DCCB and SCC Based Fast Circuit Partition Algorithm For Parallel SPICE Simulation

Xiaowei Zhou, Yu Wang * , Huazhong Yang

Abstract —*With the rapid scale growing of VLSI circuits, simulation speed and efficiency of CAD tool SPICE have turned out to be a bottleneck. Real VLSI circuit design simulation becomes unbearably time-consuming and urgent is the need to increase its efficiency. The emergence and thriving of multi-core systems in recent years offer a promising solution strategy to this problem. Circuit partition is required to these strategies, but traditional partition algorithms encounter difficulties when facing VLSI circuits for parallel simulation. This paper presents an efficient circuit partition algorithm specially designed for VLSI circuit partition and parallel simulation. The algorithm is established on recognizing DCCB and SCC. Our algorithm shows preferable solution quality and speedup for real experimental circuit designs compared with traditional ones.*[1]

Index Terms —Parallel, circuit partition, DCCB, SCC, overweight circle

I. INTRODUCTION

The growing VLSI circuit size and increasing structure complexity make the transistor level circuit simulation more and more like a mission impossible. In most transient analysis in SPICE tools, simulation of some moderate scale circuit designs takes days to accomplish. Low simulation efficiency becomes a critical bottleneck for modern CAD tools.

The rapid development of multi-core and many-core systems in recent years provides a promising way to solve this problem through parallel simulation. Synopsis HSPICE has already released its dual core parallel version, which for some simulation tasks can improve the efficiency by 70%.

A prerequisite step for parallel simulation is circuit partition. Partition problem is a classical problem in CAD research and has wide range of applications. Algorithms to solve this NPC problem are well developed in recent years[1]. But most traditional methods may encounter difficulties in two aspects when facing VLSI partition problems for multi-core parallel simulation.

For the direct algorithms[2],[3],[4]-[11], overwhelming computing time corresponding to enormous problem size may be the main problem. If taking too much time on partition process, the whole simulation efficiency is very likely to be hampered, no matter how fast the parallel simulation is accomplished. For algorithms with pre-assemble steps[12]-[19], such as clustering methods, the pretreatment processes are often based on certain mathematical search criteria, rendering the solution quality of whole algorithm varies with circuit structure, cluster size and clustering strategies. Their scalability and stability are not well expected.

In this paper we introduce a new circuit partition algorithm specially designed for VLSI partition and multi-core parallel simulation, called DCCB (Direct Current Connected Blocks) and SCC (Strong Connected Components) based partition. The proposed algorithm shows considerable improvements in efficiency. We run our algorithm on some typical circuit designs and experimental results show that the algorithm gives satisfying computing time speedups and quality improvements.

The following of this paper is organized as below: Section II describes algorithm framework. Section III presents terminologies for recognizing DCCBs. Section IV gives out the analysis and strategies for recognizing SCCs and the overweight circle. Section V presents experimental results. Section VI concludes our work with some expectations and plans for future research.

II. CIRCUIT PARTITION FRAMEWORK

To implement the proposed algorithm we establish a software framework carrying out the process the algorithm describes. **Fig.1** gives out the basic flow chart of the whole process.

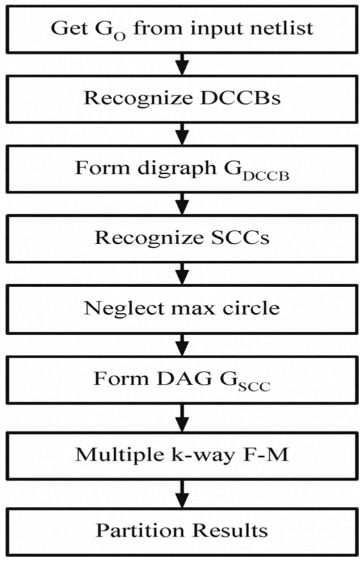


Fig. 1: Flow Chart of DCCB/SCC based partition

This work was supported by National Natural Science Foundation of China(No.60870001, No.90207002) and TNList Cross-discipline Foundation.

Xiaowei Zhou was in NICS group, EE. Department, Tsinghua University, Beijing, China. (e-mail: zhouxw3700@hotmail.com).

Yu Wang, the corresponding author, is assistant researcher in NICS group, EE. Department, Tsinghua University, Beijing, China. (e-mail:yu-wang@mail.tsinghua.edu.cn)

Huazhong Yang is professor in NICS group, EE. Department, Tsinghua University, Beijing, China. (e-mail:yanghz@tsinghua.edu.cn).

978-1-4244-3870-9/09/$25.00 ©2009 IEEE

In our partition algorithm, we first read the input netlist. With that we form the original graph G_O representing the circuit. Then we pre-partition G_O to a digraph G_{DCCB} illustrated in section III in details. After that we transform this digraph G_{DCCB} to a DAG G_{SCC} by recognizing SCC and the overweight circles. This step is seen in subsection A, B of section IV. At last we apply a classical multiple k-way partition to this DAG G_{SCC} to get a final result[11].

The advantage of the proposed algorithm can be explained in these following aspects:

1) The search space for optimization is greatly reduced thus the partition time may shrink by orders. This ensures the high simulation efficiency of our algorithm.
2) Most VLSI designs have quite clear functional structure and signal flows. Different DCCBs are often unconnected or simply coupled. Low communication cost is expected among DCCBs. On the other hand, duplication and redundancy commonly used in VLSI designs make the load balance requirement easy to be met. These two helpful characters ensure the good final solution quality our method reach.
3) Since DCCBs are commonly seen as basic functional blocks in VLSI circuits, our method is robust to most of application cases. Additionally, the recognition process is simple and determinable, few parameters would fluctuate solution quality.

III. DCCBs and Directed Cyclic Graph

A. DCCB Recognition

The input netlist of circuit design can be viewed as an original graph $G_O(V,E)$. Typically, G_O of a VLSI circuit may include tens of thousands of elements. In order to reduce problem size and speed up partition, we put MOSFETs and related passive device networks with direct current passage together to form **DCCB** (Direct Current Connected Blocks). This pretreatment method is for the first time introduced in VLSI circuit partition algorithms for adapting parallel simulation task. To better illustrate the ideas of DCCB, we first give out some definitions and classifications in **Table I**:

TABLE I
NODE ASSEMBLY DEFINITION FOR DCCB RECOGNITION

Assembly	Definition	Description
N	Node assembly	All the MOSFET D-nodes & S-nodes
N_I	Input node assembly	All nodes connected to outer input
N_P	Power node assembly	All nodes connected to V_{DD} or GND
N_{IN}	Inner node assembly	All other nodes in N than N_I or N_P

And **DCCB** can be defined as follows:

Theorem 3.1: A MOSFET i is recognized as component of DCCB j, iff either its D-node N_{di} or S-node N_{si} is directly connected (or via RLC networks) to at least one MOSFET's D-node or S-node in DCCB j, while at least one of the connection passages does not include N_P nodes.

Fig. 2 gives an example of DCCBs. All the 10 MOSFETs enclosed in the black dashed line form DCCB1, and the rest two outside the enclosure form DCCB2. **Fig. 3** gives out the pseudo code of DCCB recognition.

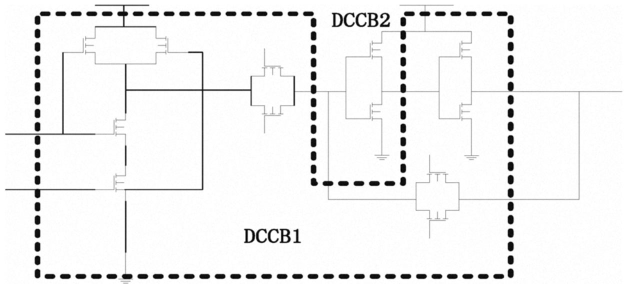
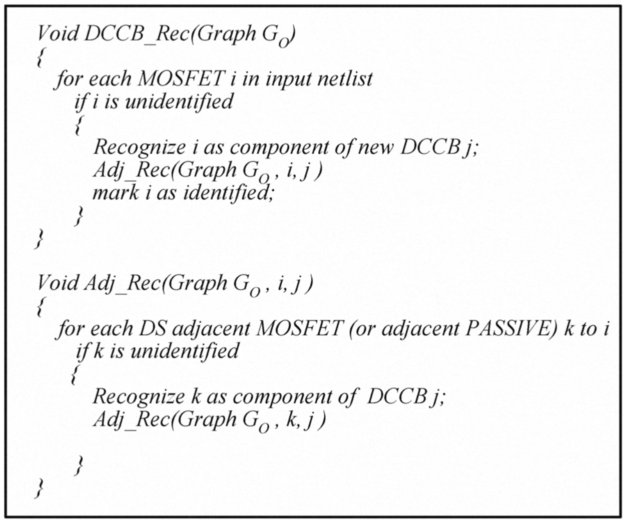


Fig. 2 Classifying DCCBs



```
Void DCCB_Rec(Graph $G_O$)
{
  for each MOSFET i in input netlist
    if i is unidentified
    {
      Recognize i as component of new DCCB j;
      Adj_Rec(Graph $G_O$, i, j )
      mark i as identified;
    }
}

Void Adj_Rec(Graph $G_O$, i, j )
{
  for each DS adjacent MOSFET (or adjacent PASSIVE) k to i
    if k is unidentified
    {
      Recognize k as component of  DCCB j;
      Adj_Rec(Graph $G_O$, k, j )

    }
}
```

Fig. 3: pseudo code of DCCB recognition

With proper data structure, this recognition process has linear time complexity. After recognizing DCCBs, we need to further identify the sequence of them to correctly represent the signal flow in the circuit. Few more corresponding definitions are given in **Table II**:

TABLE II
NODE ASSEMBLY DEFINITION FOR SCC RECOGNITION

Assembly	Definition	Description
N_O	Inner output node assembly	All D-nodes and S-nodes in DCCBs
N_{II}	Inner input node assembly	All G-nodes in DCCBs

And DCCB sequence is identified as follows:

Theorem 3.2: A DCCB i is recognized as "pointing" at a DCCB j, iff at least one of its N_O nodes is directly connected (or via RLC networks) to at least one of the N_{II} nodes in DCCB j.

By recognizing DCCBs and identifying their sequence relationship, we transform the original input circuit graph G_O to a new directed cyclic DCCB graph G_{DCCB}. **Fig. 4** gives an example of a G_{DCCB}. Root is an artificial node pointing to DCCBs not pointed at by any other DCCBs.

B. Theoretical analysis for algorithm speedup

In this subsection we try to give out some approximate estimation of the expected improvement of DCCB based pretreatment.

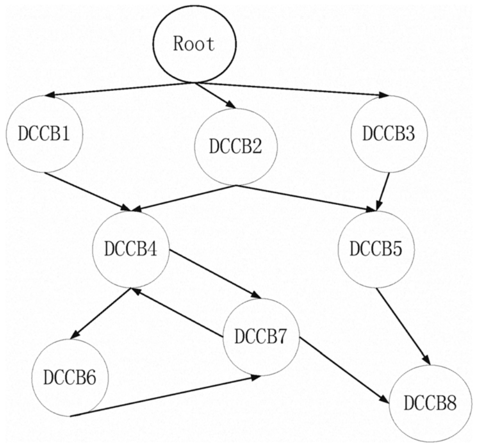


Fig. 4: Example of G_{DCCB}

For generality we assume the average time complexity of this algorithm to be $O(N^{\alpha})$. And we further assume each DCCB contains β elements on average after pre-partition. The computing time cost T_c of this classical algorithm without our DCCB recognition is:

$$T_c = k \bullet O(N^{\alpha}) \text{, where k is a constant} \qquad (1)$$

And when problem size shrinks to N/β, the T_c', representing time cost after DCCB recognition is:

$$T_c' = k \bullet O((N/\beta)^{\alpha}) = k \bullet O(N^{\alpha})/\beta^{\alpha} \qquad (2)$$

Let the average efficiency speedup to be γ, we get:

$$\gamma = T_c / T_c', \qquad (3)$$

In the estimation above we neglected the DCCB recognition time itself which has linear time complexity.Take $O(N^{\alpha})$ to be *NlgN* as an example. Based on real VLSI scale, we further assume N to be 10^4 and β to be 10. We get T_c from (1) and T_c' from (2):

$$T_c = k \bullet N \bullet \lg N = 4k \bullet 10^4,$$

$$T_c' = k \bullet (N/\beta) \bullet \lg(N/\beta) = 3k \bullet 10^3,$$

Still neglect the recognition time cost:

$$\gamma = T_c / T_c' = 16.67.$$

We see there is more than an order speedup theoretically expected for applying DCCB recognition method.

IV. Recognizing SCC and overweight circles

A. Recognizing SCC

The DCCB based graph G_{DCCB} is much more simplified than the original graph G_O, but still not suitable for applying partition algorithm, because it may contain circles, and assigning elements in one circle to more than two parts increases the communication cost. Due to this consideration, circle elements are better to be assigned in one partition. Thus we further cluster circling DCCBs to **SCC**s (Strongly Connected Components). SCC is recognized as follows:

Theorem 4.1: A DCCB *i* is recognized as component of SCC *j,* iff DCCB *i* can reach any other DCCBs in SCC *j* through a certain series of directed arcs.

By forming SCCs we transform the graph G_{DCCB} to a new DAG G_{SCC}. **Fig. 5** gives the SCC recognition result of the example in **Fig. 4**. SCCs are basic elements for partitioning in G_{SCC} and traditional k-way partition methods are used. **Fig. 6** gives pseudo codes of SCC recognition.

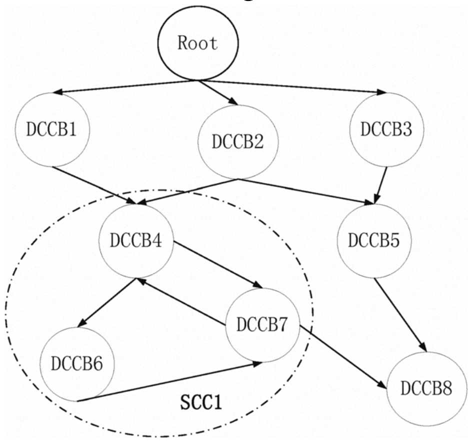


Fig. 5: SCC based DAG G_{SCC}

```
Void SCC_Detect(Graph $G_{DCCB}$, node i)
{
  Set level of node i;
  Set min_level of node i=level of node i;
  for each adjacent node j that i is pointing to
  {
    if j is unvisited
      SCC_Detect(Graph $G_{DCCB}$, node j)
    else
      if (min_level of node j<min_level of node i)
        set min_level of node i to min_level of node j;
  }
  for each node i in $G_{DCCB}$
  {
    Set k = min_level of node i;
    node i is recognized as component of SCC k;
  }
}
```

Fig. 6: pseudo codes of SCC recognition

B. Recognizing overweight circles

For some VLSI designs, there may be long feedbacks covering a wide range of stages and a large percent of the nodes in circuits. If they are put together by SCC recognition process, successive partition process may have difficulty to reach satisfying load balance. We call this kind of SCC "overweight circle". In our algorithm, overweight circles are not recognized as SCCs and are available for partition. This strategy is adopted with the following steps: first we recognize all SCCs, then for those SCCs larger than preset threshold size, we find an inner key node and cut off at least one adjacent node pointing to it in the SCC. We further recognize smaller circles in the original SCC. **Fig. 7** gives the pseudo codes of this recognition process:

```
Void OVC_Detect(Graph $G_{DCCB}$)
{
  for each SCC i larger that threshold number
    select the key node k;
    cut one arc that is pointing to k in SCC i;
    get subgraph $G_i$ from Graph $G_{DCCB}$;
    SCC_Detect(Graph $G_i$)
    OVC_Detect(Graph $G_i$)
}
```

Fig. 7: pseudo codes of recognizing overweight circle

V. Implementation and Experimental Results

In this paragraph we give out primary experimental results of our algorithm performance. We use traditional k-way F-M partition algorithm[11] as a comparison. Due to convenience of program testing and debugging, several typical mini-circuits are chosen as testbenches in the first step. All the experiments are done on an Intel 2.66GHz PC with 512M memory, on the platform of VC++6.0.

Circuit C1 is a 2-bit calculator in digital circuits. It contains 32 MOSFETs and 3 capacitors. Circuit C2 is a typical charge pump block in PLL, containing 90 MOSFETs and 6 passive devices. Algorithm performances are listed below in **Table III**. N is the problem size. The "k-way" row lists the performances given under a k-way partition task. As for C1 there are not enough DCCBs for 8-way partition and not enough SCCs for more than 4-way partition, performances are unavailable in corresponding rows.

Table III
Algorithm Performance for C1(N=35) and C2(N=96).

Algorithms / Performance		Traditional F-M		Based on G_{DCCB}		Based on G_{SCC}	
		C1	C2	C1	C2	C1	C2
Cut Cost (min)	2 way	4	2	4	2	4	2
	4 way	11	10	8	7	-	7
	8 way	19	23	-	16	-	16
Balan (min /max)	2 way	0.9	0.72	0.98	0.4	0.98	0.98
	4 way	0.63	0.53	0.14	0.12	-	0.12
	8 way	0.4	0.37	-	0.11	-	0.11
CPU Time (ms)	2 way	67	80	37	45	32	49
	4 way	83	104	54	64	-	59
	8 way	107	127	-	80	-	77

From **Table 3** we can see that:

1) Solution quality of our DCCB/SCC based algorithm appears better or not worse than direct F-M algorithm. This exceeds our theoretical expectation. Reason for it may lies in the fact that k-way F-M is very sensitive to initial solution, and our algorithm are not likely to begin with bad initial solutions.
2) The load balance of the proposed algorithm appears awful in both test circuits. It is not because of the disadvantages our algorithm has born with. It is due to lack of duplication and similar blocks in the two test circuits. In much larger real circuit design containing over 10^3 device elements, duplications and redundancies are commonly seen. We then have reason to believe that load balance requirements may be quite easily achieved in those cases.
3) Speedups are observed in all comparable cases. Notice that the speedup rate doesn't reach theoretical expectation. This is because the neglected processes of DCCB/SCC recognition still account for a certain percent of CPU time. This percentage is likely to decrease when scale N grows, as the two steps have lower time complexity compared to partition algorithm. Speedup for partitioning real VLSI circuits is expected to approach the theoretical expectation. This estimation can be to some extent approved by the speedup increase observed in C2 which is larger in problem size.

VI. Conclusions and Future Work

In this paper we present a DCCB/SCC based fast circuit partition algorithm. The DCCB/SCC recognition is simple and determinable with good stability and scalability, and the whole algorithm is hopefully suitable for VLSI parallel simulation. We run our algorithm process on some mini-circuits in real designs as first step experiment and results achieved are quite preferable. Future works may contain these following aspects:

1) Fulfill experiment results of real circuits with over 10^3 sizes.
2) Develop algorithms that properly assign every partition to a certain multi-core network with given topology restrict, in order to keep partition communication as low as possible.

References

[1] Jason Cong, Wilburt Juan Labio, and Narayanan Shivakumar, "Multiway VLSI circuit partition based on dual net representation", in IEEE Trans on CAD, Vol. 15, No. 14, April 1996.

[2] B. Kernghan and S. Lin, "An efficient heuristic procedure for partitioning of electrical circuits," Bell Syst. Tech. J., vol. 49, pp. 291-307, 1970.

[3] C. Fiduccia and R. Mattheyses, "A linear time heuristic for improving network partitions," in Proc. ACM/IEEE DAC, 1982, pp. 175-181.

[4] Yi Zou, Zhenquan Zhuang, Huanhuan Chen, "HW-SW partitioning based on genetic algorithm", in CEC, Volume 1, 19-23, June 2004, Page 628 – 633.

[5] G.F. Nan, M.Q. Li, J.S. Kou, "Two novel encoding strategies based genetic algorithms for circuit partitioning", in Proc. of ICMLC, Aug. 2004.

[6] B.Riess, K.Doll, and F.M.Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques",in 31st ACM/IEEE DAC, 1994.

[7] Yen-Chuen Wei, Chung-Kuan Cheng, "Ratio Cut Partitioning for Hierarchical Designs", IEEE Transactions on CAD, Vol.10, 1991.

[8] Yang and D.F.Wong, "Efficient network flow based min-cut balanced partitioning," in Proc. IEEE Int. Conf. CAD, Nov. 1994, pp. 50-55.

[9] W.K. Mak, "Min-cut partitioning with functional replication for technology-mapped circuits using minimum area overhead", in IEEE Trans. On CAD, Volume 21, April 2002, Page 491 – 497.

[10] Dutt S., W.Y. Deng, "Probability-based approaches to VLSI circuit partitioning", in IEEE Trans. On CAD, Volume 19 , May 2000, Page 534 – 549.

[11] L. Sanchis, "Multiple-way network partitioning", IEEE Trans. On Comput., vol. 38, pp. 62-81, 1989.

[12] Liao Qin, Li Xiwen, "The effective clustering partition algorithm based on genetic evolution", in Journals of Donghua University, vol. 23, 2006.

[13] P. Chan, M. Schlag, and J. Zien, "Spectral K-way ratio-cut partitioning and clustering," in Proc. 30th ACM/IEEE DAC, 1993, pp. 749-754.

[14] Blaschko M.B., Lampert C.H, "Correlational spectral clustering", in IEEE Conf. On CVPR, 23-28, June 2008, Page 1-8.

[15] Kan Li, Y.S.Liu, "A Spectral Clustering Algorithm Based on Self-Adaption", in ICMLC, Vol. 7, 19-22 Aug. 2007, Page3965-3968.

[16] V.Kolar, N.B.Abu-Ghazaleh, "A multi-commodity flow approach for globally aware routing in multi-hop networks", in ICPCC 2006, pp. 313– 317.

[17] S.Bethi, V..Phoha, Y.Reddy, "Clique clustering approach to detect denial-of-service attacks", in Proc.of 5th Annual IEEE SMC 2004, Page 447-448.

[18] C.J.Alpert and A.B.Kahng, "Geometric embedding for faster (and better) multi-way netlist partitioning," in Proc. 30th ACM/IEEE DAC, June 1993, pp. 743-748.

[19] S.L. Zhang, C.Q. Shi, Z.Y. Zhang, Z.Z. Shi, "A Global Geometric Approach for Image Clustering", in 18th ICPR, 2006, Vol. 4.