A tuning curve plotting the sharp peak wavelength against pump wavelength is given in Fig. 2, where the signal and idler wavelengths have been experimentally measured up to 1·7 $\mu$m. Above this wavelength, the sensitivity of the Ge detector falls off and the idler wavelength was determined by imposing the conservation of energy condition: $\omega_p = \omega_s + \omega_i$ where $\omega_p$, $\omega_s$, and $\omega_i$ represent the pump, signal, and idler frequencies, respectively. This curve indicates that we can obtain a tuning range of 1·2–2·5 $\mu$m using the wavelength variation that is possible to achieve with a temperature-tuned diode laser.

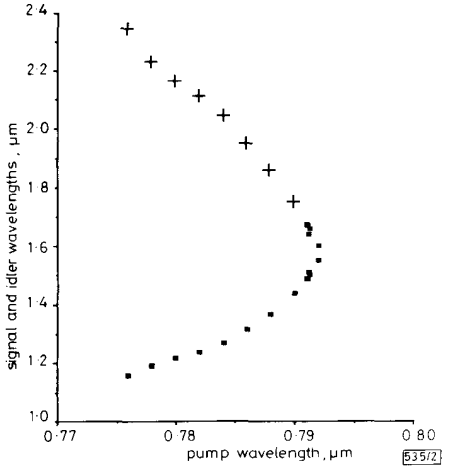


**Fig. 2** *Tuning curve of sharp fluorescence peak signal and idler wavelengths against pump wavelength at ambient temperature in 6 $\mu$m wide guide, with 19 $\mu$m domain inversion period*

■, □ measured values of signal and idler wavelengths
\+ values of idler wavelength calculated using the energy conservation condition

In conclusion, we have demonstrated efficient QPM parametric fluorescence at ambient temperature, using pump wavelengths and powers attainable with diode lasers. This indicates the possibility of realising diode laser pumped integrated optical parametric oscillators and work is continuing in this direction.

*Acknowledgment:* This work was partially supported by the Minister of Research and Technology.


*28th June 1993*

P. Baldi, S. Nouh, M. de Micheli and D. B. Ostrowsky (*Laboratoire Physique de la Matière Condensée, Université de Nice-Sophia Antipolis, Parc Valrose, 06108 Nice Cedex 2, France*)

D. Delacourt, X. Banti and M. Papuchon (*Thomson-CSF/LCR, Domaine de Corbeville, 91404 Orsay Cedex, France*)

## References

1 HAMPEL, B., and SOHLER, W.: 'Optical parametric fluorescence in Ti : $LiNbO_3$ channel waveguides'. SPIE Proc., 1986, **651**, Integrated Optical Circuit Engineering III, pp. 229–234

2 SOHLER, W., and SUCHE, H.: 'Optical parametric oscillation in Ti-diffused $LiNbO_3$ optical waveguide resonators'. Tech. Dig. 3rd Int. Conf. Integrated Opt. and Opt. Fiber Commun., 1981, San Francisco, CA, pp. 89–90

3 ARMSTRONG, J. A., BLOEMBERGEN, N., DUCUING, J., and PERSHAN, P. S.: 'Interactions between light waves in a nonlinear dielectric', *Phys. Rev.*, 1962, **127**, pp. 1918–1939

4 LIM, E. J., FEJER, M. M., and BYER, R. L.: 'Second harmonic generation of green light in periodically poled planar lithium niobate channel waveguides', *Electron. Lett.*, 1989, **25**, pp. 174–175

5 WEBJORN, J., LAURELL, F., and ARVIDSSON, G.: 'Blue light generated by frequency doubling of diode laser light in a lithium niobate channel waveguide', *IEEE Photonics Technol. Lett.*, 1989, **1**, pp. 316–317

6 LIM, E. J., HERTZ, H. M., BORTZ, M. L., and FEJER, M. M.: 'Infrared radiation generated by quasiphase-matched difference-frequency mixing in a periodically poled lithium niobate waveguide', *Appl. Phys. Lett.*, 1991, **59**, (18), pp. 2207–2209

# BUILDING BDDs WITH ORDERING-RESHUFFLE STRATEGY

F.-M. Yeh and C.-S. Lin

*Indexing terms: Boolean functions, Computer aided design*

An ordering-reshuffle strategy for building BDDs from a net list description is proposed. This algorithm dynamically modifies an initial ordering according to the encountered adverse situation to maintain the intermediate BDD under a reasonable size. The effectiveness of this strategy is demonstrated by building common-ordering BDDs for large circuits in the ISCAS85 benchmark including c7552 which hitherto has not been successful.

*Introduction:* The binary decision diagram (BDD) [1] has been widely recognised as one of the most efficient representations of Boolean functions. On the building of BDDs, the ordering of variables is recognised as the dominant factor affecting BDD size which also has a major impact on the manipulation efficiency. For example, a multiplexer of $n$ variables in its best ordering has BDD size smaller than $2n$, whereas in the worst ordering has a size larger than $2^{(n+1)}/n$. Unfortunately, the best known algorithm for finding optimal ordering has complexity $O(n^2 3^n)$.

As a result, various ordering heuristics [2, 4] based on the given netlist descriptions have been proposed. These heuristics are employed in a static way, i.e. the ordering remains intact during the building process, which restricts their ability to deal with large circuits. As a consequence, to the best of our knowledge, no results have been published for these two circuits, c2670, and c7552 of the ISCAS85 benchmark, although Reference 5 was able to build a BDD for c2670 after resynthesis.

To overcome the problem of excessive sizes of intermediate function gates during the BDD building process, an ordering-reshuffle strategy for building BDDs from netlist descriptions is proposed in this Letter. The proposed strategy dynamically modifies a given ordering according to the latest encountered adverse situation to allow an intermediate BDD to maintain a reasonable size.

*Ordering-reshuffle strategy:* Given a netlist description, a limit of maximum vertices *Max_size*, and a limit of maximum tries *Max_tries*, the ordering-reshuffle strategy is as follows. First, build a BDD from a description with a given ordering. When the building process is not successful under the given limit of vertices, the ordering is modified to accommodate the particular situation of the last encountered intermediate gate denoted as *blocking_gate*; then repeat the building process until the BDD is successfully built or an abort condition is met.

The algorithm based on our ordering-reshuffle strategy is shown in Fig. 1. We will discuss *Initial_Order*, *Reorder*, and *Reshuffle* procedures in detail. The initial variable ordering *R_ordering* is obtained from an existing ordering heuristic. Two such initial ordering heuristics are used separately: Fan-in depth first search (DFS) [2] and dynamic weight assignment (DWA) [3].

When BDD building with the initial ordering is not successful, the ordering is modified by *Reorder*(). The goal of

```
Ordering-reshuffle algorithm ()
{
given Max_size and Max_tries;
Try = 0;
R_ordering = Initial_order();
contruct BDD with R_ordering;
while((BDD_size > Max_size) and (Try < Max_tries)) do
      {
      B_ordering = Reorder(blocking_gate);
      R_ordering = Reshuffle(R_ordering,B_ordering);
      reconstruct BDD with R_ordering;
      Try = Try + 1
      }
}
```

6/4/1

**Fig. 1** *Ordering-reshuffle algorithm*

*Reorder*() is to overcome the problem of excessive sizes of the *blocking_gate* during building process. A natural solution is to reorder the variables within the fan-in cone of the *blocking_gate* into *B_ordering* which will then have higher priority than the original *R_ordering* in *Reshuffle*(). In the *Reorder* procedure, the simple but effective DFS [2] heuristic is used to reorder the variables within the fan-in cone of the given *blocking_gate*. Note that *Reorder* is to change the ordering of variables directly related to the encountered adverse situation. This is distinct from simulated annealing [6] which changes only neighbouring variables regardless of their relevance to the encountered situation.

*Reshuffle*() is then employed to accommodate both the original *R_ordering* and the new *B-ordering*. The rationale is that it is also desirable to keep the *R_ordering* as intact as possible because it is optimised for previous situations as well as final BDD size. For this purpose, *Reshuffle*() calculates the weight of each primary input in the fan-in cone of the *blocking_gate* and then reorders all primary inputs according to their weights, which will be described next.

We calculate the new weight of those primary inputs covered by the fan-in cone of the *blocking_gate* according to the following formula. For the covered *i*th primary input in *B_ordering*, let *R*[*i*] be its order in *R_ordering*; then the new weight of the primary input can be determined by

$$R[i].\,weight = \alpha * R[i].\,weight + \frac{Cone\_size * \beta * Covered\_pi}{Max\_cone * i * All\_pi}$$

where $\alpha$, $\beta$ are the two weighting factors, *All_pi* is the number of primary inputs, *Max_cone* the largest cone size of all primary outputs, *Covered_pi* the number of primary inputs in the *blocking_gate* fan-in cone, *Cone_size* the number of gates in the *blocking_gate* fan-in cone, and the original *R*[*i*] . *weight* is simply the inverse of its order in *R_ordering*.

Note all covered primary inputs have their weights updated while the weights of the remaining primary inputs remain the same. Furthermore, with $\alpha \ll 1 \ll \beta$ in practice, the *B_ordering* in the fan-in cone of the *blocking_gate* is given much higher weight. Therefore, when the new *R_ordering* is determined by sorting all primary inputs according to decreasing weight in *Reshuffle*(), the *B_ordering* will have higher priority and the new ordering is modified to accommodate the *blocking_gate*.

In the algorithm, termination conditions must also be included in employing the strategy to avoid futile attempts due to an unreasonably small vertex limit and the possible repeated tries of the same ordering. When the given reordering limit, *Max_tries*, is reached, the algorithm terminates and the users can decide whether to raise the size limit.

*Results:* The above algorithm has been employed to build BDDs for the large ISCAS85 benchmark circuits, c2670, c3540, and c7552. The other smaller circuits can be easily built with published static ordering heuristics. In this evaluation, the maximum vertex size is limited to 500 K beyond which the variable ordering is modified to accommodate the last encountered internal circuit node. The node size of our result is for shared BDD [4].

In Table 1, the results of vertex size and the numer of ordering modifications, including the three difficult larger circuits, are shown for both initial ordering heuristics, DWA and DFS, respectively. Both heuristics alone fail for these large and 'difficult' circuits, c2670, c3540, and c7552, within the 500 K limit. However, with our ordering-reshuffle strategy, all three circuits can be completed within the 500 K limit for both initial ordering heuristics. This shows the effectiveness of our ordering strategy in building BDDs. The resultant orderings in Table 1 are reported in Reference 7. It is also interesting to note that the final BDD sizes of c2670 and c7552 are quite small, 60 545 and 36 503, respectively, for DFS. In fact, both BDDs can be built even under a 300 K with our ordering strategy, yet they defy building up for more than a 2 M limit without the strategy.

**Table 1** RESULTS OF ISCAS85 BENCHMARK CIRCUITS WITH ORDERING-RESHUFFLE

| CKT | [2] [3] [4] | [5]* | [6]† | Ordering reshuffle DWA | R | DFS | R |
|---|---|---|---|---|---|---|---|
| c2670 | — | 91434 | — | 93482 | 1 | 60545 | 1 |
| c3540 | — | — | 52522 | 198622 | 2 | 199846 | 1 |
| c7552 | — | — | — | 172027 | 7 | 36503 | 6 |

* After resynthesis
† Extensive minimisation with simulated annealing
— unable to complete within bound
DWA: initial ordering heuristic DWA, DFS: initial ordering heuristic DFS, R: number of reorderings

*Conclusions:* An ordering-reshuffle strategy for building BDDs from netlist descriptions has been presented to overcome the problem of excessive intermediate size resulting from improper ordering. Our main concern has been to take the situation of the BDD building process into consideration and adapt the process accordingly, to successfully complete an entire BDD. Bearing this goal in mind, we have used the existing heuristics such as DFS and DWA in our ordering-reshuffle strategy to accomplish what these heuristics alone have not been able to. In other words, with the strategy, we have demonstrated that a BDD can be applied to circuits of larger size than hitherto thought possible.


*2nd July 1993*

F.-M. Yeh and C.-S. Lin (*Dept. 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 MALIK, S., WANG, A. R., BRAYTON, R. K., and SANGIOVANNI-VINCENTELLI, A.: 'Logic verification using binary decision diagrams in a logic synthesis environment'. Proc. IEEE Int. Conf. Computer Aided Design (ICCAD '88), 1988, pp. 6–9
3 CALAZANS, N., ZHANG, O., YERNAUX, B., and TRULLEMANS, A.-M.: 'Advanced ordering and manipulation techniques for binary decision diagrams'. Proc. IEEE Int. Conf. European Design Automation (EDAC '92), 1992, pp. 3452–457
4 MINATO, S.-I., ISHIURA, N., and YAJIMA, S.: 'Shared binary decision diagrams with attribute edges for efficient Boolean function manipulation'. Proc. IEEE Int. Conf. Design Automation (DAC '90), 1990, pp. 52–57
5 FUJITA, M., MASUNAGA, Y., and KAKUDA, T.: 'On variable ordering of binary decision diagrams for the application of multi-level logic synthesis'. Proc. IEEE Int. Conf. European Design Automation (EDAC '91), 1991, pp. 50–54
6 MERCER, M. R., KAPUR, R., and ROSS, D. E.: 'Functional approaches to generating ordering for efficient symbolic representations'. Proc. IEEE Int. Conf. Design Automation (DAC '92), 1992, pp. 624–627
7 YEH, F.-M., and LIN, C.-S.: 'Building BDDs with ordering-reshuffle strategy'. Technical Report, Dept. of EE, NTU, 1993, pp. 10–14


## IMPROVED *C*(*t*) METHOD FOR GENERATION RATE DETERMINATION IN IMPLANTED MOS STRUCTURES

R. Sorge

*Indexing terms: Ion implementation, Integrated circuits*

A method for rapidly evaluating the generation rate in the bulk of an MOS structure without requiring knowledge of the doping profile is described. The proposed technique is based on the measurement of two high frequency capacitance time transients after applying a depletion voltage step and a linear voltage ramp on the gate electrode which drives the device from depletion towards accumulation.

*Introduction:* The bulk generation rate $g(x_g)$ of minority carriers is an essential parameter in the manufacture of many semiconductor devices. The widely used pulsed MOS capacitor *C*(*t*) transient method of Zerbst [1] for determining the