OBDD variable ordering by interleaving compacted clusters

Fu-Min Yeh and Chen-Shang Lin

Indexing terms: Boolean functions, OBDD variable ordering

An effective method for variable ordering of OBDDs based on the interleaving of the compacted clusters is proposed. The novelty of this method lies in the application of the divide-and-conquer approach to efficiently find a good variable ordering for circuits with a large number of I/Os. One notable result from this method is that the OBDD is able to be built using the cs38417 circuit, the remaining unreported circuit in the ISCAS89 benchmark.

Introduction: The ordered binary decision diagram (OBDD) [1] has become one of the most efficient and versatile representations of Boolean functions in the applications of formal verification, test generation and logic synthesis. In the manipulations as well as the applications of OBDDs, the efficiency is dominated by the size of the OBDD for representing a given function. It has been observed the OBDD size strongly depends on the ordering of the input variables. However, the best known algorithm of finding optimal ordering has complexity $O(n^2 3^n)$. Heuristic methods [2–5] for good variable ordering have therefore been proposed. In particular, the dynamic sifting algorithm proposed in [3] has a higher performance of variable ordering for OBDD than previous works. By using an efficient level exchange algorithm, it is able to explore a large search space and obtain a very compact OBDD. However, in the worst case, the sifting algorithm requires $O(n^2)$ swaps of adjacent levels in dynamic reordering process where n is the variable number. For circuits with large numbers of I/Os, it may need excessive time if reordering occurs frequently.

An effective method for variable ordering of OBDDs based on interleaving the compacted clusters is proposed in this Letter. The novelty of this method is to apply the divide-and-conquer approach to efficiently finding a good variable ordering for circuits with a large number of I/Os. This divide-and-conquer approach allows us to obtain a good ordering more efficiently than existent methods for the cases with a large number of I/Os.

Variable ordering by divide-and-conquer: Our method of variable ordering for larger circuits with large number of I/Os is based on the divide-and-conquer approach. The method consists of three main steps:

(i) The large circuit is partitioned into a number of smaller circuits, or clusters such that the correlation among clusters is sufficiently low.
(ii) Each cluster is then extensively searched for a good variable ordering. A variable ordering is good in the sense that it yields a compact OBDD.
(iii) Based on the those good orderings of clusters, a global ordering is then obtained by preserving as much as possible of the original orderings of clusters.

We will describe these three steps subsequentially.

In the first step, we are to partition a given circuit into clusters so that a good ordering can be efficiently searched for each cluster. A cluster is a subcircuit consisting of some primary outputs and all the gates and primary inputs within the fan-in cone of these primary outputs. In cluster partition, the primary outputs of a cluster have no overlapping with other clusters. However, a primary input may appear in several clusters depending on the circuit topology. The correlation of primary inputs among clusters has a major impact on the last step when the orderings of clusters are to be combined. It is regulated by a *cluster_factor* to reduce such correlation. The *cluster_factor* may have its value between 0 and 1. During the partition, the factor is compared with a computed correlation value between the fan-in cone of a PO, O, and another cluster[i] defined by

$$Comon[i].ratio = \frac{\#PI\ of\ O\ included\ in\ Cluster[i]}{\#PI\ of\ O}$$

where # *PI* is the number of primary inputs. The subcircuit O is mergered into the cluster with the highest *common.ratio* only when the computed correlation is greater than the predetermined *cluster_factor*. Otherwise, it become a new cluster. Primary outputs for merging are determined by sorting them according to the decreasing number of gates in the fan-in cone. A high *cluster_factor* implies that only highly correlated primary outputs are merged into one cluster. An additional step for the procedure is to merge those primary outputs with < 16 primary inputs into a single cluster to avoid ineffective compaction due to numerous clusters. The ordering of this cluster is assigned the lowest priority.

In obtaining variable ordering of compact OBDD size for a cluster, the dynamic sifting algorithm in the CMU package [7] is modified for such a purpose. The trigger condition of reordering is modified to be adaptive to the reordering effectiveness. When the reordering is less effective, we allow more node increase before the next reordering takes place. The purpose is to reduce ineffective and time-consuming reordering when there are many intermediate gates in the same level.

Given the good orderings of individual clusters, the last step is to obtain a good global ordering for the entire function or circuit. A variable ordering of OBDD is a sequence of variables. When two OBDDs for two clusters are to be combined into a compact one, it is desirable to retain as much as possible of the good ordering of each cluster. The interleaving of two variable orderings serves such a purpose. There may be common variables between two sequences and in reality it is possible to have common variables of different orders in two sequences. We first justify the interleaving method for the case when no such conflict exists. Since the OBDD is unique for a given function, it will not change even when a new variable is inserted into the ordering regardless of the position. Therefore, when two OBDDS are combined without any conflict in the original orderings, the resultant size can only be equal to the sum of the two original sizes before merge and deletion rules are applied. After applying these two rules, the resultant size of the OBDD can be no greater than the sum of the original sizes. Therefore, we have the following result:

Theorem: Let two OBDDS have ordering(A) and ordering(B), respectively, without any conflict. Then the resultant OBDD with the interleaved variable ordering has a size no greater than the sum of the two original OBDDs.

Even when there are conflicts in the original orderings, the resultant OBDD size is generally strongly correlated to the sum of individual sizes. In this case, some compromise must be made. The ordering of cluster with higher priority determine the global ordering. The priority of each cluster is defined from the CPU time of OBDD compaction because a long compaction time is a good indicator of a hard cluster. The interleaving is therefore given by

```
Interleaving( G_ordering, L_ordering)
{ for each variable v in L_ordering from top
    { if ( v does not belong to G_ordering ) )
        add v to queue();
        if ( next_variable(v) belong to G_ordering )
            while ( ( w = dequeue()) != EMPTY)
                insert w to G_ordering just before next_variable(v);
    }
    if ( queue() != EMPTY )
        append all variables of the queue() to G_ordering
}
```

Although the principle of ordering interleaving is similar to that in [2], we apply it in a unique way for the divide-and-conquer approach. The correlation among clusters is reduced sufficiently low in cluster partition to maximise the chance of interleaving. Thereby, we are able to take better advantage of the good orderings of clusters to obtain a more compact resultant OBDD.

Table 1: Comparison results of ISCAS89 benchmark circuits

Circuit	[2]		Sift-sift [5]		Ours		
	n	Time	*n*	Time	*c*	*n*	Time
cs9234	66k	14.5	4k	336	7	6k	29
cs13207	15k	11.8	–	–	9	6k	10
cs15850	62k	22.5	37k	5059	11	37k	76
cs35932	6k	28.6	–	–	1	5k	46
cs38584	35k	41.4	–	–	54	30k	165
cs38417	>2M	–	–	–	41	607k	996

[2]: on SPARC 470 with 96M

[5]: on Dec 5000/200

Ours: on SPARC 20 with 128M

C: number of clusters

– : no data

Results and discussions: Our variable ordering method for OBDD has been employed to build OBDDs for the larger ISCAS89 benchmark circuits with the number of I/Os > 500. In Table 1, we compare our result with those of [2, 5]. The direct result of [3] is not shown here because no CPU time has been provided, which is essential in the discussion. The result of [2], on the other hand, is based on interleaving the orderings for each primary output obtained from a DFS heuristic. In this comparison result, the value of *cluster_factor* is 0.8, which is determined empirically to deliver better overall result. In this table, *n* indicates the the number of OBDD nodes and 'time' is the CPU time. From this table, we can see that the performance of our method generally falls between the fully dynamically sifting algorithm [5] and the fully interleaving-based algorithm [2]. The result of [2], although faster, produces OBDDs significantly less compact than the other two. The reason is that too many clusters of less compact size lead to more conflicts in interleaving and even larger resultant size. On the other hand, the fully dynamic sifting algorithm [5] generally has the most compact result. The main disadvantage of the fully dynamical sifting is the dramatic growth of the process time for the circuits with large numbers of I/Os such as cs15850 and cs38417. In comparison, our method generates OBDDs of the same order as [5]. However for difficult circuits with a large number of I/Os, our method is significantly faster as in cs15850. Moreover, for this circuit, our divide-and-conquer approach actually yields a more compact OBDD than does fully dynamical sifting. More significantly, we are able to construct by our method the OBDD of cs38417 which has not been reported so far in the literature. Its resultant OBDD has a size of ~607 knode and is completed in 996 s. In contrast, a recent report [6] on building a complete OBDD for cs38417 failed at the first 60% gates with > 100 million nodes allocated on 2 Gbyte virtual memories. With cs38417 solved, all the circuits in ISCAS85 and ISCAS89 benchmarks, except the multiplier c6288, have been shown to have an OBDD of a manageable size.

© IEE 1995 *20 July 1995*
Electronics Letters Online No: 19951177

Fu-Min Yeh and Chen-Shang Lin (*Department of Electrical Engineering, National Taiwan University, Taipei, 10764, Taiwan, Republic of China*)

References

1 BRYANT, R.E.: 'Graph-based algorithms for boolean function manipulation', *IEEE Trans.*, 1986, **C-35**, pp. 677–691
2 FUJII, H., OOTOMO, G., and HORI, C.: 'Interleaving based variable methods for ordered binary decision diagrams'. Proc. IEEE Int. Conf. Computer Aided Design (ICCAD '93), 1993, pp. 38–41
3 RUDELL, R.: 'Dynamic variable ordering for ordered binary decision diagrams'. Proc. IEEE Int. Conf. Computer Aided Design (ICCAD '93), 1993, pp. 42–47
4 YEH, F.-M., and LIN, C.-S.: 'Building OBDDs with ordering-reshuffle strategy', *Electron. Lett.*, 1993, **29**, (17), pp. 1540–1541
5 PANDA, S., SOMENZI, F., and PLESSIER, B.F.: 'Symmetry detection and dynamic variable ordering of decision diagrams'. Proc. IEEE Int. Conf. Computer Aided Design (ICCAD '94), 1994, pp. 628–631
6 ARSHAR, P., and CHEONG, M.: 'Efficient breadth-first manipulation of binary diagrams'. Proc. IEEE Int. Conf. Computer Aided Design (ICCAD '94), 1994, pp. 622–627
7 LONG, D.E.: 'A binary decision diagram package' 11 June 1993, unpublished

Coplanar stripline to microstrip transition

R.N. Simons, N.I. Dib and L.P.B. Katehi

Indexing terms: Coplanar waveguides, Strip line couplers, Strip lines

A new coplanar stripline to microstrip line transition is experimentally demonstrated and modelled using the finite difference time domain technique. The measured insertion loss and return loss for two back-to-back transitions with a short length of microstrip line in between are better than 2.4 and –10dB, respectively, over the frequency range of 5.1 – 6.1GHz. The bandwidth of the transition is ~18 % at a centre frequency of 5.55GHz.

Introduction: The coplanar stripline (CPS) [1] is a useful transmission line for feed networks of printed circuit antennas [2, 3]. The CPS has all the advantages of a slot line and coplanar waveguide (CPW) and in addition, the CPS makes efficient use of the wafer area and thus the die size per circuit function is small. This results in a lower cost and larger number of circuits for a given die size. Furthermore, the CPS propagation parameters are independent of the substrate thickness beyond a certain critical thickness, which simplifies heat sinking and circuit packaging. There are several applications besides feed networks for antennas where a CPS to microstrip line transition is necessary. These include: direct coupling of signals into a CPS from a microstrip monolithic microwave integrated circuit (MMIC) amplifier or oscillator, as a transition in a ground-signal (GS) wafer probe for on-wafer characterisation of two terminal semiconductor devices and as a DC block in microwave active circuits.

We present the design, experimental characterisation and modelling, using the finite difference time domain (FDTD) technique [4], of a new CPS to microstrip transition.

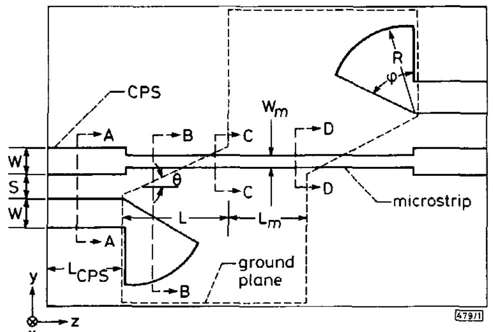

Fig. 1 *Schematic diagram of two back-to-back CPS to microstrip transitions*

D = 0.01 in, εr = 10.5, W = 0.0272in, S = 0.0073in, L_{CPS} = 0.27in, R = 0.2in, ϕ = 60°, L = 0.28in, W_m = 0.0073in, L_m = 0.5in, $\theta \simeq$ 7°

Transition design and fabrication: Two CPS to microstrip transitions in a back-to-back configuration are schematically illustrated in Fig. 1. A single transition consists of a CPS on a dielectric substrate of thickness D and relative permittivity ε_r with one of the strip conductors terminated in a microstrip radial stub of radius R and angle ϕ. The other strip conductor is extended to form the microstrip line. The CPS strip conductor width and separation are W and S, respectively. The width of the microstrip line is W_m. The resonance frequency of the radial stub depends on the radius and