WCOMP: Waveform Comparison Tool for Mixed-signal Validation Regression in Memory Design

Peng Zhang
Intel Technology Development Co., Ltd.
ASIC & System Laboratory
Fudan University
Shanghai, P. R. China
peng.zhang@intel.com

Wai-Shing Luk, Yu Song, Jiarong Tong,
Pushan Tang, Xuan Zeng
ASIC & System State-Key Laboratory
Fudan University,
Shanghai, P. R. China
{luk,songy,jrtong,pstang,xzeng}@fudan.edu.cn

Abstract— The increasing effort on full-chip validation constrains design cost and time-to-market. A waveform comparison tool named WCOMP is presented to automate mixed-signal validation regression in memory design. Unlike digital waveform comparison tools, WCOMP compares mixed-signal waveforms for functional match instead of graphical match, which tally with the requirements of full-chip validation regression. Simulations with different regression runs, process parameters, voltages and temperatures can be functionally compared. The methods are proved to be effective in Intel® Flash memory design.

I. Introduction

The increasing memory density and complexity continuously boost the attentions of improving design productivity to lower design cost and shorten time to market. One of the bottlenecks in flash memory design is full-chip validation [1] due to the following reasons. First, simulation of large memory array and control circuit inevitably requests many computing time. More importantly, hundreds of full-chip mixed-signal simulation waveforms have to be manually checked to ensure full-chip functionality including accuracy of voltage placements. Furthermore, additional full-chip regression runs at the succeeding design stage for validating small design edits multiply the efforts.

Similar case happens in performance simulations, where people simulate smaller scale circuitry along critical path to analyze performance specifications. It requests thousands of simulations for worst-case analysis with different process parameters, supply voltages and temperatures (PVT). To check whether the performance of analog function degenerates under different PVT corners largely relies on designer's manual work. As a result, full-chip validation and performance validation may take up to 70% of design loading in a product development.

There are several innovative solutions published to speed up simulation time with tolerable accuracy; see for example [2]. However, tools for functional auto-checking is largely limited to digital design. One doable way to assist analog/mixed-signal validation is waveform comparison. Currently, there are several waveform comparison tools such as ModelSim [3], SPICE-explorer [4] and WaveFormer [5]. However, their support in analog signal is very limited. More importantly, all the above waveform comparison tools are simply to perform exact comparison and then detect waveform differences from its graphical content. Nevertheless, many waveform differences are caused by slightly different set of stimuli, PVT's and clock cycle time. Therefore, existing comparison method are only useful for debugging, but not for validation regression auto-checking. To aid auto regression, we need to screen out all false alarms in which differences are not due to circuit functional errors. In this paper, we take an Intel® flash memory design as a case study. We consider the following cases as false alarms:

Case 1 : Differences of gate loading, transistor size and delay between two simulation runs introduce time offset during signal ramping.

Case 2 : Delay differences of clock synchronized events introduce mismatches between two waveforms. A mixed-signal design usually has a microprogrammed control unit (MCU) to control full-chip operations including analog blocks, such as state machines for flash memory read and write operations. For instance, we change the clock cycle time or insert "no-op" instructions in MCU control code between two simulation runs for timing adjustment. In this case, a trivial waveform comparison will report many false alarms.

Case 3 : Simulations under different PVT corners introduce many waveform differences of supply voltages, pulse widths and delays even though they look like each other and has same function.

1-4244-0630-7/07/$20.00 ©2007 IEEE.

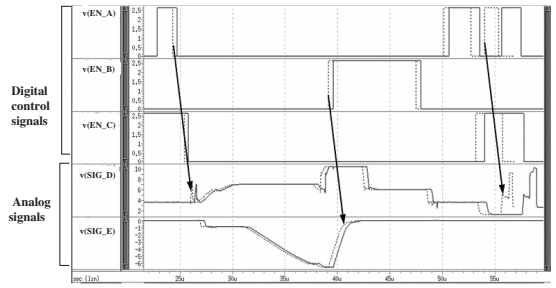
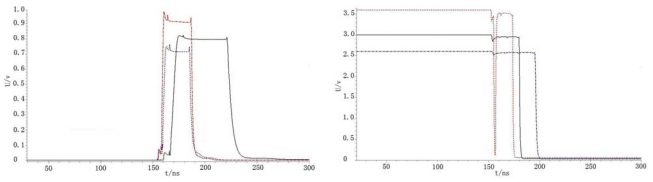


Fig. 1. The sim-comparison. Black arrows denote the enable signal shift that caused the analog waveform shift.



(a) Same signal under different PVT corners. (b) An error case with one signal has an unexpected glitch.

Fig. 2. Pattern-comparison.

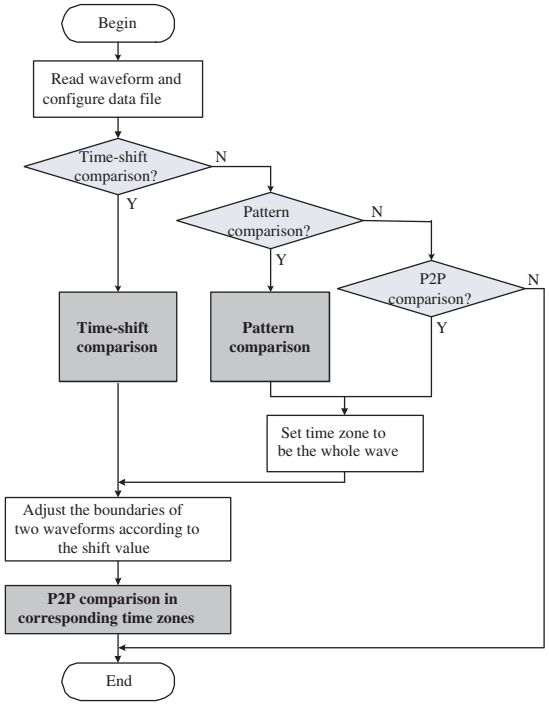


Fig. 3. Flow chart of waveform comparison tool WCOMP

In this paper, we present a waveform comparison tool named WCOMP that aims to automate analog simulation regression check and screen out all false alarms in the above cases. It tolerates gate delays, aligns analog signal with synchronous events and linearly transforms waveform respectively according to the above three cases. To the best of our knowledge, this study is the first attempt of mixed-signal waveform comparison for computer-aided validation.

WCOMP mainly consists of two components: *sim-comparison* and *pattern comparison*. The sim-comparison compares one simulation with different runs of regression regarding to analog/digital signal relevance in synchronous events, which handles Case 1 and Case 2. It is further divided into *point-to-point (p2p) comparison* and *time-shift comparison*. In considering Case 3, Pattern comparison is proposed for comparing simulations under different PVT corners and for judging the similarity of two waveforms.

Fig. 1 gives an example of the sim-comparison. Golden waveform (i.e., pre-examined waveform, solid line) and reference waveform (i.e., waveform under comparison, doted line) exhibit same analog function for SIG_D and SIG_E with several time-shift events caused by their digital control signals of EN_A, EN_B and EN_C. The time-shift comparison together with the p2p comparison will detect those shifts and compare the analog function (SIG_D & SIG_E) based on the shifting information. In a real memory design, there maybe up to 600 control signals for hundreds analog block.

Fig. 2 shows two examples of the pattern comparison. In Fig. 2(a), three waveforms from three different PVT simulations have different voltages, pulse widths and time delays but they are similar to each other and are considered as a functional "match". Fig. 2(b) illustrates a case that one of those three waveforms has a glitch. In this case it is considered as a "functional error".

The rest of this paper is organized as follows. Section II gives a overview. Sections III, IV and V will give the detail descriptions of the point-to-point (P2P), time-shift and pattern comparison respectively. Experiment results are presented in Section VI to illustrate the effectiveness of this tool.

II. Overview

The main structure of WCOMP is illustrated in Fig. 3. The key features as well as the control flow of the tool are described as follows:

1. WCOMP has an input interface for reading in data files that contain the waveforms, and a configure file where comparison parameters are defined. Configure file defines the option of comparison types (point-to-point, time-shift and pattern comparison) and names of signals to be compared and other options such as tolerance.

2. If the "point-to-point comparison" option is defined, the tool will only perform point-to-point comparison with user-defined tolerance and output the comparison results.

3. If the "time-shift comparison" option is defined, the tool will perform the time-shift comparison. First, it detects the time-shift events between two waveforms and divides both golden and reference waveforms into different time zones. Then the point-to-point comparison is used to compare the waveforms for each time zone.

4. If the "pattern comparison" option is defined, pattern comparison module will be used to extract the commonality of waveforms in different PVT.

III. Point-to-point comparison

Point-to-point comparison is a basis functionality that is often called by other components. It simply determines the graphical differences of two waveforms. To deal with time off-set of Case 1, a time tolerance window is defined by voltage tolerance and time tolerance. See Fig. 4 as an example. Note that the values of tolerances are usually given by user as an input.

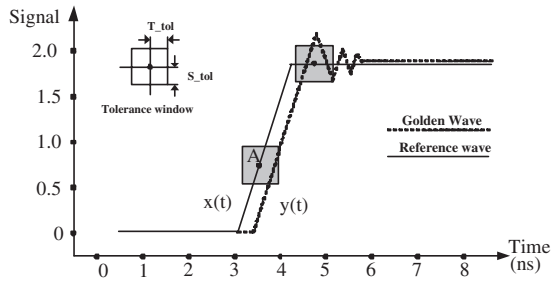


Fig. 4. Point-to-point comparison with time tolerance window

Time tolerance window is a rectangle with height of voltage tolerance (denoted by S_tol) and width of time-tolerance (denoted by T_tol). Its center is the data point of a golden or reference waveform. If any line segment of the other waveform intersects with this window, we consider two waveforms as graphical match at this data point. To the best of our knowledge, we can only find a similar algorithm mentioned in [6], in which the tolerance window is a circle. Our analysis shows that rectangle tolerance window has advantage over circle tolerance window as it has linear calculation formula and is more straightforward for user to set time-tolerance parameters.

IV. Time-shift comparison

Time-shift comparison aims to screen out false alarms of synchronous delays. It compares the analog signals when their digital enable signal is shifted for clock cycles. The flow chart of the time-shift comparison is illustrated in Fig. 6(a).

In mixed signal designs [7] [8] including memory designs, analog modules are usually controlled by digital enable signals. Enable signals are controlled by MCU and may have clock-controlled delays due to clock cycle time differences and "no-op" insertions. They are also called *Index signals*. In our cases, enable signals usually have a special prefix such as `INDEX_`. Thus they can easily be identified. Let `N` be the total number of index signals. A *state* is defined as a vector formed by all index signals (`idx_sig0,idx_sig1,...,idx_sigN`). The whole waveform can be divided into different time zones according to state vector changes. The time duration of a given state value is referred to as state time zone (see Fig. 5). The criteria that two waveforms are functionally matched

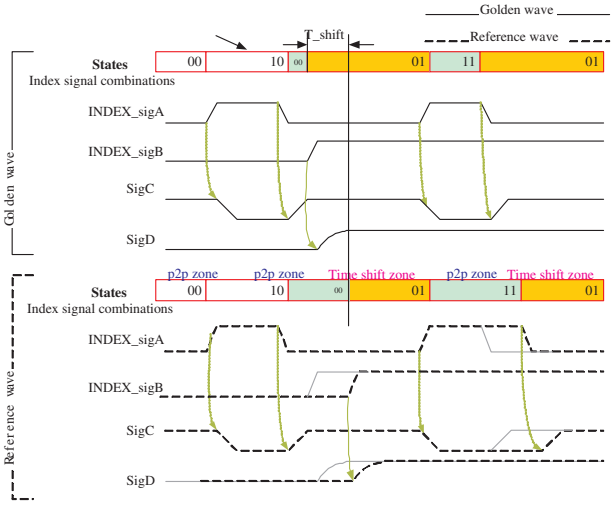
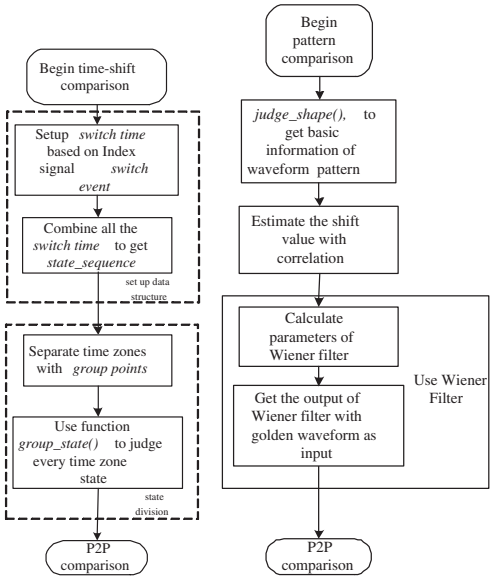


Fig. 5. Time-shift comparison and state definition

is given by, a) States are in same sequences; b) Each state has same analog function after aligning the state start time; c) The duration time of every state in two waveforms can be different as long as the their state sequences are same.

Fig. 6(a) shows the time-shift comparison flow. Note that it is not feasible to store each state vector value directly because there may be up to hundreds of index signals in a real design. To deal with it and save memory usage, an algorithm of detecting state sequence has been proposed and was detailed in [9]. In this algorithm, each index signal initial value and its *switch events* time. are captured. After generating the switching events of each index signal and its own *switching sequence*, we combine switching sequence of all index signals into a *state_sequence*. Each state begin-time and end-time defines a state *time zone*. Comparing each state time zone between golden waveform and reference waveform can determine the state time shift as well as determine whether states are in same sequence.

In a real design, some index signals may switch at the same time within a given time tolerance (see Fig. 7). Also, some index signal shifting may introduce new state, such as state (1,0,1) and (1,0,0) in Fig. 7. We use



(a) time-shift comparison (b) pattern comparison

Fig. 6. Flow chart of the time-shift comparison and pattern comparison

`group_state()` to detect such cases and group the same amount of switch events of golden/reference waveforms. Then `group_state()` will compare current state between two waveforms, for instance, if they contains the same signals within time tolerance) [9]. It returns true or false as a result of state comparison. If it returns false, a state error will be reported. After obtaining the grouped state, we can adjust the boundary of each state and perform the point-to-point comparison for each "TRUE" state time zone. In this way, all state shift errors introduced in directly waveform comparison is tolerated and only real signal behavior differences after state alignment is considered as a mismatch.

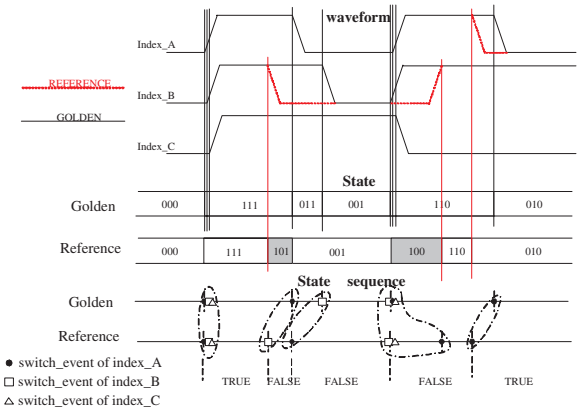


Fig. 7. Switch sequence and state

V. Pattern comparison

In the pattern comparison, we compare the waveforms from different PVT corners for functional check. If two waveforms are in the same shape, they are considered as functional match. Although the problem looks related to pattern recognition, we are cautious when the related techniques are applied for validation. In particular, the suggestion of using learning algorithms was quickly rejected by designers in our earlier development stage. For validation automation, robustness is most concerned, in the sense that designers would prefer the tool misclassifies two waveforms as different even they are in fact acceptable, rather than the tool has chances to misclassify two waveforms as match but they are in fact unacceptable. In the former situation, designers can still make their own judgments by manually debugging, whereas in the later situation, there is probably no chance for the designers to notify the error.

In this paper, we propose a method that utilizes a signal processing technique called Wiener filtering in order to linearly restore the common waveform pattern of signal waveforms under different PVT corners. This technique has been successfully applied in channel equalization, time-delay estimation and noise reduction [10].

Wiener filter is a finite duration impulse response (FIR) filter. Let $y(t)$ be a signal representing the reference waveform and $x(t)$ be a signal representing the golden waveform. The Wiener filter function to transform $y(t)$ is given by:

$$\hat{x}(m) = \sum_{k=0}^{P-1} w_k \cdot y(m-k) = w^T y, \qquad (1)$$

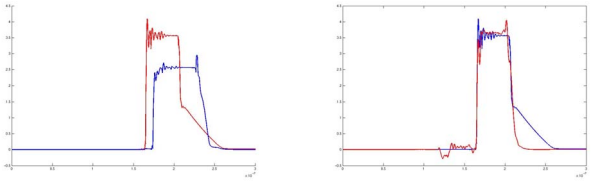
where w_k's are coefficients of Wiener filter and P is the order of the filter. $\hat{x}(m)$ represents the transformed signal at time m.

The error between $\hat{x}(m)$ and $x(m)$ is

$$e(m) = x(m) - \hat{x}(m) = x(m) - w^T y. \qquad (2)$$

Filter coefficients w_k's are calculated by solving the underlying least square linear problem such that $||e(t)||_2$ is minimized. We apply the *Conjugate Gradient Normal Residue* method (CGNR) for solving this least square problem [11]. One of the advantages of using CGNR over the standard QR decomposition method is that the underlying Toeplitz matrix does not need to be explicitly formed.

Fig. 8 shows an example that two waveforms are in different shapes. By using the Wiener Filter technique, we transform the original waveform (in blue color) in such a way that the shape is mostly preserved and the error between the resulting waveform and the transformed waveform (in red color) is minimized in least square sense. After the transformation through a Wiener filter, the point-to-point comparison will be used and reports that which portions of waveform are different.



(a) Original waveforms. (b) Waveforms after the Wiener filtering.

Fig. 8. Example of two waveforms in different shapes

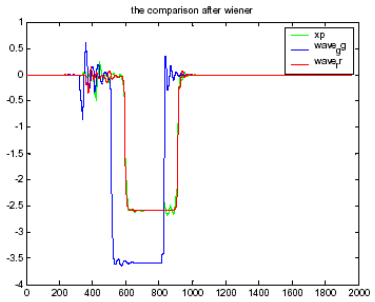


Fig. 9. Signals after transform

The flow chart of the pattern comparison is described on Fig. 6(b). The detail of the pattern comparison is summarized into following steps.

Step 1 : reads in the two waveforms under-compared and take them as two functions $F = x(t)$ and $F^{'} = y(t)$.

Step 2 : the function `Judge_shape()` examine the waveform to judge the basic waveform shape information. If the shape is a DC value, no transform is needed. If the shape is a single ramp (half pulse), we make it to a full pulse for the convenience of transform. If the shape is similar to a pulse, we need to equalize the pulse widths of two waveforms by interpolate the sample data in order to perform linear transform, which is essential as Wiener filter cannot deal with pulse width changes.

Step 3 : Calculate time relationship between two waveform by the correlation between $x(t)$ and $y(t)$ as shown in following equation, The n that maximizes C_{xy} is the time delay between $x(t)$ and $y(t)$.

$$C_{xy}(n) = \sum_{t=-\infty}^{+\infty} x(t)y(t+n) \qquad (3)$$

Step 4 : Use Wiener filter to linearly transform waveform and make two waveforms similar to each other. An example is shown in Fig. 9. The blue signal is transformed into the green signal. We can see that the transformed green signal matches with red signal very well. At last, we apply the point-to-point comparison with two waveforms to determine if they are functionally matched.

VI. Experimental Results

We implemented WCOMP in C++. All experiments were run on an IBM Workstation with 2.4GHz Intel Xeon CPU. Tables I and II show the sim-comparison results we got from WCOMP. In Table I, we performed waveform comparison with different runs of full-chip simulations and two densities (128Mb flash and 32Mb flash)full-chip simulations of a same product. It demonstrates high efficiency and fast speed (less than a minute) to compare up to 1700 signals with 17,000 data points per signal. The comparison speed is mostly proportional to the total numbers of data points for all signals. Table II shows the advantage of time-tolerance and time-shift. In Table II, there were many false alarms without time tolerance and time-shifting that makes directly wave comparison inapplicable for functional check. Adding time-tolerance and time-shifting screened out all those false alarms out. Only user cared functional errors are flagged out.

TABLE I
Comparison between two runs of full-chip simulations and two densities of simulations. RevTest is for different runs and DensTest is for two densities.

Waveform name	No. of Signals	Data points per signal	Run time (sec)	Errors detected	Real Errors
RevTest1	1760	9672	26.41	0	0
RevTest2	1699	17822	59.23	0	0
RevTest3	1699	7944	15.69	0	0
RevTest4	1699	9388	23.94	0	0
DensTest1	1760	9672	28.11	0	0
DensTest2	1699	17822	50.87	1	1
DensTest3	1699	7944	18.85	3	3
DensTest4	1699	9388	19.91	2	2

TABLE II
Comparison of false alarms without tolerance window and time-shift algorithm

Waveform name	Wave differences that detected by different algorithms			
	Without Time tol.	Without Time-shift	Our tool	Manual check
TolTest1	17	800	17	17
TolTest2	30	1153	30	30
TolTest3	6	635	3	3
TolTest4	16	1	1	1
TolTest5	55	6	6	6
TolTest6	22	0	0	0

Table III gives test result of the pattern comparison. There were nine PVT corners for each test. The run time is within 3 minutes to compare up to 24 signals, which met designers' expectations. The comparison speed is mostly proportional to the Wiener filter iteration numbers, as

TABLE III
PATTERN COMPARISONS RESULTS FOR DIFFERENT PVT SIMULATIONS

Test name	Total signal	Iterations of Wiener	Max. run time	Errors detected	Real errors
PatTest1	12	140	173s	1	1
PatTest2	12	50	78s	0	0
PatTest3	12	55	153s	1	1
PatTest4	5	12	1s	0	0

illustrated in Table III. All the errors detected by the pattern comparison are matched with real design errors. Fig. 10 and Fig. 11 show examples of pattern pass and pattern fail.

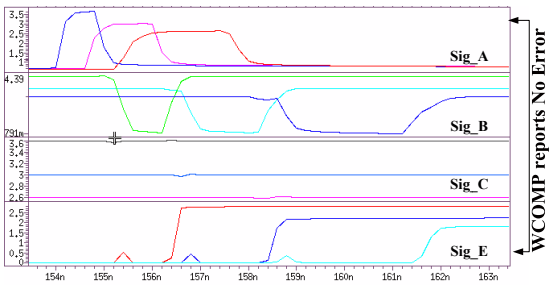


Fig. 10. Pattern compare results (Pass cases for sig_a, sig_b and sig_c)

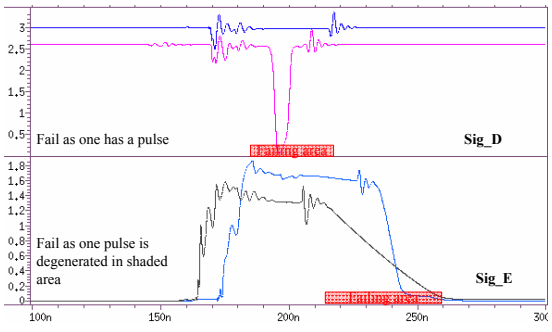


Fig. 11. Pattern comparison results (Mismatch cases for sig_d and sig_e)

VII. SUMMARY

A mixed-signal waveform comparison tool is presented with the sim-comparison for different runs of simulations and the pattern comparison for different PVT corners. This is the first attempt to apply functional match concept in waveform comparison and we have never seen it in previous waveform comparison tools. The functional match method is particularly suitable for full-chip validation regression in Intel® flash memory design practice.

ACKNOWLEDGEMENT

This work was supported by Intel Corporation. In addition, the authors are grateful to the members of Flash Product Group, Intel Technology Development (Shanghai) Ltd., for their suggestions and comments.

REFERENCES

[1] A. Conci et al. Current criticalities and innovation perspectives in flash memory design automation. *Proceedings of The IEEE*, 91, April 2003.

[2] G. G. E. Gielen. CAD tools for embedded analogue circuits in mixed signal integrated systems on chip. *IEE Proc. Computers and Digital Techniques*, 152:317–332, May 2005.

[3] Mentor Graphics Corp. ModelSim. See http://www.model.com.

[4] Sandwork Design Inc. SPICE Explorer. See http://www.sandwork.com/products_spice.htm.

[5] Synapticad Inc. WaveFormer Pro. See http://www.syncad.com/syn_v40.htm.

[6] Y.-C. Ju, V. B. Rao, and R. A. Saleh. Consistency checking and optimization of macromodels. *IEEE Transaction on computer-aided design*, 10(8):957–967, August 1991.

[7] K. Kundert et al. Design of mixed-signal systems-on-a-chip. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1561–1571, December 2000.

[8] H. A. Castro, K. Augustine, et al. A 125MHz burst mode 0.18μm 128Mbit 2 bits per cell flash memory. In *Symposium On VLSl Circuits Digest of technical papers*, pages 304–307, May 2005.

[9] Zhixi Huang et al. A waveform detecting method for extracting mixed-signal circuit states. Technical report, Department of Microelectronic, Fudan University, 2006.

[10] Saeed V. Vaseghi. *Advanced digital signal processing and noise reduction*, chapter 6. Wiley, 2nd edition, 2000.

[11] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.