A Process Monitor Based Speed Binning and Die Matching Algorithm

Conference Paper *in* Proceedings of the Asian Test Symposium · November 2011

DOI: 10.1109/ATS.2011.96 · Source: DBLP

Sreejit Chakravarty
LSI Corporation

2011 Asian Test Symposium

A Process Monitor Based Speed Binning and Die Matching Algorithm

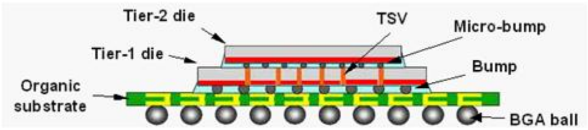
Sreejit Chakravarty
LSI Corporation
Milpitas, CA, USA

Abstract— *Speed binning groups ICs with similar performance and price point. Die matching matches dice with similar performance for system and 3D integration. A hybrid test flow combining process monitor readings and search using manufacturing test is presenteds. This flow eliminates error due to process monitor data inaccuracies and reduces test time of manufacturing test based search. Silicon data is presented.*

Keywords: Process Monitor, Speed Binning, Die Matching, 3D Integration

I. INTRODUCTION

When integrating systems, it is important to match dice with similar performance. This results in systems with wider performance range which can then be binned for different price points. Similarly, in 3D layouts, as shown in Figure **1**, 2 or more dice are stacked on top of each other. The stacked dice are connected using through silicon via (TSV) and finally packaged. Stacked dice should match in their performance. Thus dice are binned into performance bins before stacking. This problem is similar to the speed-binning problem.



Speed binning has been studied in the context of binning microprocessors, DSP etc. We address the problem of test time optimization for speed binning. The motivation is to reduce test cost.

It is difficult to find published papers on binning. The standard methodology for speed binning is to use max frequency search. It uses at-speed manufacturing tests to search for the max frequency. Tests used are system tests for system assembly, functional tests for microprocessors, and structural tests for ASICs. The search for max frequency is very time consuming and smaller speed-binning subset of the tests is used to reduce the test time. This paper proposes an alternative approach to speed binning that reduces test time without sacrificing accuracy.

We first investigate using on-chip process monitor (PM) data for max frequency prediction. PMs have been proposed for a variety of applications [1][2][3]. It has also been proposed for speed characterization [4].

For process characterization [4] process monitor accuracy may be adequate. Our interest is in its use in high volume manufacturing for mature processes for which the accuracy requirement are very different. In Section II silicon evaluation of several algorithms using process monitor reading only are presented. Data shows that the purely process monitor based algorithm needs correction, even if we increase the bin size to 10% of the base frequency.

A hybrid flow combining process monitor reading with search using manufacturing tests is presented in Section III. Such an approach has also been used for voltage binning [6]. In Section IV silicon evaluation of the hybrid algorithm is presented. Result show that the hybrid algorithm reduces test time drastically without sacrificing accuracy.

II. PROCESS MONITOR BASED METHODOLOGY

ASICs have inbuilt process monitors (PMs) and a variety of them that have been discussed in the literature. In this paper we assume that PMs are ring-oscillator driving the clock pin of a counter for a fixed duration of time. This count C is read out during test. The value of C for a faster die is larger than the value of C for a slower die. A normalized value of C, given by K = Q/C, is used in our study. The value of Q is a constant and chosen such that K is close to 1 for a nominal die. If K is small (large) then we have a fast (slow) die. The inverters in the ring oscillator used in this study uses nominal V_T transistor cells. We refer to these PMs as NOMY and use the K values from NOMY.

NOMY is used to determine the max frequency of a die as follows. First a model for mapping K to F_{MAX}, the max frequency, is developed using a small sample of the die, as described below in subsection A. This model is modified to determine the speed bin for a die as described in subsection B. The binning algorithm so derived is used to bin dice flowing through the production flow. Silicon evaluation of this PM based flow is presented in subsection C.

A. A First Pass Process Monitor Based Algorithm

The first algorithm builds a model directly mapping the K value to the F_{MAX} value for each clock domain. We use a sample of dice, with different K values, and determine the F_{MAX} of each die.

To build this model a golden reference for F_{MAX} of a die is required. If systems with sockets are available system tests can be used to determine F_{MAX}. For ASICs this poses a problem. ASIC vendors often do not have access to socketed systems. Also, ASICs are often used in different end products. In such a scenario the model will have to be built by the ASIC vendor for each of these end systems.

Functional manufacturing tests to determine the golden F_{MAX} data. Functional test suites are typically not part of the ASIC manufacturing test suite and are more appropriate for microprocessors and DSP chips for which such test suites are available.

1081-7735/11 $26.00 © 2011 IEEE
DOI 10.1109/ATS.2011.96

For ASICs used in our study F_{MAX} was determined using at-speed manufacturing structural tests.

Let $T = \{T_1, T_2 \ldots T_N\}$ be the set of all TDF and PDF tests for a clock domain C. Let $F(d, T_j)$ be the maximum frequency at which die d passes test T_j. Then, $\boldsymbol{F_{MES}(d, C)}$, the measured maximum frequency for die d for clock domain C is $Min\{F(d, T_j): T_j . T\}$. Here, we assume that all the max frequency measurement is done at nominal voltage. Lower operating voltages can also be used, based on the end application of the die.

Table 1

	Sample (385)	Total (3986)
LOT1	10	118
LOT2	75	764
LOT3	75	773
LOT4	75	784
LOT5	75	783
LOT6	75	764

In a production environment we assume that an initial set of die is used to build such a model. In our study we use the sample shown in Table 1. The total dice population was 3986 dice which were picked from different lots spanning the process spread. About 10% of the die, selected randomly from each lot as shown in Table 1, was used to build the model.

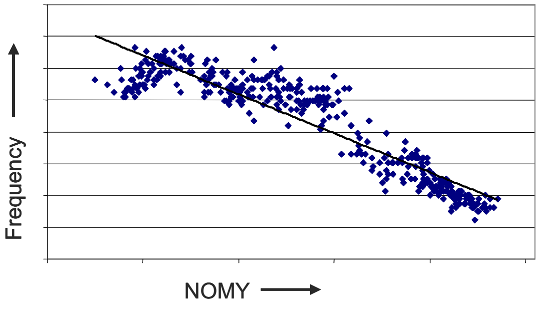


Figure 2. Model for FAST

This design had three clock domains, FAST, SLOW and VERY SLOW. This study built two models, one each for FAST and SLOW clock domain, to understand the accuracy of the process monitor data. Simple linear models, shown in Figure 2 and Figure 3, which fit the NOMY reading with the F_{MES} data were used. The model for FAST was

$$\boldsymbol{F_{CALC}} = -122.75 * K + 539.45; R^2 = 0.854 \qquad (1)$$

The model for SLOW was

$$\boldsymbol{F_{CALC}} = -211.99 * K + 472.71; R^2 = 0.7968 \qquad (2)$$

An important question is: How does F_{MES} compare with F_{CALC}. We next investigate that question.

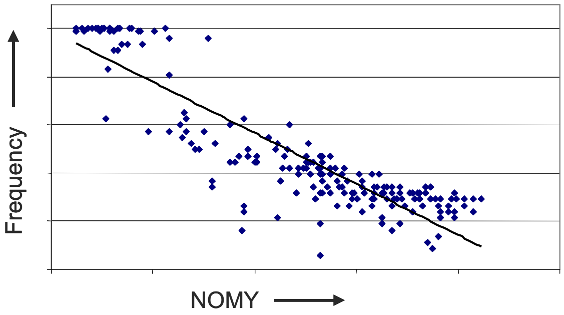


Figure 3. Model for SLOW

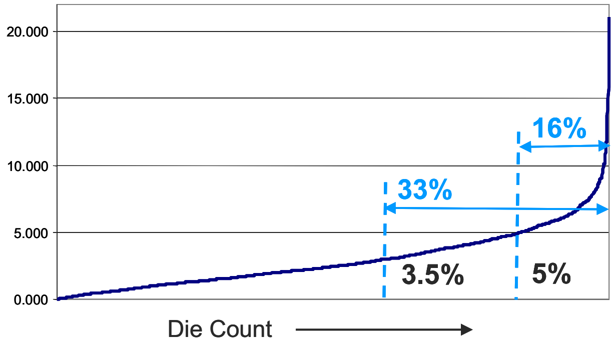


Figure 4. Absolute error for FAST

Errors observed for this sample data is analyzed in two different ways. For FAST we plot the absolute difference between FCALC and FMES in Figure 4. The plot is for the entire die population not just the sample used to build the model. If we target 5% accuracy then 16% of the die population does not satisfy the accuracy requirement. If we increase the accuracy requirement to 3.5% then 33% of the die population does not meet the requirement.

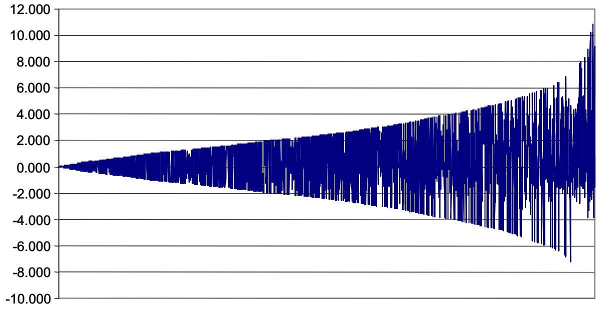


Figure 5. Error profile for FAST

In Figure 5 and Figure 6 we plot the difference between F_{CALC} and F_{MES} for the two clock domains. The most important message from these two figures is that the difference between F_{CALC} and F_{MES} can be both positive and negative. The second message is that, for some die, the error can be rather large.

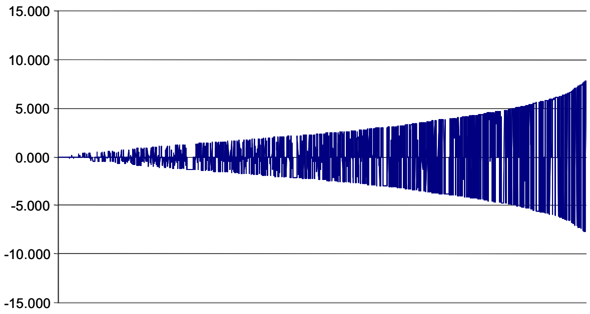


Figure 6. Error profile for SLOW

B. *The Final Process Monitor Based Algorithm*

We next investigate a different algorithm that attempts to cancel out the errors across clock domains. For that we formally define the notion of a bin.

Base frequency of a clock domain is the target frequency for that clock domain. For simplicity, assume two clock domains C1, C2 and with base frequency F1, F2.

The frequency range for a bin is based on the bin size as a percentage of the base frequency. If the bin size is m% of the base frequency, then the frequency range of the bins, for clock domain C1, is as defined below.

$BIN0_{F1}$ = {F1, (1+0.01*m)*F1} (3)
$BIN1_{F1}$ = {(1+0.01*m)*F1, (1+0.02*m)*F1}
$BIN2_{F1}$ = ...

Similarly, the frequency range of the bins, for clock domain C2, is as defined below.

$BIN0_{F2}$ = {F2, (1+0.01*m)*F2} (4)
$BIN1_{F2}$ = {(1+0.01*m)*F1, (1+0.02*m)*F1}
$BIN2_{F2}$ = ...

The frequency range for the rest of the bins can be similarly defined.

Define the measured F_{MAX} of a die *d* for these clock domains to be $F1_{MES}(d)$, $F2_{MES}(d)$. Define the calculated F_{MAX} of a die d for these clock domains to be $F1_{CALC}(d)$, $F2_{CALC}(d)$. Recall that the calculated F_{MAX} is the F_{MAX} estimated using the NOMY data and models similar to equations (1) and (2).

For a given die *d*, let $BIN_{F1}(MES, d)$, $BIN_{F2}(MES, d)$ be the bins for clock domains *C1, C2* determined using $F_{MES}(d)$. Similarly, let $BIN_{F1}(CALC, d)$, $BIN_{F2}(CALC, d)$ be the bins for clock domains *C1, C2* determined using $F_{CALC}(d)$. Then,

$BIN(MES, d) = min\{BIN_{F1}(MES, d), BIN_{F2}(MES, d)\}$ (5)
$BIN(CALC, d) = min\{BIN_{F1}(CALC, d), BIN_{F2}(CALC, d)\}$ *(6)*

Equation (5) gives the measured bin for die d, whereas equation (6) gives the calculated bin for die d.

The new PM based binning algorithm uses the following steps. We use a sample *Q* of dice in the steps outlined below.

- Use this sample *Q* to derive models similar to equations (1), (2).
- For each die *d* in *Q*, for each clock domain, compute F_{CALC} using models similar to equations (1), (2).
- For each die *d* in *Q*, calculate *BIN(CALC,d)* using equation (6)
- For each die *d* in *Q*, calculate *BIN(MES,d)* using equation (5)
- For the set of points { *(BIN(CALC,d), BIN(MES,d)): d in Q}* find the best fit linear correlation. Non-linear models could be used but the linear model was found to be adequate. Let the model derived be

*BIN(MES, d) = A * BIN(CALC,d) + B* *(7)*

Here *A, B* are constants. Since *BIN(CALC,d)* can be calculated from F_{CALC} from equations similar to (1), (2), equations (1), (2) and (8) is a model for calculating the bin from the process monitor reading K.

For the sample data of Table 1 we derived the model using the above steps. It turned out that we had a perfect correlation with A = 1 and B = 0, when we used 5% as well as 10% for the bin sizes. That is,

BIN(d) = BIN(CALC,d) *(8)*

Note that in spite of randomness in the process monitor data the binning data had a perfect correlation.

C. *Silicon Evaluation of the Process Monitor Based Algorithm*

We next evaluate the effectiveness of the above model. Calculations and measurements are done on the rest of the dice which are not used to build the model. We pick a value of *m*, i. e. the bin size, and calculate the value of *BIN(d)* using equations (1), (2) and (8). For each die, and each clock domain C we measure $F_{MES}(d, C)$. From the $F_{MES}(d,$ C) we derive the value of *BIN(MES, d)* using equations (3), (4).

We next calculate and plot the difference given by equation (9).

DIFF(d) = BIN(MES, d) – BIN(d) *(9)*

The difference plot derived using equation (9), for two bin sizes, viz. 5% and 10%, for the data of Table 1 are shown in Figure 7 and Figure 8. In Figure 7 and Figure 8 the X-axis represents the NOMY reading in increasing order. Note that in spite of the perfect correlation while building the model there are a fair number of die for which the model given by equation (9) fails to predict the correct bin.

There are a number of observations to be made. The error could be both positive and negative and it is equally likely. In addition, the error is spread across the entire process range.

However, the data shows that the new algorithm is pretty close to predicting the right bin. Most of the time the calculated bin matches the measured bin or the two of them differ by one. We use this observation in the hybrid algorithm described in Section III.

III. A Hybrid Algorithm for Speed Binning

As shown in Figure 9, the algorithm first predicts the bin using NOMY and refines it using structural test. We first explain the notations.

Assume *M* clock domains C_1, C_2... C_M with base frequency F_1, F_2... F_M. The bin size is the percentage *m* of the base frequency. From this the frequency range of the bins in the respective clock domain is defined as in equations (3), (4). Notation *[f_J] of BIN_J* denotes the vector:

$[f_J] = \{f_1, f_2 ... f_M\}$, where (10)

$f_P = F_P * (1 + J * 0.01 * m); J = 1, 2 ...$ and $P = 1, 2 ... M$

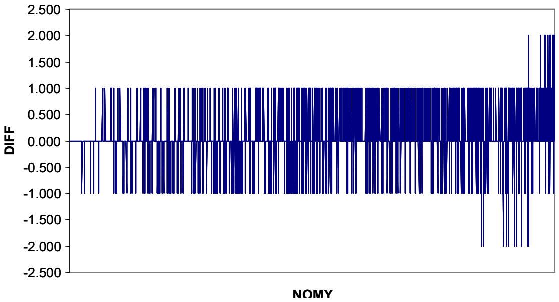


Figure 7. Bin difference plot for bin size of 5%

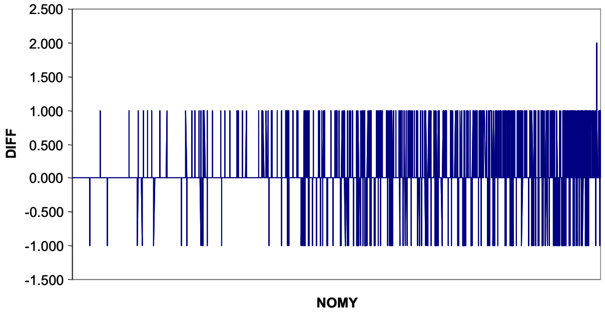


Figure 8. Bin difference plot for bin size of 10%

A die belongs to BIN_J if the die passes all tests for clock domain C_P applied at frequency f_P, defined by equation (10), where $P = 1, 2... M$. The correct bin for a die is BIN_J iff the die can belong to BIN_J but cannot belong to BIN_{J+1}.

We use the term "pass @ [f_J]" for test block S_J. Let the clock domain for test block S_J be C_P. Given this, the term implies that tests for clock domain C_P was run at frequency f_P and the die passed the test.

Similarly, we use the term "fail @ [f_J]" for test block S_J. Let the clock domain for test block S_J be C_P. Given this, the term implies that tests for clock domain C_P was run at frequency f_P and the die failed the test.

The term "Increment [f_J]" implies that $[f_J]$ is replaced by $[f_{J+1}]$ and the next round of test will try to ascertain whether or not the die belong to BIN_{J+1}, i. e. one bin higher than BIN_J.

The term "decrement [f_J]" implies that $[f_J]$ is replaced by $[f_{J-1}]$ and the next round of test will try to ascertain whether or not the die belong to BIN_{J-1}, i. e. one bin lower than BIN_J.

In the first step Figure Figure 9 of we determine an initial bin BIN_{INIT} based on the models described in Section II. This initial bin, BIN_{INIT} is close but may be higher or lower than the correct bin. We next perform a local search to determine the correct bin starting from BIN_{INIT} as described below.

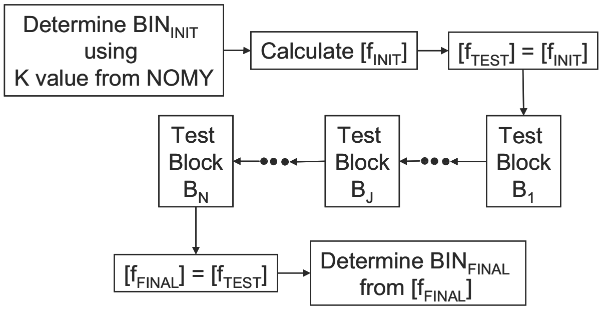


Figure 9. Optimal Hybrid Algorithm

The local search for the correct bin is done using structural tests. Structural tests differ in their effectiveness in determining the correct bin. If a test block *S* is the best structural test block to determine the correct bin then we use that block first in the test flow. When we say that *S* is the best test block it implies that for the largest number of dice, compared to the other test blocks, the bin determined by *S* is the correct bin for that die. We assume N structural test blocks and $<S_1, S_2... S_N>$ is the optimal ordering of these tests. Here, S_I is more effective than S_J provided $I < J$. The die under test passes through the set of test blocks in the order $B_1, B_2... B_N$. Test block B_j uses structural test S_j.

The intuition behind this is that if we determine the correct bin using the most effective test block *S* first then we will, with a high probability, have to run the rest of the structural test blocks only once to determine the correct bin. This one pass through all the structural tests is needed in any case since we have to certify that the die is defect free and runs at the identified bin frequency.

The methodology to determine the optimal ordering of the set of structural tests $S = \{T_1, T_2... T_N\}$ is as follows. The same sample Q of die used to build the model described in Section **Error! Reference source not found.** is used for this purpose. Assume *m* to be the bin size. For each die d in Q, measure $F_{MAX}(T_j, d)$ the maximum frequency determined by a search using structural test T_j. From that we compute $BIN(MES, T_j, d)$ the measured bin using structural test T_j. $BIN(MES,d) = min \{ BIN(MES, T_j, d) \}, j = 1, 2, ..., N$. If T_j determines $BIN(MES,d)$ for most of the dice d in Q then T_j is the most effective structural test for determining the correct bin and we set S_1 to T_j. The next most effective structural test is set as S_2 and so on.

In our experiment we used a slight variation of the above algorithm to determine the ordering of the structural tests. For each test we calculated the average bin number given by equation (11). In equation (11), $|Q|$ is the cardinality of the sample set of dice Q.

$$BIN_{AVE}(T_j) = \left(SUM_{d \text{ in } Q} \{BIN(MES, T_j, d)\}\right)/|Q| \quad (11)$$

Figure 10 processes block B_1, which uses the most effective test block S_1. Assume test S_1 is in clock domain C_P. Then f_P, which is a component of $[f_{TEST}]$, is the frequency at which the die is tested using test S_1. If the die passes the test then the PM based algorithm have determined a bin that is lower than the correct bin.. We check this and, if required, increase the bin. This is done by repeatedly increasing the test frequency and hopping to the next higher bin until the test fails. At this point the test frequency is decreased to the previous passing frequency. This now becomes the starting test frequency for the next block.

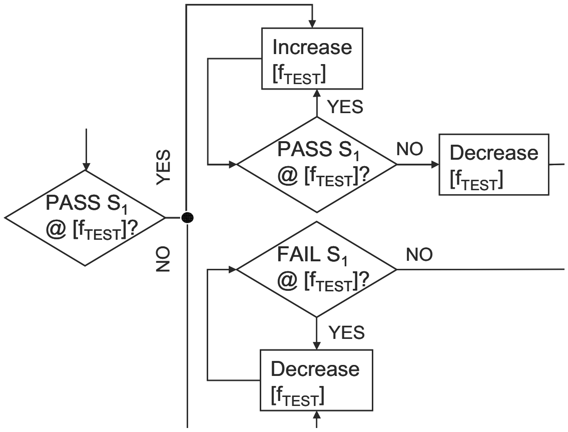


Figure 10. Details of block B_1

On the other hand, if the PM based algorithm determines the bin to be higher than the correct bin for the die then test S_1 will fail. In that case, the bin calculation needs to be corrected by reducing the test frequency by repeatedly hopping to the next lower bin until the die passes test S_1. The first passing frequency is the starting test frequency for the next block.

Details of the rest of the blocks B_J that uses structural test S_J is shown in Figure 11. This flow is quite different from the one for block B_1. The reason for this is that the correct bin will never be higher than the bin computed by the previous blocks and can only decrease.

The above algorithm can be further by making it more adaptive. For each block we keep count of the number of dice that goes through the fail loop and lead to a decrease in the value of $[f_{TEST}]$. Let those values be $X_1, X_2... X_N$. At periodic intervals we inspect these count values and reorder the test blocks depending on the relative values of these counts. The block with the largest count value goes to the front of the test flow and the block with the lowest count value is put at the end of the test flow.

We evaluated the non-adaptive optimal algorithm using the silicon data of Table 1. We compare its performance with three versions of purely structural test base binning: VERY_SIMPLE, SIMPLE and SIMPLE_ORDERED.

A. *Structural Test Based Binning Algorithm*

VERY_SIMPLE. For each die d run each structural test T_J starting with the base frequency for that clock domain. If d passes the test we increase the test frequency to the next bin and re-test d at the higher frequency. If it passes we continue the process after increasing the test frequency one more notch to the next bin. If the test fails we terminate the process for test block T_J and note the last passing frequency and the corresponding bin *BIN(MES,T_J,d)*. Finally, *BIN(d) = min (BIN(MES,T_1,d), ..., BIN(MES,T_N,d))*.

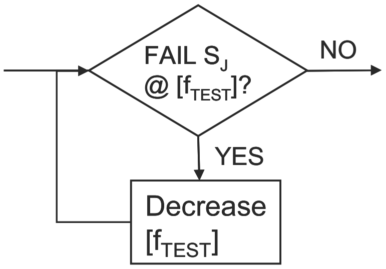


Figure 11. Details of block B_J

IV. Experimental Results

SIMPLE. This is also a purely structural test based approach. In VERY_SIMPLE we determine the bin of each die d using a structural test independent of the bin that was determined by the structural tests processed prior to it. In SIMPLE, for T_1 we determine bin BIN_1 as in VERY_SIMPLE. The frequency $[f_1]$ corresponding to BIN_1 is the starting frequency for test T_2. Starting from this, the bin is lowered for test T_2 while test T_2 keeps failing in a manner similar to the process described in Figure 11. At the end of this we end with bin BIN_2. The frequency $[f_2]$ corresponding to BIN_2 is the starting frequency for test T_3. This process continues till we have completed processing the last structural test T_N.

SIMPLE_ORDERED. This algorithm is similar to algorithm SIMPLE with the difference that the structural tests are processed in the order of their effectiveness. The set of structural tests $S = \{T_1, T_2... T_N\}$ is first ordered with respect to their effectiveness as was discussed in the context of the optimal hybrid algorithm that uses the process monitor data.

A. *Silicon Results*

For the silicon data of Table 1, we had 9 structural tests *{T1, T2, T3, T4, T5, T6, T7, T8, and T9}*. These tests were ordered using the sample used to build the models of Section II. The average bin size computed by these tests for bin sizes given by 5% and 10% are shown in Table 2.

Table 2. Average bin data for the sample subset of dice

	T1	**T2**	**T3**	**T4**	**T5**	**T6**
5%	11.48	11.47	10.27	11.44	11.48	11.43
10%	5.72	5.72	5.18	5.71	5.72	5.72
	T7	**T8**	**T9**			
5%	11.46	11.17	6.82			
10%	5.72	5.61	3.57			

From this table, for both 5% and 10% bin sizes, the ordering of the tests derived was: *{T9, T3, T8, T6, T4, T7, T2, T1 and T4}*.

Table 3. Average bin data for the entire set of dice

	T1	**T2**	**T3**	**T4**	**T5**	**T6**
5%	11.52	11.51	10.19	11.43	11.51	11.35
10%	5.75	5.74	5.09	5.72	5.74	5.68

	T7	T8	T9			
5%	11.51	11.35	5.08			
10%	5.74	5.68	3.48			

We use t_j to denote the test time for structural test T_j. The average bin size for the entire population of dice for VERY_SIMPLE is tabulated in Table 3. The number of times each test is executed is one more than the average size. The total time for VERY_SIMPLE is given by equations (12), (13) for 5% and 10% bin sizes respectively.

VERY_SIMPLE(5%) = $12.48 * t1 + 12.47 * t2 + 11.27 * t3 + 12.44 * t4 + 12.48 * t5 + 12.43 * t6 + 12.46 * t7 + 12.17 * t8 + 7.82 * t9$ *(12)*

VERY_SIMPLE(10%) = $6.72 * t1 + 6.72 * t2 + 6.18 * t3 + 6.71 * t4 + 6.72 * t5 + 6.72 * t6 + 6.72 * t7 + 6.61 * t8 + 4.57 * t9$ *(13)*

SIMPLE uses knowledge of the bin determined by the previous test. We use the random ordering *T1... T9*. The untime is given by equations (14), (15) for two bin sizes.

SIMPLE(5%) = $12.48 * t1 + t2 + 2 * t3 + t4 + t5 + t6 + t7 + t8 + 4.65 * t9$ *(14)*

SIMPLE(10%) = $6.72 * t1 + t2 + t3 + t4 + t5 + t6 + t7 + t8 + 3.15 * t9$ *(15)*

Note that by forwarding the knowledge of the bin computed in the previous step the search time for the correct bin has reduced considerably. The next algorithm uses the optimal ordering determined above. The test time for this algorithm are given by equations (16) and (17).

SIMPLE_ORDERED(5%) = $7.82 * t9 + t3 + t8 + t6 + t4 + t7 + t2 + t1 + t5$ *(16)*

SIMPLE_ORDERED(10%) = $4.57 * t9 + t3 + t8 + t6 + t4 + t7 + t2 + t1 + t5$ *(17)*

Note that for the silicon data we used one test, viz. t9 was extremely effective in determining the correct bin for the entire set of dice. As a result, once that pattern was used to determine the bin all other tests were used only once. This may not occur all the time and the gains observed in equation (16), (17) may not be as significant for other test cases.

The optimal algorithm, that uses knowledge of the process monitor data attempts to reduce this further. The test times for the optimal algorithm are given by equations (18), (19).

OPTIMAL(5%) = $1.25 * t9 + t3 + t8 + t6 + t4 + t7 + t2 + t1 + t5$ *(18)*

OPTIMAL(10%) = $1.19 * t9 + t3 + t8 + t6 + t4 + t7 + t2 + t1 + t5$ *(19)*

If we compare equation (18) with equation (16) the gain in test time is in the number of times test T9 is used on an average once the process monitor reading is used for an initial guess. Correction is required for about 20 to 25% of the die and one or two extra passes are adequate for correction. Similarly, the gain between equations (17), (19) should be fairly obvious. In any manufacturing flow each test will have to be used at least once in order to certify that the part is operational at the stated bin frequency. From equations (18), (19) we note that except for T9 all other patterns are executed only once. So, the added test time for finding the correct bin is 0.25*t9 and 0.19*t9 respectively for the 5% and 10% case.

V. Summary

In this paper we studied the problem of determining the correct bin using process monitor data. An algorithm based on process monitor data only was presented, analyzed and evaluated using silicon data. Results show that this algorithm requires some refinements. A hybrid algorithm that combines the process monitor based algorithm with manufacturing test was presented. We presented silicon data that showed that this algorithm improved the test time when compared with other purely manufacturing test based algorithms. One improvement that can be made in a future study is to collect similar results where max frequency search using memory tests are also included.

Acknowledgment

We acknowledge the contribution of Joel Lurkins and AJ Haas for providing us with the max frequency measurements

References

[1] K. Brand, S. Mitra, E. Volkerink, E. McCluskey, "Speed Clustering of Integrated Circuits", in *Proceedings 2004 IEEE International Test Conference*, Oct. 2004, pp. 1128-1137.

[2] Lohit *S.* Dutta and Thomas Hillmann-Ruge, "Application of Ring Oscillators to Characterize Transmission Lines in VLSI Circuits", *IEEE TRANSACTIONS ON COMPONENTS, PACKAGING, AND MANUFACIZTRING TECHNOLOGY-PART B*, VOL. 18, NO. 4, NOVEMBER 1995.

[3] Benjamin Jun and Paul Kocher, "The Intel Random Number Generator," white paper 1999, [Online] http://cryptography.com/resources/whitepapers/IntelRNG.pdf

[4] Linda Milor, Larry Vu, and Bill Liu, "LOGIC PRODUCT SPEED EVALUATION AND FORECASTING DURING THE EARLY PHASES OF PROCESS TECHNOLOGY DEVELOPMENT USING RING OSCILLATOR DATA," *2nd International Workshop on* Statistical Metrology, 1997 pp. 20 – 23

[5] Niiyama, T.; Piao Zhe; Ishida, K.; Murakata, M.; Takamiya, M.; Sakurai, T, "Dependence of Minimum Operating Voltage (V_{DDmin}) on Block Size of 90-nm CMOS Ring Oscillators and its Implications in Low Power DFM" *9th International Symposium on Quality Electronic Design*, 2008, pp.133 – 136

[6] N. Parris, J. Healey, C. Hawkins, "A Simple Approach to Diagnose Localized Thermal and IR Drop Effects on a Microprocessor Core Using On- Chip Synthesizable Ring Oscillators", *Proceedings of the First Silicon Debug and Diagnosis Workshop*, Session 3.1, May 2004.

[7] J. Rearick, "Calibrating Clock Stretch during AC Scan Testing", in *Proceedings 2005 IEEE International Test Conference*, November 2005.

[8] S. Chakravarty, B. Dang, D. Escovedo, A. J. Haas, "Optimal Manufacturing Flow to Determine Minimum Operating Voltage," to appear, *IEEE International Test Conference, 2011*.