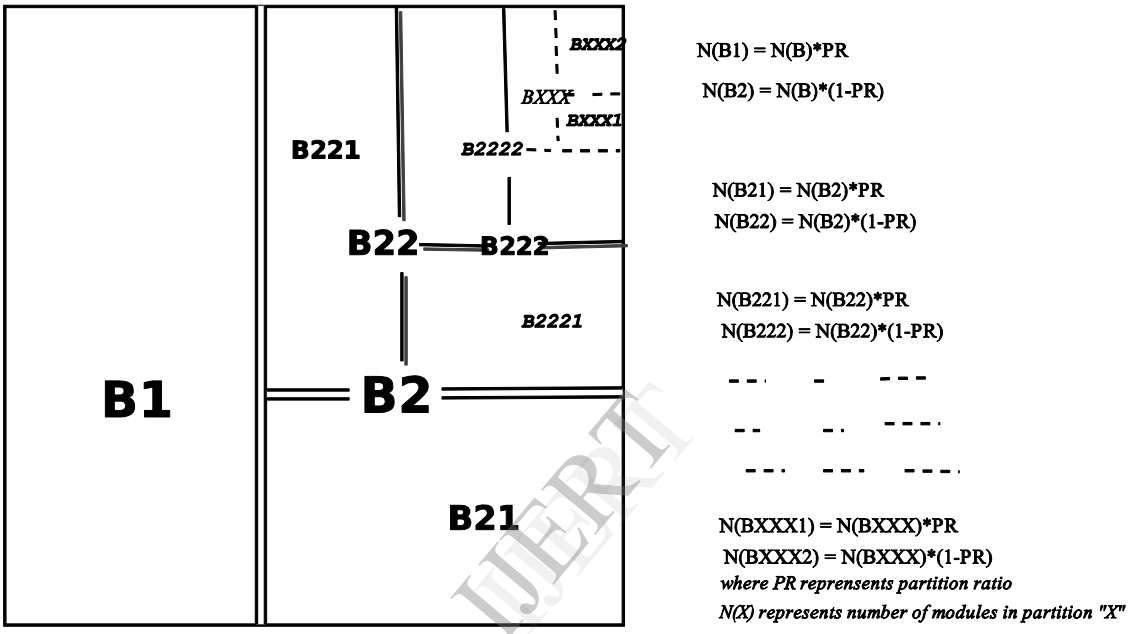

Fig. 1. Unequal sized recursive min-cut partitioning

location.

II. UNEQUAL SIZED RECURSIVE PARTITIONING

Traditional min-cut placers divide the given circuit recursively into equal sized partitions till each module is mapped to a particular location of the chip. The Fig. 2 shows a typical example of min-cut

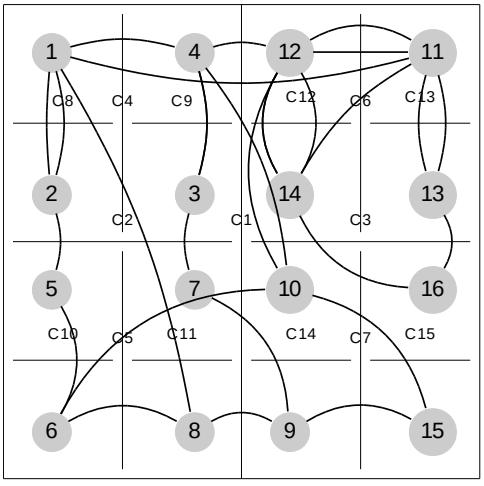

Fig. 2. Conventional equal sized min-cut partitioning

placement by equal sized partitioning. The circuit is partitioned with equal number of modules and equal areas with minimum net cut at each level of partitioning. The cutline C1 divides the modules into two sets of eight modules each with the number of interconnections between them as 6. This is the first level of partitioning. In the second of level of partitioning the horizontal cut-lines C2 and C3 divide the blocks obtained in the previous level into blocks of four modules each. The process of division is continued to these sub-blocks further till each module is mapped to unique

Contrary to the traditional method of equal sized partitioning, the modules are divided into unequal sized partitions and mapped to unequal sized chip areas as shown in Fig. 1 for unequal sized recursive partitioning. The number of modules allotted to the block B1 is shown as N(B)×partition ratio. If partition ratio is defined as 0.4, then the left smaller block B1 gets 0.4×N(B) , while the right bigger block gets 0.6×N(B). In the next partitioning level, the block B2 is considered for partitioning leaving its complementary block B1. The algorithm recursively apply the same partition ratio to this block B2 and divides into B21 and B22, where B21 gets 0.4×N(B2) modules and B22 gets 0.6×N(B2) modules. This process is repeated recursively till all the modules are allotted to unique locations of the chip.

The Fig. 3 represents a typical layout region with alternate vertical and horizontal cut-lines for unequal sized partitioning. Each of the cut-lines divides the partition area repeatedly into two partitions of areas of 40% and 60%. At each partitioning step, the total number of modules is divided into partitioning

sizes of 40% and 60% and the modules are assigned to the

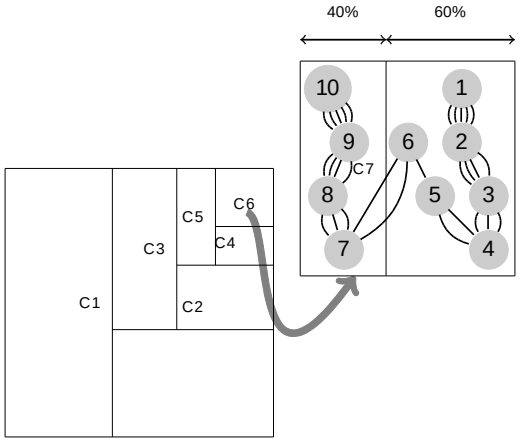

Fig. 3. A chip layout partitioned alternatively with vertical and horizontal cutlines with partition ratio of 0.4

respective partitioning areas. From this figure, it can be observed that we reached a stage where the circuit to be partitioned have total number of ten modules. The vertical cut line C7 cuts the partitioning area into two partitions of areas of 40% and 60% of total area. The modules are divided into two partitions of four modules and six modules each such that the cut size is two nets.

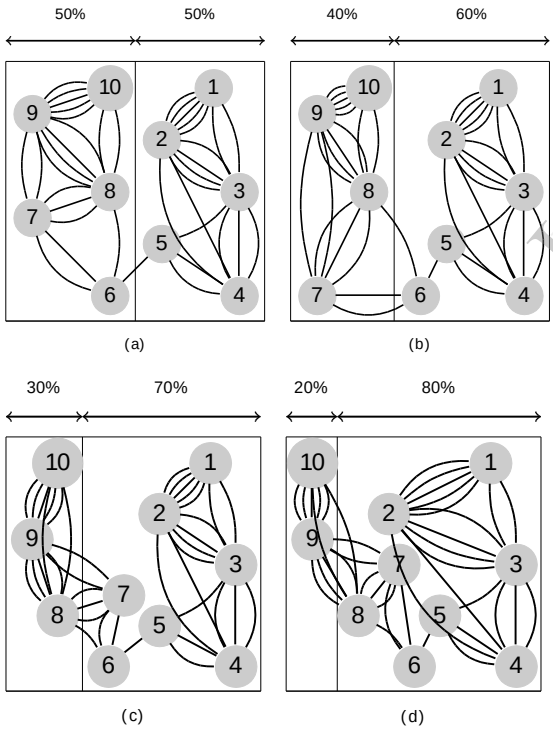

Fig. 4. A circuit with ten modules partitioned with different partition ratios of 0.5, 0.4, 0.3 and 0.2

The Fig. 4 represents different unequal partitioning strategies for a circuit having ten modules. The Fig. 4(a) shows conventional partitioning for VLSI cell placement where partition areas and sizes are equal sized. The Fig. 4(b) represents a typical sub-circuit of unequal partition sized placement. Here the sub-circuit area is divided into two partitions such that the left one has 40% of the total number of modules of the sub-circuit and the right one accommodates the remaining 60% of the modules. The cut line is placed at 0.4 times the length of x-axis of the sub-circuit from left most bottom corner. Here the cut size is two nets. And for a horizontal cut the cutline is placed at 0.4 times the length of the y-axis of the sub-circuit from the left most bottom corner. The Fig. 4(c),(d) represent 30-70 partition and 20-80 partition, and the process of placement is same as above with variation in partition ratios

III. Implementation Details

In VLSI CAD, the circuits are represented using hypergraphs. A hypergraph is represented by G = (V, E), where V is a set of circuit gates and E is a set of signal nets. A hyperedge e ∈ E connects any number of vertices in V. The Fig. **4** shows a circuit with seven logic gates and five edges and its hyper graph. The gates are represented as a set vertices V ={A,B,C,D,E,F} with each vertex v ∈ V having a size s(v) and the interconnection wires are represented as a set of hyper-edges E= {e1,e2,e3,e4,e5} with each hyper-edge e ∈ E having a weight w(e).

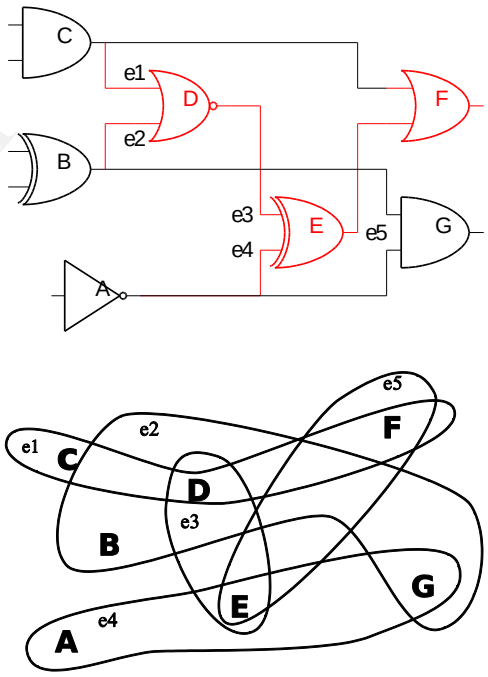

Fig. 5 . A simple circuit with seven logic gates and its hypergraph

The objective of the min-cut algorithm is to partition the given circuit into two sub-circuits such that the interconnections between the two partitions are minimum. This task of partitioning can be easily implemented using the hypergraph partitioning.

The hypergraph partitioning problem divides the set of vertices V into subsets V1 and V2 such the total weight of hyperedges cut is minimum. Also the total weights of vertices from V1 and V2 are maintained almost equally with an allowed imbalance.

Mathematically, the cost for Min-cut can be defined as:

$$\text{CutCost}(V1,V2)= \sum_{e \in E \text{ s.t. } e \cap V1 \neq \emptyset \ \Lambda \ e \cap V2 \neq \emptyset} w(e) \quad (1)$$

$$\gamma - \varepsilon \sum_{v \in V1} s(v) / \sum_{v \in V} s(v) \leq \gamma + \varepsilon \quad (2)$$

where the ratio parameter γ and tolerance parameter ε are used to specify the size constraint on V1

$$\sum_{v \in V2} s(v) = \sum_{v \in V} s(v) - \sum_{v \in V1} s(v) \quad (3)$$

represents the size constraint on V2

The Half-Perimeter Wire length (HPWL) net model is used in the algorithm wire length estimation. This method calculates the perimeter of the bounding box of the modules under consideration and approximates the wire length by the half-perimeter. HPWL is most effective for 2-pin and 3-pin nets and since most of the nets in any circuit are either 2 or 3 pin nets this method is widely adopted in the industry.

```
Procedure{Critical_path_minimization}
{Timingpaths}

   If
{Critical_path_minimization==False}
          Exit from the procedure
   Else
        For{Each TimingPath}
           For{Each net in the
TimingPath}
             Increase the net weight
geater than the value set by
optimization parameter}
           EndFor
        EndFor
   EndIf

EndProcedure
```

Fig. 6. Min-cut Algorithm for Critical path minimization

The partitioning process is done at different levels in a top-down fashion called multilevel partitioning. In this experiment, a multilevel hypergraph partitioning software package called Hmetis [*22*] is used which is fast and scalable. For this hmetis program to specify the imbalance between the partitions a command line argument called UBfactor is used. UBfactor is one of the command line arguments to hmetis program, which is used to modify the allowed imbalance between the partitions. This argument UBfactor is used to modify partition ratios ranging from 0.1 to 0.9.

The algorithm shown in Fig. **6** is implemented in C++. Eclipse, an integrated development environment (IDE) is used to develop and carry the experiment. The operating system Ubuntu 12.04.4 LTS is used to run Eclipse. The hardware used include a processor of 4x Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz with memory of 4041MB. MCNC benchmark circuits *fract, primary1, industry1* and *biomed* are used to test our hypothesis. The Table **1** shows the characteristics of the benchmark circuits used. The experiment is carried in two phases. The first phase is carried out with unconstrained critical path with different partition ratios from 0.1 to 0.9 for all the above mentioned benchmarks. In the second phase, the critical path elements are given higher weights as shown in Fig. **6** and the experiment is carried with different partition ratios from 0.1 to 0.9.

Table 1. Characteristics of MCNC benchmark

Circuit	#Cells	#Nets	#Pins	#Rows
Fract	125	163	454	6
Primary1	752	1266	3303	16
Industry1	2271	2479	8024	64
Biomed	6417	5742	26947	46

IV. EXPERIMENTAL RESULTS

In Table **2**, the total wire lengths and maximum delays of MCNC benchmark circuit *fract* for various partition ratios are compared for both constrained and unconstrained critical paths. It clearly shows that the optimal wire lengths and optimal maximum delays are at partition ratio of 0.7. The Table **3** compares the wire lengths for benchmark *primary1*, the results show that optimal wire length is at partition ratio of 0.5 for constrained critical path and at partition ratio of 0.6 for unconstrained critical path. The Table **4** compares the wire lengths for benchmark *industry1*, the results show that optimal wire length is at partition ratio of 0.5 for constrained critical path and at partition ratio of 0.4 for unconstrained critical path.

Table 2. Comparison of wire lengths and maximum delays for MCNC benchmark *Fract*

Partition Ratio	Unconstrained Critical path		Constrained Critical path	
	Wire length	Maximum Delay	Wire length	Maximum Delay
0.1	1097.37	95.08	1085.12	91.8
0.2	957.75	90.84	955.31	90.81
0.3	957.75	90.84	955.31	90.81
0.4	1048.38	91.07	1028.78	91.64
0.5	955.3	91.1	947.95	92.22
0.6	955.3	91.1	947.95	92.22
0.7	911.21	88.89	889.16	87.62
0.8	957.75	89.65	962.64	89.44
0.9	957.75	89.65	962.64	89.44
Improvement				
	4.62%	2.43%	6.20%	4.99%

Table 3. Comparison of wire lengths and maximum delays for MCNC benchmark Primary1

Partition Ratio	Unconstrained Critical path		Constrained Critical path	
	Wire length	Maximum Delay	Wire length	Maximum Delay
0.1	11245.05	182.48	11431.63	182.28
0.2	11671.96	184.39	10587.3	182.02

0.3	11646.66	181.64	11561.28	182.23
0.4	9980.14	180.99	9907.41	181.42
0.5	9983.31	181.16	9003.01	183.88
0.6	9341.36	180.64	9138.98	180.36
0.7	9550.07	179.81	9853.65	181.27
0.8	9777.76	180.77	9796.73	181.55
0.9	10059.2	180.58	181.21	181.21
Improvement				
	6.43%	0.75%	0.0%	1.91%

Table 4. Comparison of wire lengths and maximum delays for MCNC benchmark *industry1*

Partition Ratio	Unconstrained Critical path		Constrained Critical path	
	Wire length	Maximum Delay	Wire length	Maximum Delay
0.1	51409.14	2697.53	50805.15	2808.79
0.2	43022.78	968.22	45498.85	1399.97
0.3	37080.86	494.25	38216.12	592.69
0.4	35120.25	483.08	36606.52	481.31
0.5	35123.41	528.81	34845.13	520.77
0.6	35895.01	465	35547.16	497.14
0.7	38842.25	463.54	38687.3	535.79
0.8	41400.53	630.31	42203.75	1228.07
0.9	43917.25	1867.16	42810.91	1504.05
Improvement				
	0.01%	12.34%	0.0%	7.58%

Table 5. Comparison of wire lengths and maximum delays for MCNC benchmark *biomed*

Partition Ratio	Unconstrained Critical path		Constrained Critical path	
	Wire length	Maximum Delay	Wire length	Maximum Delay
0.1	147128.12	31540.95	150095.49	29549.66
0.2	114007.53	9747.41	106288.04	5997.79
0.3	86107.77	2070.48	84853.04	1424.23
0.4	76336.73	841.21	68721.81	551.72
0.5	70351.51	491.98	71746.86	684.93
0.6	69994.56	541.13	72532.87	656.66
0.7	76827.08	702.85	75680.52	755.7
0.8	100594.88	4102.1	103097.13	4568.24
0.9	119304.08	12371.03	120641.74	13402.97
Improvement				
	0.51%	0.00%	4.22%	19.45%

The Table 5 compares the wire lengths for benchmark *biomed* the results show that optimal wire length is at partition ratio of 0.4 for constrained critical path and at partition ratio of 0.6 for unconstrained critical path. Also from the tables, we can observe that the improvement in wire length is significant with all of the benchmark circuits with an average improvement of 2.7%

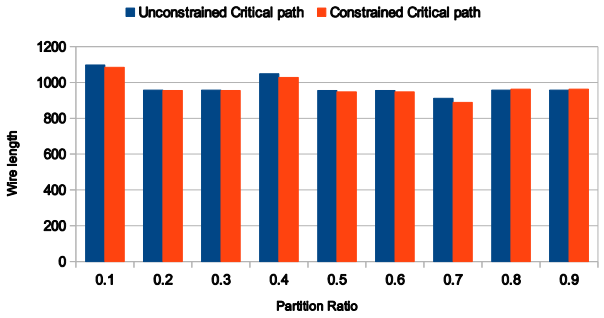

Fig. 7. Wire length variation for different partition ratios for fract

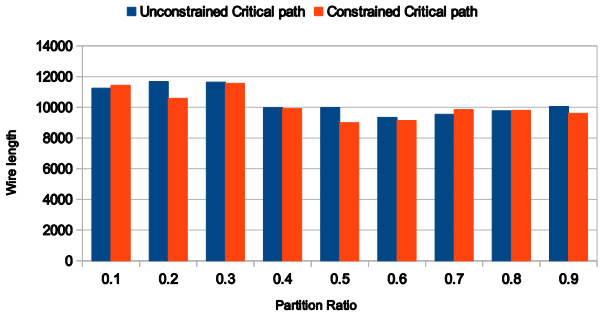

Fig. 8. Wire length variation for different partition ratios for primary1

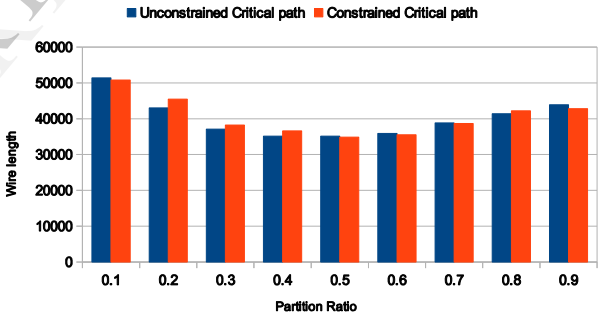

Fig. 9. Wire length variation for different partition ratios for industry1

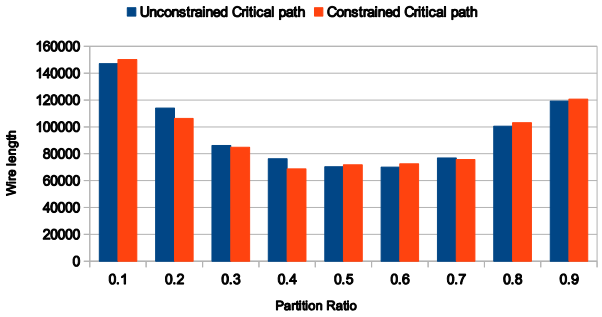

Fig. 10. Wire length variation for different partition ratios for biomed

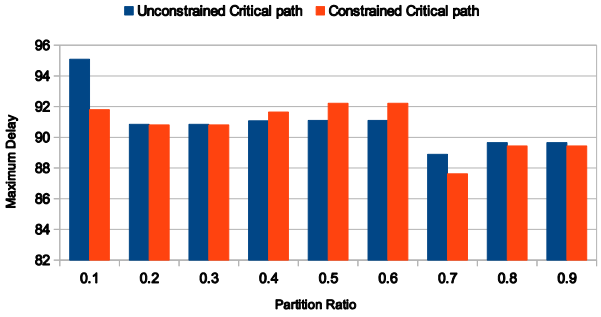

Fig. 11. Maximum Delay variation for different partition ratios for fract

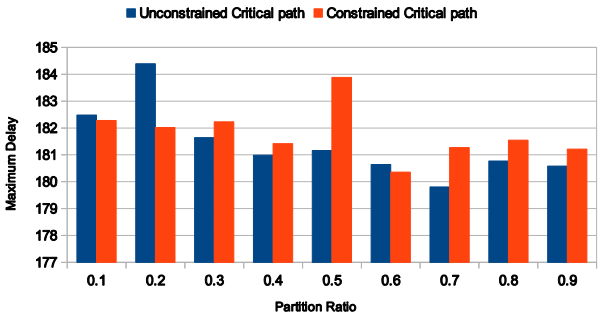

Fig. 12. Maximum Delay variation for different partition ratios for primary1

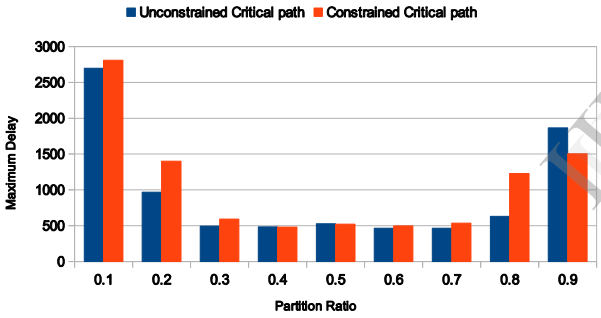

Fig. 13. Maximum Delay variation for different partition ratios for indusrty1

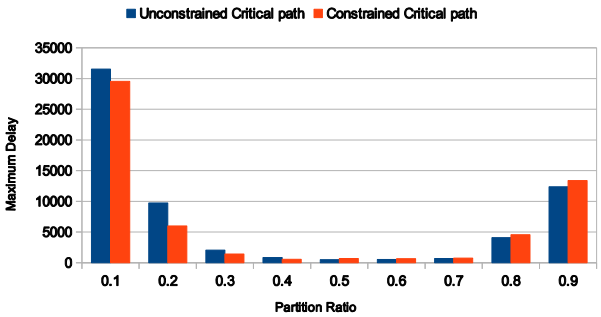

Fig. 14. Maximum Delay variation for different partition ratios for biomed

Fig. 7 to Fig. 10 show the variation of wire lengths for various partition ratios for MCNC benchmarks. While Fig. 11 to Fig. 14 show the variation of Maximum delay values for various partition ratios. From these figures, it can be conveyed that each of the circuit behave differently without any uniform variation to different partition ratios. The experimental results show that some of the circuits yield optimal wire lengths for equal sized partitioning while other circuits yield optimal wire length for unequal sized partitioning.

V. CONCLUSIONS

This paper examines the behavior of the circuits for critical path minimization for unequal sized recursive partitioning. Firstly modeling of unequal sized partitioning is discussed and then these models are applied to MCNC benchmark circuits. These results show that equal sized partitioning do not always give optimal placement results. The minimal wire length can occur at any partition ratio and purely a characteristic of the circuit topology. These results necessitate checking all the partition ratios to find which partitioning strategy gives optimal solution.

REFERENCES

[1] Michael, and Mary N. Youssef Burstein, "Timing influenced layout design," in *Design Automation, 1985*, 1985.

[2] Alfred E., Vishwani D. Agrawal, David N. Deutsch, M. F. Jukl, Patrick Kozak, and Manfred Wiesel. Dunlop, "Chip layout optimization using critical path weighting," in *In Proceedings of the 21st Design Automation Conference*, 1984, pp. 133-136.

[3] Tim Tianming Kong, "A novel net weighting algorithm for timing-driven placement," in *In Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, 2002, pp. 172-176.

[4] Haoxing, David Zhigang Pan, and David S. Kung Ren, "Sensitivity guided net weighting for placement-driven synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 24, no. 5*, pp. 711-721, 2005.

[5] Tong, Pravin M. Vaidya, and C. L. Liu Gao, "A performance driven macro-cell placement algorithm," in *Design Automation Conference*, 1992.

[6] Ren-Song, and Juergen Koehl Tsay, "n analytic net weighting approach for performance optimization in circuit placement," in *Proceedings of the 28th ACM/IEEE Design Automation Conference. ACM*, 1991.

[7] Majid, David Knol, and Gustavo Tellez Sarrafzadeh, "Unification of budgeting and placement," in *Proceedings of the 34th annual Design Automation Conference. ACM*, 1997.

[8] Karthik, et al Rajagopal, "Timing driven force directed placement with physical net constraints," *Proceedings of the 2003 international symposium on Physical design. ACM*, 2003.

[9] Bill, C. Y. Chen, and Naresh Sehgal Halpin, "Timing driven placement using physical net constraints.," in *Proceedings of the 38th annual Design Automation Conference. ACM*, 2001.

[10] Sung Woo, and John Lillis Hur, "Mongrel: hybrid techniques for standard cell placement.," *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design.*, 2000.

[11] Michael AB, and Ernest S. Kuh Jackson, "Performance-driven placement of cell based IC's," in *Proceedings of the 26th ACM/IEEE Design Automation Conference. ACM*, 1989.

[12] Arvind, Kamal Chaudhary, and Ernest S. Kuh Srinivasan, "RITUAL: A performance driven placement algorithm for small cell ICs," in *Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers., 1991 IEEE International Conference on. IEEE 1991*, 1991.

[13] William, and Carl Sechen Swartz, "Timing driven placement for large standard cell circuits," in *Design Automation, 1995. DAC'95. 32nd Conference on. IEEE*, 1995.

[14] Amit, Karthik Rajagopal, Satish Venkatesan, Tung Cao, Vladimir Tiourin, Yegna Parasuram, and Bill Halpin. Chowdhary, "How accurately can we model timing in a placement engine?," in *In Proceedings of the 42nd annual Design Automation Conference*, 2005, pp. 801-806.

[15] Maogang, Xiaojian Yang, and Majid Sarrafzadeh. Wang, "Dragon2000:

standard-cell placement tool for large industry circuits," *In Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, pp. 260-263, 2000.

[16] Ameya R., Satoshi Ono, and Patrick H. Madden. Agnihotri, "Recursive bisection placement: feng shui 5.0 implementation details.," *Proceedings of the 2005 international symposium on Physical design.*, 2005.

[17] Zhe-Wei, Tung-Chieh Cheny, Tien-Chang Hsuy, Hsin-Chen Chenz, and Yao-Wen Changyz. Jiang, "NTUplace2: A hybrid placer using partitioning and analytical techniques.," *In Proceedings of the 2006 international symposium on Physical design*, pp. 215-217, 2006.

[18] Jarrod A., David A. Papa, Saurabh N. Adya, Hayward H. Chan, Aaron N. Ng, James F. Lu, and Igor L. Markov. Roy, "Capo: robust and scalable open-source min-cut floorplacer.," *In Proceedings of the 2005 international symposium on Physical design*, pp. 224-226, 2005.

[19] Dennis J-H., and Andrew B. Kahng Huang, "Partitioning-based standard-cell global placement with an exact objective.," *In Proceedings of the 1997 international symposium on Physical design*, pp. 18-25, 1997.

[20] Melvin A Breuer, "A class of min-cut placement algorithms," *In Proceedings of the 14th Design Automation Conference*, pp. pp. 284-290, 1977.

[21] Wilm E. Donath, "Complexity theory and design automation.," *In Proceedings of the 17th Design Automation Conference*, pp. 412-419, 1980.

[22] George, and Vipin Kumar. Karypis, "hmetis: A hypergraph partitioning package, version 1.5. 3 user manual 23," 1998.