

國立交通大學

電信工程學系碩士班

碩士論文

可變超取樣率三角積分類比數位轉換器之低面

積降頻器電路設計與實現

Design and implementation of a small area decimator for  
programmable oversampling ratio sigma-delta A/D converters

研究生：唐江俊

指導教授：闕河鳴 博士

西元二〇〇八年九月

可變超取樣率三角積分類比數位轉換器之低面積降頻器電路設計與實現  
Design and implementation of a small area decimator for programmable  
oversampling ratio sigma-delta A/D converters

研究生：唐江俊

Student: Chiang-Chun Tang

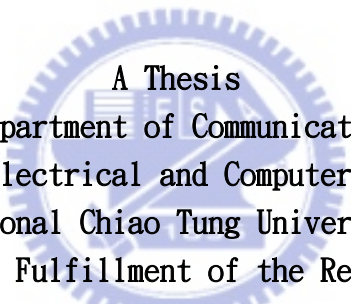
指導教授：闕河鳴 博士

Advisor: Dr. Herming Chiueh

國立交通大學

電信工程學系碩士班

碩士論文



A Thesis  
Submitted to Department of Communication Engineering  
College of Electrical and Computer Engineering  
National Chiao Tung University  
In Partial Fulfillment of the Requirements  
For the Degree of  
Master of Science  
in

Communication Engineering

September 2008

Hsinchu Taiwan

西元二〇〇八年九月

# 可變超取樣率三角積分類比數位轉換器之低面積降頻器電路設計與實現

研究生：唐江俊

指導教授：闕河鳴 博士

國立交通大學

電信工程學系碩士班

## 中文摘要

在很多領域，三角積分類比數位轉換器近來非常受歡迎。而三角積分類比數位轉換器主要由類比電路(三角積分調變器)和數位電路(降頻器)所構成。然而，數位電路部分佔據了整個三角積分類比數位轉換器的絕大多數積體電路面積。而且對於可變超取樣率的三角積分類比數位轉換器而言，需要不同頻寬的數位濾波器去取出所要的訊號頻段，這樣的需求也會導致額外的數位電路面積消耗。

在此，一個針對可變超取樣率三角積分類比數位轉換器的低面積降頻器被設計與實現。而最主要的改良是在於裡頭的高階有限脈衝響應濾波器。對於調換結構並採用多相分解的有限脈衝響應濾波器，可藉由使用摺疊和儲存元件共享技巧，並且在此主要配合使用特別的控制電路去改變計算程序以重複使用儲存元件來達到降低面積的目的。在此提出的電路架構，與廣泛採用的直接結構摺疊架構相比，由於只需使用一半的儲存元件，因此可得到較小的電路面積。此外，在此提出的架構不因節省面積而對電路的其他特性有所損傷，也就是本架構除了面積較小外，關鍵路徑也較短，等待週期也較少並且尖峰功率消耗也較小(平均功率在相同的水平)。

由於只需使用一半的儲存元件，本架構相對於直接結構摺疊架構而言，可使降頻器裡的高階有限脈衝響應濾波器減少 24.6% 的面積，進而達到整體使用四個階段降頻器 15.8% 的面積節省(對於三個階段降頻器，整體面積則可減少 20.9%)。

# **Design and implementation of a small area decimator for programmable oversampling ratio sigma-delta A/D converters**

Student: Chiang-Chun Tang

Advisor: Dr. Herming Chiueh

Department of Communication Engineering  
National Chiao Tung University  
Hsinchu, Taiwan

## **Abstract**

The sigma-delta modulation (SDM) has become a very popular analog to digital conversion technique in many fields. A sigma–delta A/D converter consists of analog circuits (sigma–delta modulator, SDM) and digital circuits (decimator). However, the silicon area of sigma-delta A/D converters is governed largely by the digital parts. Moreover, the distinct bandwidth digital lowpass filters are required to perform selecting-signal for programmable oversampling ratio SDM, which results in extra filters hardware consumption in digital part of SDM A/D converters.

The small area decimator for programmable oversampling ratio SDM A/D converters is designed and implemented. The main improvement in this thesis is focused on the high order FIR filter of decimator. Combing the folding and the storage elements sharing techniques for decimation FIR filters using polyphase decomposition in transposed-form as well as changing the computation procedures mainly to reuse storage elements by using extra control circuits, the area reduction compared with the widely used folded FIR filter architecture in direct-form is obtained due to half storage elements (registers) requirement. In addition, the extra advantages of my proposed folded decimation FIR filter architecture based on transposed-form are shorter critical path, smaller peak power (average power in the same level), and shorter latency.

As a result of half registers requirement, the 24.6% area reduction for high order (126<sup>th</sup>-order) FIR filter is obtained, which result in 15.8% area reduction for the 4-stages decimator (20.9% area reduction for 3-stages decimator).

# *Acknowledgments*

首先，得感謝我的指導教授闕河鳴博士，在進實驗室之初，傳授 VLSI 相關方面的知識，使得學生可以銜接上研究所課程，也因此最後能夠完成在 CIC 的下線。另外，平日報告與開會時，老師也會竭力給予指導與建議以彌補學生此方面能力的不足。除此之外，在論文撰寫期間，老師也花了很多時間精力在協助學生修改論文並提供許多寶貴意見與硬體設備上的支援，使得此論文得以順利的完成，在此由衷感謝。

其次，非常感謝蘇韋力學長、張紹宣學長、蔡佐昇學長，平日在學業上的解惑，幫助本人克服此不熟悉的領域(晶片設計)。

另外，在此枯燥煩悶的研究生生活裡，幸虧有呂秉勳、蘇品翰、游凱迪、賴明君、吳春慧、林信太、吳俊誼、劉嘉儀、林順華學長還有玄奘朋友們的陪伴，使得我三年的研究生生活不感到孤單，謝謝你們。

最後，我得感謝我的父母，無論是精神上或是經濟上都給予最大的支持，使得我無後顧之憂得以盡心完成學業，也由於父母的支持，才讓我能有動力完成這份論文。

唐江俊  
Sep. 2008

# Contents

中文摘要	I
English Abstract	II
Acknowledgments	III
Contents	IV
List of Tables	VI
List of Figures	VIII
List of Abbreviations and Symbols	XIII
<b>Chapter1 Introduction</b>	<b>1</b>
1.1 Motivation	2
1.2 Fundamentals	2
1.2.1 Sampling Theorem	3
1.2.2 Principle of Sigma-Delta A/D Converter	5
1.2.3 Decimator	15
1.3 A Brief Introduction of Proposed Solution	22
1.4 Thesis Organization	23
<b>Chapter2 Decimator Architecture and Design</b>	<b>25</b>
2.1 Considerations about SDM Quantization Noise	25
2.2 Decimator Architecture	27
2.3 First Decimation Stage	29
2.3.1 Introduction to Modified Comb Filters	31
2.3.2 Stage1 Design	35
2.4 Decimation FIR Filters	39
2.4.1 Stage2 Design	40
2.4.2 Stage3 Design	43
2.5 Compensation FIR Filter	46
2.5.1 Stage4 Design	48
2.6 Specification Summary	49
<b>Chapter3 Decimator Implementation</b>	<b>53</b>
3.1 Implementation and Verification Flow	53
3.2 Previous Work Comparison	56
3.2.1 Comb Filter	56

3.2.2 FIR Filter	59
3.3 Overall Decimator System	73
3.4 Clock Divider Circuit	74
3.5 The First Decimation Stage: Comb Filter	75
3.5.1 Gain Control	75
3.5.2 Pipelined Comb Filter	76
3.6 Proposed Circuit for FIR Filters	77
3.7 Comparisons	84
3.8 Implementation Results	90
3.8.1 Pad Assignment	92
3.9 Decimator Simulation Result	93
3.9.1 OSR=128	93
3.9.2 OSR=64	95
3.10 Specification Table	97
3.11 Paper Comparison	98
<b><u>Chapter4 Testing and Measurement Results</u></b>	<b>99</b>
4.1 Introduction to Digital IC Testing using Agilent 93000 in CIC	99
4.2 Shmoo Plot	104
4.2.1 OSR=128	104
4.2.2 OSR=64	107
4.3 Timing Diagram	110
4.4 Measured Power	111
4.5 CHIP Summary	112
<b><u>Chapter5 Conclusions</u></b>	<b>114</b>
<b><u>Appendix A: Filter Coefficients</u></b>	<b>116</b>
<b><u>Appendix B: Test-Patterns for Agilent 93000</u></b>	<b>119</b>
<b><u>References</u></b>	<b>126</b>

# *List of Tables*

Table 1.1 Ideal Peak SNR with 1-bit quantizer (N=1)	14
Table 2.1 Optimized rotated angle $\alpha$ represented by $q$ , which is independent to $f_b$	34
Table 2.2 Ideal in-band quantization noise power	36
Table 2.3 Aliasing power using comb filter and MCFs with different order for 1-bit 4 <sup>th</sup> -order SDM, OSR=128 and OSR=64 (calculating in Matlab)	36
Table 2.4 Quantization noise power which cannot be remove by latter filter	40
Table 2.5 Aliasing power2 with different stop-band attenuation	41
Table 2.6 Quantization noise power in in-band after 2nd decimation stage	44
Table 2.7 Aliasing power3 with different stop-band attenuation (Rs) , given a narrow transition-band (0.02).	44
Table 2.8 FIR orders and aliasing power3 with different widths of transition-band	44
Table 2.9 Pass-band ripple with different order of compensation filter for decimation ratio=128	48
Table 2.10 Pass-band ripple with different order of compensation filter for decimation ratio=64	48
Table 2.11 Summary of decimation filter specifications	49
Table 2.12 Aliasing power, remaining quantization noise power and obtained SNR at each stage for OSR=128	50
Table 2.13 Aliasing power, remaining quantization noise power and obtained SNR at each stage for OSR=64	50
Table 3.1 required word-length for these two structures	57
Table 3.2 The computation of each cycle in Figure 3.22(a)	81
Table 3.3 The algorithmic operations of proposed folded architecture is equivalent to the unfolded decimation FIR filter in transposed-form using polyphase	82
Table 3.4 Comparison of decimator using the described implementations of FIR filter	84
Table 3.5 Detail area comparison of decimator using the described implementations of FIR filters	85
Table 3.6 Area normalized to direct-form folding (27-bits word-length)	85
Table 3.7 Detail Power Comparison	86
Table 3.8 Detail area comparison with 16-bits word-length	87
Table 3.9 Area normalized to direct-form folding (16-bits word-length)	87
Table 3.10 Detail power comparison using 16-bits word-length	88
Table 3.11 Pad and silicon area of the chip (decimator)	91
Table 3.12 Specification	97



Table 3.13 Paper comparison	98
Table 4.1 Specification of Agilent 93000	100
Table 4.2 Measured power consumption	111
Table 4.3 Chip summary	112
Table A.1 Coefficients of the 2nd-stage FIR filter	116
Table A.2 Coefficients of the 3rd-stage FIR filter	116
Table A.3 Coefficients of the 4th-stage compensation FIR filter	118



# *List of Figures*

Figure 1.1 (a) Continuous-time signal $x_c(t)$	3
(b) its (continuous-time) Fourier transform $X_c(f)$	3
Figure 1.2 (a) Continuous-time to discrete-time conversion system	
(b) periodic impulse train $s(t)$	3
Figure 1.3 (a) Sampled signal $x_s(t)$	
(b) its (continuous-time) Fourier transform $X_s(f)$	4
Figure 1.4 (a) Discrete-time sequence $x[n]$	
(b) its discrete-time Fourier transform (DTFT) $X(f_d)$	5
Figure 1.5 Transfer Curve of a quantizer	6
Figure 1.6 Power spectral density of signal and quantization noise	
(a) Nyquist sampling (b) oversampling	8
Figure 1.7 Power spectral density of signal and quantization noise	
(a) Nyquist sampling	
(b) oversampling and removing out-of-band quantization noise	8
Figure 1.8 Power spectral density of signal and quantization noise after down-sampling	9
Figure 1.9 Procedure to obtain Nyquist rate high SNR signal	9
Figure 1.10 Decimator components	10
Figure 1.11 Power spectral density of signal and quantization noise for noise-shaping	11
Figure 1.12 Power spectral density of signal and quantization noise for oversampling and noise-shaping	12
Figure 1.13 Ideal noise transfer function (NTF) for different order SDM	15
Figure 1.14 (a) Decimator components	
(b) Behavior of decimator for SDM in time domain	16
(c) Behavior of decimator for SDM in frequency domain	17
Figure 1.15 (a) Aliasing-band of signal	
(b) cut-off-band of low-pass filter designed to prevent aliasing	18
Figure 1.16 Downsampler	19
Figure 1.17 (a) $X(f)$	
(b) $X_d(f)$ aliased by other copies of $X(f/D-i/D)$ (i.e., $i=1,2,3$ ), $D=4$	19
Figure 1.18 The bands in aliasing band $[0.5/D, 0.5]$ alias to wanted signal (green band), called folding-band, $D=4$ .	20

Figure 1.19 The bands over $[-0.5, 0.5]$ overlap with wanted signal band and are folding-bands, which are found from the trace-back process in this demonstration.	20
Figure 1.20 There are $D-1$ folding-bands for down-sampling $D$ .	21
Figure 1.21 Filter specifications for low-pass filter	22
Figure 2.1 (a) frequency over $[-0.5, 0.5]$ (b) frequency over $[0, 1]$ (c) frequency over $[0, 0.5]$ . The signals represented by these figure are the same.	26
Figure 2.2 Decimator architectures	28
Figure 2.3 For 1-bit 4th-order OSR-128 SDM, decimator architectures in terms of SNR, number of required multiplication operations per second and pass-band ripple are shown.	29
Figure 2.4 zero-pole plot for 5th-order comb filter with $D=32$ ( $5*32$ zeros around unit circle, 5 poles in $z=1$ and the other poles in $z=0$ )	30
Figure 2.5 Magnitude frequency response of 5th-order comb filter with $D=32$ (half periodic spectrum have been depicted, so there are $D/2=16$ notches in spectrum)	31
Figure 2.6 zero-pole plot of (a) 1st-order comb filter (b) counterclockwise rotated of 1st-order comb filter (c) clockwise rotated of 1st-order comb filter (d) 3rd-order modified comb filter, with $D=4$ .	33
Figure 2.7 Magnitude response (dB) of 3rd-order modified comb filter (MCF3) with $D=4$	34
Figure 2.8 Magnitude responses of Comb3 and MCF3 in folding-band The MCF3 can suppress more quantization noise power in folding-band.	35
Figure 2.9 The flow for quantization noise power spectral density of SDM in first decimation stage	37
Figure 2.10 Quantization noise power spectral density of 1-bit 4th-order sigma-delta modulator	38
Figure 2.11 First decimation filter (comb filter with $D=32$ for the left graph and 16 for the right graph, respectively). Folding-bands of these two magnitude response are both in the notch position, each $D-1$ folding-bands	38
Figure 2.12 Quantization noise PSD after first decimation filter	38
Figure 2.13 Quantization noise PSD after down-sampling	39
Figure 2.14 Folding-band for 2nd decimation stage	40
Figure 2.15 Flow for quantization noise power spectral density in $2^{\text{nd}}$ decimation stage	41

Figure 2.16 Quantization noise PSD after down-sampling 1	41
Figure 2.17 Magnitude response of 2nd stage decimation filter (FIR1) with quantized coefficients	42
Figure 2.18 Quantization noise PSD after 2nd decimation filter	42
Figure 2.19 Quantization noise PSD after down-sampling 2	43
Figure 2.20 Folding-band of 3rd decimation stage	43
Figure 2.21 Flow for quantization noise power spectral density in 3rd decimation stage	45
Figure 2.22 Quantization noise PSD after down-sampling 2	45
Figure 2.23 Magnitude response of 3rd stage decimation filter (FIR2) with quantized coefficients	45
Figure 2.24 Quantization noise PSD after 3rd decimation filter	46
Figure 2.25 Quantization noise PSD after down-sampling	46
Figure 2.26 Pass-band drop of comb filter with $D=32$ for $OSR=128$ ( $fb=0.5/OSR=3.90625e-3$ )	47
Figure 2.27 Pass-band drop of comb filter with $D=16$ for $OSR=64$ ( $fb=0.5/OSR=7.8125e-3$ )	47
Figure 2.28 Magnitude response of compensation filter in the 4th stage	49
Figure 2.29 Magnitude response of each stage	50
Figure 2.30 Quantization noise power spectral density (dB) of 1-bit 4th-order SDM	51
Figure 2.31 Magnitude response of equivalent single stage low-pass filter for decimation ratio 128	52
Figure 2.32 Magnitude response of equivalent single stage low-pass filter for decimation ratio 64	52
Figure 3.1 The cell-based implementation flow of digital IC	54
Figure 3.2 Simulation and verification procedures of decimator	55
Figure 3.3 Commutative rule: the system in (a) is equivalent to the system in (b).	56
Figure 3.4 Comparison of recursive and non-recursive algorithm of comb filter in terms power, area, speed, power speed product	58
Figure 3.5 System A is equivalent to system B (polyphase decomposition; efficient implementation for FIR filter followed by down-sampling).	61
Figure 3.6 Efficient implementation of decimation FIR filter with $M=2$	62
Figure 3.7 Decimation FIR filter (126th-order) with polyphase decomposition in direct-form	63
Figure 3.8 The meaning of switched arrow where $f$ denotes the sampling rate	64
Figure 3.9 The folded architecture of FIR filter with polyphase decomposition in direct-form	66

Figure 3.10 The decimation FIR filter (126th-order) with polyphase decomposition in transposed-form	67
Figure 3.11 The values stored in storage elements	68
Figure 3.12 Folded architecture of k-tap FIR filter in transposed-form without polyphase decomposition (i.e. no down-sampling) (for linear-phase, the feature of FIR filter coefficients: $h_n=h_{k-1-n}$ )	70
Figure 3.13 Corresponding unfolded FIR filter in transposed form	70
Figure 3.14 Folded architecture of decimation FIR filter using polyphase decomposition	72
Figure 3.15 Block diagram of overall decimator system	73
Figure 3.16 Circuit of clock divider by 2	74
Figure 3.17 Timing diagram for the circuit of clock divider by 2	74
Figure 3.18 Components of first decimation stage	75
Figure 3.19 Pipelined comb filter (only integrators part needed to be pipelined due to its critical timing)	76
Figure 3.20 Retiming to reduce the registers usage	77
Figure 3.21 Implementation of first decimation stage	77
Figure 3.22 (a) Proposed folded architecture of decimation FIR filter based on transposed-form using polyphase decomposition (b) the unfolded one (c) Timing diagram of my proposed architecture	79
Figure 3.23 Parts of control circuits	83
Figure 3.24 Trade-off between unfolded and folded FIR Filter	89
Figure 3.25 Trade-off between the three folded architectures	90
Figure 3.26 Layout of decimator	91
Figure 3.27 Pad assignment	92
Figure 3.28 The post-layout gate-level simulation and verification flow for decimator	93
Figure 3.29 Post-layout gate-level simulation result with decimation factor=128 at nWave of workstation	94
Figure 3.30 Verification in time domain and frequency domain for decimation ratio 128 using Matlab	94
Figure 3.31 Post-layout gate-level simulation result with decimation factor=64 at nWave of workstation	96
Figure 3.32 Verification in time domain and frequency domain for decimation ratio 64 using Matlab	96
Figure 4.1 P600 test system of Agilent 93000 SoC Series	100
Figure 4.2 Test development flow	101

Figure 4.3 (a) DUT board on test-head of Agilent 93000	101
(b) chip in socket of DUT board	
(c) the reverse-side of DUT board wired the core-power and io-power of the chip to power-supplies pins	
(d) Software (SmarTest) used to manipulate the Agilent 93000 in workstation (unix-system)	102
Figure 4.4 Test-pattern for Agilent 93000 composed of drive vector (input of DUT) and expected vector (expected output of DUT)	103
Figure 4.5 Flow of function test	103
Figure 4.6 Shmoo plot (128)	104
Figure 4.7 The spectrums for drive vector (IN) and expected vector (OUT), decimation factor 128	105
Figure 4.8 Spectrums of decimator input and output over the frequency range [100Hz 25.6MHz] (128)	106
Figure 4.9 Shmoo plot (64)	107
Figure 4.10 The spectrums for drive vector (IN) and expected vector (OUT), decimation factor 64	108
Figure 4.11 Spectrums of decimator input and output over the frequency range [100Hz 12.8MHz] (64)	109
Figure 4.12 Timing diagram (decimation ratio 128) plotted by Agilent 93000	110
Figure 4.13 Timing diagram (decimation ratio 64) plotted by Agilent 93000	111
Figure B.1 Corresponding waveforms of state characters of signals in my design	119
Figure B.2 Parts of test vectors for SDM with OSR 128	120
Figure B.3 Corresponding post-layout-simulation of parts' test vectors for SDM with OSR 128	121
Figure B.4 The corresponding timing diagram measured by Agilent 93000 (128)	122
Figure B.5 The corresponding post-layout-simulation (128)	122
Figure B.6 Parts of test vectors for SDM with OSR 64	123
Figure B.7 Corresponding post-layout-simulation of parts' test vectors for SDM with OSR 64	124
Figure B.8 The corresponding timing diagram measured by Agilent 93000 (64)	125
Figure B.9 The corresponding post-layout-simulation (64)	125

# *List of Abbreviations and Symbols*

ADC	Analog-to-Digital Converter
APR	Auto Place&Route
AWG	Arbitrary-Waveform-Generator
BW	Band-Width
CIC	Chip Implementation Center
CMOS	Complementary Metal-Oxide Semiconductor
CTFT	Continuous Time Fourier Transform
DAC	Digital-to-Analog Converter
DFT	Discrete Fourier Transform
DPS	Device-Power Supplies
DSP	Discrete-time Signal Processing
DTFT	Discrete-time Fourier Transform
DUT	Device Under Test
ENOB	Effective Number Of Bits
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
F <sub>p</sub>	End Frequency of Pass-band
F <sub>s</sub>	Beginning Frequency of Stop-band
f <sub>s</sub>	Sampling frequency
IIR	Infinite Impulse Response
LSB	Least Significant Bit
OSR	Over-Sampling Ratio
PCB	Printed Circuit Board
PSD	Power Spectral Density
R <sub>p</sub>	Maximum Pass-band Ripple
R <sub>s</sub>	Minimum Stop-band Attenuation
SDM	Sigma-Delta Modulator
SNR	Signal-to-Noise Ratio
VLSI	Very Large Scale Integrated

---

# CHAPTER

# 1

---

# Introduction

---



With the advance in VLSI technology, sigma-delta modulation (SDM) has become a very popular analog to digital conversion technique in many fields, such as voice, audio, telecommunication (wireless: 3G and 4G mobile terminals; wire-line: xDSL moderns ), etc. Since high resolution of sigma-delta A/D converters can be achieved by techniques, over-sampling and noise-shaping, even using 1-bit quantizer in the A/D converter [1]. That relieves the analog circuit design, which means that no accurate multi-bit quantizer is needed, like 16-bit or 24-bit quantizer, and a wide transition-band of anti-aliasing analog filter can be accepted due to over-sampling (imply that analog filter is easy to design and its cost is low). However, it needs digital hardware to finish the remaining A/D conversion jobs, which are removing out-of-band quantization noise, converting 1-bit to multi-bit (such as 16-bit) and down-sampling to Nyquist rate. In other words, the sigma-delta is one kind of A/D conversion method which moves the high resolution difficulty encountered in analog part to digital part. So the high resolution sigma-delta A/D converters are more attractive and applicable than other A/D conversion methods as the VLSI technology advances.



In addition, wireless communication devices demand multi-standard operation, which means that different signal bandwidth and different dynamic range requirements are needed. And a sigma-delta A/D converter is a best choice to perform baseband channel select filtering in digital domain as well as to meet these different bandwidth and dynamic range requirements by changing sampling rate and selecting over-sampling ratio (OSR), respectively.

## 1.1 Motivation

A sigma–delta A/D converter consists of analog circuits (sigma–delta modulator, SDM) and digital circuits (decimator, namely decimation filter and down-sample circuit). Although the resolution of the sigma-delta A/D converter is typically determined by the analog modulators, silicon area of the sigma-delta A/D converter is governed largely by the digital decimation filters. For example, the digital part of sigma-delta A/D converter governs 78% area in [2]. So it is more important to reduce the not crucial part’s silicon area, namely digital part’s silicon area.

Furthermore, for a programmable OSR sigma-delta A/D converter, a decimation filter with programmable decimation ratios is needed. That means different low-pass filters are needed to obtain different spectrums for down-sampling. Therefore, digital hardware would increase by several times. As a result of that, digital parts of the sigma-delta A/D converter would govern more silicon area percentage.

For cost concerns, the silicon area of the sigma-delta A/D converter must be minimized. Of course, the silicon area of the digital part (decimator) is main part needed to be improved, which dominates the silicon area of whole A/D converter. And typically, decimator consists of comb filter and several stages finite-impulse-response filters (FIR filters). The high order FIR would dominate the decimator silicon area. For instance, area of high order FIR filter would govern 80% decimator area in a three stages decimator case (comb, 18<sup>th</sup>-order FIR, and 126<sup>th</sup>-order FIR).

Now, it is quite obvious that area of high order FIR filter is the main part this thesis wants to improve as well as to keep the programmable decimation ratio decimator area overhead minimum.

## 1.2 Fundamentals

This section would introduce the concepts of signal processing and show the meaning of signal processing terminologies, such as sampling theorem, aliasing, folding-band, etc. Also, principle of sigma-delta A/D converter would be described.

### 1.2.1 Sampling Theorem [3]

Let  $x_c(t)$  be a band-limited continuous-time signal with

$$X_c(f)=0 \quad \text{for } |f| \geq f_B$$

$X_c(f)$  is continuous-time Fourier transform of  $x_c(t)$ . And  $x_c(t)$  and  $X_c(f)$  are shown in Figure 1.1(a) and Figure 1.1(b), respectively.

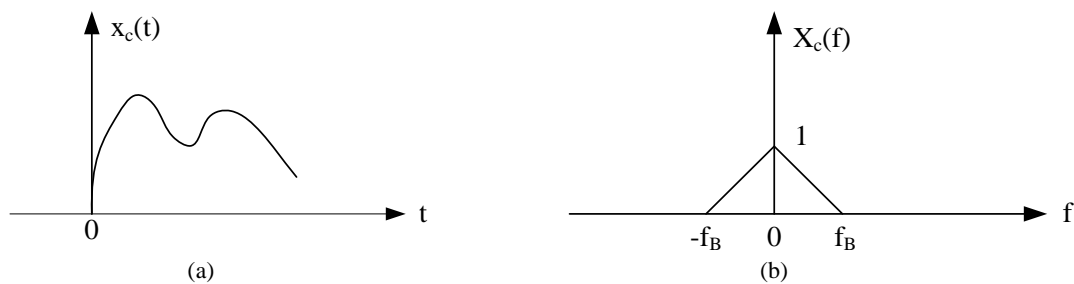
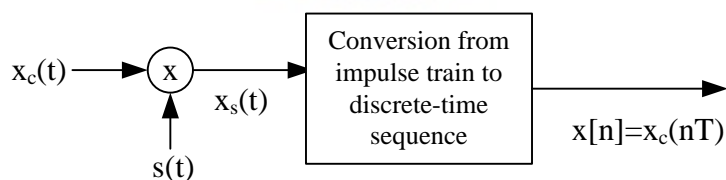
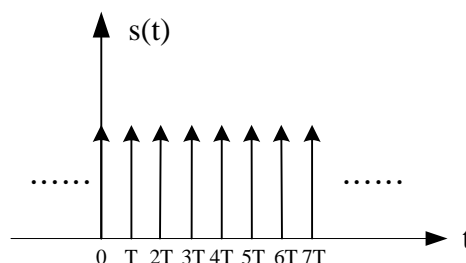


Figure 1.1 (a) Continuous-time signal  $x_c(t)$  (b) its (continuous-time) Fourier transform  $X_c(f)$

It is convenient to understand the continuous-time to discrete-time conversion mathematically in two stages depicted in Figure 1.2, namely sampling process [3].



(a)



(b)

Figure 1.2 (a) Continuous-time to discrete-time conversion system (b) periodic impulse train  $s(t)$

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{1.1}$$

$$x_s(t) = x_c(t)s(t) = x_c(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{1.2}$$

$$x_s(t) = \sum_{n=-\infty}^{\infty} x_c(nT)\delta(t - nT) \tag{1.3}$$

And, continuous-time Fourier transform of  $s(t)$  and  $x_s(t)$  are  $S(f)$  and  $X_s(f)$ , respectively.

$$S(f) = f_s \sum_{k=-\infty}^{\infty} \delta(f - kf_s) \tag{1.4}$$

where  $f_s=1/T$

$$X_s(f) = X_c(f) * S(f) = f_s \sum_{k=-\infty}^{\infty} X_c(f - kf_s) \tag{1.5}$$

The time domain and continuous-time frequency domain of signal  $x_s$  are shown in Figure 1.3.  $T$  is periodic sampling period, and its reciprocal,  $f_s=1/T$ , is the sampling frequency.

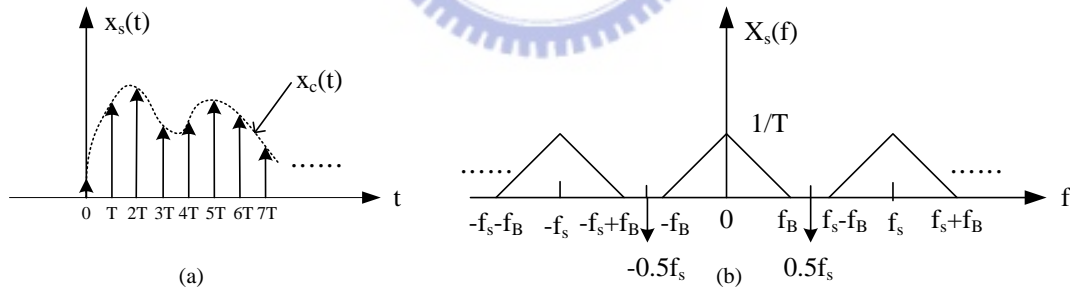


Figure 1.3 (a) Sampled signal  $x_s(t)$  (b) its (continuous-time) Fourier transform  $X_s(f)$

$X_s(f)$  consists of periodically repeated copies of  $X_c(f)$ , which are shifted by integer multiples of sampling frequency. It is obvious that when  $f_s - f_B > f_B$  the replicas of  $X_c(f)$  do not overlap, which means the signal  $x_c(t)$  could be recovered from  $x_s(t)$  with an ideal low-pass filter. The minimum sampling rate for non-overlap (no aliasing) is  $f_s=2f_B$ , which is called Nyquist rate.

For discrete-time signal processing, discrete-time sequence,  $x[n]$ , is a better representation for computer and digital system design (including digital filter). Also, the discrete-time Fourier transform would be introduced, which is suitable for discrete-time signal. And its relation is shown below in Figure 1.4.

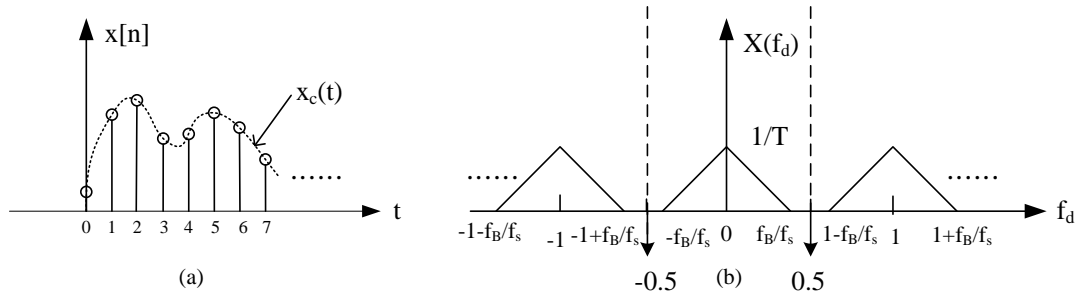


Figure 1.4 (a) Discrete-time sequence  $x[n]$  (b) its discrete-time Fourier transform (DTFT)  $X(f_d)$

$$f_d = \frac{f}{f_s}$$

$$x[n] = x_c(nT) \quad -\infty < n < \infty \quad \text{where } n \text{ is an integer.}$$

$f_d$  is digital frequency for discrete-time sequences, which means frequency for discrete-time Fourier transform.  $f_d$  is frequency normalized to sampling frequency,  $f_s$ . Because there is no time information on the discrete-time sequence  $x[n]$ , there is no frequency information (Hz) for discrete-time Fourier transform (only normalized frequency between  $-0.5 \sim 0.5$ ). And 1 in  $f_d$  represents the sampling frequency.

For conveniences, the digital frequency  $f_d$  will be used in following chapters to design and illustrate digital filter spectrum. And if  $x[n]$  is a real number sequence, the  $X(f_d)$  will be even function, which means that only frequency range between 0 and 0.5 needs to be depicted in spectrum graphs.

### 1.2.2 Principle of Sigma-Delta A/D Converter [1]

Previous section introduces the continuous-time to discrete-time conversion, namely sampling. However, the analog to digital (A/D) conversion consists of sampling and quantization, which are discrete in time and amplitude respectively.

It is called a quantization process that an infinite number of input amplitude values are mapped into finite number of output amplitude values, which is shown in Figure 1.5.

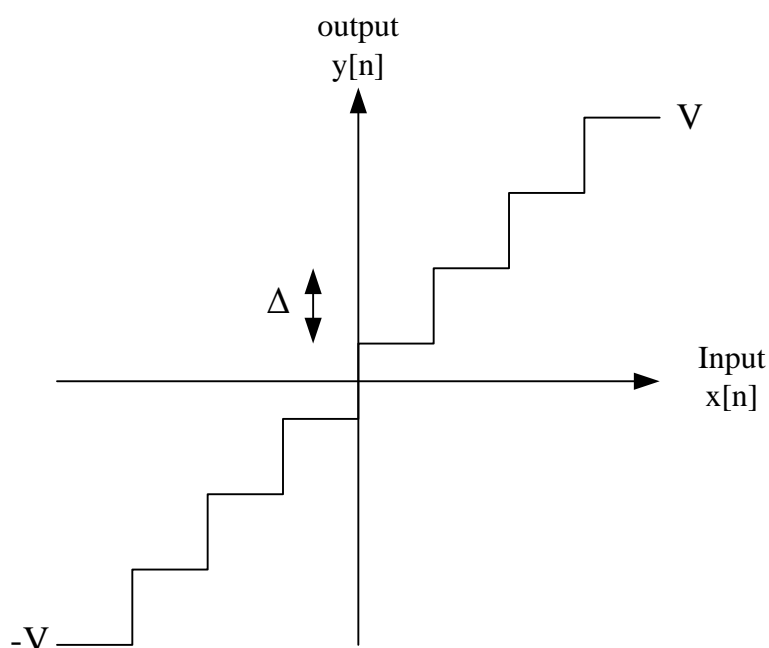


Figure 1.5 Transfer Curve of a quantizer

For a N-bit quantizer with quantization levels  $L=2^N$ , quantization error between output and input do not exceed half a least significant bit (LSB).

$$\Delta = 2V / (L - 1) = \text{LSB}$$

$$-\Delta/2 \leq e \leq \Delta/2$$

$e$  is quantization error, i.e.,  $e = \text{output} - \text{input}$ . That implies

$$y[n] = x[n] + e[n] \quad (1.6)$$

In order to simplify the analysis of quantization error, some assumptions about noise process due to quantization are made:

- The error sequence,  $e[n]$ , is a sample sequence of the stationary random process.
- The error sequence,  $e[n]$ , is uncorrelated with the input.
- The probability density function of random process  $e[n]$  is uniform distributed over  $[-\Delta/2, \Delta/2]$ .
- The random variables of random process  $e[n]$  are uncorrelated, i.e., the random process  $e[n]$  is a white noise process, which means that the power spectrum density of  $e[n]$  is uniform distributed over  $[-0.5, 0.5]$  in  $f_d$ .

These assumptions are reasonable when  $N$  is large, quantizer is not overloaded, and the successive signal values are not extremely correlated [1].

Under those assumptions, the analysis of quantization error is quite simple. For

example, the power of  $e[n]$  is its variance  $\sigma_e^2$

$$\sigma_e^2 = \frac{\Delta^2}{12} = \frac{\left(\frac{2V}{L-1}\right)^2}{12} = \frac{\left(\frac{2V}{2^N-1}\right)^2}{12} \cong \frac{\left(\frac{2V}{2^N}\right)^2}{12} \quad (1.7)$$

And then, the signal to noise ration, SNR is

$$\text{SNR} = 10 \log \left( \frac{\sigma_x^2}{\sigma_e^2} \right) \quad (1.8)$$

where  $\sigma_x^2$  is signal power. For sinusoidal input, amplitude is  $V$ , and then signal power  $\sigma_x^2$  is  $V^2/2$ .

$$\text{SNR} = 10 \log \left( \frac{\sigma_x^2}{\sigma_e^2} \right) = 6.02N + 1.76 \text{ (dB)} \quad (1.9)$$

The meaning of this Equation 1.9 is that SNR would increase about 6dB when one bit increased in  $N$ . However, when  $N$  is larger than 10-bit, the precision of quantizer is hard to maintain due to the very small  $\Delta$  (LSB). For example, 10-bit quantizer means that the quantization levels is  $L=2^{10}$ , which implies that  $\Delta=2V/(L-1)=2*1.8/(2^{10}-1)=3.52 \times 10^{-3}$  (Volt) for  $V=1.8$  (Volt). Any component mismatches and process variation would cause quantization error greater than  $\Delta$ , which means the  $N$ -bit resolution could not be obtain.

To obtain high resolution, two techniques, oversampling and noise-shaping, can be used to overcome the difficulties encountered in above situations.

### Oversampling

Oversampling means that signal samples are acquired from analog signal waveform much faster than Nyquist rate. For quantization noise assumptions, the noise would uniform distributed over  $[-0.5, 0.5]$  in  $f_d$ . And then the technique, oversampling, would change the distribution of signal power spectral density in  $f_d$ . For example, the signal power spectral density (PSD) would distributed over  $[-0.125, 0.125]$  for over-sampling-ratio (OSR=4), which is different from signal PSD distributed over  $[-0.5, 0.5]$  for Nyquist rate sampling. Also the magnitude of signal PSD would change according to sampling frequency (see section 1.2.1,  $1/T$  in Figure 1.4), which makes the signal power identical with different sampling frequency or OSR. The PSD of signal and quantization noise with different sampling frequency are shown in Figure 1.6.

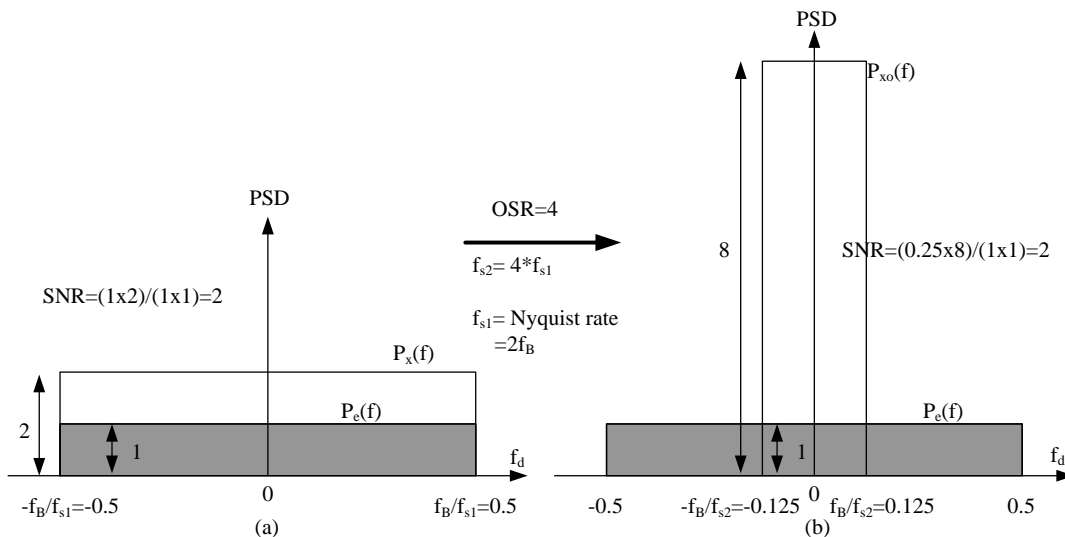


Figure 1.6 Power spectral density of signal and quantization noise  
 (a) Nyquist sampling (b) oversampling

Before further digital signal processing, the SNR of Figure 1.6(a) and Figure 1.6(b) are the same. However, oversampling makes the distribution of signal PSD different. For  $OSR=4$ , the signal PSD is distributed over  $[-0.125, 0.125]$ , which means that a digital low-pass filter could be utilized to remove the quantization noise out of the range  $[-0.125, 0.125]$  and higher SNR can be obtained. The improved SNR is illustrated in Figure 1.7.

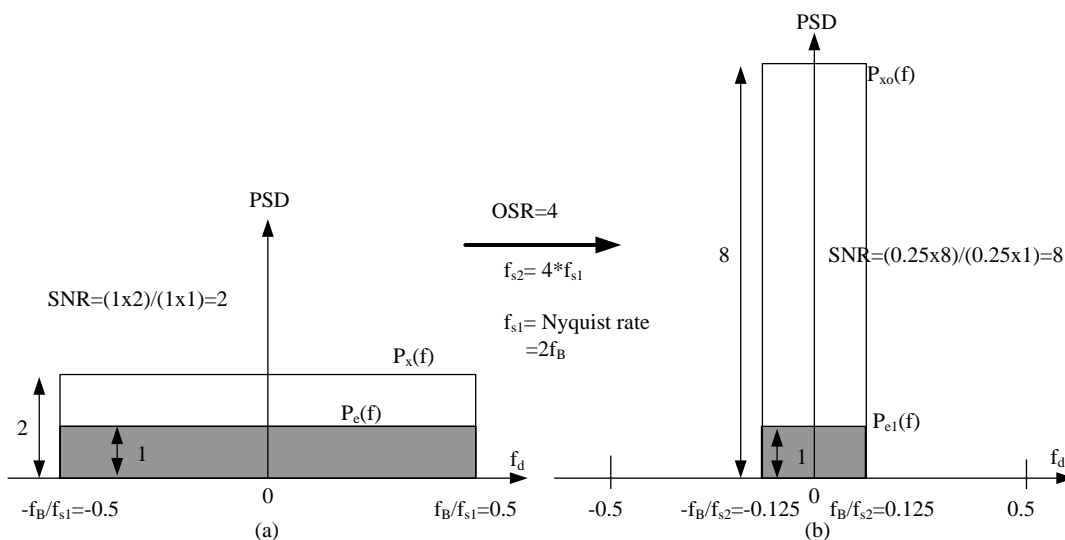


Figure 1.7 Power spectral density of signal and quantization noise  
 (a) Nyquist sampling (b) oversampling and removing out-of-band quantization noise

The improved SNR by oversampling = the original SNR \* OSR

or

The improved SNR by oversampling = the original SNR + 3.01\*log<sub>2</sub>OSR (dB)

Because no signal information on the frequency [-0.5, -0.125] and [0.125, 0.5], that means the lower sampling frequency, such as Nyquist rate (2\*f<sub>B</sub>), can be used to represent the signal well. And then the PSD is changed as Figure 1.8

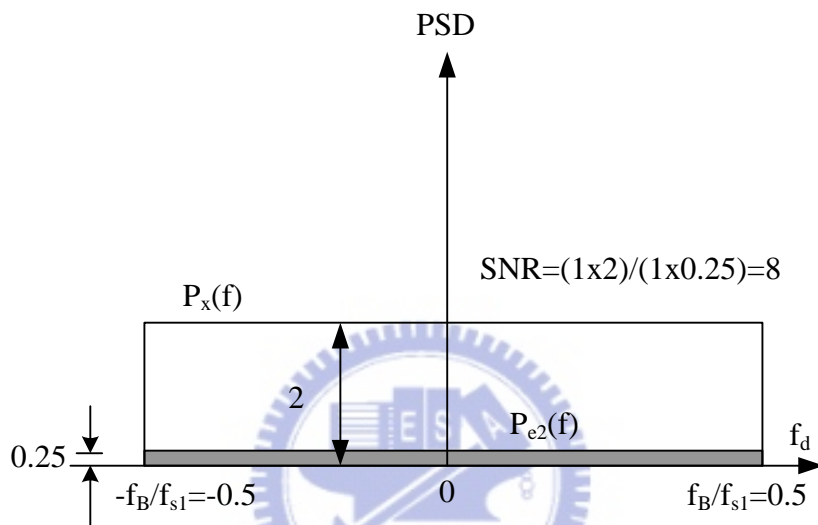


Figure 1.8 Power spectral density of signal and quantization noise after down-sampling

Now, Figure 1.9 (from Figure 1.6(b) to Figure 1.7(b) and then to Figure 1.8) demonstrates the function of the decimator, which is composed of digital low-pass filter and downsampler (circuit of lowering the sampling rate) shown in Figure 1.10.

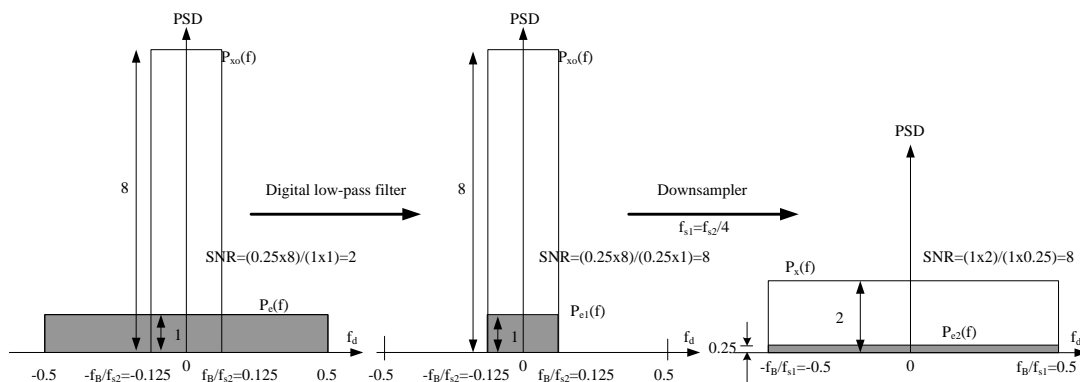


Figure 1.9 Procedure to obtain Nyquist rate high SNR signal



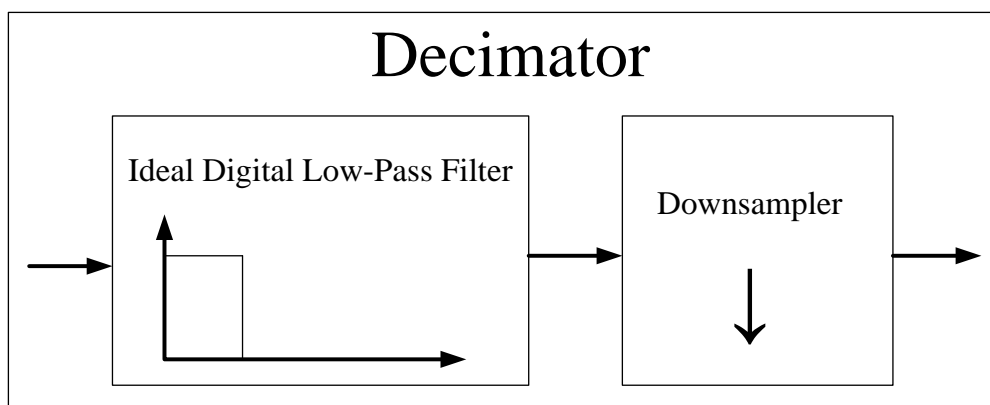


Figure 1.10 Decimator components

### Noise Shaping

In above quantization noise assumptions, the quantization noise PSD is uniform distributed over  $[-0.5, 0.5]$  in  $f_d$ . And noise-shaping is a modulation technique to change the shape of the quantization noise PSD.

Now, for easily understanding,  $z$  domain representations of signal would be introduced. Z-transform is a best representation for discrete-time signal and systems as Laplace transform for continuous-time. Also  $z$ -transform has a similar relationship to the corresponding Fourier transform, which is  $z = e^{j2\pi f_d}$ . For conveniences, the  $z$ -transform of input ( $x[n]$ ), output ( $y[n]$ ), and quantization error ( $e[n]$ ) would be used and relationship of A/D could be expressed as follows:

$$Y(z) = X(z) + E(z) \quad (1.10)$$

where  $Y(z)$ ,  $X(z)$ , and  $E(z)$  are  $z$ -transform of  $y[n]$ ,  $x[n]$ , and  $e[z]$ , respectively.

Generally, some modulation could be used during the A/D conversion, so the relationship between output, input, and quantization error could be rewrite as follow:

$$Y(z) = X(z)H_x(z) + E(z)H_e(z) \quad (1.11)$$

Noise-shaping is a technique to change the distribution of quantization noise PSD over  $[-0.5, 0.5]$  in  $f_d$  as Figure 1.11.

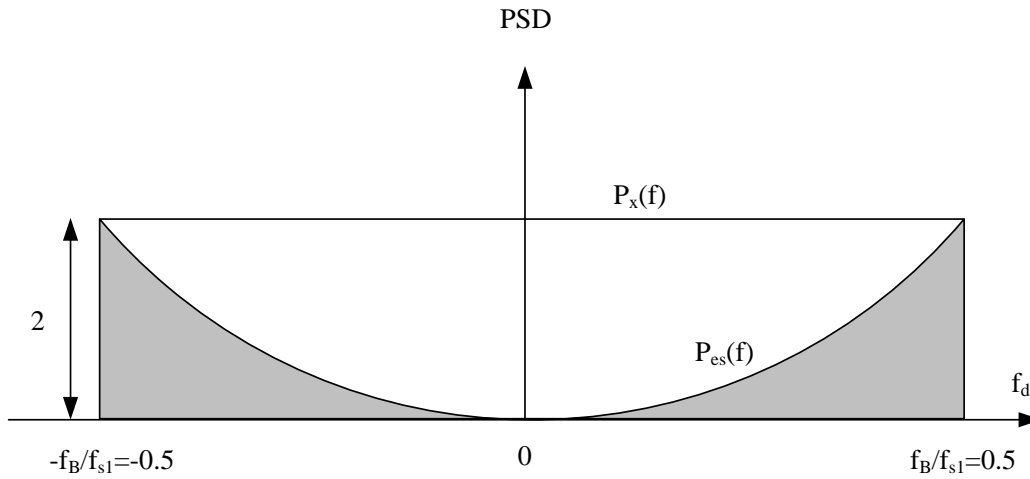


Figure 1.11 Power spectral density of signal and quantization noise for noise-shaping

### Sigma-Delta A/D converters

Sigma-delta A/D converters is based on two techniques, oversampling and noise-shaping. Combining oversampling and noise shaping, sigma-delta A/D converters could obtain a very high resolution (SNR). The general form for  $k^{\text{th}}$ -order sigma-delta modulator could be written as:

$$Y(z) = X(z)z^{-k} + E(z)(1-z^{-1})^k \quad (1.12)$$

$$H_x(z) = z^{-k} \quad (1.13)$$

$$H_e(z) = (1-z^{-1})^k \quad (1.14)$$

Now the peak SNR for sinusoidal signal can be derived according to quantization bit  $N$ , OSR, and  $k^{\text{th}}$ -order SDM noise transfer function: ( $f$  represents normalized frequency,  $f_d$ , for following analysis)

$$z = e^{j2\pi f} \quad , \text{ the relationship between } z \text{ domain and frequency domain} \quad (1.15)$$

$$H_x(f) = e^{-j2\pi kf} \quad (1.16)$$

$$\begin{aligned} H_e(f) &= (1 - e^{-j2\pi f})^k = \left( e^{-j\pi f} (e^{j\pi f} + e^{-j\pi f}) \right)^k \\ &= 2^k e^{-jk\pi f} \sin^k(\pi f) \end{aligned} \quad (1.17)$$

For  $Y(f)=X(f)H(f)$  in frequency domain, i.e.,  $y[n]=x[n]*h[n]$  in time domain, and then the relationship of the power spectral density (PSD) between input, output, transfer function is  $P_y(f) = P_x(f)|H(f)|^2$ . So,

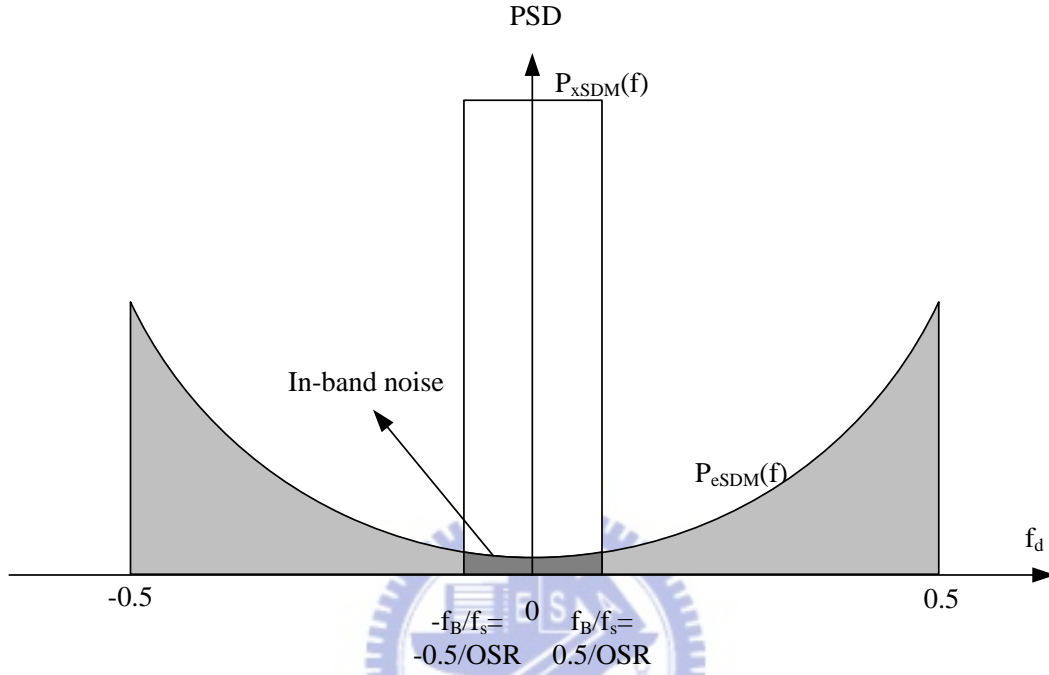


Figure 1.12 Power spectral density of signal and quantization noise for oversampling and noise-shaping

$$\begin{aligned}
 P_{xSDM}(f) &= P_x(f)|H_x(f)|^2 \\
 &= P_x(f)|e^{-j2\pi kf}|^2 \\
 &= P_x(f)
 \end{aligned} \tag{1.18}$$

$$\begin{aligned}
 P_{eSDM}(f) &= P_e(f)|H_e(f)|^2 \\
 &= P_e(f) \left| (1 - e^{-j2\pi f})^k \right|^2 \\
 &= P_e(f) \left| 2^k e^{-jk\pi f} \sin^k(\pi f) \right|^2 \\
 &= P_e(f) 2^{2k} \sin^{2k}(\pi f)
 \end{aligned} \tag{1.19}$$

$P_{xSDM}(f)$ : Signal PSD of SDM over  $[-0.5/OSR, 0.5/OSR]$  in  $f$ , i.e.  $f_d$ .

$P_{eSDM}(f)$ : Quantization noise PSD of SDM over  $[-0.5, 0.5]$  in  $f$ , i.e.  $f_d$

$$P_x(f) = \frac{\sigma_x^2}{1/OSR} = \frac{V^2}{2} OSR \quad (1.20)$$

$$P_e(f) = \frac{\frac{\Delta^2}{12}}{1} = \frac{(\frac{2V}{2^N-1})^2}{12} \cong \frac{(\frac{2V}{2^N})^2}{12} \quad (1.21)$$

Signal power is still the same  $PWR_{signal} = \sigma_x^2 = \int_{-0.5/OSR}^{0.5/OSR} \frac{V^2}{2} OSR df = \frac{V^2}{2}$

As a result of that signal spectrum is distributed over  $[-f_B/f_s, f_B/f_s]$  (or  $[-0.5/OSR, 0.5/OSR]$ ). So inband quantization noise power  $QN_{in-band}$  :

$$\begin{aligned} QN_{in-band} &= \int_{-0.5/OSR}^{0.5/OSR} P_e(f) |H_e(f)|^2 df \\ &= \frac{(\frac{2V}{2^N})^2}{12} 2^{2k} \int_{-0.5/OSR}^{0.5/OSR} \sin^{2k}(\pi f) df \\ &= \frac{(\frac{2V}{2^N})^2}{12} 2^{2k} \int_{-0.5/OSR}^{0.5/OSR} (\pi f)^{2k} df \\ &= \frac{(\frac{2V}{2^N})^2}{12} 2^{2k} \frac{1}{2k+1} \frac{1}{\pi} \left[ \left(\frac{0.5\pi}{OSR}\right)^{2k+1} - \left(\frac{-0.5\pi}{OSR}\right)^{2k+1} \right] \\ &= \frac{(\frac{2V}{2^N})^2}{12} 2^{2k} \frac{1}{2k+1} \frac{2}{\pi} \left(\frac{0.5\pi}{OSR}\right)^{2k+1} \\ &= \frac{(\frac{2V}{2^N})^2}{12} \frac{1}{2k+1} \frac{1}{\pi} \left(\frac{\pi}{OSR}\right)^{2k+1} \\ &= \frac{(\frac{2V}{2^N})^2}{12} \frac{\pi^{2k}}{2k+1} \left(\frac{1}{OSR}\right)^{2k+1} \\ &= \frac{1}{3} \frac{V^2}{2^{2N}} \frac{\pi^{2k}}{2k+1} \left(\frac{1}{OSR}\right)^{2k+1} \end{aligned} \quad (1.22)$$

For  $x \approx 0$ , and then  $\sin(x) \approx x$ , so for  $OSR > 8$ , i.e.,  $0.5/OSR \approx 0 \Rightarrow \sin(\pi f) = \pi f$

$$SNR_{peak} \text{ (dB)} = 10 \log \left( \frac{PWR_{signal}}{QN_{in-band}} \right)$$

$$\begin{aligned}
 &= 10 \log \left( \frac{\frac{v^2}{2}}{\frac{1}{32} \frac{v^2}{2} \pi^{2k} \left(\frac{1}{\text{OSR}}\right)^{2k+1}} \right) \\
 &= 10 \log \left( \frac{3}{2} \frac{2^{2N} (2k+1)}{\pi^{2k}} (\text{OSR})^{2k+1} \right) \\
 &= 10 \log \left( \frac{3}{2} \right) + 10 \log(2^{2N}) + 10 \log(2k+1) \\
 &\quad + 10 \log((\text{OSR})^{2k+1}) - 10 \log(\pi^{2k}) \\
 &= 1.76 + 6.02N + (20k+10) \log(\text{OSR}) + 10 \log\left(\frac{2k+1}{\pi^{2k}}\right) \\
 &= 1.76 + 6.02N + \frac{(20k+10)}{3.32} \log_2(\text{OSR}) + 10 \log\left(\frac{2k+1}{\pi^{2k}}\right) \\
 &= 1.76 + 6.02N + (6.02k + 3.01) \log_2(\text{OSR}) + 10 \log\left(\frac{2k+1}{\pi^{2k}}\right)
 \end{aligned} \tag{1.23}$$

From Equation 1.23, it is obvious that sigma-delta can obtain a very high SNR, such as SNR=167.2 (dB) for a 1-bit, OSR=128 4<sup>th</sup>-order SDM. And for every doubling of OSR, the SNR improves by (6k+3) dB. That means high order SDM could improve more SNR for doubling OSR. The higher order of SDM implies the better noise-shaping (noise attenuation in signal-band), and you can see that in Figure 1.13. So, it is a good idea to combine the two techniques, noise-shaping and oversampling.

According to Equation 1.23, a SNR table (Table 1.1) with different OSR and order of SDM using one bit quantizer is shown below. Also, SNR values for other number of quantizer bit N is easy to obtain by Table 1.1. For an N-bit quantizer, the new SNR values table would increase 6.02\*(N-1) dB to Table 1.1. For example, N=3, the SNR values table with different OSR and order of SDM using 3-bit quantizer is SNR values of Table 1.1 increasing 12.04 dB.

Table 1.1 Ideal Peak SNR with 1-bit quantizer (N=1)

SNR	k=1	k=2	k=3	k=4
OSR=16	38.7318 dB	55.0897 dB	70.6904 dB	85.9212 dB
OSR=32	47.7627 dB	70.1412 dB	91.7625 dB	113.0139 dB
OSR=64	56.7936 dB	85.1927 dB	112.8346 dB	140.1066 dB
OSR=128	65.8245 dB	100.2442 dB	133.9067 dB	167.1993 dB
OSR=256	74.8554 dB	115.2957 dB	154.9788 dB	194.2920 dB

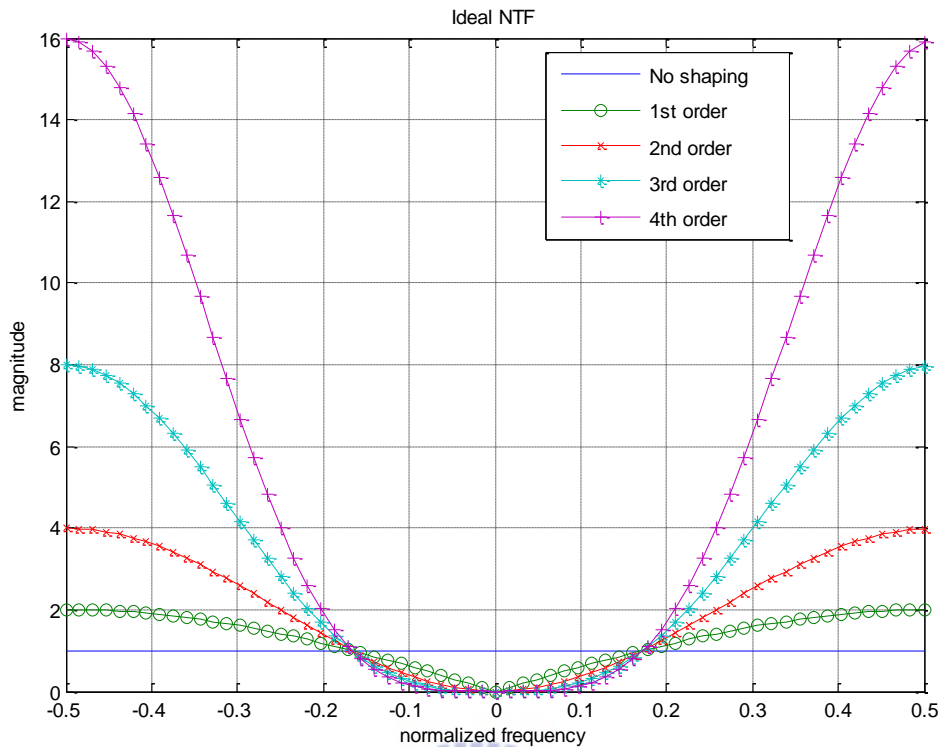


Figure 1.13 Ideal noise transfer function (NTF) for different order SDM

### 1.2.3 Decimator

The decimator is a circuit used to lower the sampling rate of the oversampling A/D converters (the sigma-delta A/D converter is one of them), and for preventing aliasing, a pre-filter (low-pass filter) is needed before down-sampling. So, a decimator is composed of digital lowpass filter and downsampler, shown in Figure 1.14 (a). And the functions of decimator are removing out-of band quantization noise to obtain high SNR signal (such as, expanding one bit resolution to multi-bit), preventing aliasing (keep the aliasing power minimum), and lowering the sampling rate (from oversampling to Nyquist rate), which are also mentioned in above section 1.2.2 (the PSD differences in decimator are shown in Figure 1.9).

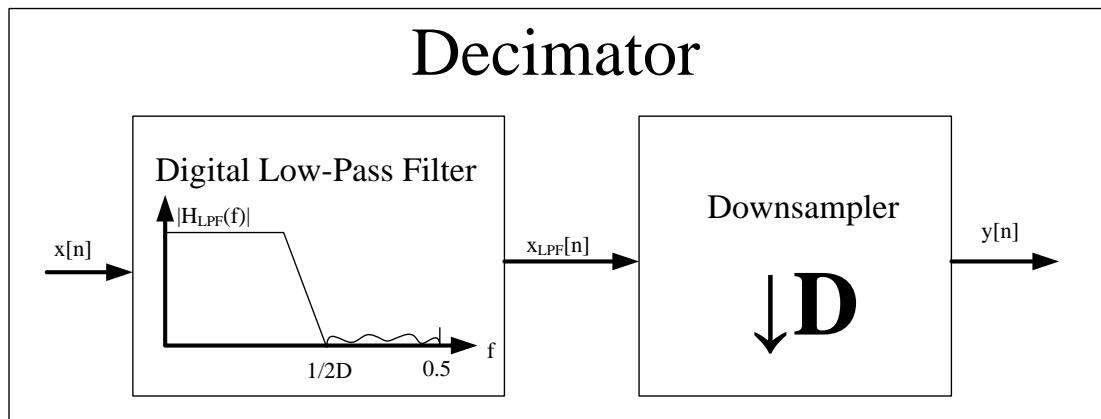


Figure 1.14 (a) Decimator components

The decimator behavior in time and frequency domain for the sigma-delta A/D converter is illustrated in Figure 1.14 (b) and Figure 1.14 (c)

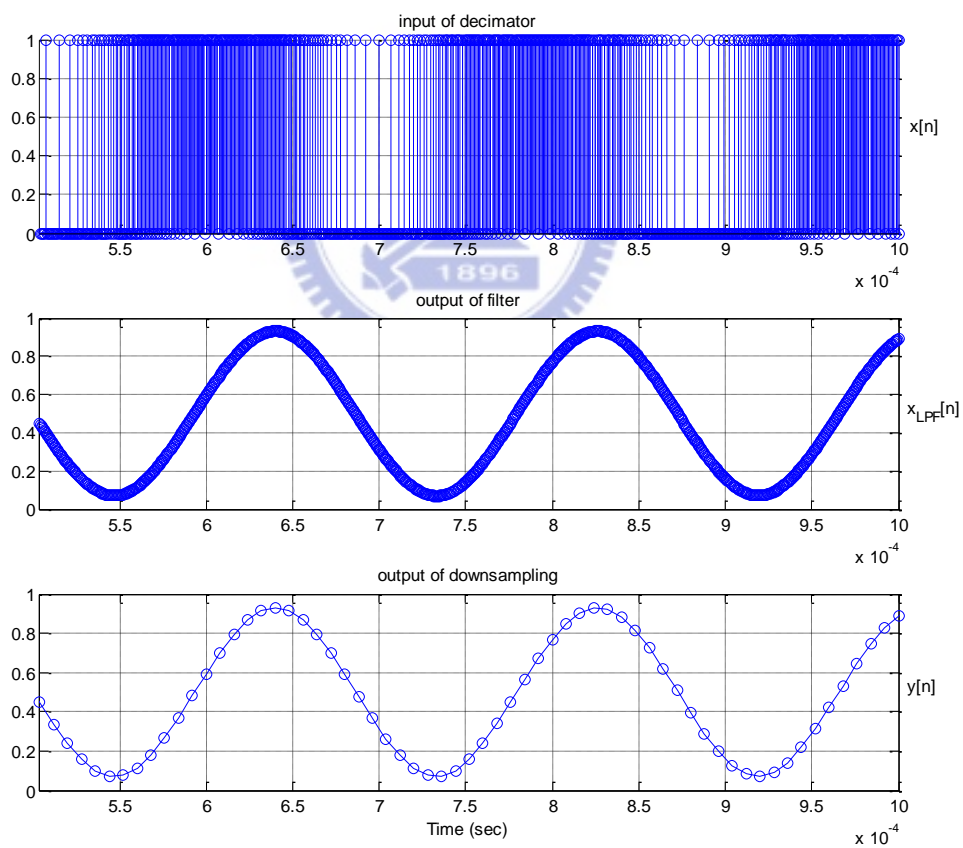


Figure 1.14(b) Behavior of decimator for SDM in time domain

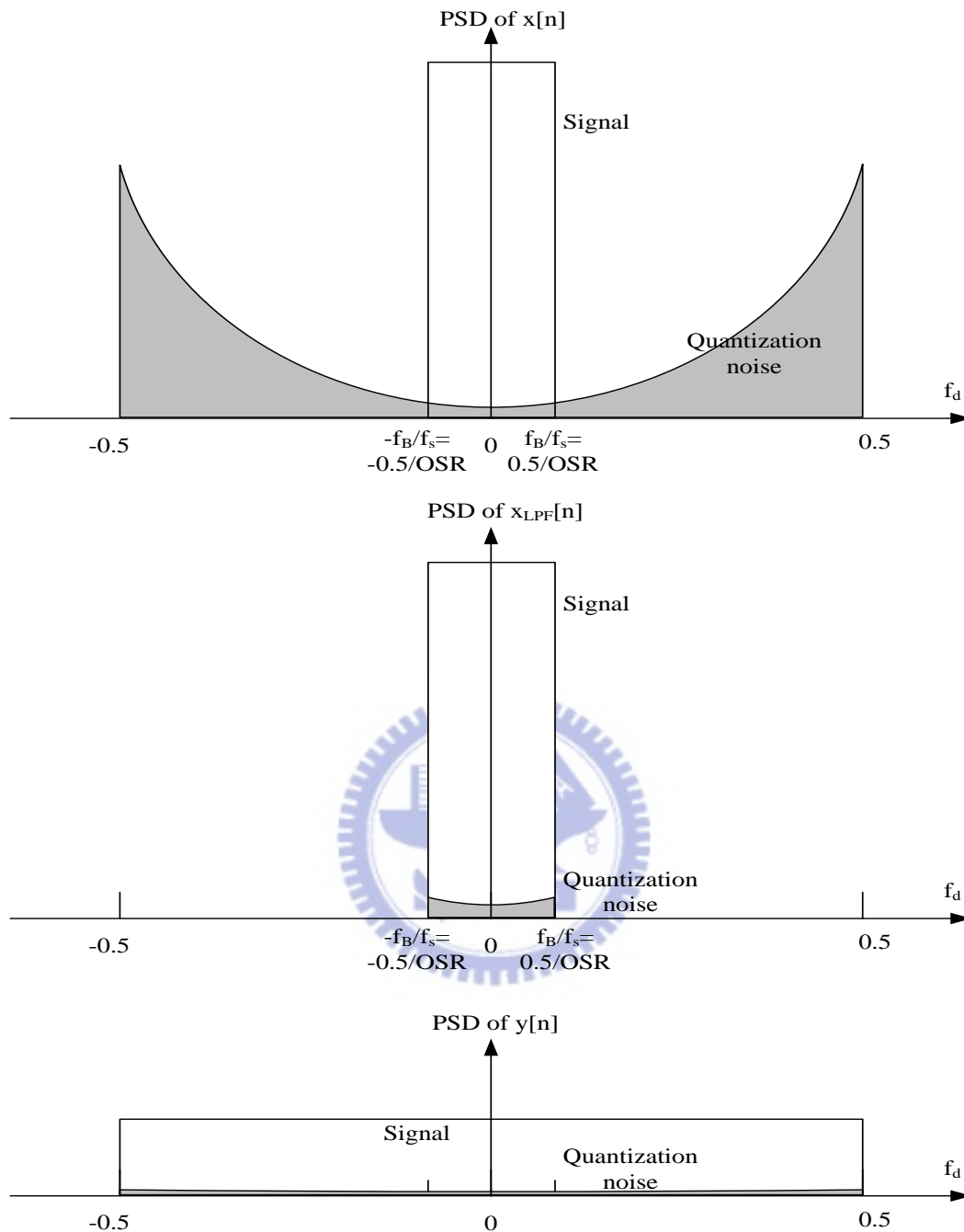


Figure 1.14(c) Behavior of decimator for SDM in frequency domain

In this section, the terminologies related to design a decimator would be introduced, which include the parameters for designing low-pass filter and some considerations in down-sample procedure. For down-sampling  $D$ , the aliasing-band of signal and cut-off-band of low-pass filter designed to prevent aliasing is shown below.



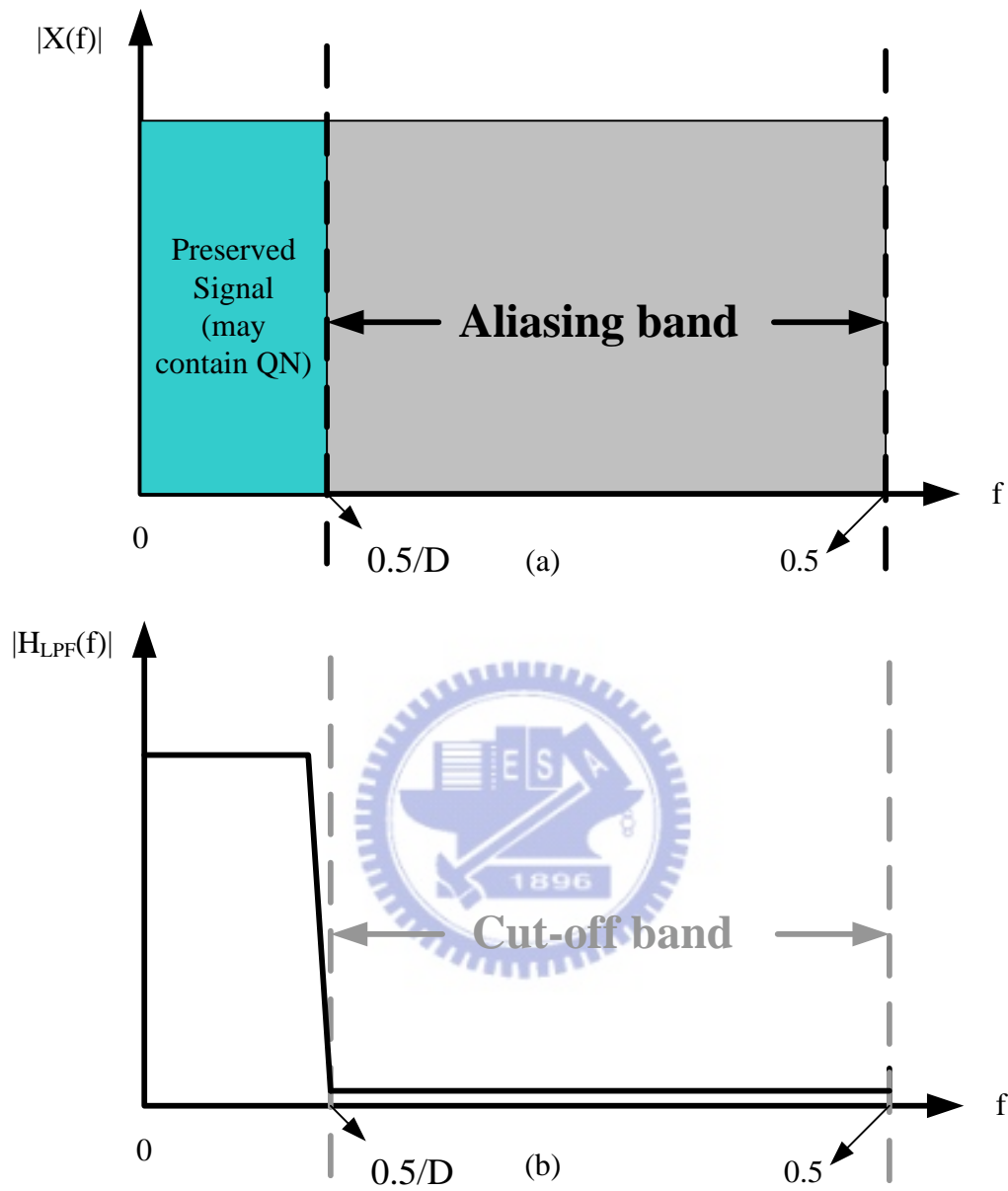


Figure 1.15 (a) Aliasing-band of signal (b) cut-off-band of low-pass filter designed to prevent aliasing

From above Figure 1.15, it is clear that the signal in aliasing band is required to be removed due to the relationship between  $x[n]$  and  $x_d[n]$  expressed by Equation 1.24 and Equation 1.25.  $x_d[n]$  is a down-sample sequence of  $x[n]$ , see Figure 1.16.

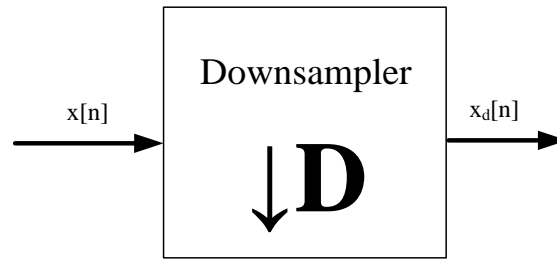


Figure 1.16 Downsampler

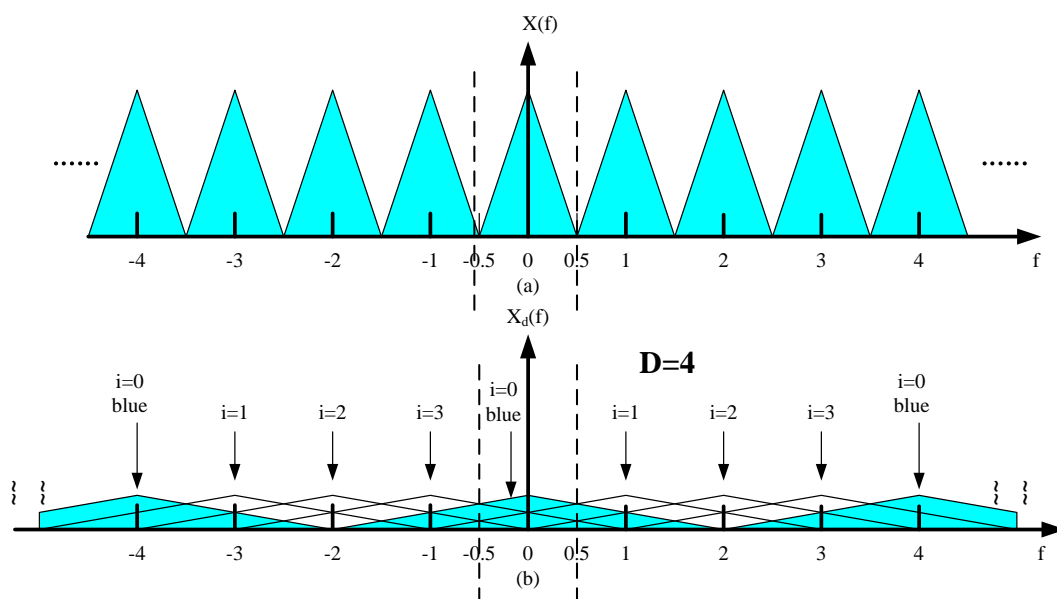
$$x_d[n] = x[nD] \quad (1.24)$$

And the relationship between  $x[n]$  and  $x_d[n]$  in frequency domain (discrete-time Fourier transform) is

$$X_d(f) = \frac{1}{D} \sum_{i=0}^{D-1} X\left(\frac{f}{D} - \frac{i}{D}\right) \quad (1.25)$$

$X(f)$  and  $X_d(f)$  are discrete-time Fourier transform of  $x[n]$  and  $x_d[n]$ , respectively.

Now, from Equation 1.25, it could explain why the aliasing-band is in the range  $[0.5/D, 0.5]$  in Figure 1.15(a) because  $X_d(f)$  is  $D$  copies of  $X(f)$  with expanding  $D$  times in frequency domain and shifted by integer multiples, which imply that  $0.5/D$  in frequency domain would be expanded to  $0.5$  and thus larger than  $0.5/D$  in frequency domain would overlap (expanded to  $>0.5$ ) with other copies  $X(f/D-i/D)$ . For example  $D=4$ ,  $X_d(f)$  aliased by other copies of  $X(f/D-i/D)$  (i.e.,  $i=1,2,3$ ) is shown below.

Figure 1.17 (a)  $X(f)$  (b)  $X_d(f)$  aliased by other copies of  $X(f/D-i/D)$  (i.e.,  $i=1,2,3$ ),  $D=4$

From above Figure 1.17, it is obvious that there are  $D-1$  copies ( $D=4$ ) aliased to the band  $[-0.5, 0.5]$ . In a word, frequency of signal larger than  $0.5/D$  in  $[0, 0.5]$  would cause aliasing. Now, considering a case  $OSR > D$  (first few stages of multi-stages decimator), aliasing would still exist. However, only a few bands in aliasing band  $[0.5/D, 0.5]$  alias to the wanted signal due to  $OSR > D$ , shown in Figure 1.18. And these few bands in aliasing-band aliasing to wanted signal are called folding band. Aliasing-band exclude folding-bands would alias to the unwanted signal (such as, quantization noise), which could be removed latter by remaining filters.

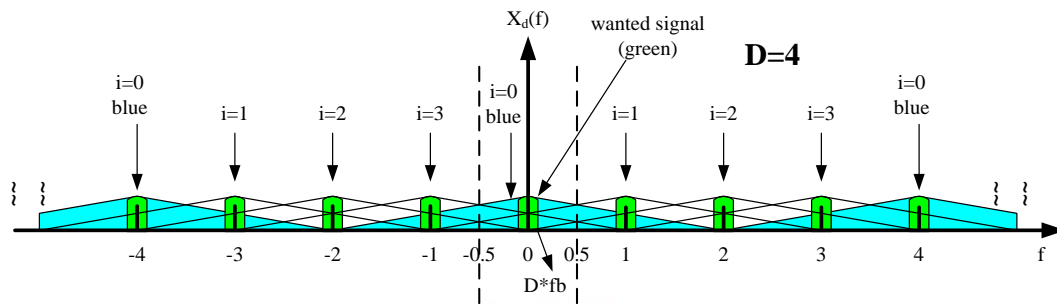


Figure 1.18 The bands in aliasing band  $[0.5/D, 0.5]$  alias to wanted signal (green band), called folding-band,  $D=4$ .

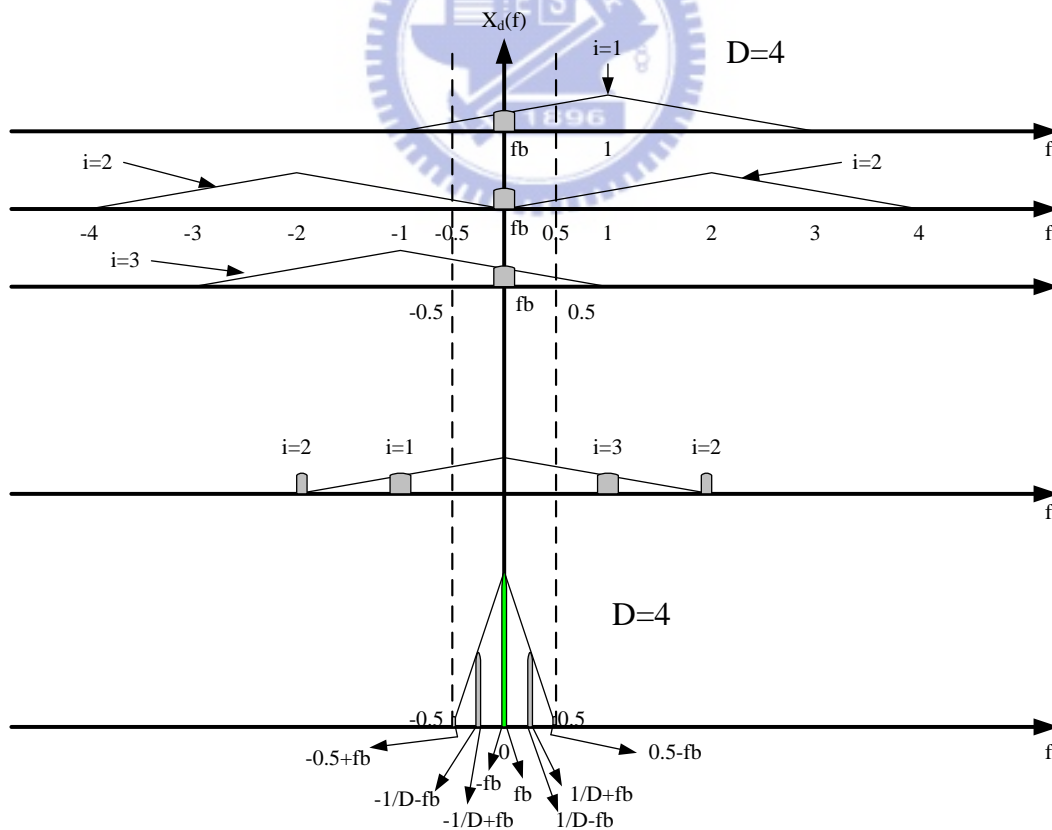


Figure 1.19 The bands over  $[-0.5, 0.5]$  overlap with wanted signal band are folding-bands, which are found from the trace-back process in this demonstration.

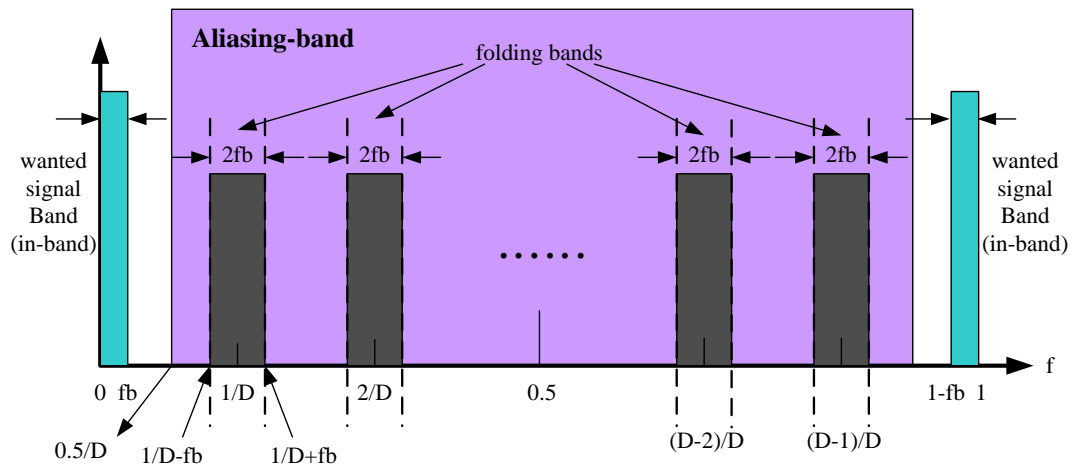


Figure 1.20 There are  $D-1$  folding-bands for down-sampling  $D$ .

Signal (might be quantization noise) on folding-bands would alias to wanted signal (in-band signal), which could not separate and recover by remaining filter, so signal on folding-bands must be attenuated more. Folding-bands of signal are shown in Figure 1.20. Also, these bands are derived from Equation 1.25. Note that the frequency range in Figure 1.20 is  $[0, 1]$ , which is convenient for calculating aliasing power by FFT in matlab and understanding from Equation 1.25. On the other hand, the frequency range  $[0, 1]$  makes the folding-band not split at frequency 0.5, which is the reason why the frequency range  $[-0.5, 0.5]$  is usually chosen to depict signal (make the signal-band continuity at frequency 0).

Now that the folding-bands are known, the parameters in filter design (filter specifications), especially for low-pass FIR filter, could be introduced.

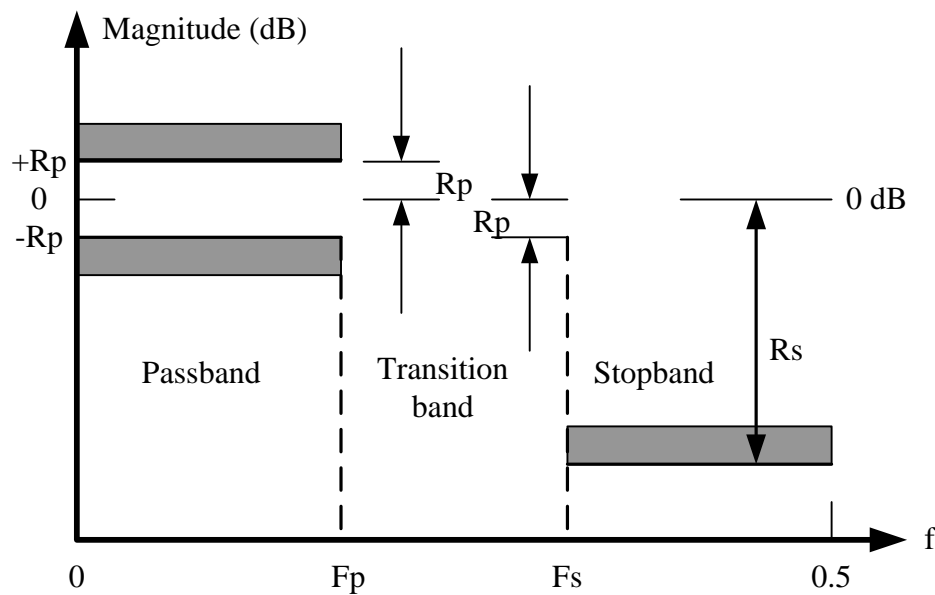


Figure 1.21 Filter specifications for low-pass filter

$R_p$ : pass-band ripple (dB), maximum deviation in pass-band

$R_s$ : stop-band minimum attenuation (dB)

$F_p$ : end frequency of pass-band

$F_s$ : beginning frequency of stop-band

These are parameters in designing low-pass filter, and the decisions of these parameters influence the aliasing power of folding-bands, which are discussed further in Chapter 2.

### 1.3 A Brief Introduction of Proposed Solution

As mentioned in motivation (Section 1.1), the high order FIR filter, which comprises many multipliers, adders and registers, dominates large silicon area in decimator of the sigma-delta A/D converter. A technique, time-multiplexing (folding), could be used to reduce the number of functional units (such as multipliers and adders), so as to minimize the silicon area of integrated circuits. The basic idea of folding is to execute multiple algorithm operations on a single functional unit by time-multiplexing (to finish the same operations by more clock cycles using fewer functional units), so the number of functional units is reduced. For example, it could be time-multiplexed as one multiplication operation finished in each cycle using 100 times faster clock, which only demand one multiplier, if 100 multiplication operations are required to finish in one clock cycle. And the technique, folding, is very suitable

for decimator due to the much lower sampling-rate at the input of high order FIR filter, which imply that there are many clock cycles could be used by each sample and a faster clock is not needed.

In order to minimize the silicon area of high order FIR filter, the technique, folding, is used to obtain acceptable minimum multipliers and adders. Furthermore, the transposed-form structure has been adopted to reduce half registers, which could merge together in polyphase decomposition. Now, the basic idea to obtain minimum hardware is achieved. However, it is hard to use folding technique to transposed-form structure with acceptable power consumption. This thesis proposed a design methodology based on transposed-form folding, which change the computation procedure to preserve the half register benefit and maintain lower power consumption by using extra control circuits, for FIR filter with polyphase decomposition.

For the programmable decimation ratio requirement, IIR-FIR structure of comb filter is adopted in the first stage of decimator to ease the design of the different low-pass filter spectrums, which are different in pass-band edge. Moreover, the pass-band drop of designed filter spectrums by IIR-FIR comb filter could be compensated by the same compensation filter. These means that no extra filter hardware is needed to produce different low-pass filter spectrum to prevent aliasing.

This thesis focuses on comparison of different folding implementation in area, power, speed and etc. So, the specification of sigma-delta modulator is not a main issue wanted to discuss in this thesis. A case of 1-bit, OSR 128 and 64, 4<sup>th</sup>-order SDM is chosen as a specification of the sigma-delta A/D converter. And then the decimator would be designed to meet the requirements of that SDM specification. Usually, the designed decimator could be used for most SDM specifications, lower than 4<sup>th</sup>-order and OSR=128 or 64. As a result of that one bit SDM A/D converter don't require D/A circuit, one bit SDM is often chosen to implement. These make designed decimator more useful.

## **1.4 Thesis Organization**

In Chapter 2, architecture (number of stages and decimation ratio of each stage) and filters specification of decimator are decided to meet the requirements of determined sigma-delta modulator, 1-bit 4<sup>th</sup>-order, OSR=128,64 SDM.

Hardware implementation methods of the decimator are discussed and compared

in Chapter 3. Also the advantages of this thesis proposed implementation method for high order FIR filter are shown in Chapter 3. Finally, it has been verified that the proposed implementation methodology is suitable for any order, number of quantizer bit, and OSR SDM as well as its advantages would still exist.

In Chapter 4, testing environment and instrument are introduced. And function testing results and electrical characteristic of decimator, which is fabricated in TSMC 0.18um CMOS mixed signal RF general purpose MiM Al 1P6M process, would be plotted and summarized.

Finally, conclusions of this work are given in Chapter 5.



---

# CHAPTER

# 2

---

## Decimator Architecture and Design

---

In this chapter, decimator architecture (number of decimation stages and decimation ratio of each stage) and decimation filter specifications (order of each filter according to designed pass-band ripple, pass-band edge, stop-band attenuation, and stop-band edge; these definitions see Figure 1.21.) are decided so as to meet the SNR requirements of pre-defined sigma-delta modulator specifications (1-bit 4<sup>th</sup>-order, OSR=128, 64 SDM) with efficient decimator hardware in terms of functional units and power consumption.

### 2.1 Considerations about SDM Quantization Noise

These in-band quantization noise power of 1-bit 4<sup>th</sup>-order, OSR=128, 64 SDM are  $9.5134 \times 10^{-18}$  and  $4.8711 \times 10^{-15}$  (see section 1.2.2 and Equation 1.22, using  $V=1$  for convenience), respectively. And sinusoidal signal power with maximum amplitude, namely  $V=1$ , is 0.5. As a result of that, the ideal peak SNR are  $10\log(0.5/9.5134 \times 10^{-18})=167.2$  dB and  $10\log(0.5/4.8711 \times 10^{-15})=140.1$  dB for OSR=128 and OSR=64, respectively. However, the quantization noise power at SDM output (before decimator) is (similar to Equation 1.22):

$$\begin{aligned}
 \text{QN} &= \int_{-0.5}^{0.5} P_e(f) |H_e(f)|^2 df \\
 &= \frac{(\frac{2}{2N})^2}{12} 2^{2k} \int_{-0.5}^{0.5} \sin^{2k}(\pi f) df \quad (2.1)
 \end{aligned}$$



N: number of quantizer-bit

k: order of SDM

$$\int \sin^n u \, du = -\frac{\sin^{n-1} u \cos u}{n} + \frac{n-1}{n} \int \sin^{n-2} u \, du \quad (2.2)$$

The result of Equation 2.1 could be obtained by using integral formula of Equation 2.2 [4].

Furthermore, numerical methods could be used to solve the Equation 2.1 in Matlab for known N and k. In this case (N=1 and k=4), quantization noise power of SDM output calculated in Matlab is 5.83, which is much larger than in-band quantization noise power ( $9.5134 \times 10^{-18}$  with OSR=128 and  $4.8711 \times 10^{-15}$  with OSR=64).

As a result of that, the decimator design procedure must take care of quantity of quantization noise power, especially in folding-bands (alias to in-band in decimation procedure). The target aliasing power in my decimator design procedure is ten times as small as in-band quantization noise power, so the SNR would not degrade more than 0.41 dB ( $10 \log \left( \frac{\text{signal power}}{1.1 * \text{QN}_{\text{in-band}}} \right) = \text{SNR} - 0.41 \text{ dB}$ ) after decimator.

Before proceeding, some concepts of spectrum in discrete-time signal or digital filter must be reminded. For example, shaped quantization noise power spectral density could be shown as Figure 2.1(a) or Figure 2.1(b) or Figure 2.1(c):

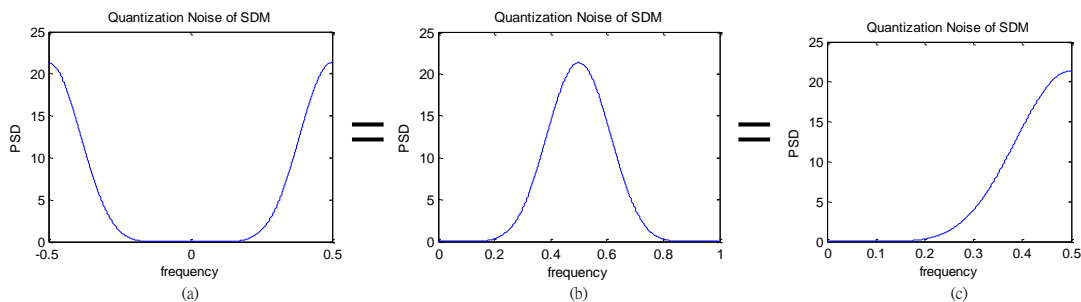


Figure 2.1 (a) frequency over [-0.5, 0.5] (b) frequency over [0, 1] (c) frequency over [0,0.5]. The signals represented by these figure are the same.

Although the graphs of Figure 2.1 are quite different, the signals shown by these figure are the same, which could be recognized over the range [0, 0.5]. Because

spectrum of discrete-time signal is periodically repeated, only one period should be depicted, which means that the ranges over  $[-0.5, 0.5]$  or  $[0, 1]$  could be chosen to be depicted. For real number signal, its spectrum is even function, which means  $X(-f)=X(f)$ , and then only half periodic spectrum is needed to be depicted, i.e. the range over  $[0, 0.5]$ .

In the following design procedure, signal or filter spectrum would be depicted over the range  $[0, 1]$  (like Figure 2.1 (b)), which match the FFT points  $[0, N-1]$  in Matlab and is easy to illustrate folding-bands of signal.

## 2.2 Decimator Architecture

Decimation is often performed in several stages [5], which reduces the number of algorithmic operations per second and the required functional units in decimator, especially for high OSR ( $OSR > 4$ ). First of all, the components of multi-stages decimator must be decided as well as the number of decimation stages. The most efficient choice for first decimation stage is comb filter (also called sinc filter), which don't require multiplier and attenuates aliasing-band signal enough (especially attenuate more for folding-bands signal) [6]. According to analysis in [7], the appropriate decimation ratio of comb filter is  $OSR/4$  for sigma-delta modulation (32 and 16 for  $OSR=128$  and 64, respectively), which results in most efficient algorithmic bit-operations per second considering word-length of comb filter and the required sampling rate at each components of comb filter.

Furthermore, the remaining four times Nyquist rate would be decimated by FIR filters, which could approximate to an ideal low-pass filter that would perfectly preserve in-band signal and exactly remove out-of-band quantization noise due to its sharp roll-off if the order of FIR filter is high enough. According to analysis in [5], two stages FIR filters structure with each decimation ratio 2 is an efficient implementation for decimating four times Nyquist rate signal.

Three stages decimator architecture is an efficient implementation, which is good enough for attenuating aliasing power to obtain required SNR. However, the pass-band drop due to comb filter is slightly severe, which make the in-band signal perfectly preserved by FIR filters worse, especially order and decimation ratio of comb filter are high. For example, 5<sup>th</sup>-order comb filter with decimation ratio 32 would cause the pass-band drop of OSR-128 signal more than 1 dB, which destroys the effort made by FIR filter in terms of suppressing pass-band ripple. Usually, the compensation filter is introduced and combined with low-pass filter, which means that a single FIR filter in

the 3<sup>rd</sup>-stage of decimator is used to remove quantization noise, prevent aliasing and compensate the pass-band drop due to comb filter. The hardware complexity is not increased; however, the features of compensation filter and low-pass filter slightly conflict with each other over the frequency [0.23, 0.25] because one (the compensation filter) need to go up in magnitude to compensate pass-band drop and the other (the low-pass filter) need to go down to prevent aliasing. As a result of that, the quantization noise over [0.25, 0.27] didn't be removed very well and the pass-band drop also didn't be compensated very well over entire in-band (worse near 0.23 in frequency).

In this work, the compensation filter and the low-pass filter are separated to provide a better solution in terms of suppressing pass-band ripple and attenuating aliasing power.

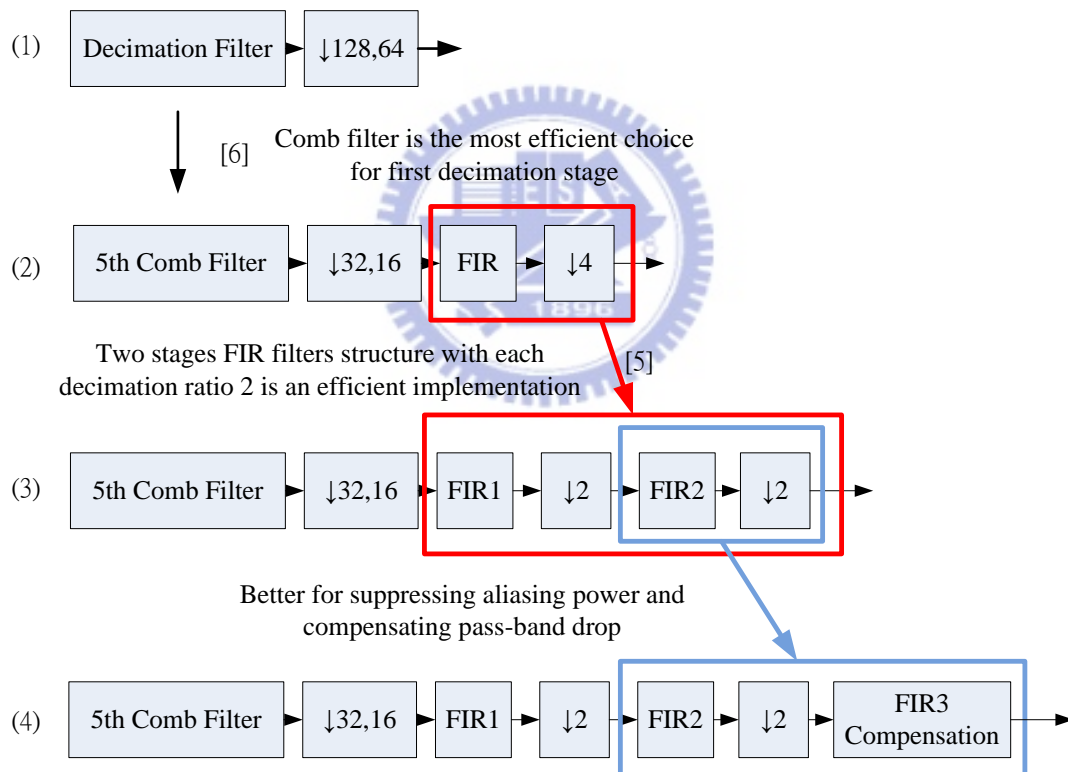


Figure 2.2 Decimator architectures

A numerical demonstration of the different decimator architectures for 1-bit 4<sup>th</sup>-order OSR-128 is shown below:

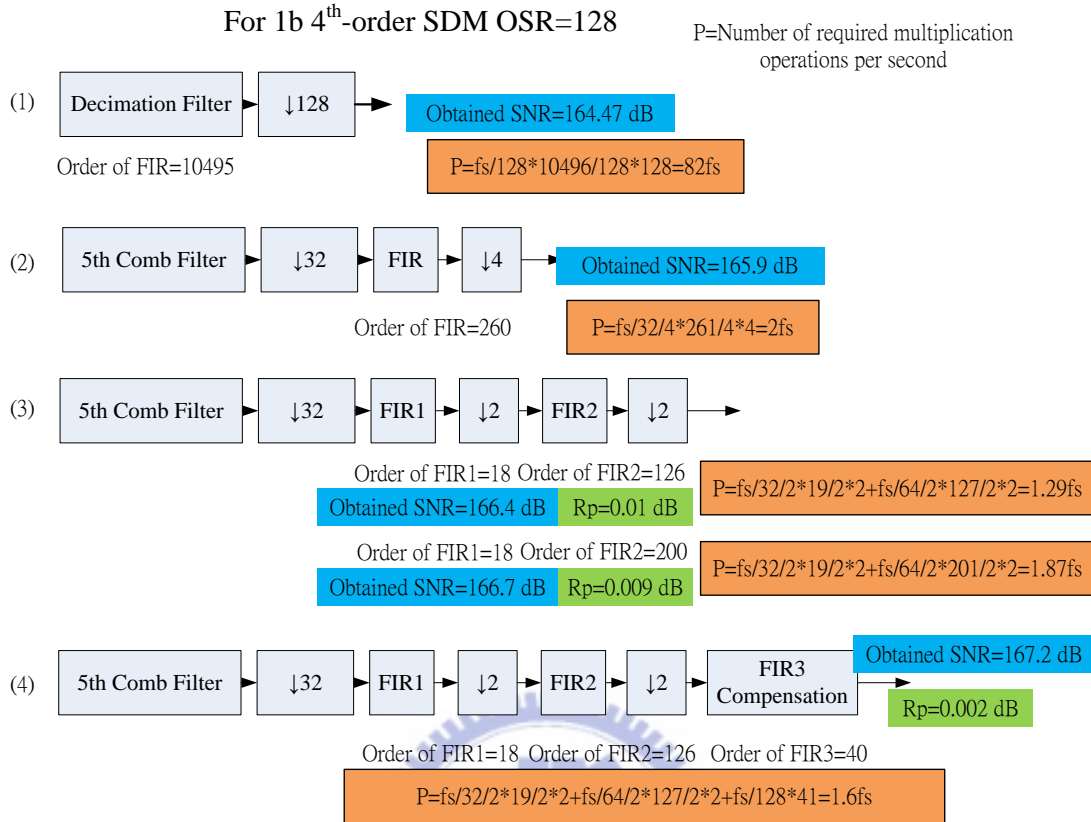


Figure 2.3 For 1-bit 4<sup>th</sup>-order OSR-128 SDM, decimator architectures in terms of SNR, number of required multiplication operations per second and pass-band ripple are shown.

### 2.3 First Decimation Stage

As mentioned in previous section, the features of comb filter, which don't require multiplier, suppress folding-bands signal excellently, attenuate out-of-band signal well and could have large decimation ratio to lower the sampling rate for later FIR filter, are suitable for the first decimation stage operating in the fast sampling rate. The transfer function of comb filter (also called sinc filter) is:

$$H_{\text{comb}}(z) = \frac{1}{D^k} \left( \frac{1-z^{-D}}{1-z^{-1}} \right)^k \tag{2.3}$$

$$= \frac{1}{D^k} \prod_{i=0}^{(\log_2 D)-1} (1+z^{-2^i})^k \tag{2.4}$$

$$= \frac{1}{D^k} \left( \sum_{i=0}^{D-1} z^{-i} \right)^k \tag{2.5}$$

k: order of comb filter

D: decimation ratio

For k=5 and D=32, its zero-pole plot and frequency response are show below:

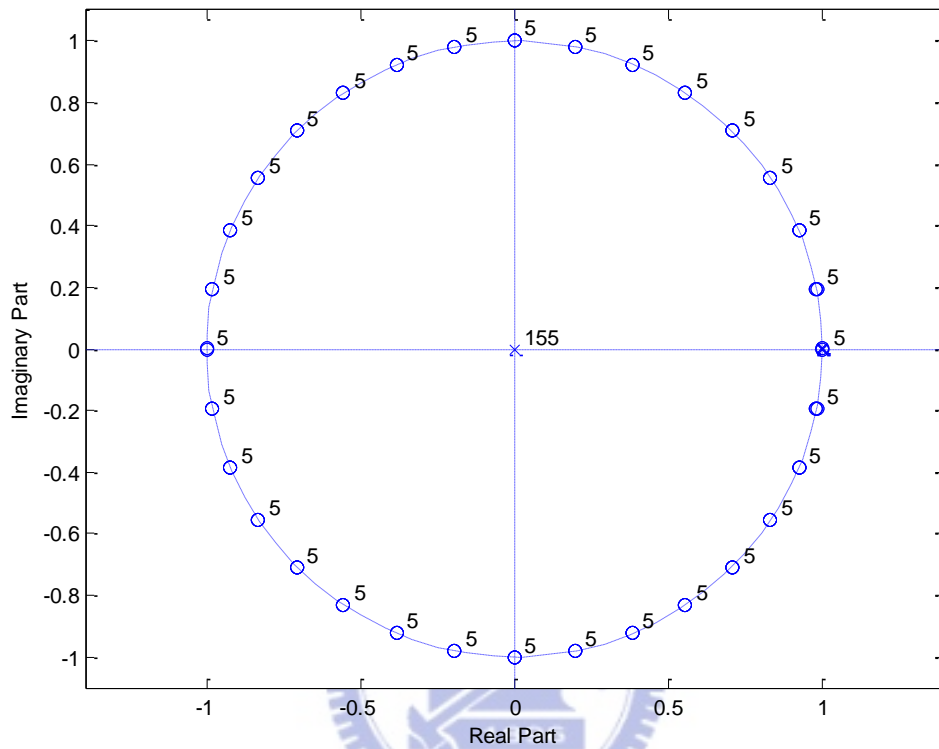


Figure 2.4 zero-pole plot for 5<sup>th</sup>-order comb filter with D=32  
(5\*32 zeros around unit circle, 5 poles in z=1 and the other poles in z=0)

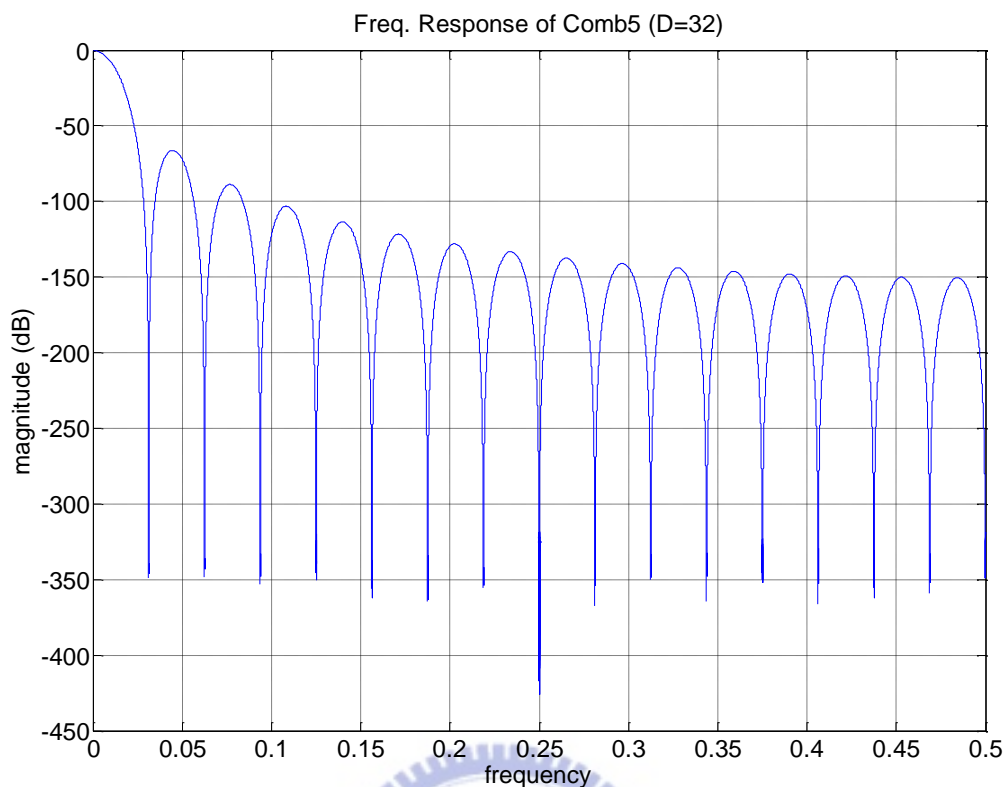


Figure 2.5 Magnitude frequency response of 5<sup>th</sup>-order comb filter with D=32 (half periodic spectrum have been depicted, so there are D/2=16 notches in spectrum)

Recent researches on first decimation stage filter are also based on comb filters, which would be introduced in following section.

### 2.3.1 Introduction to Modified Comb Filters

Novel decimation schemes based on comb filter are proposed by [8]-[10]. The basic idea is to rotate the zeros of comb filter to obtain a better rejection around folding-bands. As seen in Figure 2.4, there are 5 zeros in the same position for 5<sup>th</sup>-order comb filter. If the zeros could be distributed around folding-bands (not all zeros in the same position (middle of folding-band)), a better rejection around folding-bands can be achieved.

The transfer function of counterclockwise rotated 1<sup>st</sup>-order comb filter could be defined as [9]:

$$H_{q^+}(z) = \frac{1}{D} \frac{1 - z^{-D} e^{j\alpha D}}{1 - z^{-1} e^{j\alpha}} \quad (2.6)$$

And the transfer function of clockwise rotated 1<sup>st</sup>-order comb filter could be defined as:

$$H_{q^-}(z) = \frac{1}{D} \frac{1 - z^{-D} e^{-j\alpha D}}{1 - z^{-1} e^{-j\alpha}} \quad (2.7)$$

In order to obtain real number coefficient, the Equation 2.6 and Equation 2.7 must be combine together to form the transfer function:

$$H_q(z) = H_{q^+}(z)H_{q^-}(z) = \frac{1}{D^2} \frac{1 - 2\cos(\alpha D)z^{-D} + z^{-2D}}{1 - 2\cos(\alpha)z^{-1} + z^{-2}} \quad (2.8)$$

The transfer function of modified comb filter consists of  $H_{\text{comb}}(z)$  and  $H_q(z)$ . The k<sup>th</sup>-order modified comb filters (MCF) mean that there are k zeros distributed around the folding-band. And the transfer functions of different order MCF are defined as follow:

$$H_{\text{MCF}3}(z) = H_{\text{comb}1}(z)H_q(z) \quad (2.9)$$

$$H_{\text{MCF}4}(z) = H_{\text{comb}2}(z)H_q(z) \quad (2.10)$$

$$H_{\text{MCF}5}(z) = H_{\text{comb}1}(z)H_{q_1}(z)H_{q_2}(z) \quad (2.11)$$

$$H_{\text{MCF}6}(z) = H_{\text{comb}2}(z)H_{q_1}(z)H_{q_2}(z) \quad (2.12)$$

MCF3, MCF4, MCF5 and MCF6 denote 3<sup>rd</sup>-order, 4<sup>th</sup>-order, 5<sup>th</sup>-order and 6<sup>th</sup>-order modified comb filters, respectively. Also,  $H_{\text{comb}1}(z)$  and  $H_{\text{comb}2}(z)$  denote the transfer function of 1<sup>st</sup>-order and 2<sup>nd</sup>-order comb filter, respectively.

A zero-pole plot of 3<sup>rd</sup>-order modified comb filter (MCF3) is shown below:

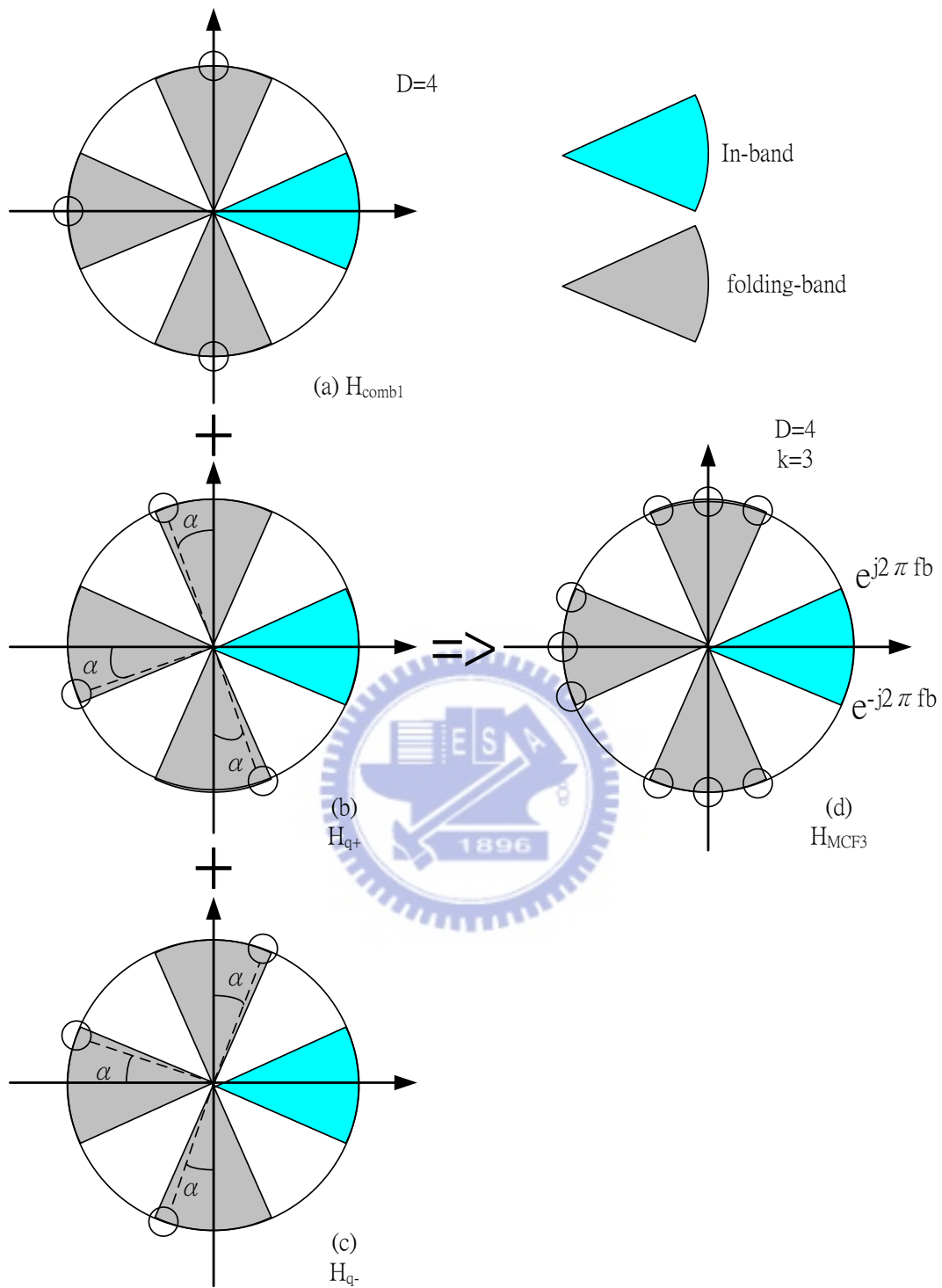


Figure 2.6 zero-pole plot of (a) 1<sup>st</sup>-order comb filter (b) counterclockwise rotated of 1<sup>st</sup>-order comb filter (c) clockwise rotated of 1<sup>st</sup>-order comb filter (d) 3<sup>rd</sup>-order modified comb filter, with  $D=4$ .



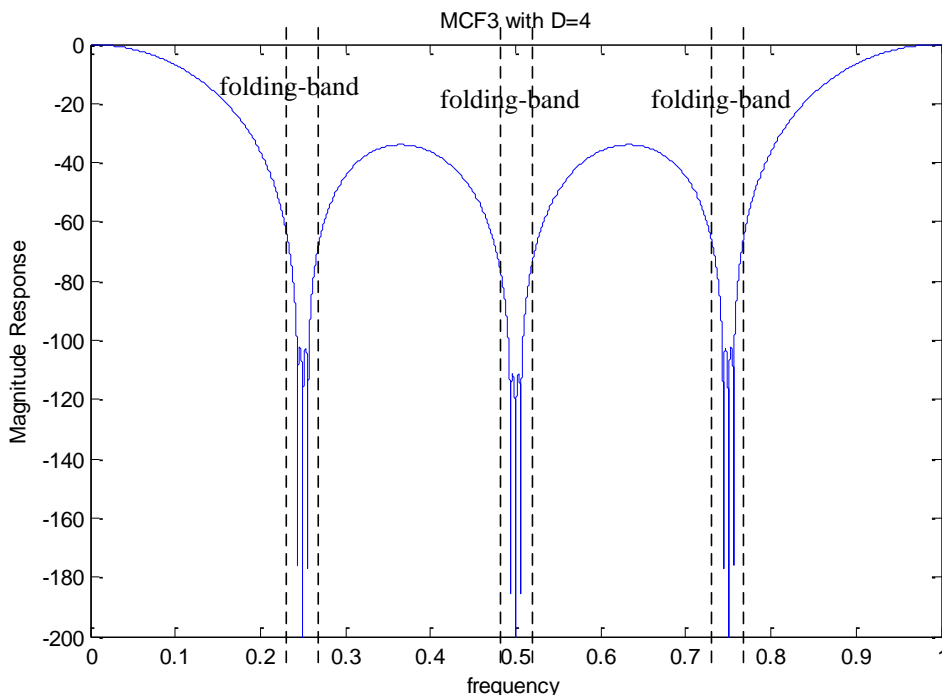


Figure 2.7 Magnitude response (dB) of 3<sup>rd</sup>-order modified comb filter (MCF3) with D=4

The optimized rotated angle  $\alpha$  to obtain maximum rejection around folding-band has been presented by [9]. Now,  $f_b$  denotes the in-band edge. The optimized rotated angle  $\alpha$  can be rewrite as follow:

$$\alpha = q2\pi f_b \tag{2.13}$$

then optimized q for different orders MCF are [9]:

Table 2.1 Optimized rotated angle  $\alpha$  represented by q, which is independent to  $f_b$ .

$H_{MCFk}$	q1	q2	Gain (dB)
$H_{MCF3}$	0.78	-	8 dB
$H_{MCF4}$	0.85	-	13 dB
$H_{MCF5}$	0.54	0.93	18 dB
$H_{MCF6}$	0.63	0.92	23 dB

$$\text{Gain} = \frac{\text{quantization noise of SDM in folding bands after comb filter}}{\text{quantization noise of SDM in folding bands after MCF}}$$

$$= \frac{\sum_{i=1}^{D-1} \int_{\frac{i}{D}-fb}^{\frac{i}{D}+fb} |H_{\text{comb}-k}(f)|^2 P_{e\text{SDM}}(f) df}{\sum_{i=1}^{D-1} \int_{\frac{i}{D}-fb}^{\frac{i}{D}+fb} |H_{\text{MCF}-k}(f)|^2 P_{e\text{SDM}}(f) df} \quad (2.14)$$

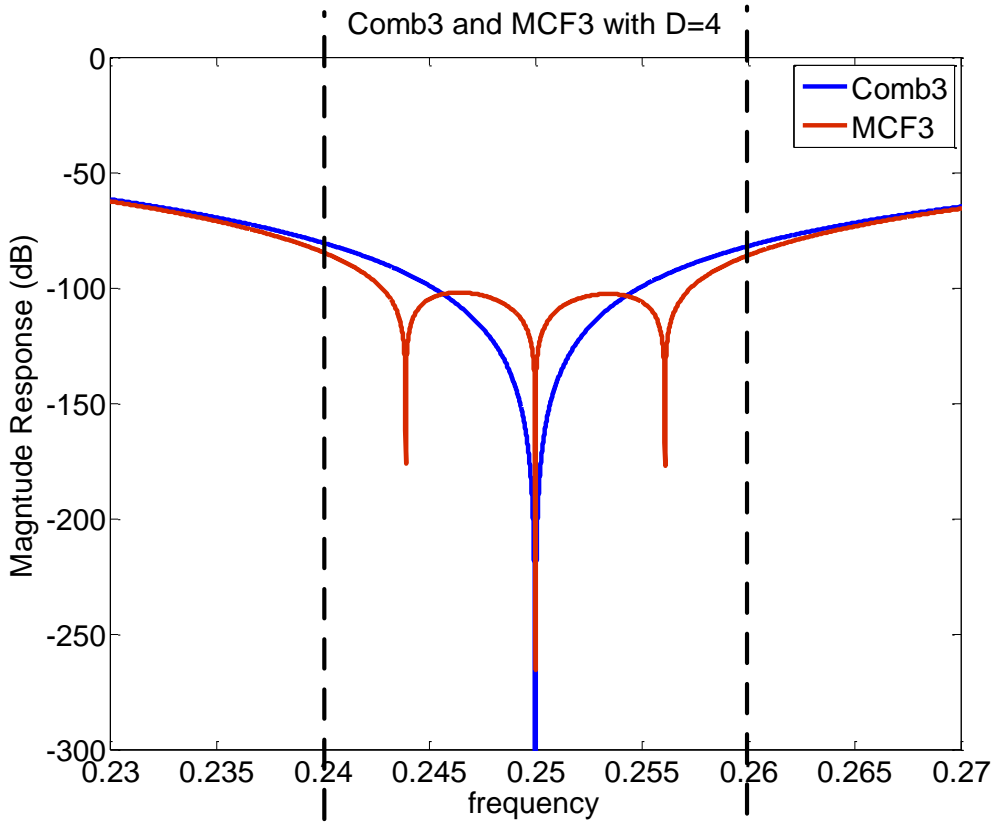


Figure 2.8 Magnitude responses of Comb3 and MCF3 in folding-band. The MCF3 can suppress more quantization noise power in folding-band.

### 2.3.2 Stage1 Design

As mentioned in the introduction of Chapter 2, aliasing power is the main concern in the design procedure of decimation filter. The quantization noise are  $9.5134 \times 10^{-18}$  and  $4.8711 \times 10^{-15}$  for 1-bit 4<sup>th</sup>-order, OSR=128, 64 SDM, respectively. The aliasing power of first decimation filters must be smaller than the ideal in-band quantization noise power of SDM. And four times Nyquist rate would be left for later FIR filters to decimate [7], i.e.  $D=OSR/4$ .

Table 2.2 Ideal in-band quantization noise power

1-bit 4 <sup>th</sup> -order SDM	OSR=128	OSR=64
<b>ideal in-band noise power</b>	$9.5134 \times 10^{-18}$	$4.8711 \times 10^{-15}$

$$\text{Aliasing Power1} = \sum_{i=1}^{D-1} \int_{\frac{i}{D}-fb}^{\frac{i}{D}+fb} |H_{\text{First decimation filter}}(f)|^2 P_{\text{eSDM}}(f) df \quad (2.15)$$

Aliasing Power1 is the SDM quantization noise power in folding-bands filtered by first decimation filter, which would alias to in-band signal and would not be removed by latter filters.

Table 2.3 Aliasing power using comb filter and MCFs with different order for 1-bit 4<sup>th</sup>-order SDM, OSR=128 and OSR=64 (calculating in Matlab).

		<b>D=32 for OSR=128</b>		<b>D=16 for OSR=64</b>	
		Comb Filter	MCF	Comb Filter	MCF
aliasing power1	order=3	1.1549e-012	1.8162e-013	7.3879e-011	1.1631e-011
	order=4	2.4977e-016	2.0120e-017	6.1846e-014	4.9951e-015
	order=5	3.3085e-019	5.7030e-021	1.6880e-016	2.9137e-018
	order=6	3.0518e-021	1.9477e-023	1.6022e-018	1.0238e-020

It is obvious that the modified comb filters have more rejection around folding-bands to obtain smaller aliasing power than the same order comb filter as [8]-[10] claimed. And the minimum required orders of MCF and comb filter are both 5 to keep the aliasing power of first decimation stage smaller than ideal in-band SDM QN power. The requirement of aliasing power in this work is to keep it 10 times smaller than in-band QN power.

Both 5<sup>th</sup>-order comb filter and 5<sup>th</sup>-order modified comb filter (MCF) meet the requirement of aliasing power. And modified comb filter is better in suppressing aliasing power. However, MCF requires multiplication operations, which make the hardware complexity increase much. Furthermore, 6<sup>th</sup>-order comb filter could suppress the aliasing power to the same level suppressed by 5<sup>th</sup>-order MCF and overhead of 6<sup>th</sup>-order comb filter compared with 5<sup>th</sup>-order comb filter are two adders (one addition and one subtraction) and two word registers, whose hardware complexity is much lower than MCF. Moreover, the additional suppressed aliasing

power by MCF is hard to maintain because the aliasing power of latter decimation stages are in the same level with 5<sup>th</sup>-order comb filter. Finally the SNR of SDM improved by MCF compared with comb filter is:

$$\begin{aligned}
 & \text{SNR}_{\text{MCF}} - \text{SNR}_{\text{comb}} \\
 &= 10 \log \left( \frac{0.5}{9.5134 \times 10^{-18} + 5.7030 \times 10^{-21}} \right) \\
 & - 10 \log \left( \frac{0.5}{9.5134 \times 10^{-18} + 3.3085 \times 10^{-19}} \right) \\
 &= 167.2 \text{ dB} - 167.06 \text{ dB} = 0.14 \text{ dB}
 \end{aligned}
 \tag{2.16}$$

$$\text{SNR} = 10 \log \left( \frac{\text{signal power}}{\text{ideal inband QN} + \text{aliasing power} \cdot 1} \right)
 \tag{2.17}$$

So, comb filter is still an efficient implementation for first decimation stage.

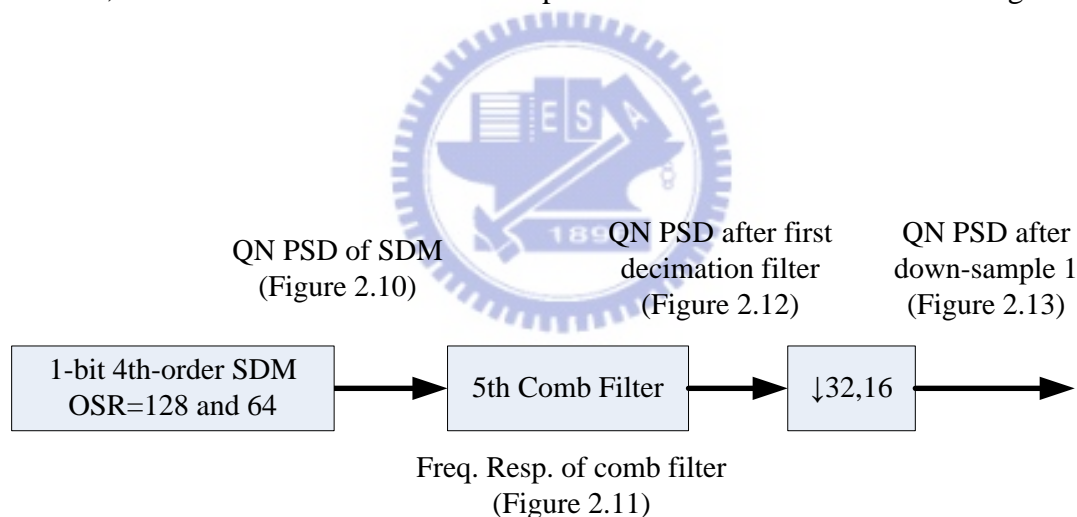


Figure 2.9 The flow for quantization noise power spectral density of SDM in first decimation stage

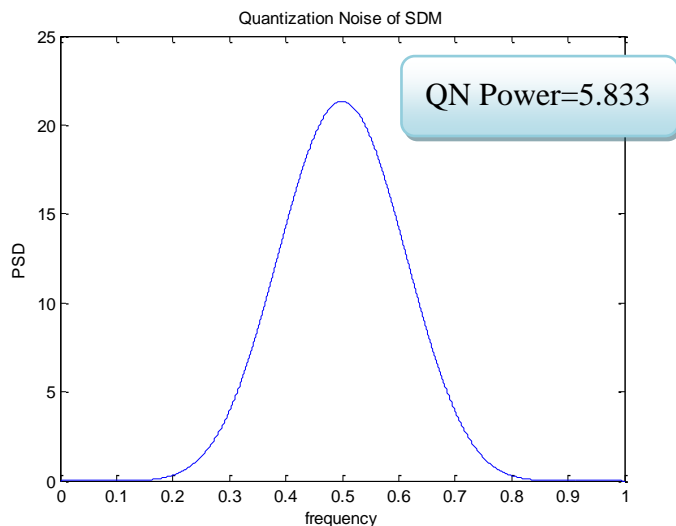


Figure 2.10 Quantization noise power spectral density of 1-bit 4<sup>th</sup>-order sigma-delta modulator

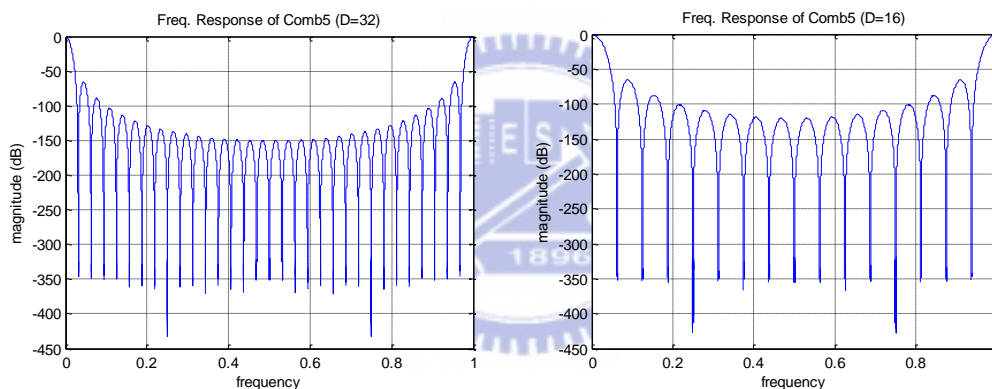


Figure 2.11 First decimation filter (comb filter with D=32 for the left graph and 16 for the right graph, respectively). Folding-bands of these two magnitude response are both in the notch position, each D-1 folding-bands

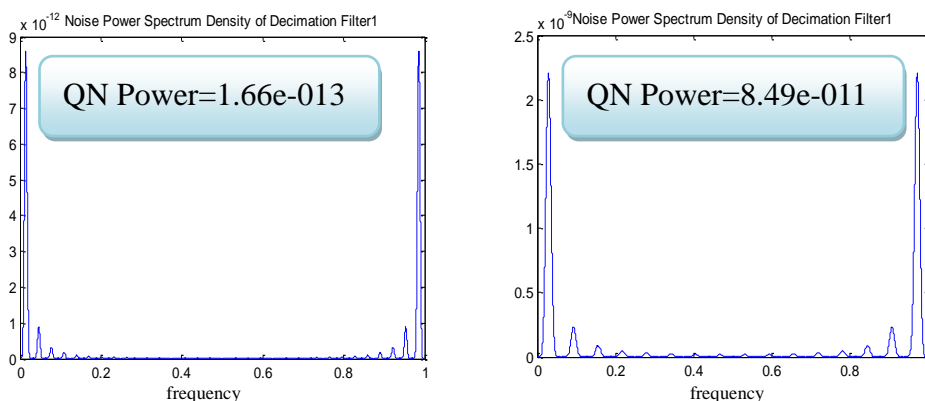


Figure 2.12 Quantization noise PSD after first decimation filter

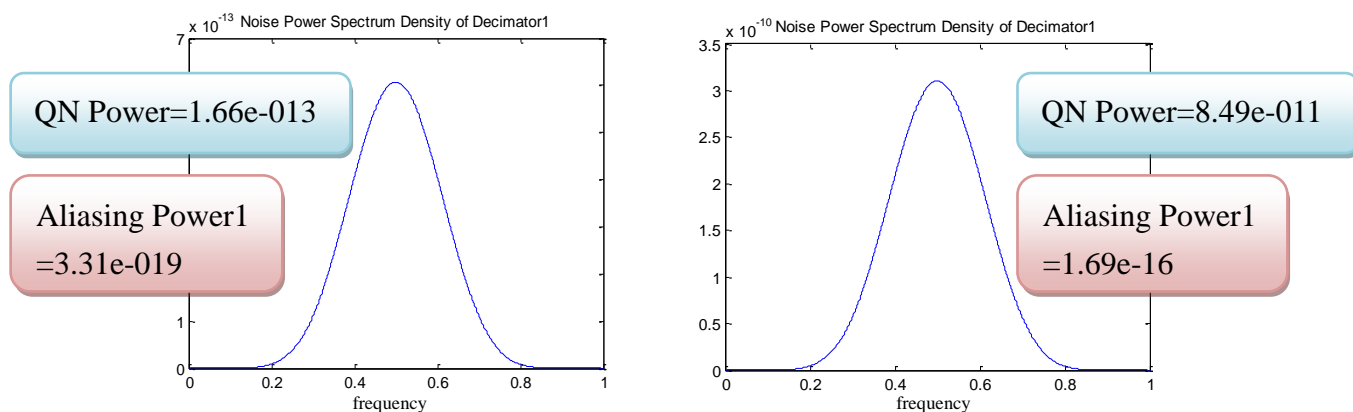


Figure 2.13 Quantization noise PSD after down-sampling

## 2.4 Decimation FIR Filters

Although comb filter don't require multiplication operations and provides excellent rejection around folding-bands, the disadvantages of comb filter are non-flat in-band (signal band) and gradual roll-off. So, further filters are needed to compensate the in-band signal and to remove the remaining quantization noise near in-band edge.

FIR low-pass filter is a sharp roll-off low-pass filter which could remove quantization noise near in-band edge as well as provides small pass-band ripples. These desired features demand many addition and multiplication operations. Furthermore, FIR filter can be designed to compensate the pass-band drop due to comb filter. The main advantage of FIR filters (finite-impulse response filters) compared with IIR filters (infinite-impulse response filters) is the linear phase characteristic, which is important to most applications.

The transfer-function of  $N^{\text{th}}$ -order FIR filter is:

$$H_{\text{FIR}}(z) = \sum_{i=0}^N b_i z^{-i} \quad (2.18)$$

$N$ = order of FIR filter

So, there are  $(N+1)$  coefficients (also called  $N+1$  taps).

The following decimation FIR filters in each decimation stage ( $2^{\text{nd}}$  and  $3^{\text{rd}}$ ) are designed to keep the aliasing power smaller than ideal in-band quantization noise power. And the  $4^{\text{th}}$ -stage of decimator is designed to compensate the pass-band drop due to comb filter.

### 2.4.1 Stage2 Design

The folding-band for 2<sup>nd</sup> decimation stage is shown below:

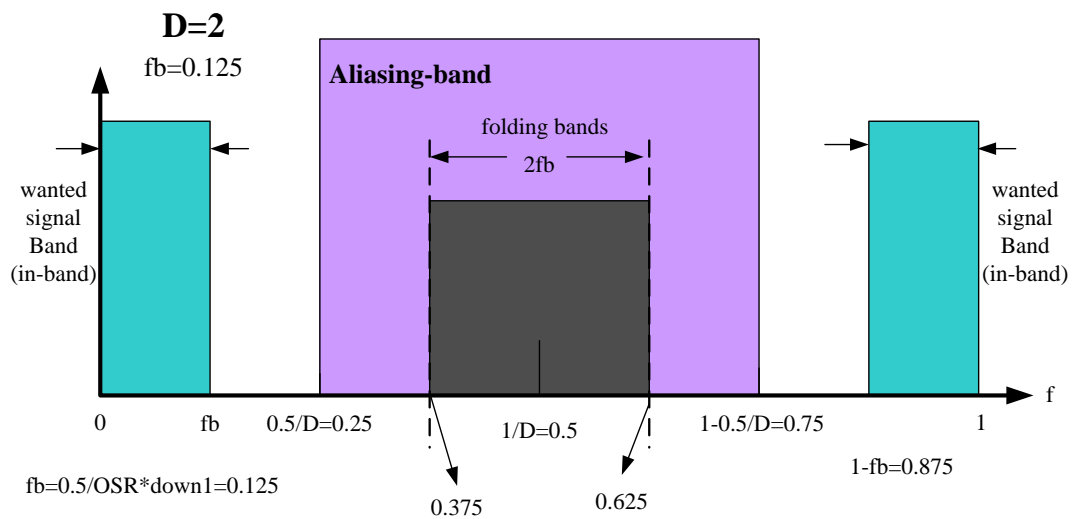


Figure 2.14 Folding-band for 2<sup>nd</sup> decimation stage

For 2<sup>nd</sup> decimation stage,  $D=2$  (down-sampling=2) and  $fb=0.5/OSR*down1=0.5/128*32$  or  $0.5/64*16=0.125$ , folding-band could be plotted according to Figure 1.20.

From the above figure, pass-band edge  $F_p$  must be chosen larger than  $fb$  (0.125) in the left half spectrum to preserve in-band signal and stop-band edge  $F_s$  must be chosen smaller than 0.375 in the left half spectrum to reject noise in folding-band. In order to completely remove quantization noise in folding-band, a slightly smaller  $F_s$  (0.365) is chosen. For releasing the hardware complexity, a possible large transition-band would be used, i.e.  $F_p=0.125$  and  $F_s=0.365$ .

Reminded that ideal in-band quantization noise powers of SDM in Table 2.2 are:

Table 2.4 Quantization noise power which cannot be remove by latter filter

1-bit 4 <sup>th</sup> -order SDM	OSR=128	OSR=64
ideal in-band noise power	9.5134e-018	4.8711e-015
Aliasing power1	3.3085e-019	1.6880e-016

Table 2.5 Aliasing power2 with different stop-band attenuation

		OSR=128	OSR=64	Require FIR order
Aliasing Power2	Rs=40 dB	4.8495e-018	2.4829e-015	14
	Rs=50 dB	8.5409e-019	4.3728e-016	16
	Rs=60 dB	1.6135e-019	8.2605e-017	17
	Rs=70 dB	2.6691e-020	1.3665e-017	18
	Rs=80 dB	1.0944e-021	5.6031e-019	20

In order to keep the total aliasing power (aliasing power1+aliasing power2) 10-times smaller than in-band noise power, the minimum stop-band attenuation (Rs) is 60 dB. However, even order is required to obtained coefficient symmetric for polyphase decomposition. So, the minimum stop-band attenuation (Rs) 70 dB is chosen to obtain even FIR filter order (18) (with pass-band ripple Rp=0.0001 dB, which is not crucial for SNR, almost don't affect FIR filter order and is already smaller than the compensated pass-band ripple due to comb filter).

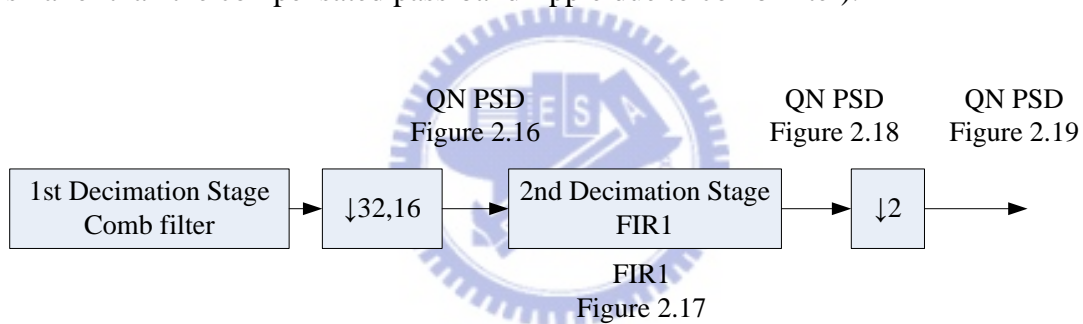


Figure 2.15 Flow for quantization noise power spectral density in 2<sup>nd</sup> decimation stage

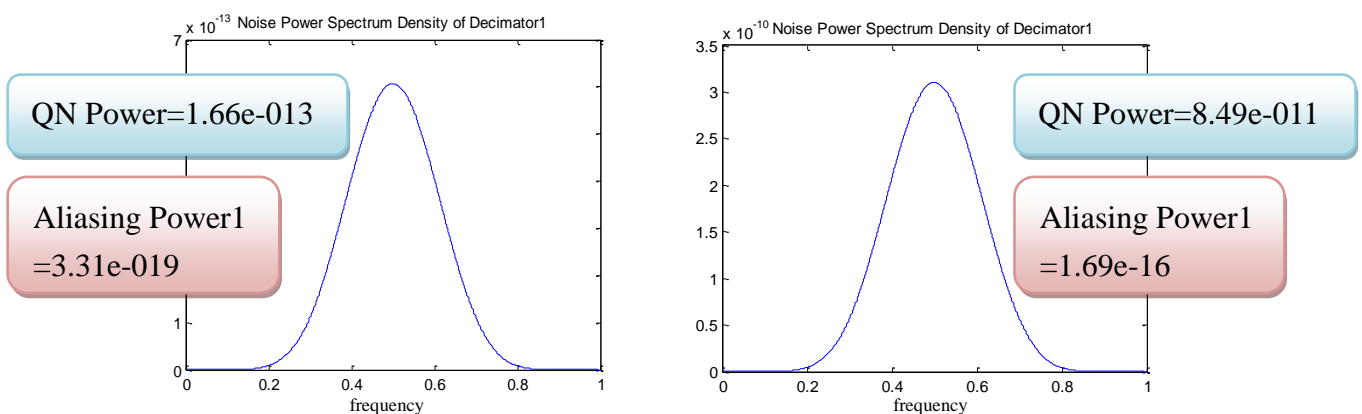


Figure 2.16 Quantization noise PSD after down-sampling1



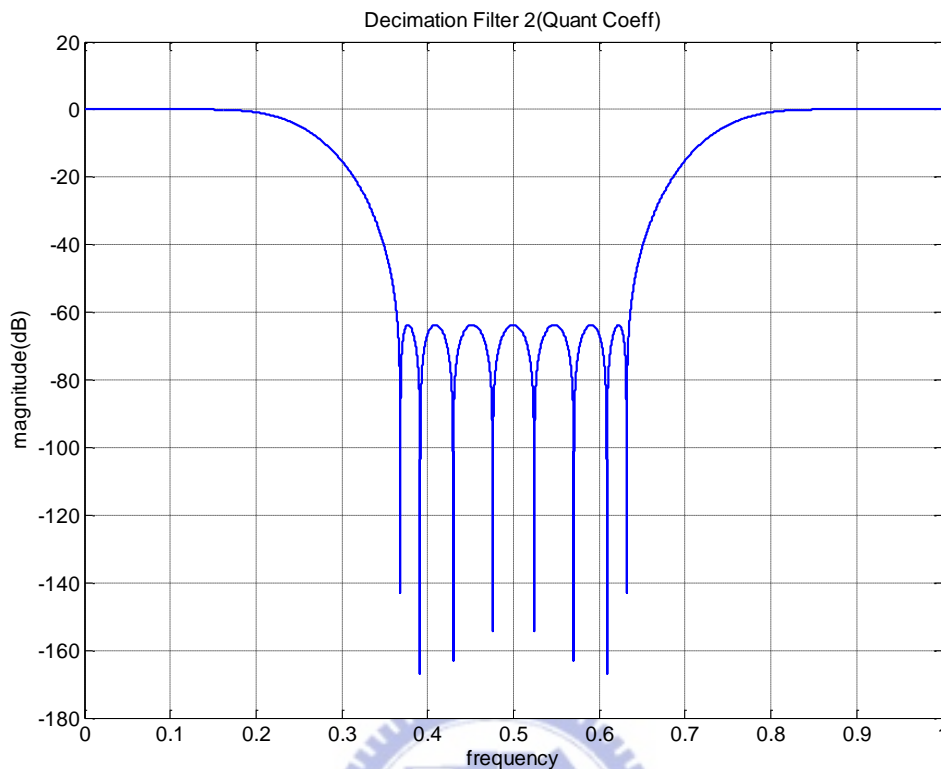


Figure 2.17 Magnitude response of 2<sup>nd</sup> stage decimation filter (FIR1) with quantized coefficients

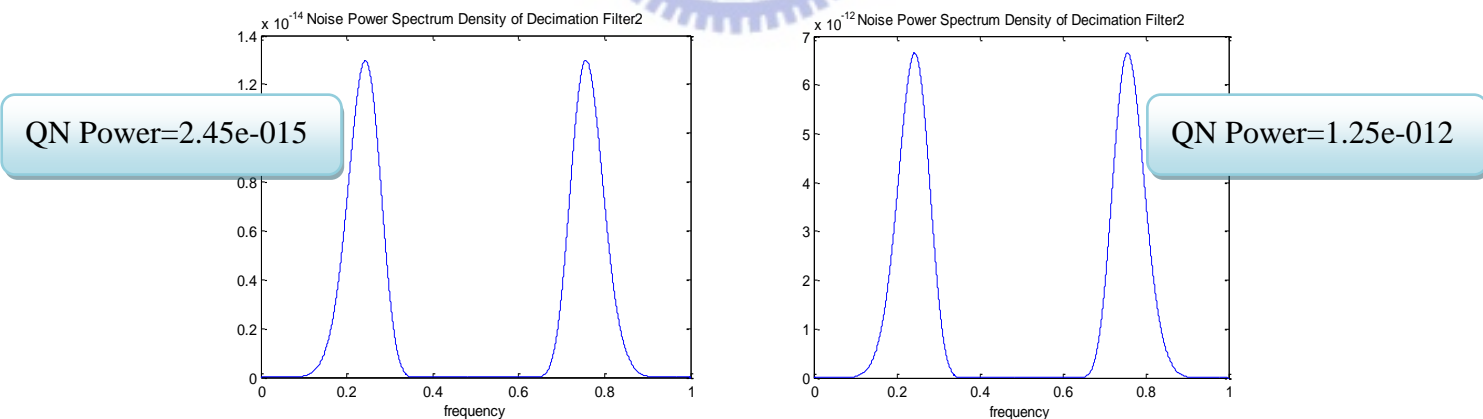


Figure 2.18 Quantization noise PSD after 2<sup>nd</sup> decimation filter

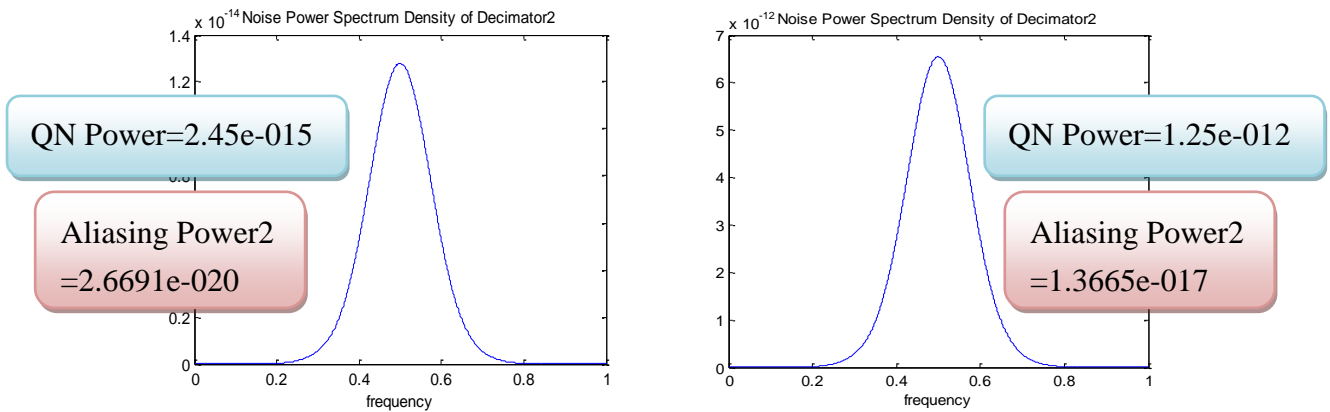


Figure 2.19 Quantization noise PSD after down-sampling 2

### 2.4.2 Stage3 Design

For final decimation, the folding-band is equal to aliasing band shown as follow:

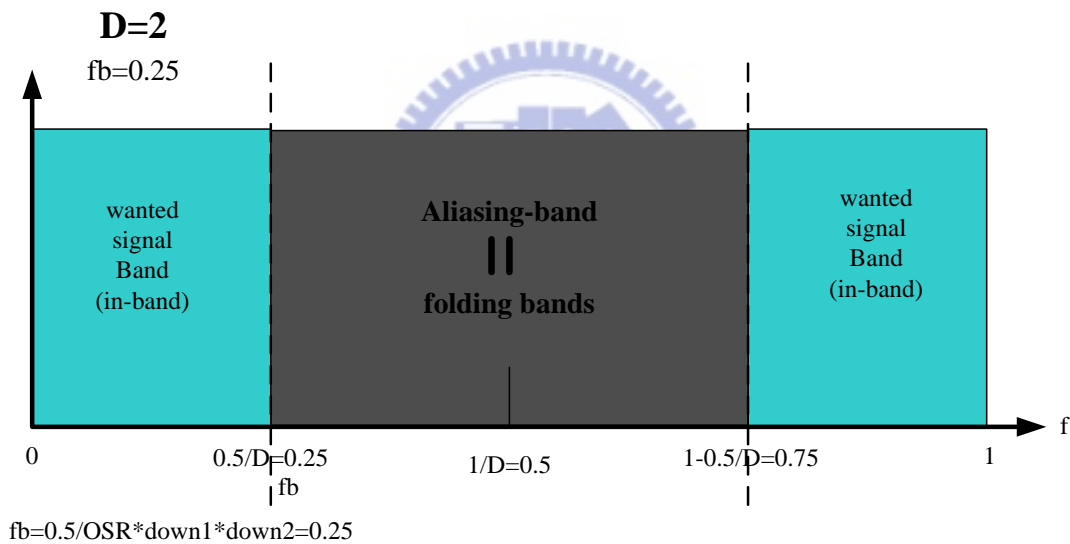


Figure 2.20 Folding-band of 3<sup>rd</sup> decimation stage

In order to preserve signal, pass-band edge ( $F_p$ ) must be chosen larger than 0.25 in the left half spectrum. Also, the stop-band edge ( $F_s$ ) must be chosen smaller than 0.25 in the left half spectrum to suppress noise power around folding-band. The only solution is  $F_p = F_s$  for low-pass filter, which require  $F_p \leq F_s$ . However, that is impossible because it requires infinite hardware to accomplish  $F_p = F_s$  (ideal low-pass filter). So, a transition band is allowed to relax the hardware with acceptable aliasing power.

To keep the affected signal band minimum,  $F_p=0.25-0.5*\text{transition-band}$  and  $F_s=0.25+0.5*\text{transition-band}$  are set. Furthermore, aliasing power is affected by stop-band attenuation ( $R_s$ ) and the width of transition-band both.

Table 2.6 Quantization noise power in in-band after 2<sup>nd</sup> decimation stage

1-bit 4 <sup>th</sup> -order SDM	OSR=128	OSR=64
<b>ideal in-band noise power</b>	9.5134e-018	4.8711e-015
<b>Aliasing power1</b>	3.3085e-019	1.6880e-016
<b>Aliasing power2</b>	2.6691e-020	1.3665e-017

Table 2.7 Aliasing power3 with different stop-band attenuation ( $R_s$ ), given a narrow transition-band (0.02).

		OSR=128	Require FIR order
<b>Aliasing Power3</b>	$R_s=30$ dB	1.3474e-018	151
	$R_s=40$ dB	5.5301e-019	168
	$R_s=50$ dB	3.4788e-019	185
	$R_s=60$ dB	2.4815e-019	202

Stop-band attenuation of 40 dB is good enough; however, for other application like telecommunication, some channel signal near in-band would exist. So, the 60 dB stop-band attenuation is chosen to remove neighbor channel signal as well as the quantization noise of A/D converters. However, the order of FIR is still too high. For folding techniques with OSR=64, a maximum FIR even order is 126. So, a wider transition-band must be used to lower the FIR order with acceptable aliasing power.

Table 2.8 FIR orders and aliasing power3 with different widths of transition-band

		Transition-band	OSR=128	Require FIR order
<b>Aliasing Power3</b>		0.020	2.4815e-019	202
		0.025	3.1171e-019	162
		0.030	3.8010e-019	135
		0.035	4.4907e-019	116

As a result of that, width of transition-band is fine tuned to 0.032 to obtain FIR order 126 and then an acceptable aliasing power, width of transition-band and FIR order are all achieved.

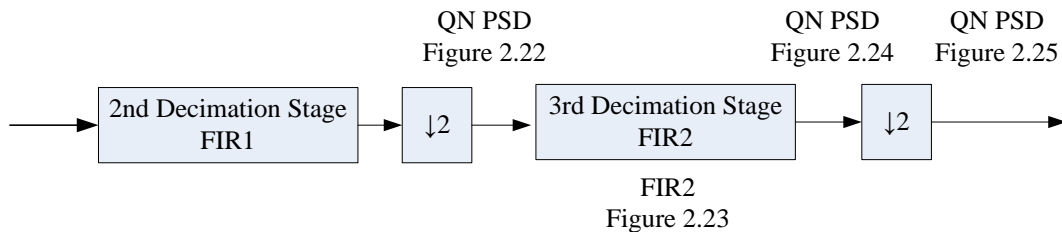


Figure 2.21 Flow for quantization noise power spectral density in 3<sup>rd</sup> decimation stage

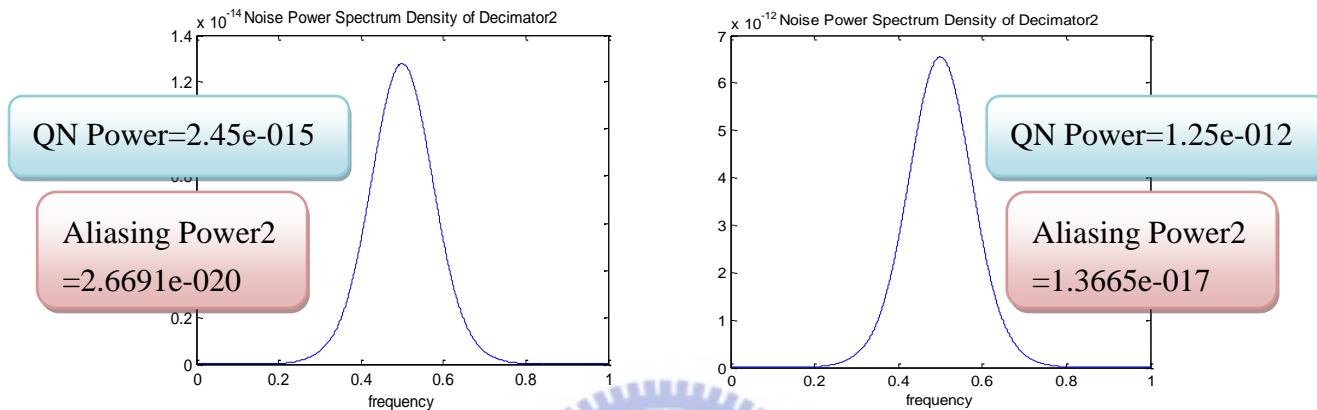


Figure 2.22 Quantization noise PSD after down-sampling 2

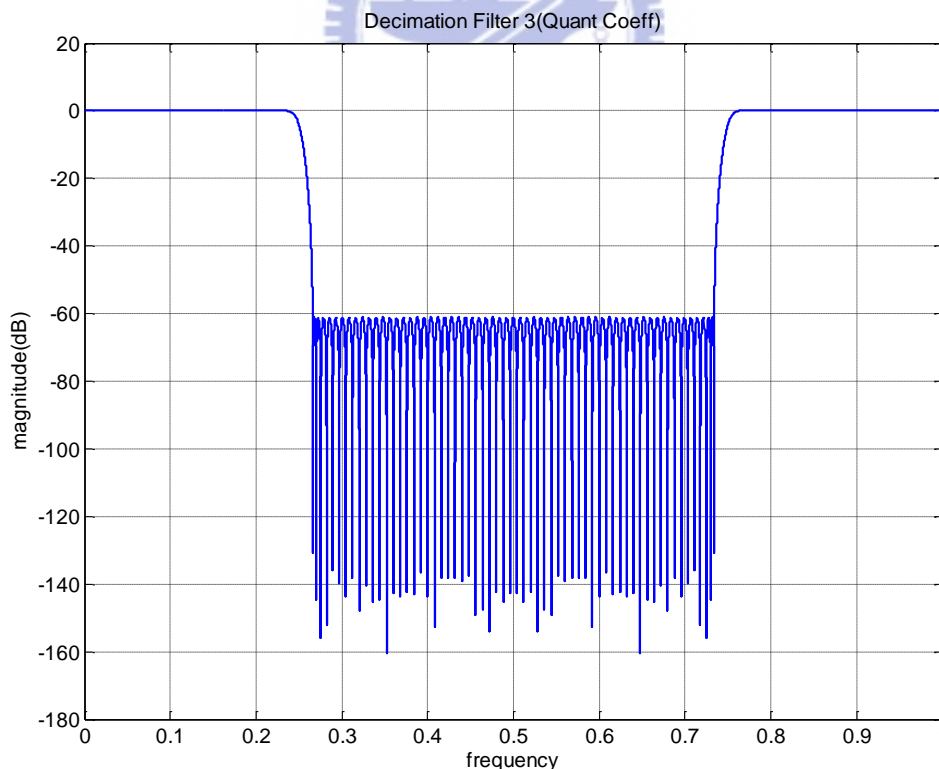


Figure 2.23 Magnitude response of 3<sup>rd</sup> stage decimation filter (FIR2) with quantized coefficients

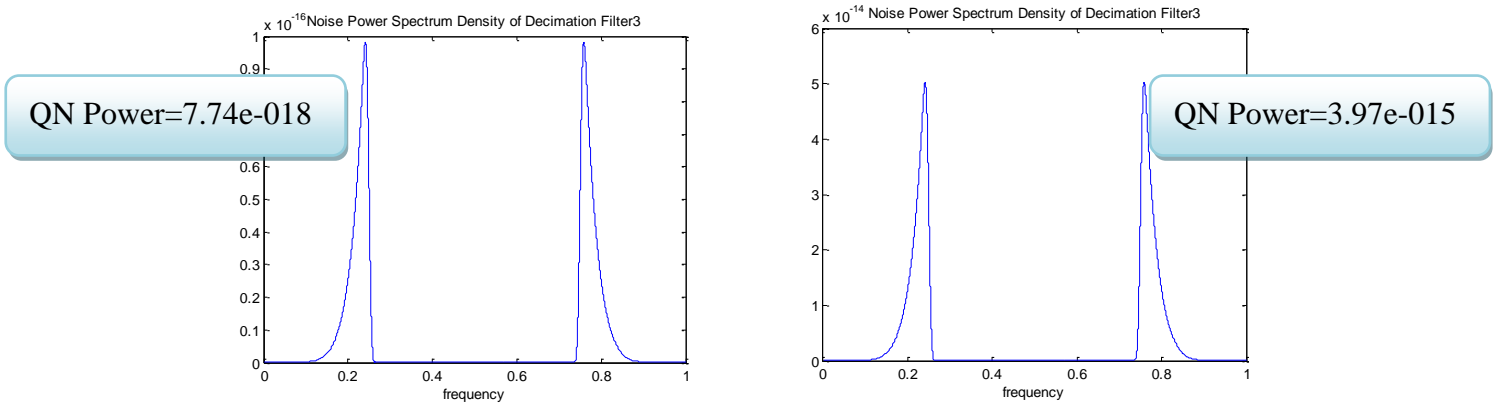


Figure 2.24 Quantization noise PSD after 3<sup>rd</sup> decimation filter

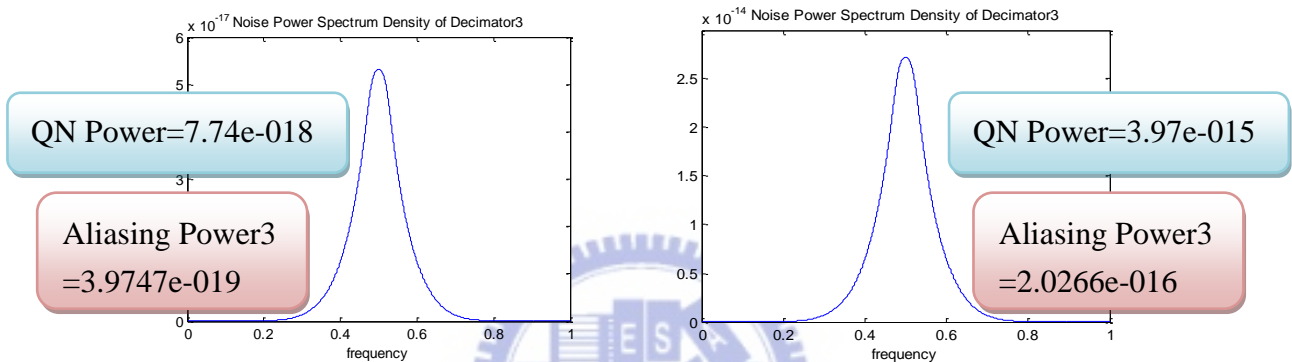


Figure 2.25 Quantization noise PSD after down-sampling

Note that some ideal in-band quantization noise has been removed by the 3<sup>rd</sup> decimation filter due to  $F_p < 0.25$ , which cause total quantization noise power smaller than ideal in-band quantization noise power.

## 2.5 Compensation FIR Filter

The compensation filter is to compensate the pass-band drop of comb filters with  $D=32$  (for  $OSR=128$ ) and  $D=16$  (for  $OSR=64$ ), which are shown below (Figure 2.26 and Figure 2.27):

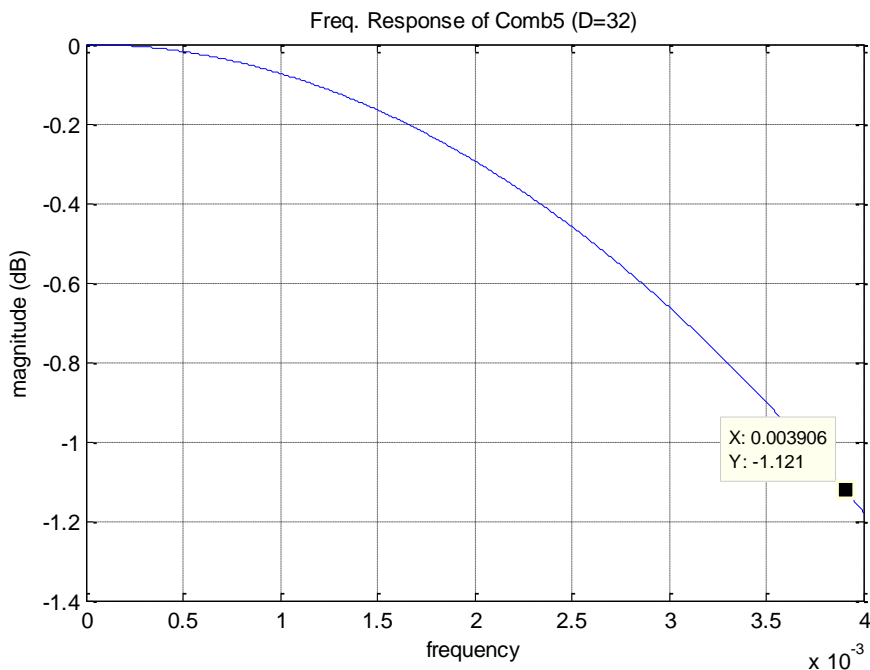


Figure 2.26 Pass-band drop of comb filter with D=32 for OSR=128  
 (fb=0.5/OSR=3.90625e-3)

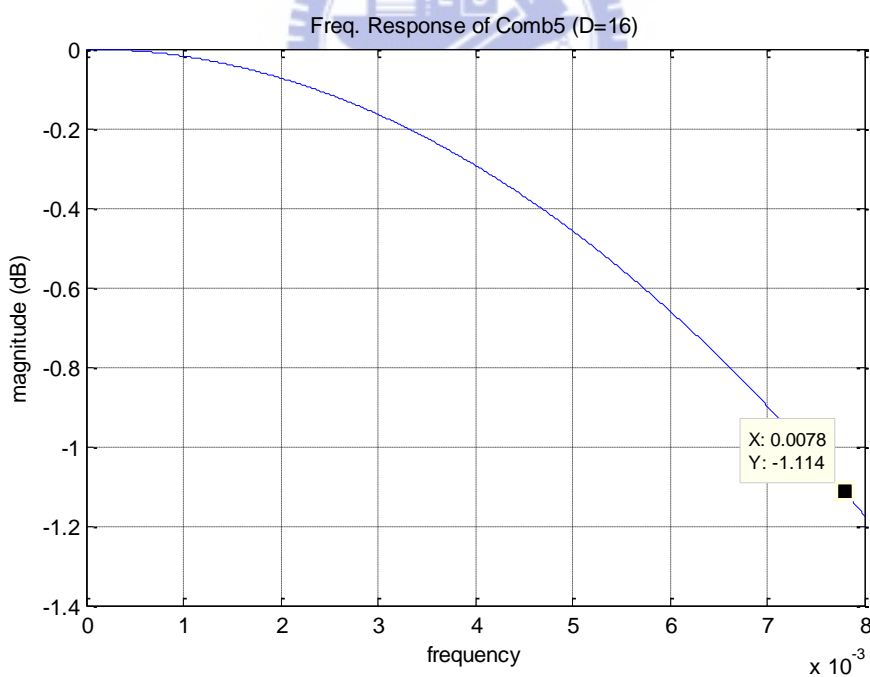


Figure 2.27 Pass-band drop of comb filter with D=16 for OSR=64  
 (fb=0.5/OSR=7.8125e-3)

### 2.5.1 Stage4 Design

According to the pass-band drop of comb filter, the compensation filter is designed using Matlab function (fir2) and the compensated results are shown in Table 2.9 and Table 2.10 for decimation ratio 128 and 64, respectively.

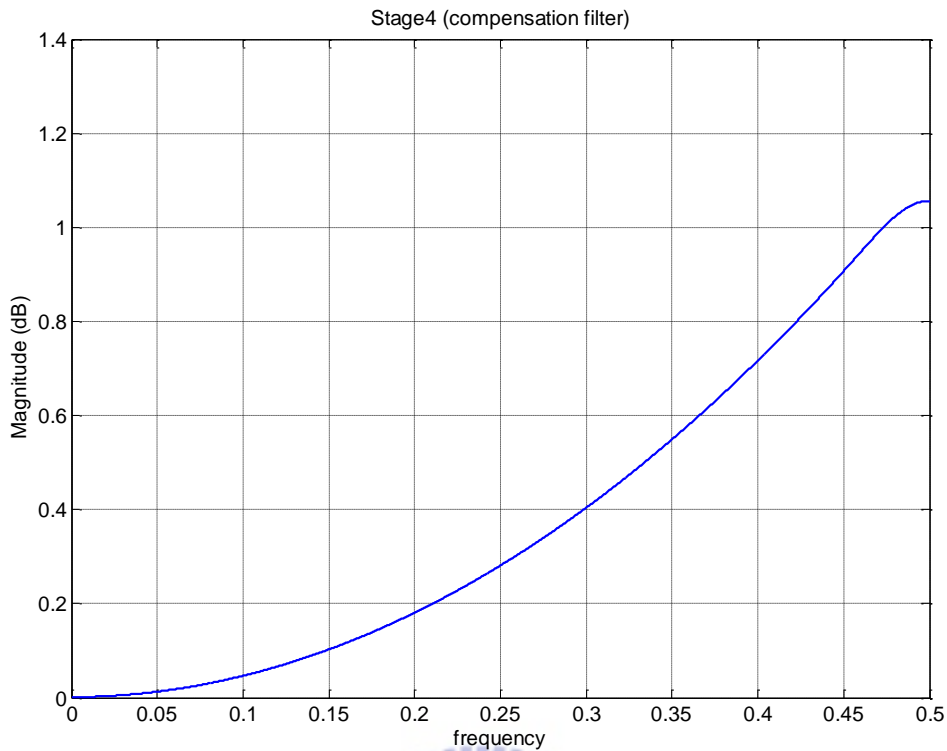
Table 2.9 Pass-band ripple with different order of compensation filter for decimation ratio=128

order of compensation FIR filter	pass-band ripple of 90% in-band	pass-band ripple of 95% in-band
NN4=20	0.0065 dB	0.0886 dB
NN4=40	0.0025 dB	0.0546 dB
NN4=60	0.0020 dB	0.0520 dB

Table 2.10 Pass-band ripple with different order of compensation filter for decimation ratio=64

order of compensation FIR filter	pass-band ripple of 90% in-band	pass-band ripple of 95% in-band
NN4=20	0.0074 dB	0.0867 dB
NN4=40	0.0031 dB	0.0527 dB
NN4=60	0.0023 dB	0.0495 dB

Order of 40 is chosen to compensate the 90% in-band ripple to 0.0025 dB and 0.0031 dB for decimation ratio 128 and 64, respectively. And the magnitude response of compensation filter in the 4<sup>th</sup> stage is shown in Figure 2.28.

Figure 2.28 Magnitude response of compensation filter in the 4<sup>th</sup> stage

## 2.6 Specification Summary

In this section, the information about filters used in the decimator are summarized as below tables and figures, such as filters specification listed in Table 2.11, magnitude frequency response shown in Figure 2.29 (the coefficients of each FIR filter are shown in Appendix A), aliasing-powers at each stages for decimation ratio 128 and 64 listed in Table 2.12 and Table 2.13, respectively.

Table 2.11 Summary of decimation filter specifications

Stage	Filter Type	Decimation Ratio	Transition-band (frequency)	Pass-band ripple Rp	Stop-band attenuation	tap
1	5 <sup>th</sup> -order Comb	32 and 16	-	drop 1.12 dB	60-150 dB	0
2	FIRpm	2	0.125~0.365	0.0001 dB	70 dB	17
3	FIRpm	2	0.2339~0.2661	0.0006 dB	60 dB	128
4	FIR2	0	Compensation pass-band drop to 0.0025 and 0.0031 dB			41



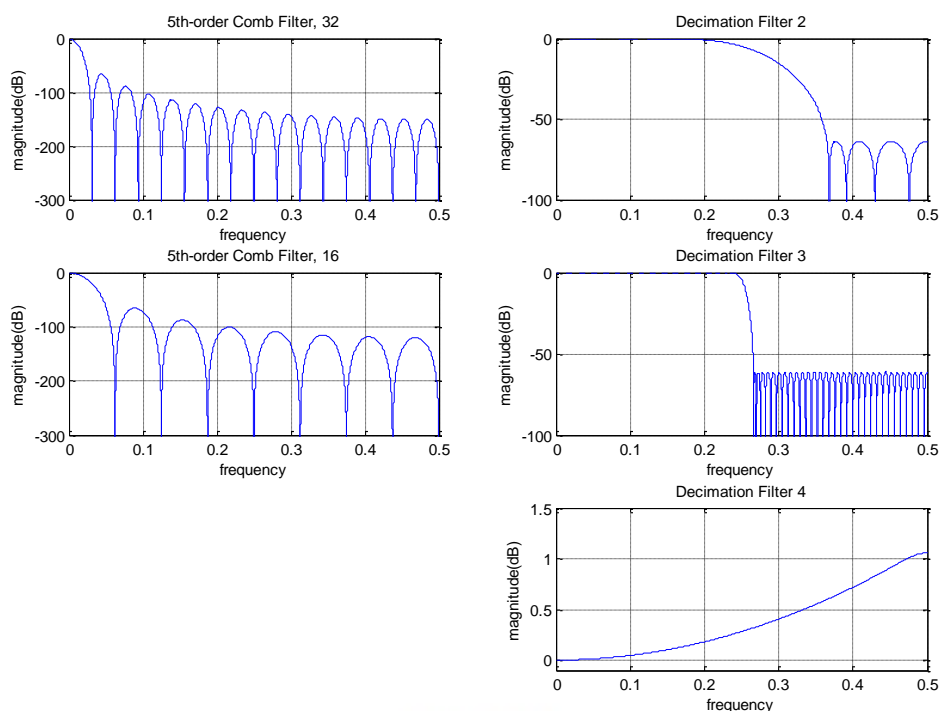


Figure 2.29 Magnitude response of each stage

Table 2.12 Aliasing power, remaining quantization noise power and obtained SNR at each stage for OSR=128

Stage	Aliasing power	Total aliasing power	Remaining QN	Obtained SNR
1	3.3085e-019	3.3085e-019	1.6579e-013	124.79 dB
2	2.6691e-020	3.5754e-019	2.4506e-015	143.09 dB
3	3.9747e-019	7.5501e-019	7.7448e-018	168.09 dB
4	-	-	9.5168e-018	167.20 dB

Table 2.13 Aliasing power, remaining quantization noise power and obtained SNR at each stage for OSR=64

Stage	Aliasing power	Total aliasing power	Remaining QN	Obtained SNR
1	1.6880e-016	1.6880e-016	8.4886e-011	97.70 dB
2	1.3665e-017	1.8247e-016	1.2547e-012	116.00 dB
3	2.0266e-016	3.8513e-016	3.9653e-015	141.01 dB
4	-	-	4.8726e-015	140.11 dB

Finally, the magnitude response of the equivalent single stage low-pass decimation filters for decimation ratio 128 and 64 are shown in Figure 2.30 and Figure 2.31, respectively. The equivalent single stage decimation filters are composed of four decimation filters in efficient way to preserve the SNR of the SDM output. The quantization noise power spectral density of 1-bit 4<sup>th</sup>-order SDM is shown in Figure 2.29, where noise is increasing with the increase of frequency. Thus, the stop-band attenuation of the equivalent single stage decimation filters also increases more at high frequency. The expected SNR can be preserved by my decimator if the harmonic-tones near in-band edge (not in in-band) of SDM output spectrum do not exceed 60 dB to noise-floor due to only 60 dB stop-band attenuation for equivalent decimation filter near in-band edge.

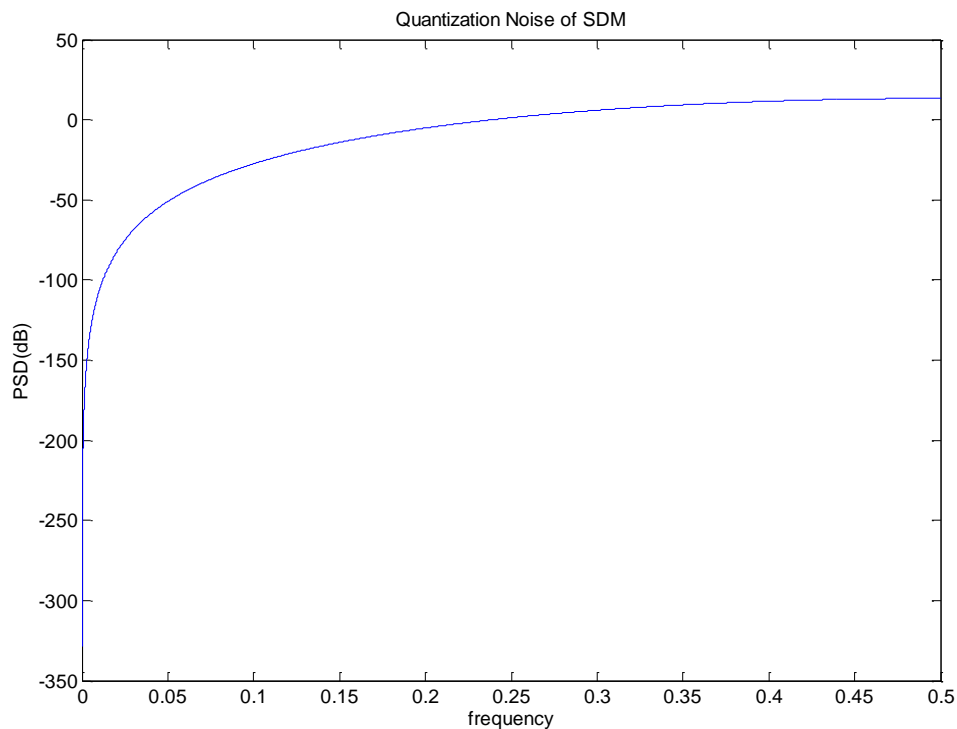


Figure 2.30 Quantization noise power spectral density (dB) of 1-bit 4<sup>th</sup>-order SDM

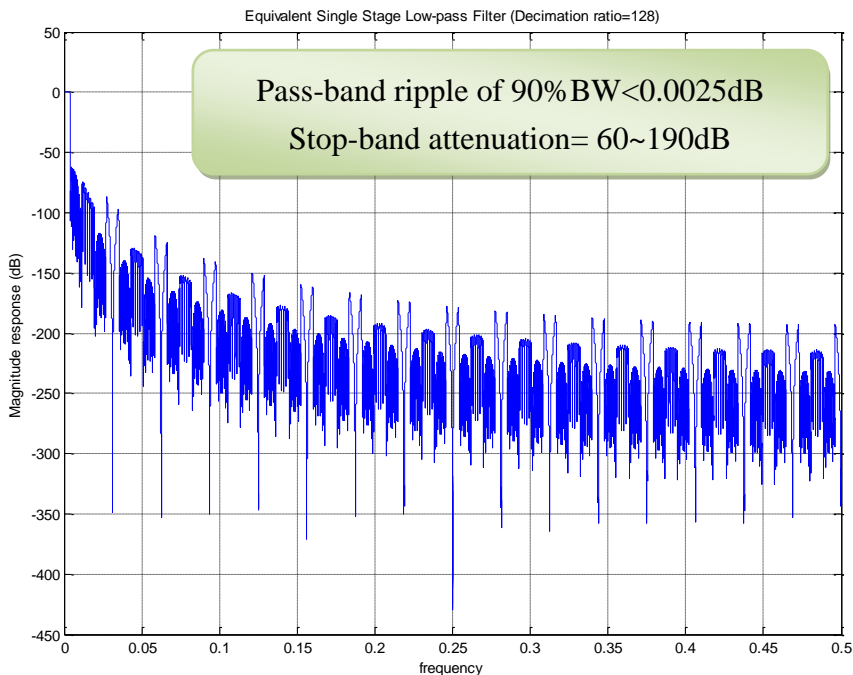


Figure 2.31 Magnitude response of equivalent single stage low-pass filter for decimation ratio 128

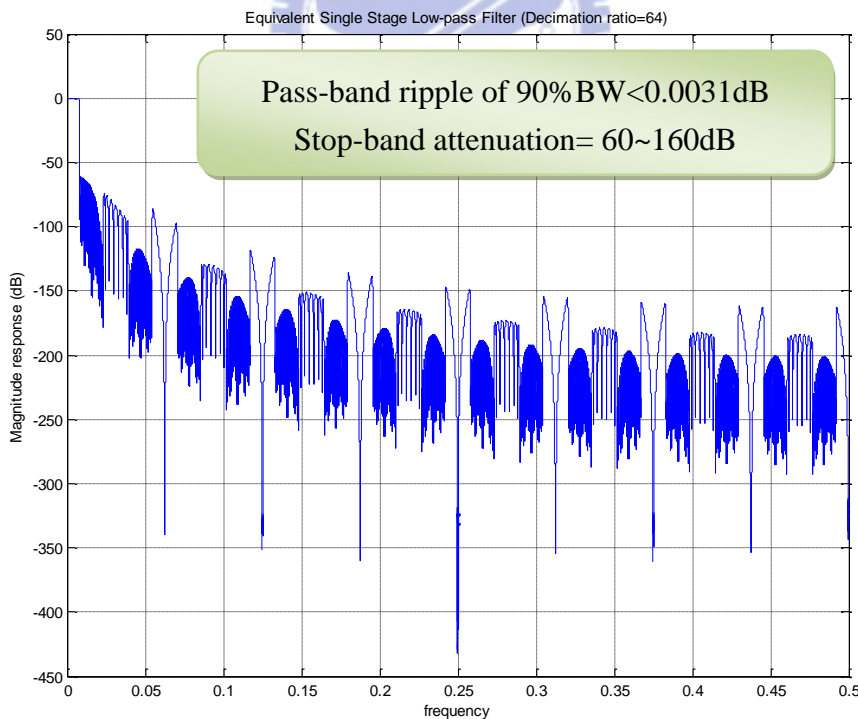


Figure 2.32 Magnitude response of equivalent single stage low-pass filter for decimation ratio 64

# CHAPTER

# 3

---

## Decimator Implementation

---

The hardware implementations of each filter are discussed and decided in this Chapter in terms of area, power and speed (throughput). Also, the circuit of overall decimator with programmable decimation ratio would be described. However, the most important part is the high order FIR filter which dominates the silicon area, consumes the highest power and limits the operating frequency due to requiring many multiplication operations per second. So, the implementation comparisons focus on the high order FIR filter, which is the main part this thesis wants to improve. Afterward the decimator is implemented step by step to become integrated circuits fabricated in TSMC 0.18um CMOS mixed signal RF general purpose MiM Al 1P6M process.

### 3.1 Implementation and Verification Flow

The implementation steps of digital IC cell-based design flow is shown in Figure 3.1. The system level of Figure 3.1 is the behavior of decimator characterized by Matlab, see Chapter 2. And then the circuit of the decimator is addressed by verilog (a hardware description language) in RTL level, which could be further synthesized to gate level by a logic synthesis tool. After logic synthesis, the logic cell could be placed and routed (physical synthesis) by an APR tool. In addition, the function and behavior of the circuit is verified by simulation using SDM bit-streams at each implementation step.

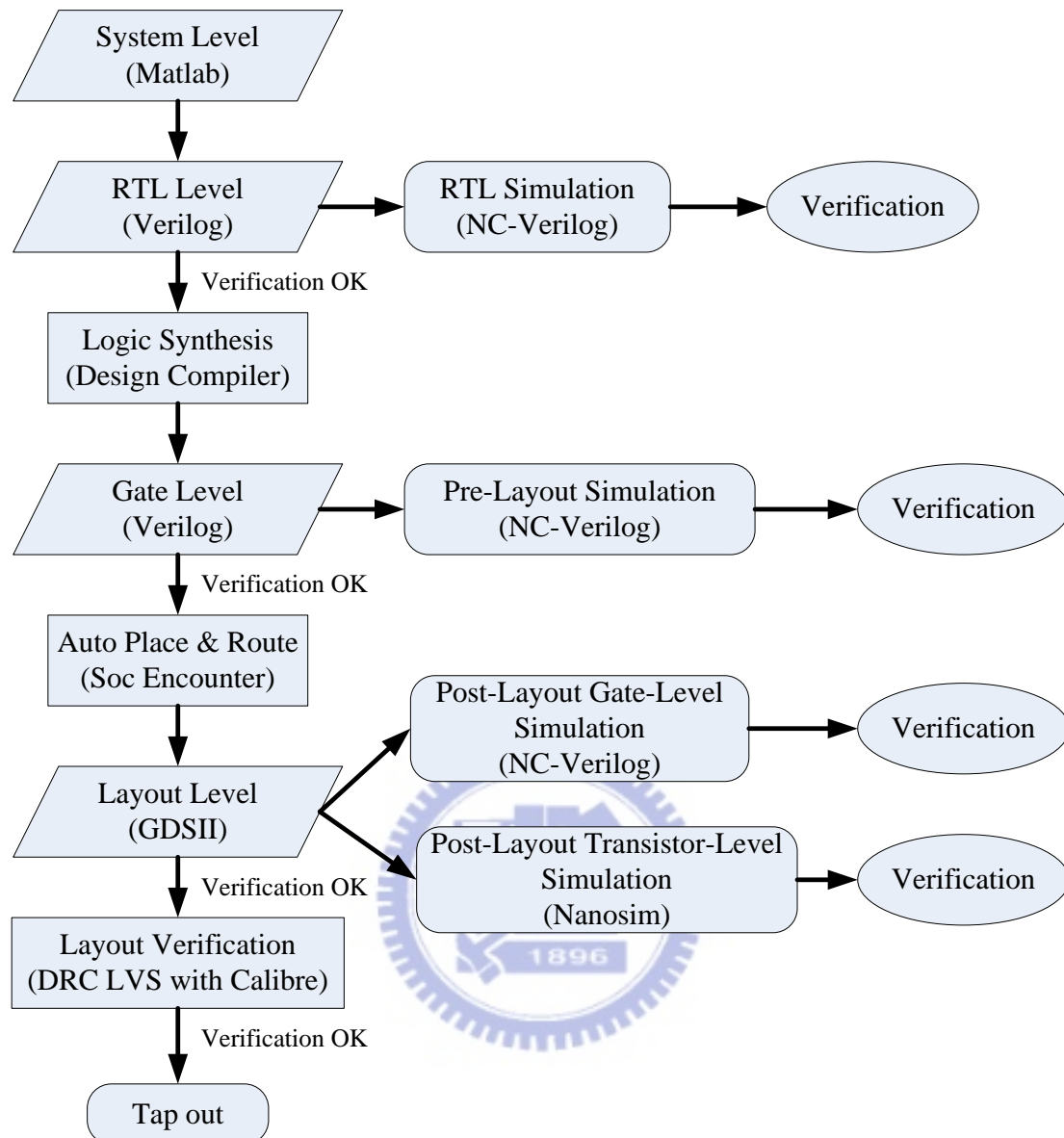


Figure 3.1 The cell-based implementation flow of digital IC

In my implementation flow, the logic synthesis and APR (auto place & route) tools used in the flow are Design Compiler of Synopsys and SOC Encounter of Cadence, respectively. Furthermore, the simulators used at each step are NC-Verilog of Cadence for circuit described by verilog (required SDF) and Nanosim of Synopsys for post-layout transistor-level simulation. After verifying the functions and behaviors of layout (decimator) are correct, further layout verifications, DRC (design rule check) and LVS (layout versus schematic), are required to confirm whether the chip layout satisfies a series of recommended design rules required by semiconductor manufacturers and whether the integrated circuit layout corresponds to the circuit simulated at post-layout simulation by using Calibre of Mentor.

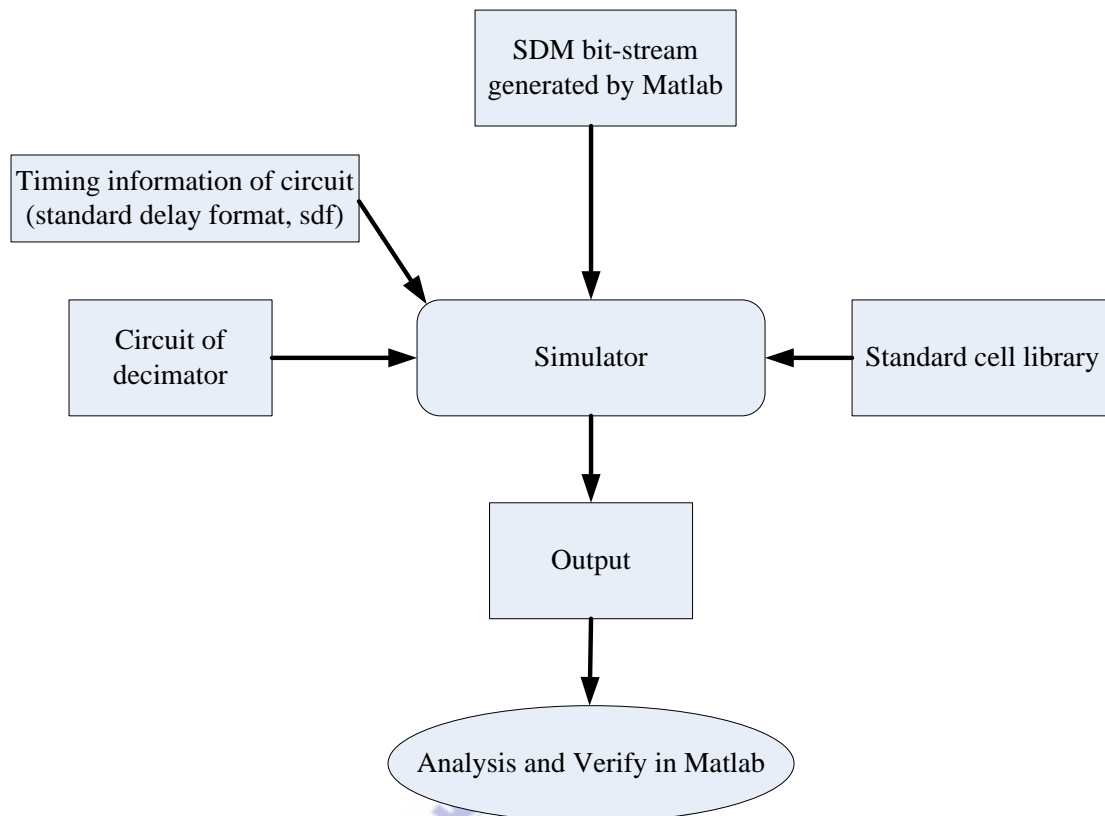


Figure 3.2 Simulation and verification procedures of decimator

The simulation and verification procedures are shown in Figure 3.2. Both time-domain and frequency-domain of decimator's output would be verified, which would compare with results filtered and down-sample by designed decimator in Chapter 2 using Matlab (floating point calculation) to confirm that the circuit (decimator) behaviors are what we expected. Furthermore, the magnitude frequency response of decimator's output (circuit) could directly compare with in-band magnitude frequency response of SDM bit-streams to determine whether the function of circuit (decimator) is correct or not, because the function of decimator is to preserve the in-band signal of SDM output with Nyquist-rate, which imply that no large aliasing and no big pass-band ripple distort in-band signal except signal at in-band edge.

No matter where the implementation step is, the logic values are the same for digital circuits with correct behaviors. So, only the post-layout simulation results would be shown and verified in this thesis to prevent repetition.

### 3.2 Previous Work Comparison

In this section, the previous works on implementations of comb filter and FIR filters are discussed in terms of silicon area, power consumption, and highest operating frequency. An appropriate implementation of comb filter is chosen to implement the first decimation stage. And the features and behaviors of previous work in FIR filters are described in order to compare with my proposed implementation for FIR filters with polyphase decomposition.

#### 3.2.1 Comb Filter

From the transfer function of comb filter mentioned in Section 2.3 and shown again in Equation 3.1-3.3, there are three structures which could be chosen to implement the comb filter.

$$H_{\text{comb}}(z) = \frac{1}{D^k} \left( \frac{1-z^{-D}}{1-z^{-1}} \right)^k \quad (3.1)$$

$$= \frac{1}{D^k} \prod_{i=0}^{(\log_2 D)-1} (1+z^{-2^i})^k \quad (3.2)$$

$$= \frac{1}{D^k} \left( \sum_{i=0}^{D-1} z^{-i} \right)^k \quad (3.3)$$

Furthermore, the first decimation stage is the comb filter followed by down-sample  $D$ . Applying the commutative rule [3] shown in Figure 3.3, the required delay elements (registers) and the number of addition operations per second could be reduced much.

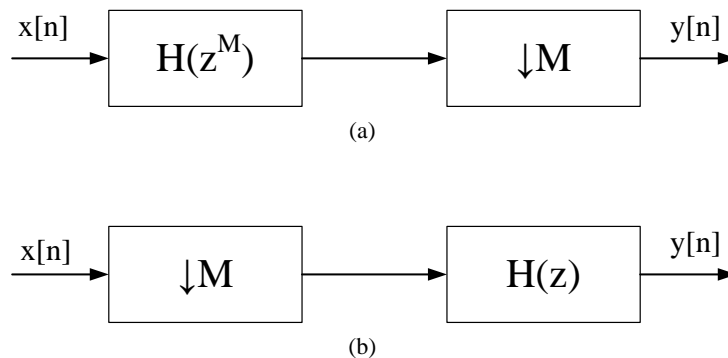


Figure 3.3 Commutative rule: the system in (a) is equivalent to the system in (b).

Equation 3.1 and Equation 3.2 combining down-sampling  $D$  could be derived to Equation 3.4 and Equation 3.5, respectively. A comparison of comb decimation filter based on the recursive algorithm (Equation 3.4) and the non-recursive algorithm

(Equation 3.5) are presented in [11]. The structure of Equation 3.3 is not an option to implement comb filter due to its many adders and delay elements (registers) requirements, which could not exploit commutative rule shown in Figure 3.3 to improve.

$$\begin{aligned}
 H_{\text{comb}}(z) \downarrow D &= \frac{1}{D^k} \left( \frac{1-z^{-D}}{1-z^{-1}} \right)^k \downarrow D \\
 &= \frac{1}{D^k} \left( \frac{1}{1-z^{-1}} \right)^k (1-z^{-D})^k \downarrow D \\
 &= \frac{1}{D^k} \left( \frac{1}{1-z^{-1}} \right)^k \downarrow D (1-z^{-1})^k \tag{3.4}
 \end{aligned}$$

$$\begin{aligned}
 H_{\text{comb}}(z) \downarrow D &= \frac{1}{D^k} \prod_{i=0}^{(\log_2 D)-1} (1+z^{-2^i})^k \downarrow D \\
 &= \frac{1}{D^k} (1+z^{-1})^k (1+z^{-2})^k (1+z^{-4})^k \dots (1+z^{-2^{(\log_2 D)-1}})^k \downarrow D \\
 &= \frac{1}{D^k} (1+z^{-1})^k \downarrow 2 (1+z^{-1})^k \downarrow 2 (1+z^{-1})^k \downarrow 2 \dots (1+z^{-1})^k \downarrow 2 \\
 &\quad \text{Stage1} \quad \text{Stage2} \quad \text{Stage3} \quad \dots \quad \text{Stage } (\log_2 D) \\
 \text{There are } \log_2 D \text{ decimation stages of } &(1+z^{-1})^k \downarrow 2 \tag{3.5}
 \end{aligned}$$

According to [6] [11], the word-length required for these two structures by Equation 3.4 and Equation 3.5 are shown in Table 3.1

Table 3.1 required word-length for these two structures

Word-length	
Recursive algorithm of comb filter in both parts (integrator and comb) (Equation 3.4)	Non-recursive algorithm of comb filter in the i-th Stage (Equation 3.5)
<b>m+klog<sub>2</sub>D</b>	<b>m+k*i</b>

m: number of input bit

k: order of comb filter

D: decimation ratio for comb filter

i: denotes the i-th stage in comb filter for non-recursive algorithm, see Equation 3.5.



In my decimator design,  $m=1$ ,  $k=5$ ,  $D=32$  and  $16$ . Although  $D=32$  and  $D=16$  are required in my programmable decimation ratio decimator, only the implementations of comb filter with  $D=32$  need to be discussed and compared, which is crucial than comb filter with  $D=16$  and determine the hardware complexity, critical path and power consumption in first decimation stage.

So, the comparison of recursive and non-recursive structure for comb filter with  $D=32$ ,  $k=5$ ,  $m=1$  are discussed in terms of power, area and speed using Table 3.1, which are shown in Figure 3.4.

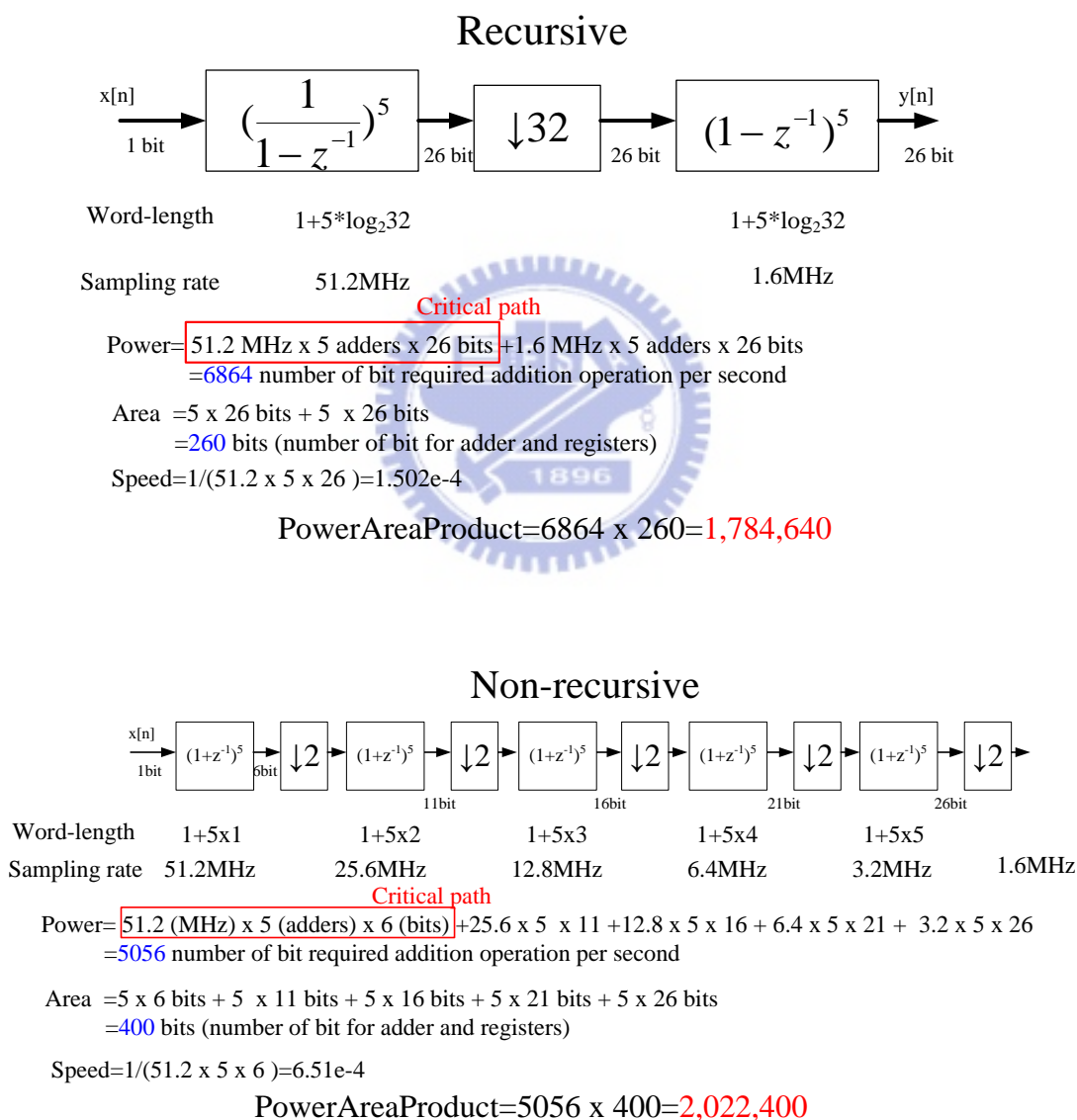


Figure 3.4 Comparison of recursive and non-recursive algorithm of comb filter in terms power, area, speed, power speed product

As claimed by [11], the advantage of non-recursive structure for comb filter is smaller power consumption and higher operating frequency especially when decimation ratio and order of comb filter are high. However, a factor, power-area-product is introduced in this thesis to judge which structure is better and a small power-area-product structure is a better implementation choice. So, the recursive structure of comb filter is adopted to implement the first decimation stage of decimator due to its smaller power-area-product.

As regards the highest operating frequency, the longer critical path of recursive structure would not limit the highest operating frequency of decimator because the critical path of decimator is in the 3<sup>rd</sup> decimation stage (high order FIR filter). Moreover, the critical path of recursive structure of comb filter can be improved by pipelining and retiming without any hardware overhead (see Section 3.53), which make the speed of recursive structure (speed=1.502e-4 x5=7.51e-4) faster than non-recursive (speed=6.51e-4). The overhead of pipelined CIC (cascaded-integrator-comb; i.e. recursive structure) comb filter is only four-clock-cycle latencies, which could be ignored in decimator because it is far smaller than the group delay (126/2\*128=8064) of high order FIR filter.

### 3.2.2 FIR Filter

The transfer-function of N<sup>th</sup>-order FIR filter is:

$$H_{FIR}(z) = \sum_{i=0}^N b_i z^{-i} \tag{3.6}$$

$$= b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + \dots + b_{N-2} z^{-(N-2)} + b_{N-1} z^{-(N-1)} + b_N z^{-N} \tag{3.7}$$

$$= b_0 + z^{-1}(b_1 + z^{-1}(b_2 + z^{-1}(b_3 + z^{-1}(\dots + z^{-1}(b_{N-2} + z^{-1}(b_{N-1} + z^{-1}(b_N))\dots))) \tag{3.8}$$

N= order of FIR filter;

So, there are (N+1) coefficients (also called N+1 taps).

The transfer function of FIR filter can be represented by direct-form structure (as Equation 3.7) or transposed-form structure (as Equation 3.8), which affect the implementation of FIR filter. The relationship between output and input of the system (FIR filter) represented in z-domain is  $Y(z)=X(z)H_{FIR}(z)$ . The corresponding difference equation is  $y[n]=x[n]*h[n]$  (\* denotes linear-convolution). So,

$$y[n]=b_0x[n]+b_1x[n-1]+b_2x[n-2] + \dots + b_{N-1}x[n-(N-1)]+b_Nx[n-N] \quad (3.7a)$$

The time-shifting property [3] is

$$x[n - k] \stackrel{Z}{\Leftrightarrow} X(z) z^{-k} \quad \text{if } x[n] \stackrel{Z}{\Leftrightarrow} X(z)$$

Considering the decimation FIR filter in the 2<sup>nd</sup> and 3<sup>rd</sup> stage, the polyphase decomposition could be explored to eliminate the redundant algorithmic operations for FIR filter followed by down-sampling.

**Polyphase Decomposition**

Let h[n] is the impulse response of the system and the corresponding transfer function in z-domain is H(z). Considering a causal system (h[n]=0 for n<0) [3],

$$\begin{aligned} H(z) &= \sum_{n=0}^{\infty} h[n]z^{-n} \\ &= \begin{matrix} h[0] & +h[M]z^{-M} & +h[2M]z^{-2M} & +\dots \\ + h[1]z^{-1} & +h[M+1]z^{-(M+1)} & +h[2M+1]z^{-(2M+1)} & +\dots \\ + h[2]z^{-2} & +h[M+2]z^{-(M+2)} & +h[2M+2]z^{-(2M+2)} & +\dots \\ +\dots\dots\dots & & & \\ + h[M-1]z^{-(M-1)} & +h[2M-1]z^{-(2M-1)} & +h[3M-1]z^{-(3M-1)} & +\dots \end{matrix} \\ &= \begin{matrix} \sum_{k=0}^{\infty} h[kM] & (z^M)^{-k} \\ +z^{-1} \sum_{k=0}^{\infty} h[kM + 1] & (z^M)^{-k} \\ +z^{-2} \sum_{k=0}^{\infty} h[kM + 2] & (z^M)^{-k} \\ +\dots\dots\dots \\ +z^{-(M-1)} \sum_{k=0}^{\infty} h[kM + M - 1] & (z^M)^{-k} \end{matrix} \\ &= \sum_{i=0}^{M-1} z^{-i} E_i(z^M) \end{aligned} \quad (3.9)$$

Where  $E_i(z)= \sum_{k=0}^{\infty} h[kM + i]z^{-k}$  is called the i-th polyphase component (3.10)

Applying the commutative rule [3] shown in Figure 3.3, the computational cost of the system followed by downsampler is reduced.

As a result, the system B shown in Figure 3.5 is equivalent to system A shown in Figure 3.5; however, the system B (polyphase decomposition) is an efficient way to implement the system (FIR filter followed by down-sampling) in terms of power consumption and required operating speed of circuit.

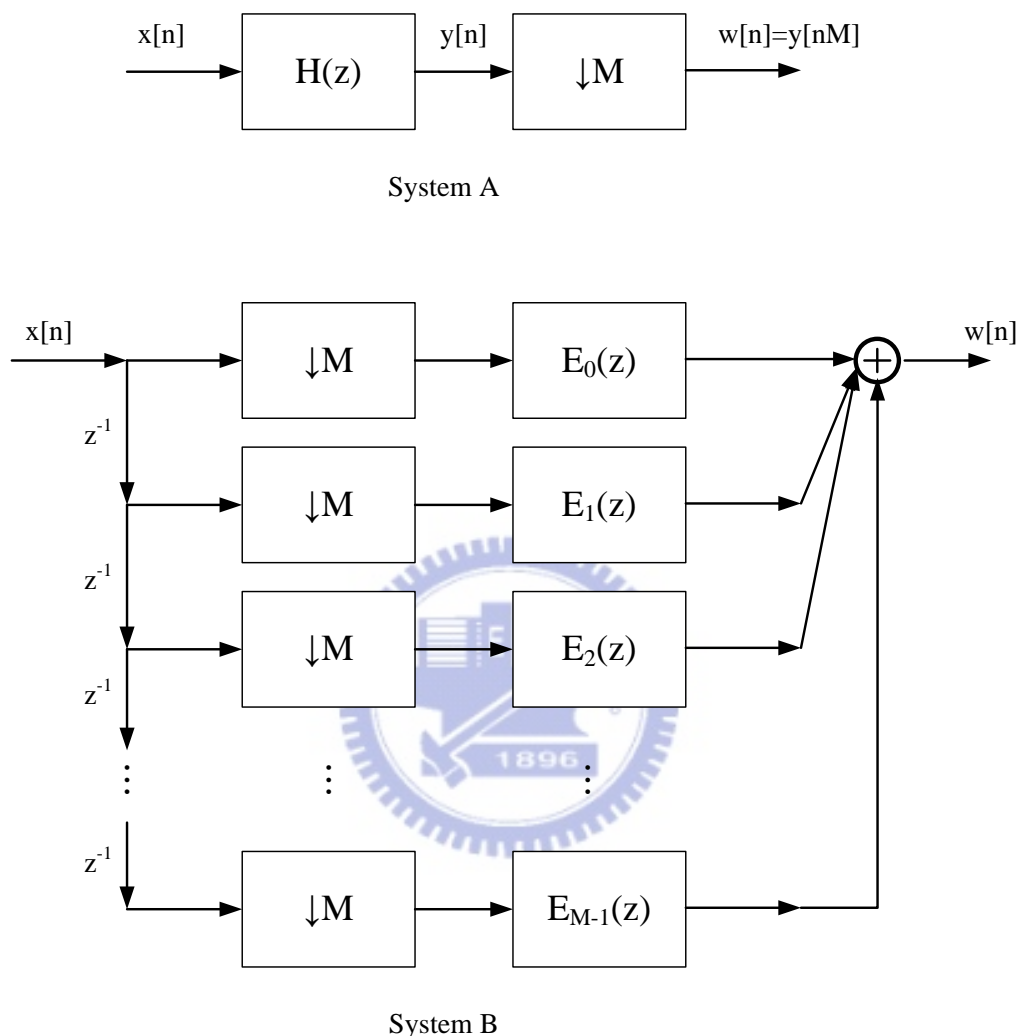


Figure 3.5 System A is equivalent to system B (polyphase decomposition; efficient implementation for FIR filter followed by down-sampling).

In the 2<sup>nd</sup> and 3<sup>rd</sup> decimation stages, the decimation ratios are both two ( $M=2$ ). So, the implementation of decimation FIR filters with polyphase decomposition with  $M=2$  is shown in Figure 3.6.

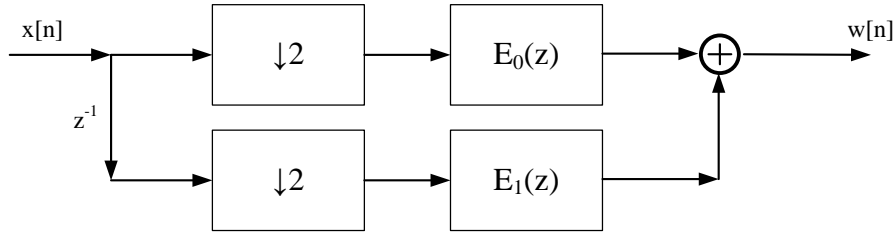


Figure 3.6 Efficient implementation of decimation FIR filter with M=2

, where  $E_0(z) = b_0 + b_2z^{-1} + b_4z^{-2} + \dots + b_{N-2}z^{-(N-2)/2} + b_Nz^{-N/2}$  (3.11)

$$E_1(z) = b_1 + b_3z^{-1} + b_5z^{-2} + \dots + b_{N-3}z^{-(N-4)/2} + b_{N-1}z^{-(N-2)/2}$$
 (3.12)

, for N is even number

, where  $E_0(z) = b_0 + b_2z^{-1} + b_4z^{-2} + \dots + b_{N-3}z^{-(N-3)/2} + b_{N-1}z^{-(N-1)/2}$  (3.13)

$$E_1(z) = b_1 + b_3z^{-1} + b_5z^{-2} + \dots + b_{N-2}z^{-(N-3)/2} + b_Nz^{-(N-1)/2}$$
 (3.14)

, for N is odd number

In my decimator design, N is assumed as a even number to make the coefficients of  $E_0(z)$  and  $E_1(z)$  are still symmetric, which implies that  $b_k = b_{N-k}$  for Equation 3.11 and Equation 3.12, because the coefficients are symmetric  $b_k = b_{N-k}$  for FIR filters with linear-phase regardless of even or odd order (N).

Generally speaking, the polyphase decomposition technique would definitely be exploited to reduce power consumption and relieve circuit speed for decimation FIR filter due to no any hardware overhead. So, the further implementation comparison for FIR filters will all base on polyphase decomposition except the 4<sup>th</sup> stage (the compensation filter; no down-sampling).

The high order FIR filter requires many multiplication operations per sample, which limits the sample throughput and consumes large power. Besides, it consumes large silicon area due to requiring many multipliers, adders and registers for a straightforward implementation. Thus, the main object in this thesis is to reduce the silicon area of FIR filter. Although there are many techniques that could be used to reduce silicon area, the folding technique could reduce the silicon area of high order FIR filter more than others. So, previous works on folded FIR filter will be introduced and briefly compared (detail compared with my proposed folded design in latter section). Both direct-form and transposed-form structures will be considered.

### Direct-Form Structure

In my decimator design, the highest order of FIR linear-phase filter is 126, which imply that there are 127 coefficients with the symmetric feature, i.e.  $h_k=h_{126-k}$ . So, there are only 64 distinct coefficients for the 126<sup>th</sup>-order FIR linear-phase filter. Usually  $h[n]$  denotes the impulse response of system (FIR filter), i.e.  $h[n]=b_n$  in Equation 3.6. The decimation FIR filter with polyphase decomposition in direct-form is shown in Figure 3.7. And all the following implementations would be illustrated using 126<sup>th</sup>-order FIR filter with polyphase decomposition.

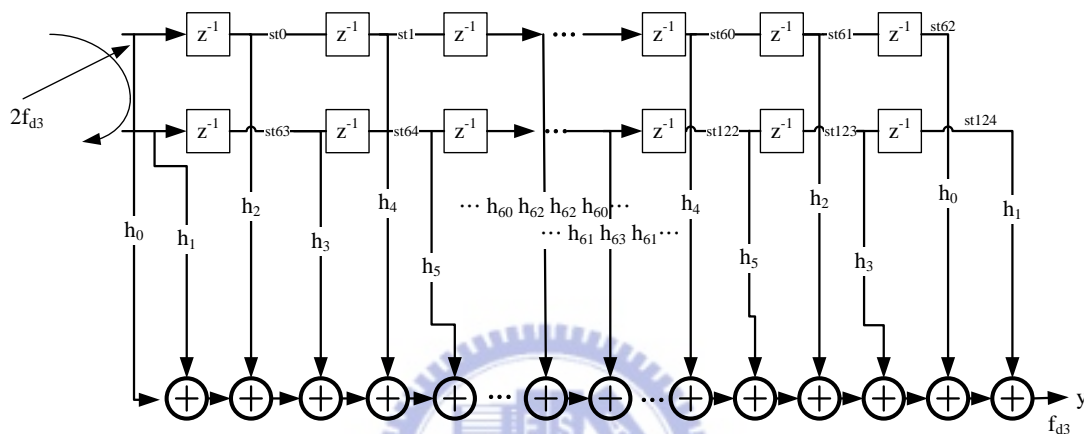


Figure 3.7 Decimation FIR filter (126<sup>th</sup>-order) with polyphase decomposition in direct-form

The coefficients  $h_k$  with  $k>63$  will be replaced by  $h_{126-k}$  with  $k>63$  due to  $h_k=h_{126-k}$ , which implies that  $h_{125}$  will be shown with  $h_1$ . The Figure 3.7 exhibits a direct implementation of Equation 3.7 modified as Figure 3.6 (combining down-sampling 2 and using polyphase decomposition) and the Figure 3.8 shows the meaning of the switched arrow at input of Figure 3.7. From these figures, it is obvious that the straightforward implementation for 126<sup>th</sup>-order FIR filter requires 127 multipliers, 126 adders and (125+1) storage elements (126 x 27-bits registers if word-length of each sample is 27-bits), which consumes large silicon area.

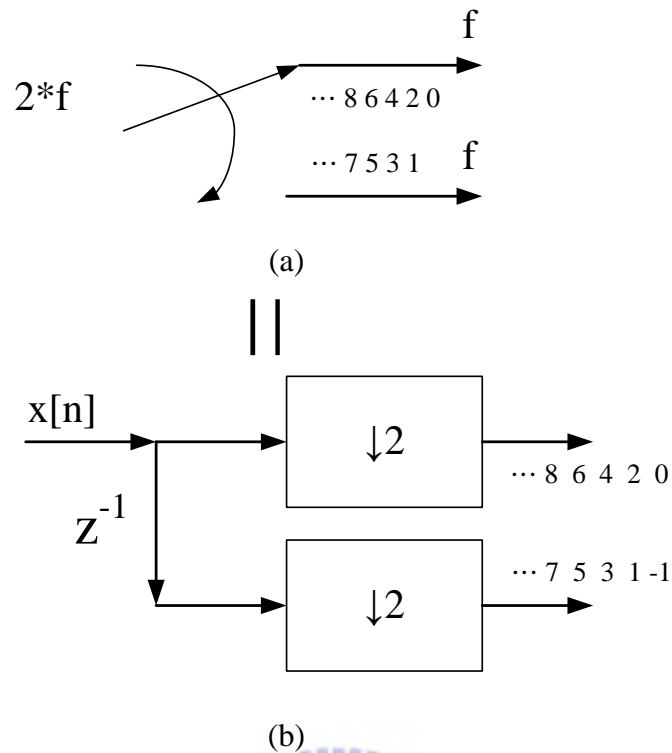


Figure 3.8 The meaning of switched arrow where  $f$  denotes the sampling rate

### Direct-Form Folding

Therefore, a folded architecture of FIR filter with polyphase decomposition in direct-form is shown in Figure 3.9. Circuits based on the direct-form folding architecture for FIR filters could be seen in [12] [13] [14] [15]. The folded architecture is an architecture where the algorithmic operations are performed by time-multiplexing so as to reduce the functional units. In Figure 3.9, one multiplication and two addition operations are performed at each clock cycle using one multiplier and two adders. Utilizing the symmetry of coefficients (127 coefficients), only 64 multiplication operations needed to calculate per sample, which implies that an output sample is calculated using 64 clock cycles for the 126<sup>th</sup>-order FIR filter. Although the number of adder could be reduced further (i.e., using only one adder) for the direct-form folding FIR filter architecture, it is not an implementation option due to the twice power (requiring 127 clock cycles per sample because of not utilizing coefficients symmetry) and the negligible area reduction (one adder).

It is easy to understand the behavior of the direct-form folded FIR architecture. The even samples of  $x$  (i.e.  $x[\text{even}]$ ) are stored in the shift registers D-R0 (data registers) in descending order ( $x[2(n-1)]$   $x[2(n-2)]$ .....  $x[2(n-62)]$   $x[2(n-63)]$ ); the oldest sample at the right end of shift registers due to right-shifting and input from left) and the odd samples of  $x$  are stored in D-R1 as well ( $x[2(n-1)-1]$   $x[2(n-2)-1]$ ...). The D-R0 are composed of  $x_0$ ,  $st_0$ ,  $st_1$ , ..... , and  $st_{62}$  as well as the D-R1 are composed of  $x_1$ ,  $st_{63}$ ,  $st_{64}$ , ..... , and  $st_{124}$ . As seen in Figure 3.9, the value of  $x[2(n-1)]$  is stored in the registers (storage-element)  $st_0$ ,  $x[2(n-1)-1]$  is stored in  $st_{63}$ , etc.

A new sample of  $x$  is coming every 32 clk cycles and it must be synchronized by extra data registers (implicitly in Figure 3.9). The shift registers (D-R0 and D-R1) are clocked by  $clk_{d3}$ , so they are shifted one-time every 64 clk cycles. The multiplication operations with  $h_{\text{odd}}$  coefficients of FIR filter are computed at the first 32 clk cycles of one output-sample-cycle (period of 64 clk cycles) and those with  $h_{\text{even}}$  coefficients are computed at the last 32 clk cycles. One output sample is produced every 64 clk cycle (=1 output-sample-cycle=2 input-sample-cycle). The term  $(x[2n-1]+x[2(n-62)-1])h_1$  is computed at the first clk cycle,  $(x[2(n-1)-1]+x[2(n-61)-1])h_3$  at the 2<sup>nd</sup> clk cycle, ..... ,  $x[2(n-31)-1]h_{63}$  at the 32<sup>th</sup> clk cycle,  $(x[2n]+x[2(n-63)])h_0$  at the 33<sup>th</sup> clk cycle, ..... , as well as the term  $(x[2(n-31)]+x[2(n-32)])h_{62}$  is computed at the 64<sup>th</sup> clk cycle. The result (term described above) of each clk cycle is accumulated and then the output sample is produced at the end of 64<sup>th</sup> clk cycle.

The advantages of direct-form folding are fewer functional units (one multiplier and two adders in Figure 3.9) compared with direct-form structure (non-folding; 127 multipliers and 126 adders in Figure 3.7).



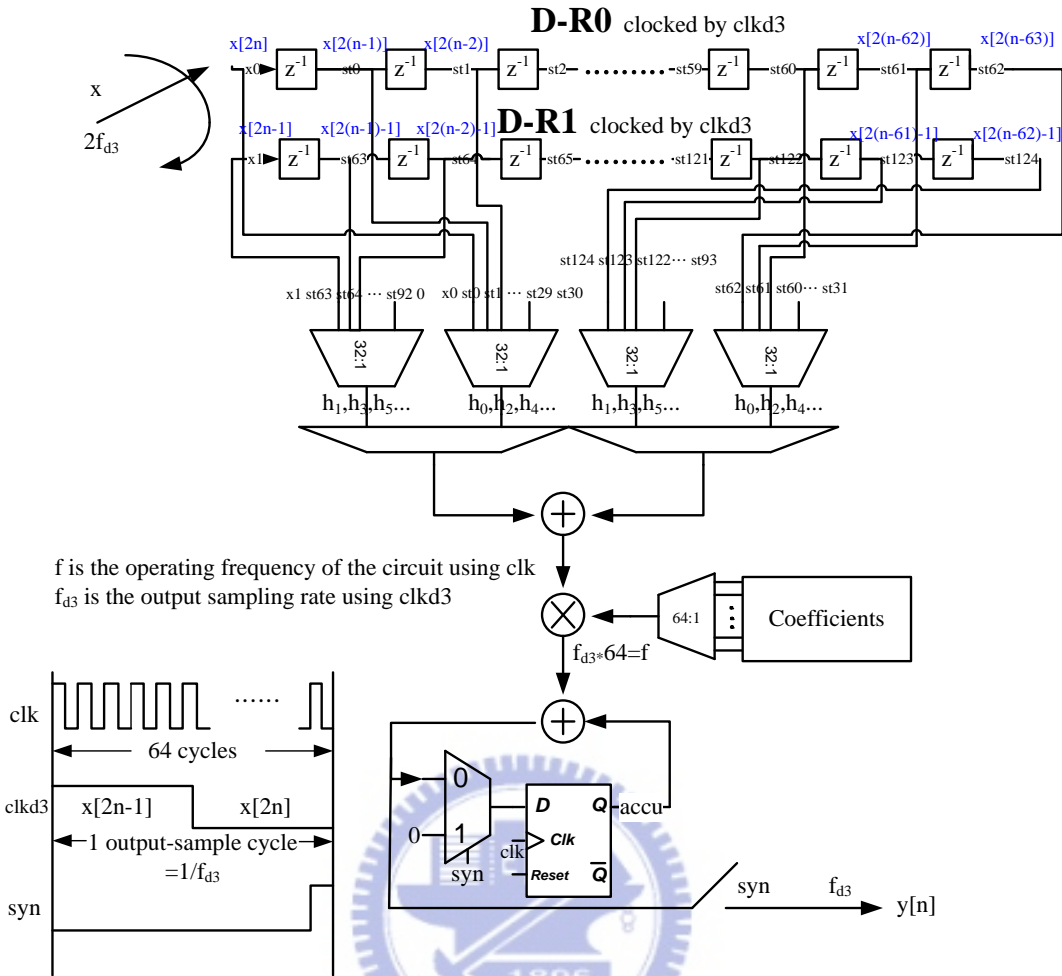


Figure 3.9 The folded architecture of FIR filter with polyphase decomposition in direct-form

### Transposed-Form Structure

The decimation FIR filter (126<sup>th</sup>-order) with polyphase decomposition in transposed-form (Equation 3.8 modified further using polyphase-decomposition as Figure 3.6) is shown in Figure 3.10.

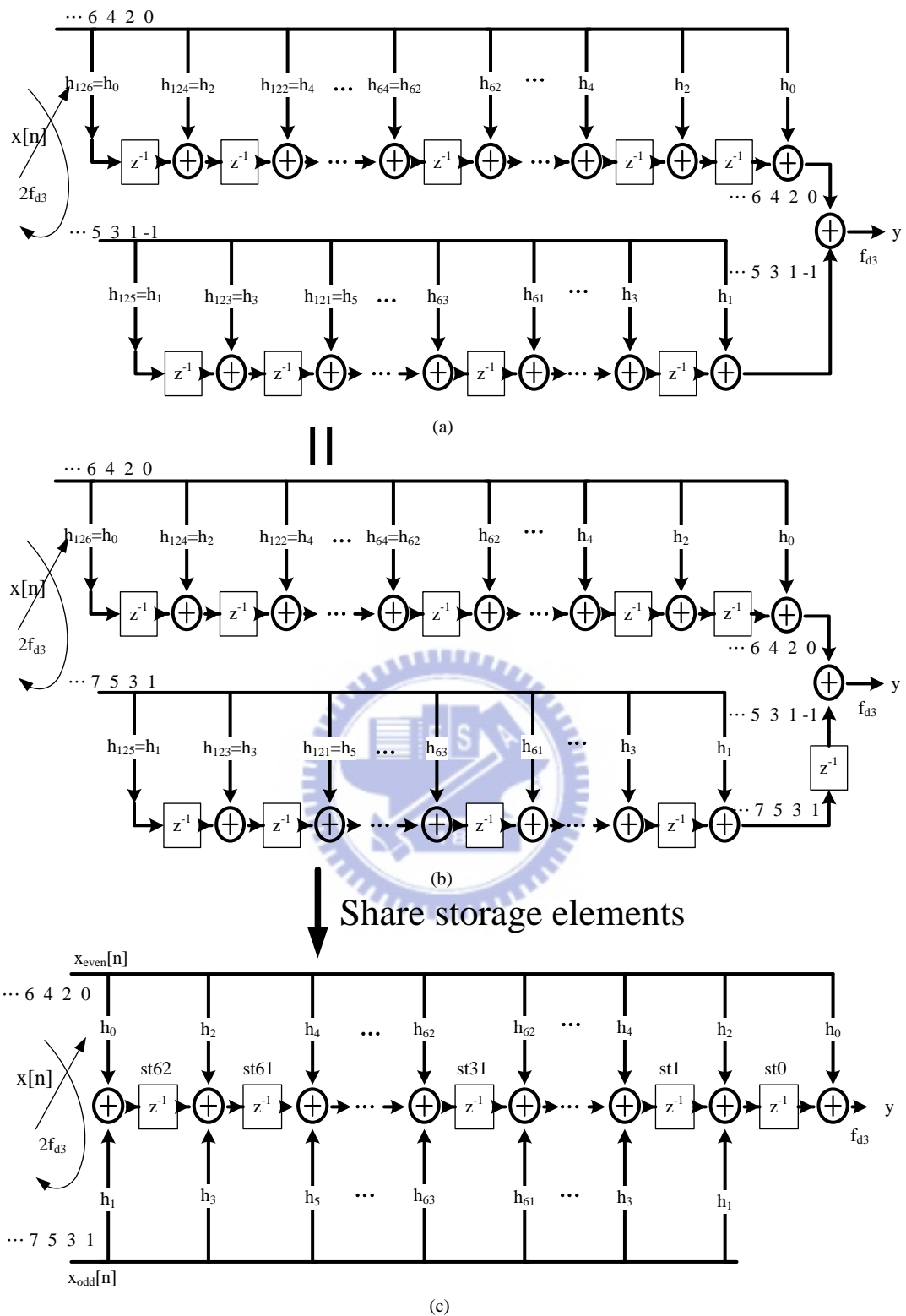


Figure 3.10 The decimation FIR filter (126<sup>th</sup>-order) with polyphase decomposition in transposed-form

The structure (a) in Figure 3.10 is equivalent to structure (b) in Figure 3.10. Note that the input-sequences allocated at the phase0-filter and phase1-filter (i.e., filter  $E_0$  and filter  $E_1$  in Figure 3.6) for structure (a) and for structure (b) are different in the beginning; however, the sequences before the final adder are the same due to the delay element used in structure (b). The storage elements of phase0-filter and phase1-filter could be merged together to halve the usage of registers [16] [17]. For convenience, the value stored in storage elements would only be represented by  $st_0$ ,  $st_1$ , and so forth because the values stored in storage elements include many delay versions of input sample multiplying with FIR filter coefficients shown in Figure 3.11. The stored values in transposed-form are similar to the values in parentheses of Equation 3.8 needed to modify further for polyphase decomposition. Consequently, the output value  $y[n]$  in transposed-form is equivalent to output value in direct-form.

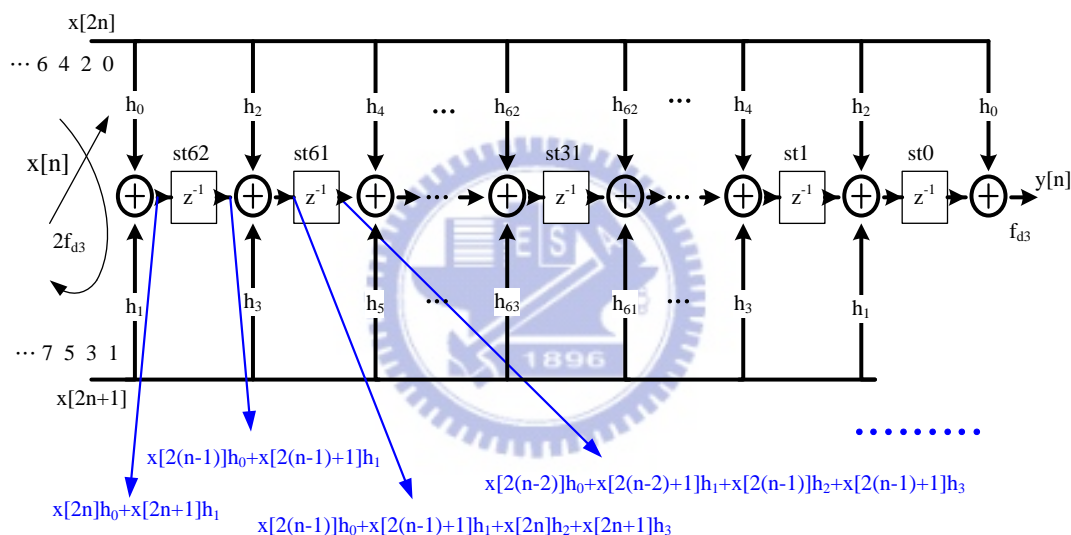


Figure 3.11 The values stored in storage elements

The advantages of the decimation FIR filter with polyphase decomposition in transposed-form (structure (c) in Figure 3.10) compared with direct-form are

1. Short critical path
2. Half storage elements (registers)

However, the amount of the function units (multipliers and adders) required by  $126^{\text{th}}$ -order FIR filter in transposed-form (Figure 3.10) are still too large.

## Transposed-Form Folding

Therefore, a folded architecture of FIR filter with polyphase decomposition in transposed-form is needed to reduce the functional units of the high order FIR filter. The structure (c) in Figure 3.10 is the folding object. However, the folded FIR filter architecture for transposed-form is quite different from the folded architecture for direct-form.

### *Encounter problem*

Note that the multipliers' output are stored to the different storage elements (st62, st61, ..., etc.) after extra addition operations for transposed-form FIR filter seen in Figure 3.10, which is different from direct-form where the multipliers' output are merely stored to the accumulator-registers (accu in Figure 3.9). As a result, the folded architecture using one multiplier for transposed-form must encounter the problem that the result of multiplier's output after addition operation should be stored to diverse storage elements and only some (two) of them (st62, st61, ..., etc.) fetch the calculated result (multiplication and addition operation) each cycle, which implies that the storage elements (registers) are hard to control using single clock due to the above described behavior of the registers even if the calculated result each cycle might use de-multiplexer to choose the destination (target storage element) (in addition, for using de-multiplexer, the non-destination storage elements still fetch zeros and then the stored value would be cleared if the registers are trigger by the same clock).

The 2<sup>nd</sup> encounter problem for the folded FIR filter architecture in transposed-form (using one multiplier, i.e. one multiplication operation each cycle) is that the calculated result (multiplication and addition operations) could not store the result to the target storage element because the original value of target storage element must be read at latter clock cycle for another addition with the result of different coefficient multiplication. For example, the calculated result related to multiplication operation with coefficient  $h_2$  and addition with st62 and st1 are calculated first and then stored to the target storage elements, st61 and st0 respectively. In another clock cycle, however, the calculated result related to coefficients  $h_4$  or  $h_0$  are wrong because they required reading the original value stored in st61 and st0 respectively. The situation could not be solved even if the operation sequences related to coefficients each cycle are changed. It may be solved using extra registers to store the calculated result or the original value stored in storage element, but the advantage of half registers for FIR filter in transposed-form will disappear.

These two problems, which both resulted from storing the calculated result to storage element, could be solved by shifting the value stored in storage elements every cycle and not utilizing the coefficient symmetry to calculate the multiplication results from the right to left end (in Figure 3.11) as the folded architecture of FIR filter in transposed-form mentioned in [18], seen in Figure 3.12.

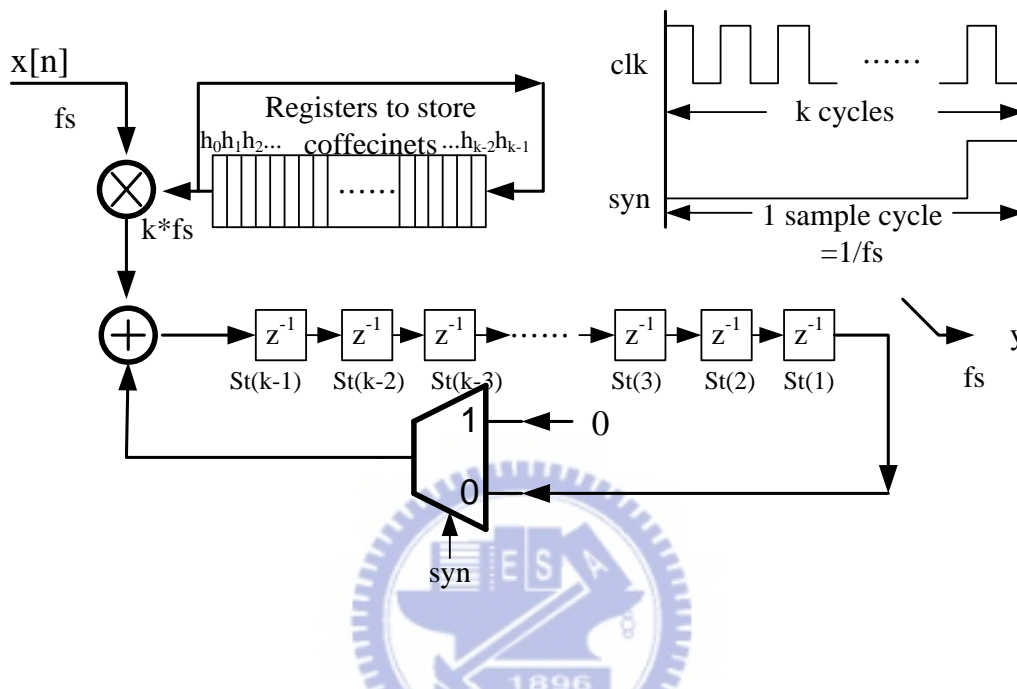


Figure 3.12 Folded architecture of k-tap FIR filter in transposed-form without polyphase decomposition (i.e. no down-sampling) (for linear-phase, the feature of FIR filter coefficients:  $h_n=h_{k-1-n}$ )

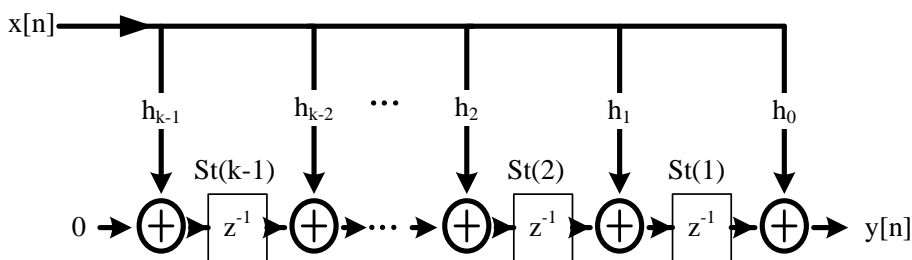


Figure 3.13 Corresponding unfolded FIR filter in transposed form

The unfolded FIR filter in transposed-form is also shown in Figure 3.13 to illustrate the behavior of folded architecture in transposed-form shown as Figure 3.12. The input sample  $x[n]$  enter the circuit at the first clock cycle (clk) and remain constant for the k clock (clk) cycles. The coefficients stored in registers in ascending order ( $h_0$  at the left end of the coefficients registers and  $h_{k-1}$  at the right end). So, the

result of  $x[n]h_0+St(1)$  is computed first and stored in the  $St(k-1)$  at the end of the first clock cycle. The original value of  $St(k-1)$  is moved forward to  $St(k-2)$  as well as the other original value contained in  $St(n)$  are moved forward to  $St(n-1)$ , i.e. the behavior of shift registers, which preserve the uncalculated (unused) value stored in  $St$  not to be modified. Because of the data shifting, the value stored in  $St(1)$  is the original value of  $St(2)$  (i.e., the value in  $St(2)$  seen in Figure 3.13) in 2<sup>nd</sup> clock cycle. In 2<sup>nd</sup> clock cycle, the result  $x[n]h_1+St(2)$  seen in Figure 3.13 is calculated and stored to  $St(k-1)$  seen in Figure 3.12 at the end of 2<sup>nd</sup> clock cycle. The data are still right shifting so the result calculated in last clock cycle is shifting to  $St(k-2)$  and not modified. One tap's computation (from right to left end in Figure 3.13, i.e.  $x[n]h_0+St(1)$ ,  $x[n]h_1+St(2)$ , ...,  $x[n]h_{k-1}+0$ ) is calculated each clock cycle. After  $k-1$  clock cycles, the result  $x[n]h_0+St(1)$  (the value seen in Figure 3.13) is shifting to  $St(1)$  in Figure 3.12 and could be output (at the same time, the result  $x[n]h_1+St(2)$  in Figure 3.13 is shifting to  $St(2)$  in Figure 3.12 and so on). After  $k$  clock cycles, the result  $x[n]h_0+St(1)$  is shifting out, and the result  $x[n]h_1+St(2)$  is shift to the  $St(1)$  as well as the other results are shifting to the individual destination storage elements like  $x[n]h_2+St(3)$  is shift to the  $St(2)$ . Thus, the computations required by a sample are finished and stored to the right destination storage element after  $k$  clock cycles.

The behavior of folded architecture of FIR filter is described as above. Now, for comparison, the folded architecture of FIR filter must be modified using polyphase decomposition (i.e. a folded architecture based on Figure 3.11 not Figure 3.13) and the FIR coefficients would not use the shift registers to store and output, which is designed for programmable FIR filter coefficients and cause large area overhead.

The behavior of the folded architecture for FIR filter using polyphase decomposition is almost the same as the behavior in Figure 3.12. The result of  $x[2n]h_0+st_0$  seen in Figure 3.11 is calculated in the 1<sup>st</sup> clock cycle and stored to  $st_62$ . The result of  $x[2n]h_2+st_1$ ,  $x[2n]h_4+st_2$  and so forth are calculated in the 2<sup>nd</sup> cycle, the 3<sup>rd</sup> cycle, etc, respectively. After 63 cycles, the result of  $x[2n]h_0+st_0$  is shifting to  $st_0$  and could be output in the 64<sup>th</sup> cycle. In the 64<sup>th</sup> cycle, the result  $x[2n]h_0+0$  (the left end computation; no storage element result needed to add) is calculated by switching the input of multiplexer to zero for addition operation (controlled by  $syn$ ). After 64 clock cycles, the result of  $x[2n]h_2+st_1$  seen in Figure 3.11 is shift to  $st_0$  seen in Figure 3.14. So, in the 65<sup>th</sup> cycle, the result of  $x[2n]h_2+st_1+x[2n+1]h_1$  seen in Figure 3.11 is calculated and stored to  $st_62$  seen in Figure 3.14 at the end of 65<sup>th</sup> cycle, where the

$x[2n]h_2+st1$  is read from  $st0$  of Figure 3.14. Also the result with other odd coefficient is calculated one by one each cycle. At the end of the 127<sup>th</sup> cycle, the result  $x[2n]h_2+st1+x[2n+1]h_1$  is stored to  $st0$  and the computations required by a output-sample are all finished and stored the data to the right storage elements.

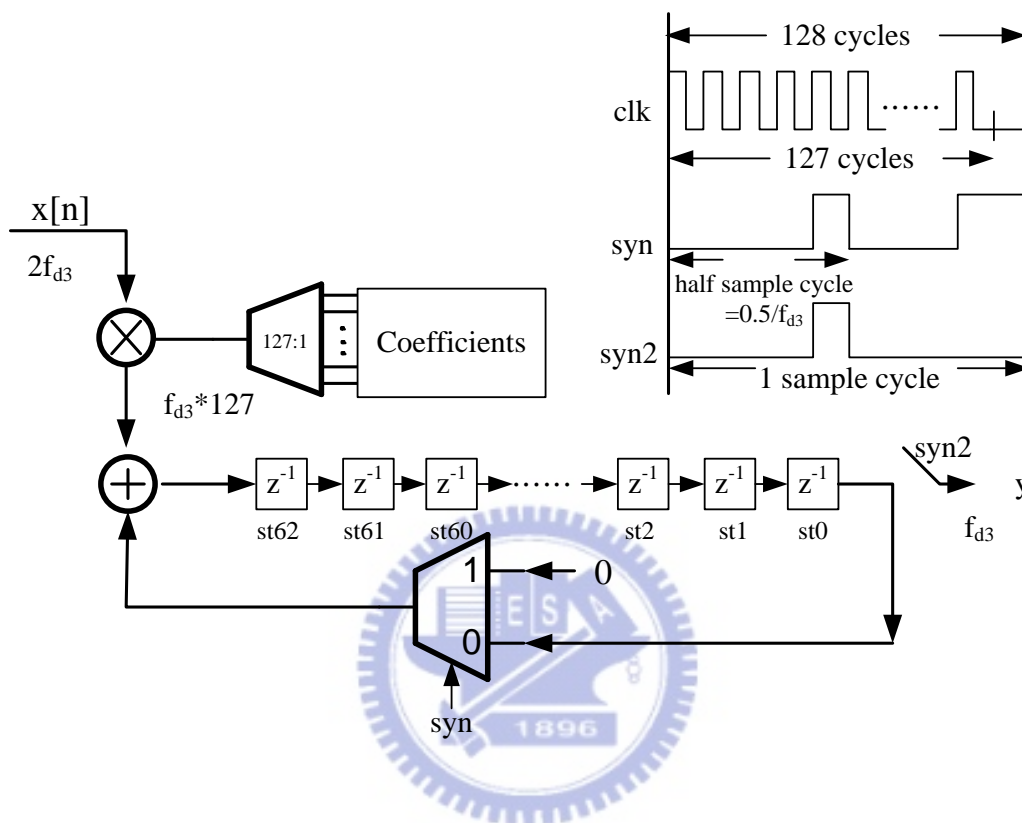


Figure 3.14 Folded architecture of decimation FIR filter using polyphase decomposition

The drawbacks of the folded architecture [18] of decimation FIR filter with polyphase decomposition shown as above are shifting data every operating cycle ( $clk$ ) and not utilizing the coefficients symmetry which implies that it requires double clock cycle compared with direct-form folding mentioned above to accomplish the computation required by a output-sample. Those will result in large power consumption (at least twice).

The advantages of folded architecture in transposed-form shown in Figure 3.14 [18] are one adder and half register reduction compared with the folded architecture in direct-form shown in Figure 3.9 [12] [13] [14] [15]. However, the silicon area required by transposed-form folding [18] is not definitely smaller than silicon area required by direct-form folding because of the twice operating frequency requirement

for transposed-form folding to maintain the throughput (output sampling-rate) (∴ trading silicon area for speed).

The implementations of decimation FIR filter using polyphase decomposition have been described above; the folded architecture could obtain the smallest silicon area due to few functional units especially for high order FIR filters. Also the advantages and disadvantages of folded architecture based on direct-form shown in Figure 3.9 and transposed-form shown in Figure 3.14 are addressed which would be compared with my proposed folded architecture based on transposed-form in latter section.

### 3.3 Overall Decimator System

The block diagram of overall decimator system and related clocks is shown in Figure 3.15. The 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> stages are FIR filters and the 2<sup>nd</sup> and 3<sup>rd</sup> stages are followed by down-sampling 2, which could be polyphase decomposed to halve the power consumption as mentioned in Section 3.2.2.

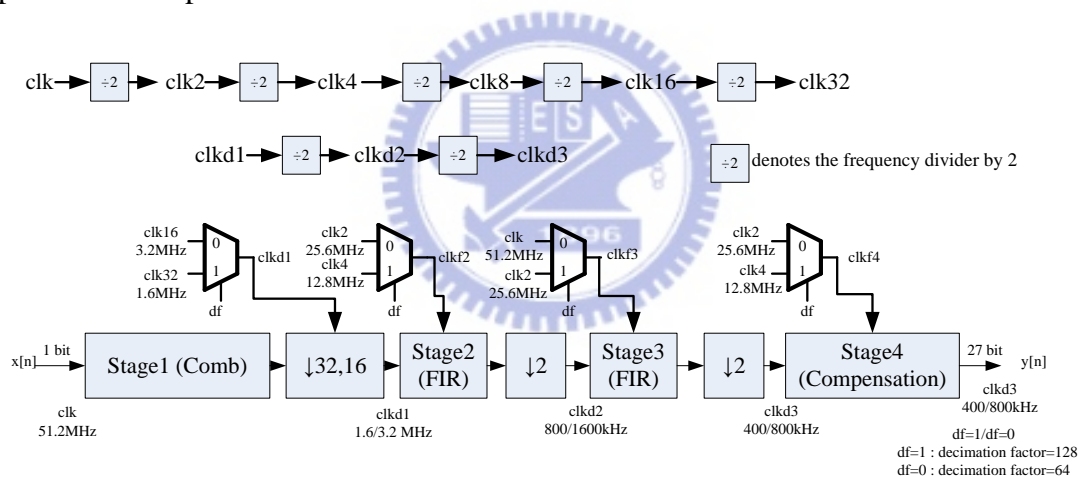


Figure 3.15 Block diagram of overall decimator system

In Figure 3.15, the sampling rate of input and output at each stage are shown for decimation ratio 128 and 64 as well as the clocks required by each stage. In order to maintain that the number of clock cycles required by a output-sample to finish the computations are equal at decimation ratio 128 and 64 for circuits using folded architecture (stage 2, 3 and 4), the operating clock for the 2<sup>nd</sup>, 3<sup>rd</sup>, and 4<sup>th</sup> stages must be chosen according to the decimation ratio. For example, the folded architecture for the 3<sup>rd</sup> stage, 64 clock cycles (64 multiplication operations; 127 taps using coefficients symmetry) are needed for an output-sample's computations, which implies that the operating frequency of the circuit to output sampling-rate must fix to 64 no matter the decimation ratio of decimator are 128 or 64. That makes the circuit of folded



architecture operate correctly.

### 3.4 Clock Divider Circuit

The circuit of clock divider by 2 is shown in Figure 3.16 as well as the timing diagram is shown in Figure 3.17.

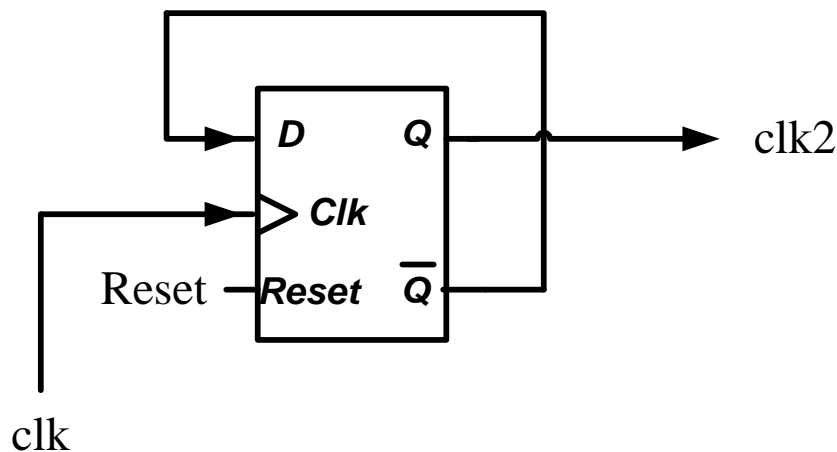


Figure 3.16 Circuit of clock divider by 2

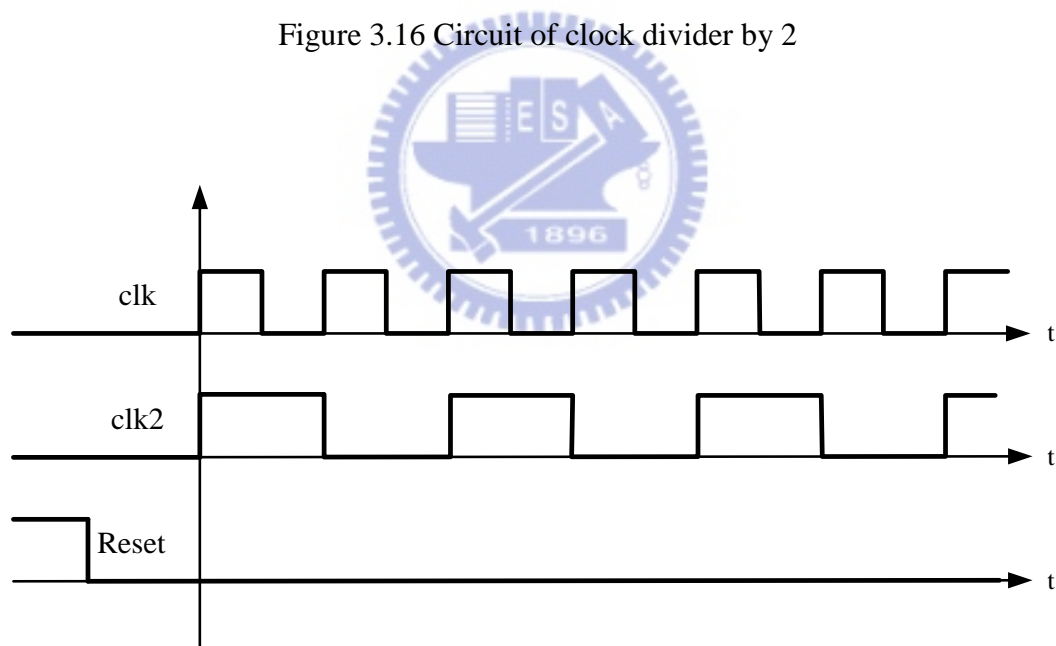


Figure 3.17 Timing diagram for the circuit of clock divider by 2

In Figure 3.16, the data (logic value: 0 or 1 or unknown) in pin D (is equal to pin  $\bar{Q}$  which is the inverse of Q) is fetched to pin Q (i.e. port clk2) at positive edge of clock (clk) so the port clk2 invert when encountering the positive edge of clock (clk). As a result of that, the frequency divider by 2 is obtained.

### 3.5 The First Decimation Stage: Comb Filter

As mentioned in Section 3.2.1, the recursive (IIR-FIR) structure of comb filter is adopted to implement the first decimation stage due to its smaller power-area product compared with non-recursive structure.

The transfer functions of comb filter followed by down-sampling with decimation ratio 32 and 16 determined in Chapter 2 are

$$H_{\text{comb } 32}(z) = \frac{1}{32^5} \left(\frac{1}{1-z^{-1}}\right)^5 \downarrow 32 (1 - z^{-1})^5 \quad \text{for decimation ratio 32} \tag{3.15}$$

$$H_{\text{comb } 16}(z) = \frac{1}{16^5} \left(\frac{1}{1-z^{-1}}\right)^5 \downarrow 16 (1 - z^{-1})^5 \quad \text{for decimation ratio 16} \tag{3.16}$$

The first decimation stage is composed of gain control ( $\frac{1}{32^5}$  and  $\frac{1}{16^5}$ ), integrator ( $\frac{1}{1-z^{-1}}$ )<sup>5</sup>, downsampler ( $\downarrow 32$  and  $\downarrow 16$ ) and differentiator  $(1 - z^{-1})^5$ . The programmable decimation ratio (128 and 64) of decimator is mainly controlled by the downsampler of first decimation stage.

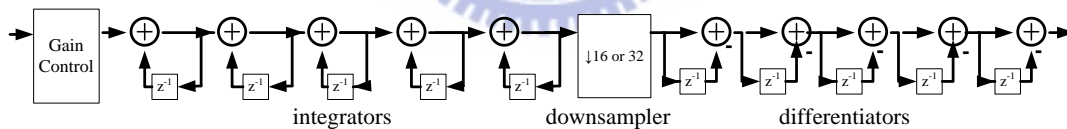


Figure 3.18 Components of first decimation stage

#### 3.5.1 Gain Control

In order to remove the DC gain (i.e. let DC gain=0 dB), the gain control is needed and then the data (value of sample) could be treated the same for later FIR filters no matter the decimation ratio are 128 or 64. Furthermore, to prevent the DC signal existing permanently, the data (value of input sample) would be represented by 2's complement (i.e. 1 of input sample is treated as +value and 0 is treated as -value). Moreover, the 'enable' signal is introduced to prevent the window effect seen [3] when enable=0 the data (value of input sample) is treated as zero which make the zero padding exist to prevent window effect.

In addition, the word-length of adder and registers is  $1+5*\log_2 32=26$  bits [6] for

unsigned representation in the first decimation stage. However, for using 2's complement representation, the word-length required in the first decimation stage is needed one more bit (27-bits).

The output of gain control circuit is list below:

- enable=0: output of gain control is 27-bits zeros .
- enable=1, in=1 and df=1: output of gain control is  $0.5*(1/32)^5$   
=27'b0000\_0000\_0000\_0000\_0000\_0000\_001.
- enable=1, in=1 and df=0: output of gain control is  $0.5*(1/16)^5$   
=27'b0000\_0000\_0000\_0000\_0000\_0100\_000.
- enable=1, in=0 and df=1: output of gain control is  $-0.5*(1/32)^5$   
=27'b1111\_1111\_1111\_1111\_1111\_1111\_111.
- enable=1, in=0 and df=0: output of gain control is  $-0.5*(1/16)^5$   
=27'b1111\_1111\_1111\_1111\_1111\_1100\_000.

### 3.5.2 Pipelined Comb Filter

As a result of the clock skew, the positive edges of clk32 and clk16 (seen in Figure 3.15; clk32 is produced by clk16 so the delay is inevitable) are not triggered at the same time, which make downsampler (registers clocked by clk16 composed of clk32 and clk16) easily fetch the unready signal (output of integrators) due to the long critical path of integrators which used most of the cycle time to calculate the result (i.e. the cycle time governed by the signal ready time is small).

In order to make the signal (output of integrators) ready earlier, the critical path of integrators must be shortened. So, comb filter is pipelined to meet the requirement seen in Figure 3.19. The cycle time of differentiators is 16 or 32 longer than the cycle time of integrators. Thus, the differentiator part is unnecessary to be pipelined.

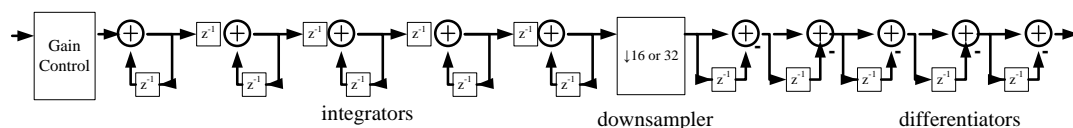


Figure 3.19 Pipelined comb filter (only integrators part needed to be pipelined due to its critical timing)

The pipelined integrators part could be retiming seen in Figure 3.20 to reduce the registers usage.

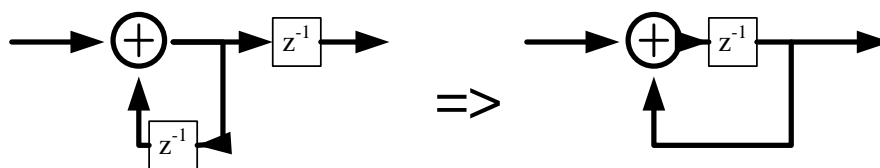


Figure 3.20 Retiming to reduce the registers usage

Finally, the first decimation stage is shown as Figure 3.21.

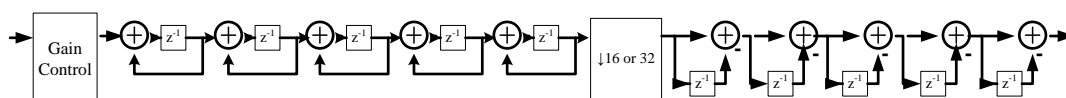


Figure 3.21 Implementation of first decimation stage

### 3.6 Proposed Circuit for FIR Filters

The basic ideas of my proposed architecture are to preserve the advantages of direct-form folding seen in Figure 3.9 [12] [13] [14] [15] and transposed-form folding seen in Figure 3.14 [18]. My proposed folded architecture is based on transposed-form FIR filter with polyphase decomposition to obtain half registers reduction compared with direct-form as well as my architecture utilizes the coefficients symmetry to halve the multiplication operation cycles (i.e. half power consumption) compared with transposed-form seen in Figure 3.14.

Furthermore, my proposed folded architecture based on transposed-form solves the encounter problems of folding on transposed-form FIR filter seen Section 3.2.2 by using extra control circuits and changing the computation procedures to reuse registers to store data instead of shifting all data each cycle seen in Figure 3.14. As a result of that, the power consumption of my proposed folded architecture based on transposed-form FIR filter is much less than half power of folded FIR architecture in transposed-form seen in Figure 3.14 [18].

My proposed folded architecture of decimation FIR filter based on transposed-form using polyphase decomposition shown in Figure 3.22(a). For convenience, the unfolded decimation FIR filter with polyphase decomposition is shown in Figure 3.22(b) so as to explain the behavior of my folded architecture. And

the timing diagram of my proposed architecture is shown in Figure 3.22(c).

As usual, the folded architecture is illustrated by the 126<sup>th</sup>-order FIR filter (the 3<sup>rd</sup> stage with highest order FIR filter). The 126<sup>th</sup>-order FIR filter's output sample requires 64 clock cycles to perform the 64 coefficients multiplications: the first 32 clock cycles are associated with  $x[2n]$  and  $h_{\text{even}}$ ; the last 32 clock cycles are associated with  $x[2n+1]$  and  $h_{\text{odd}}$ .

Note that my folded architecture requires an extra storage element st63 (registers) to store result  $(x[2n]h_0)$  at the end of first clock (clkf3) cycle compared with unfolded structure because the result  $(x[2n]h_0)$  cannot store to st62 which must be read in latter clock cycle for my folded architecture.

The result of  $x[2n]h_0+st0$  (the expected output result; right side) and  $x[2n]h_0$  (left side) seen in Figure 3.22(b) ( $x[2n]h_0$  seen in Figure 3.22(b) is computed by  $x[2n]h_0+st63$  where  $st63=0$  seen in Figure 3.22(a)) are computed in the first clock cycle (the clock is clkf3; first clock cycle implies that the value of counter is zero) and then stored to st0 and st63 respectively seen in Figure 3.22(a) at the end of first clock cycle. In the 2<sup>nd</sup> clock (clkf) cycle, the result of  $x[2n]h_2+st1$  and  $x[2n]h_2+st62$  seen in Figure 3.22(b) are computed and then stored to st1 and st62 seen in Figure 3.22(a), respectively. The entire computations required by an output sample are separately calculated at each cycle and the computation of each cycle in my proposed architecture seen in Figure 3.22(a) is listed in Table 3.2. After the first 32 clock cycles, the multiplication operations correlated with  $x[2n]$  and coefficients  $h_{\text{even}}$  have been finished.

And the input sample,  $x[2n+1]$ , appears after the first 32 clock cycles. In the 33<sup>rd</sup> clock (clkf3) cycles (counter = 32), the results of  $x[2n]h_0+x[2n+1]h_1$  and  $x[2n]h_2+x[2n+1]h_1+st1$  seen in Figure 3.22(b) are computed by  $x[2n+1]h_1+st63$  and  $x[2n+1]h_1+st1$  seen in Figure 3.22(a) where the contents of st63 and st1 are  $x[2n]h_0$  and  $x[2n]h_2+st1$  respectively. These two results are temporarily stored to the st63 and st1 respectively seen in Figure 3.22(a) at the end of the 33<sup>rd</sup> clock cycle. At the end of the 64<sup>th</sup> clock cycle, the results of  $x[2n]h_0+x[2n+1]h_1$  and  $x[2n]h_2+x[2n+1]h_1+st1$  are shifted to st62 and st0 so as to finish the required computations, respectively. The other computation correlated with  $x[2n+1]$  and  $h_{\text{odd}}$  at each cycle could be seen from Table 3.2.

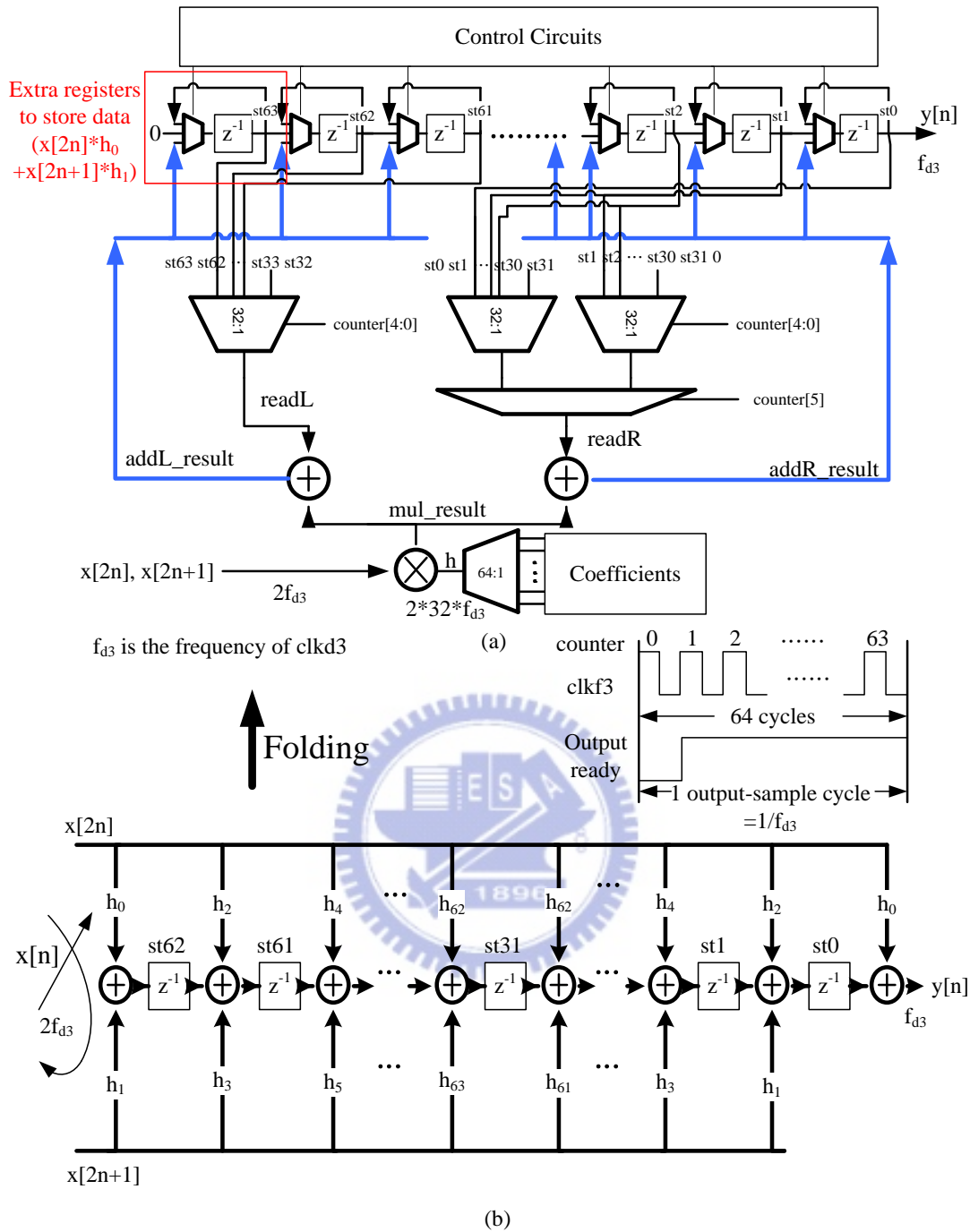


Figure 3.22 (a) Proposed folded architecture of decimation FIR filter based on transposed-form using polyphase decomposition (b) the unfolded one

For the below timing diagram of the my folded architecture, the contents of storage elements, st0, st1, st2, ....., st61, st62 and st63, in the beginning of shown output sample cycle are st0, st1, st2, ....., st61, st62 and 0, respectively.

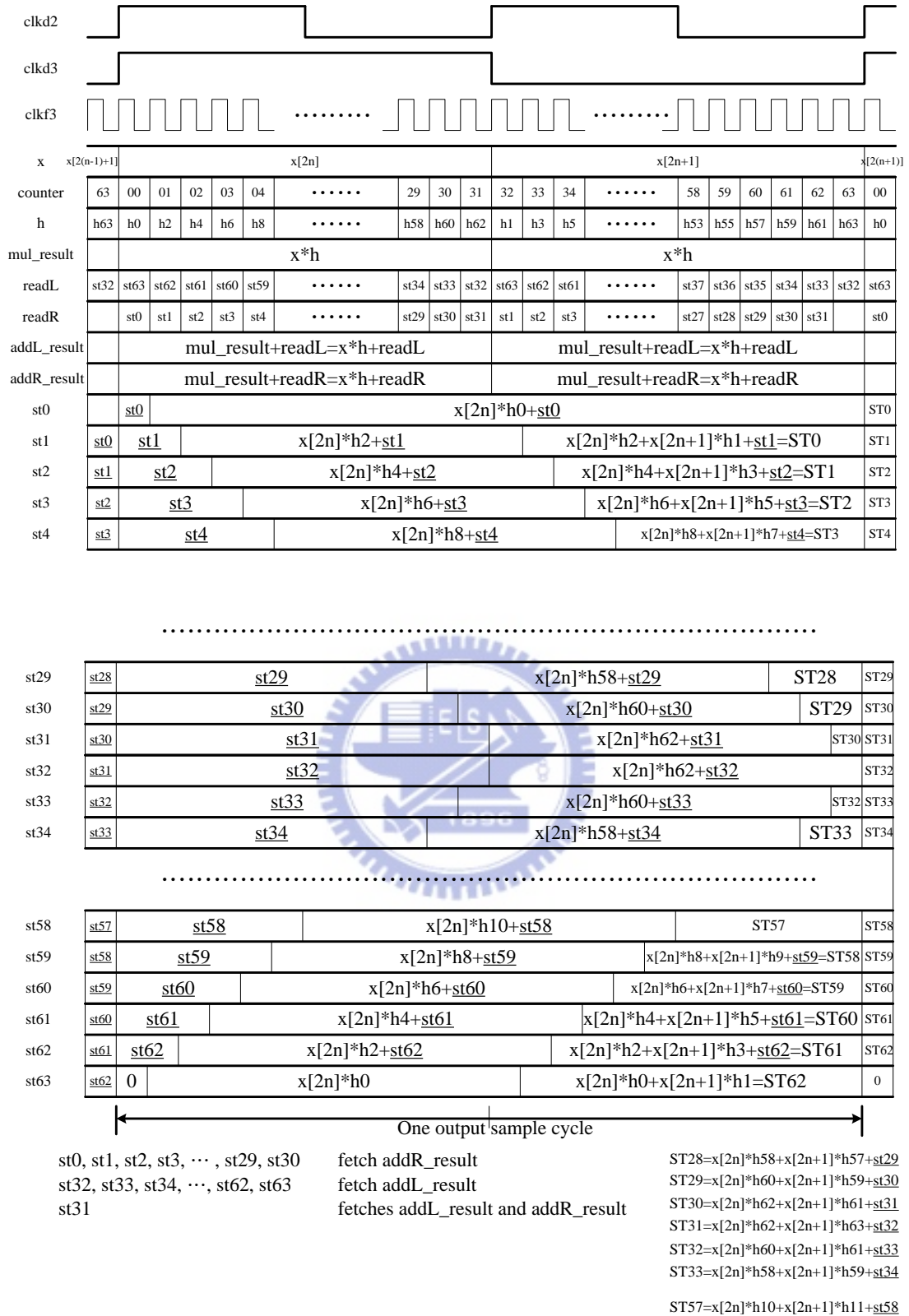


Figure 3.22(c) Timing diagram of my proposed architecture

Table 3.2 The computation of each cycle in Figure 3.22(a)

Left						Right				
<b>x[2n]</b>										
<b>counter</b>	0	1	...	30	31	31	30	...	1	0
<b>h<sub>even</sub></b>	h <sub>0</sub>	h <sub>2</sub>	...	h <sub>60</sub>	h <sub>62</sub>	h <sub>62</sub>	h <sub>60</sub>	...	h <sub>2</sub>	h <sub>0</sub>
<b>read</b>	st63	st62	...	st33	st32	st31	st30	...	st1	st0
<b>write</b>	st63	st62	...	st33	st32	st31	st30	...	st1	st0
<b>x[2n+1]</b>										
<b>counter</b>	32	33	...	62	63	62	61	...	32	
<b>h<sub>odd</sub></b>	h <sub>1</sub>	h <sub>3</sub>	...	h <sub>61</sub>	h <sub>63</sub>	h <sub>61</sub>	h <sub>59</sub>	...	h <sub>1</sub>	
<b>read</b>	st63	st62	...	st33	st32	st31	st30	...	st1	
<b>write</b>	st63	st62	...	st33	st31	st31	st30	...	st1	

At the end of the counter=63 cycle, all data shift (behavior of shift registers), which means that  $st63 \gg st62 \gg st61 \gg st60 \gg \dots \gg st2 \gg st1 \gg st0$ .

For example, in the Table 3.2, the columns with counter=62 illustrate that  $x[2n+1]h_{61}+st33$  is computed in the 63<sup>rd</sup> clock (clkf) cycle and then stored to st33 at the end of 63<sup>rd</sup> clock (clkf) cycle for the left column as well as  $x[2n+1]h_{61}+st31$  is computed in the 63<sup>rd</sup> cycle (clkf) cycle and then stored to st31 at the end of 63<sup>rd</sup> clock (clkf) cycle for the right column.

Note that the result of  $x[2n+1]h_{63}+st32$  for counter=63 (i.e. in the 64<sup>th</sup> clock cycle) is computed and then stored to st31 (not st32) at the end of the 64<sup>th</sup> clock cycle due to the data shifting requirement.

The entire computations listed in Table 3.2 are equivalent to the computations in Figure 3.22(b), which could be seen in Table 3.3. All required computations are listed in the left column of Table 3.3 for circuits shown in Figure 3.22(b) and the corresponding computations are listed in the right column of Table 3.3 for my proposed circuits shown in Figure 3.22(a). For example, the result of  $x[2n]*h_2+x[2n+1]*h_3+st62$  is computed and stored to st61 in Figure 3.22(b) during a clkd3 cycle. The corresponding computations,  $x[2n]*h_2+st62$ ,  $x[2n]*h_2+x[2n+1]*h_3+st62$ , and the result of  $x[2n]*h_2+x[2n+1]*h_3+st62$  stored to st61 are finished separately in the 2<sup>nd</sup>, 34<sup>th</sup> clock cycle and at the end of 64<sup>th</sup> clock cycle respectively for my circuit shown in Figure 3.22(a).



Table 3.3 The algorithmic operations of proposed folded architecture is equivalent to the unfolded decimation FIR filter in transposed-form using polyphase decomposition.

Non-Folding (Transposed-Form) (64 multipliers for Stage3)	Proposed Folding (1 multiplier for Stage3)
Clocked by clkd3	Clocked by clkf3
$x[2n]*h_0+st0 \Rightarrow y$ (the output data)	counter= =0 (in the first clock cycle)
$x[2n]*h_2+x[2n+1]*h_1+st1 \Rightarrow st0$	$x[2n]*h_0+st0 \Rightarrow y$
$x[2n]*h_4+x[2n+1]*h_3+st2 \Rightarrow st1$	.....
.....	.....
.....	counter= =1 (in the 2 <sup>nd</sup> clock cycle)
.....	$x[2n]*h_2+st62 \Rightarrow st62$
$x[2n]*h_4+x[2n+1]*h_5+st61 \Rightarrow st60$	(st62: $x[2n]*h_2+st62$ )
$x[2n]*h_2+x[2n+1]*h_3+st62 \Rightarrow st61$	counter= =33 (in the 34 <sup>th</sup> clock cycle)
$x[2n]*h_0+x[2n+1]*h_1 \Rightarrow st62$	$x[2n+1]*h_3+st62 \Rightarrow st62$
	(st62: $x[2n]*h_2+st62+x[2n+1]*h_3$ )
	counter= =63 (in the 64 <sup>th</sup> clock cycle)
	.....
	(at the end of the 64 <sup>th</sup> clock cycle)
	st62 $\Rightarrow$ st61
	.....

These operations require complicated control circuits which only govern little silicon area. And parts of control circuit used to arrange which result store to registers are shown in Figure 3.23.

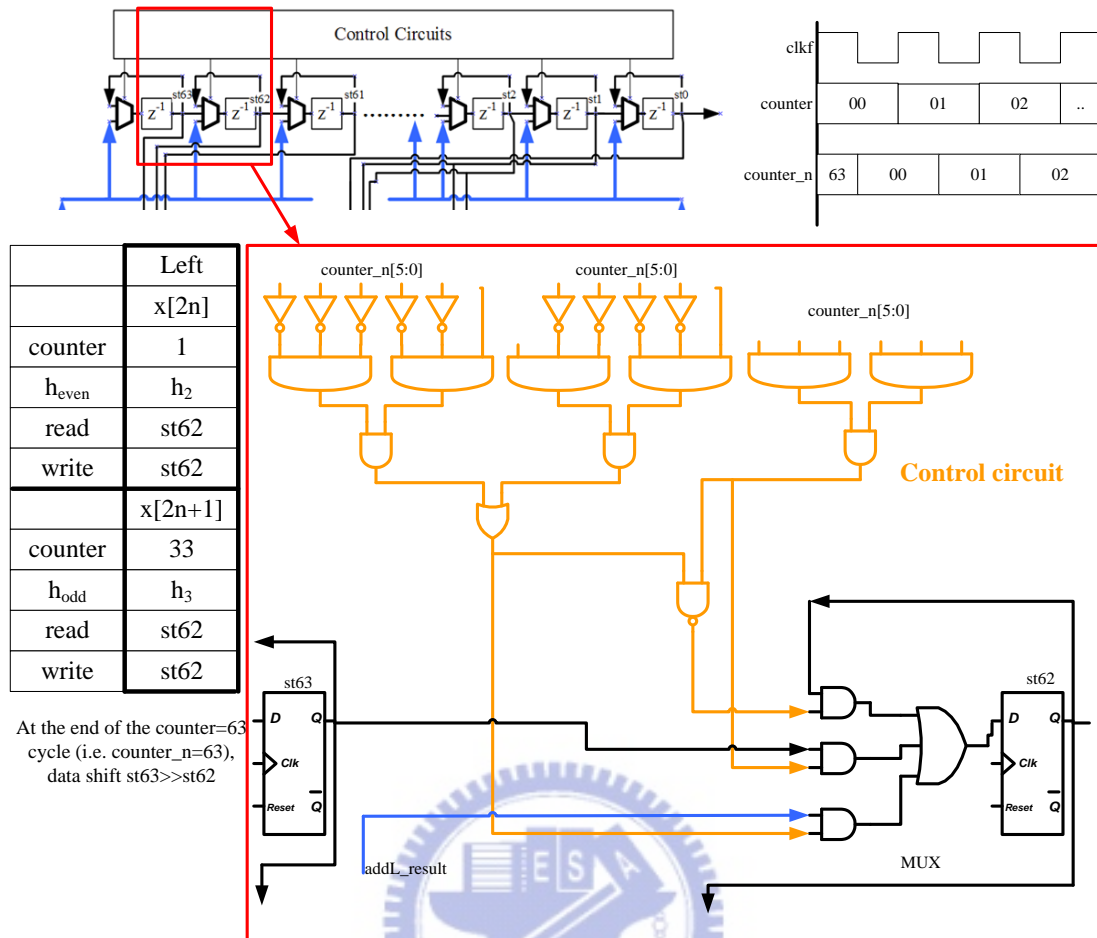


Figure 3.23 Parts of control circuits

The operations performed by parts of control circuits shown in Figure 3.23 are listed below:

```

counter_n=1
  x[k]*h2+st62=>st62
counter_n=33
  x[k+1]*h3+st62=>st62
counter_n=63
  st63=>st62
    
```

The parts of control circuits shown in Figure 3.23 control which data will store to the register st62 at the end of each clock cycle and the other registers are controlled by its individual control circuits as well. The counter\_n, a half clock cycle delay of counter, is used to be a control signal for multiplexers and prevent race condition because the value of counter and registers are both changed (or fetch value) at positive edge of clock (i.e., the value of counter is not constant around positive edge of clock,

which would cause race condition). In the Figure 3.23 case, addL\_result would store to st62 when counter\_n is 1 or 33, value stored in st63 would store to st62 (shifting operation) when counter\_n is 63 and the value stored in st62 would store to st62 (i.e., the value stored in registers is not changed) when counter\_n are other values.

### 3.7 Comparisons

Now, my proposed folded architecture will be compared with the previous works on decimation FIR filters with polyphase decomposition described in Section 3.2.2.

The comparison result of overall, detail area and detail power of decimator are listed in Table 3.4, Table 3.5 and Table 3.7 as well as the area reduction percentage of my proposed folded architecture compared with other implementations architecture is listed in Table 3.6.

Process: TSMC 0.18um

Logic Synthesis Tool: Synopsys Design Compiler

Clock Cycle Time: 17.5 ns

Wire Load Model: tsmc18\_wl10 (worst case)

Core level power before APR OSR64@25MHz by Prime Power

**Word-length of samples: 27-bits**

**Word-length of FIR filter coefficients: 20-bits**



Table 3.4 Comparison of decimator using the described implementations of FIR filter

		Cell Area ( $\mu\text{m}^2$ )	Average Power	Peak Power	Speed (throughput)
<b>Unfolded</b>	Direct-Form	2,042,184	5.526mW	0.9876W	2 <sup>nd</sup>
	Transposed-Form	1,882,399	5.542mW	1.183W	Fast
<b>Folded</b>	Direct-Form [12] [13] [14] [15]	693,410	16.96mW	0.2803W	4 <sup>th</sup>
	Transposed-Form (This design)	583,466	14.95mW	0.1905W	3 <sup>rd</sup>

Another folded architecture of FIR filter in transposed-form mentioned in Figure 3.14 (modified from straightforward implementation of a folded FIR filter [18]) doesn't meet the timing constraint; the slack is  $-0.87\text{ns}$ , cell area is  $533,350\text{um}^2$  and total power is  $327.5\text{mW}$ . Because it doesn't meet the timing requirement, it couldn't compare with other circuit architecture for justice.

Table 3.5 Detail area comparison of decimator using the described implementations of FIR filters

Area ( $\text{um}^2$ )	Order (FIR)	Direct	Transposed	Direct Folding	Transposed Folding (this design)
Stage1		42,205	42,195	42,215	42,221
Stage2	18	267,644	242,192	92,223	78,735
Stage3	126	1,562,255	1,427,906	388,782	292,761
Stage4	40	169,284	169,297	169,430	169,108
<b>Total</b>		2,042,184	1,882,399	693,410	583,466

[12][13][14][15]

Table 3.6 Area normalized to direct-form folding (27-bits word-length)

Cell Area	Direct	Transposed	Direct Folding	This Design
Stage2	2.90	2.63	1	0.85
Stage3	4.01	3.67	1	0.75
<b>Overall Decimator</b>	2.95	2.71	1	0.84

Table 3.7 Detail Power Comparison

Power (mW)	Direct	Transposed	Direct Folding	Transposed Folding (this design)
<b>Stage1</b>	0.6428	0.6397	0.6421	0.6423
<b>Stage2</b>	0.4160	0.4370	1.966	2.283
<b>Stage3</b>	1.554	1.540	11.14	9.130
<b>Stage4</b>	2.896	2.907	3.193	2.869
<b>Total (average)</b>	5.526	5.542	16.96	14.95
<b>Total (peak)</b>	987.6	1183	280.3	190.5

[12][13][14][15]

From the above comparison tables, they reveal that my proposed folded architecture requires the smallest silicon area and requires 24.6% less hardware (silicon area) compared with folded architecture in direct-form [12] [13] [14] [15] for the high order (126<sup>th</sup>-order) decimation FIR filter with polyphase decomposition. In addition, the other advantages of my proposed folded architecture compared with folded architecture in direct-form [12] [13] [14] [15] are shorter critical path (-one adder delay), shorter latency (-order/2 cycles) and smaller peak power (-30%). The average power of my design is in the same level with direct-form folding.

In order to make sure that the proposed folded architecture is still suitable for sample word-length with 16-bits, the comparison of detail area, area reduction percentage of my proposed architecture and detail power for 16-bits word-length are listed below in Table 3.8, Table 3.9 and Table 3.10, respectively.

Process: TSMC 0.18um

Logic Synthesis Tool: Synopsys Design Compiler

Clock Cycle Time: 16.5 ns

Wire Load Model: tsmc18\_wl10 (worst case)

Core level power before APR OSR64@25MHz by Prime Power

**Word-length of samples: 16-bits**

**Word-length of FIR filter coefficients: 20-bits**

Table 3.8 Detail area comparison with 16-bits word-length

Area (um <sup>2</sup> )	Order (FIR)	Direct	Transposed	Direct Folding	Transposed Folding (this design)
Stage1	-	41,526	41,526	41,486	41,496
Stage2	18	218,687	206,185	55,265	48,316
Stage3	126	1,401,173	1,319,290	224,320	180,251
Stage4	40	101,042	101,042	101,036	101,874
<b>Total</b>	-	1,763,251	1,668,867	422,940	372,590

[12][13][14][15]

Table 3.9 Area normalized to direct-form folding  
(16-bits word-length)

Cell Area	Direct	Transposed	Direct Folding	This Design
Stage2	3.96	3.73	1	0.87
Stage3	6.2	5.88	1	0.80
<b>Overall Decimator</b>	4.17	3.95	1	0.88

The area reduction percentage of my proposed architecture compared with direct-form folding architecture is decreased because the multiplier governs more percentage of silicon area in the 2<sup>nd</sup> and 3<sup>rd</sup> stage, which is resulted from the non-decreased word-length of FIR filter coefficients. The main advantage of my proposed folded architecture compared with folded architecture in direct-form is half registers usage so the decrease in the percentage of registers' silicon area will result in that area reduction percentage of my proposed architecture decrease, too.

Generally speaking, the word-length of samples (bits resolution) is decreased; the word-length of FIR filter coefficients must be decreased, too. Because the resolution of sample is low, the resolution (word-length) of FIR filter coefficients required by the sample resolution is unnecessary so accurate (i.e. word-length of coefficient could be smaller even smaller than word-length of sample).

Thus the area reduction percentage of my proposed architecture shown in Table 3.6 is more reasonable than Table 3.9.

Table 3.10 Detail power comparison using 16-bits word-length

Power (mW)	Direct	Transposed	Direct Folding	Transposed Folding (this design)
<b>Stage1</b>	0.6409	0.6391	0.6412	0.6405
<b>Stage2</b>	0.4720	0.4471	0.9586	0.8928
<b>Stage3</b>	1.778	1.706	4.384	3.826
<b>Stage4</b>	1.593	1.614	1.775	1.651
<b>Total (average)</b>	4.501	4.424	7.778	7.032
<b>Total (peak)</b>	988.7	1110	178.1	130.1

[12][13][14][15]

Basically, the advantages of FIR filters in transposed-form are shorter critical path due to inserting the storage elements (registers) between adders and half registers requirements due to sharing storage elements using polyphase decomposition compared with FIR filters in direct-form.

The FIR filters using unfolded structure require many functional units (multipliers and adders), which result in large silicon area no matter what the forms (structures) are. However, the power consumed by FIR filters using unfolded structure is much smaller than folded architecture because the logic gates required by a multiplier of unfolded structure are much less than logic gates required by the multiplier of folded architecture so as to obtain small power consumption. The reasons why the logic gates of a multiplier for unfolded structures are less than for folded architecture are that one input port of multiplier for unfolded structure is constant (fixed FIR filters coefficients) and the cycle time for each multiplier of unfolded structure to calculate is much longer than cycle time for the folded architecture's multiplier. The trade-off between unfolded and folded FIR filters is shown in Figure 3.24.

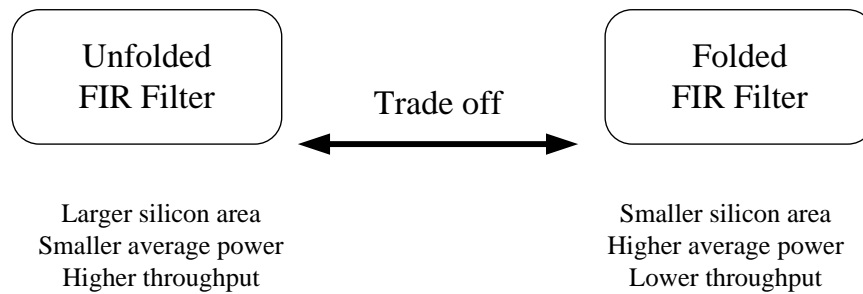


Figure 3.24 Trade-off between unfolded and folded FIR Filter

For comparison on folded architectures, the advantages of my proposed folded architecture are summarized below:

- Compared with direct-form folding FIR [12][13][14][15]
  - Area reduction [half registers] (-19% @18<sup>th</sup>-order ~ -24% @126<sup>th</sup>-order)
  - Short critical path (-one adder delay).
  - Short latency (-order/2 cycles).
  - Small peak power (-30%, same level average power).
- Compared with other transposed-form folding FIR [18]:
  - Much less than half power.

Because my proposed folded architecture is based on transposed-form, it results in the advantages of shorter critical path (registers are between adders), shorter latency (output is produced after a multiplication and an addition operation) and smaller peak power (half data shift in the same time due to half registers requirement) compared with folded FIR filter architecture in direct-form [12][13][14][15]. In addition, the half register requirement is obtained by changing computation procedures and using extra control circuits described in Section 3.6.

The differences between my folded architecture and folded architecture mentioned in [18] (seen in Figure 3.12 without polyphase decomposition or Figure 3.14 with polyphase decomposition) are that my architecture doesn't require 2-times fast clock to maintain throughput and doesn't shift all data each cycle so as to obtain less than half power consumption compared with folded architecture mentioned in [18]. The requirement of 2-times fast multiplication operation also make the timing of folded architecture seen in Figure 3.14 more critical. However, my architecture overheads compared with [18] are one adder, multiplexers and control circuits.

The trade-off between these folded architectures is shown in Figure 3.25.



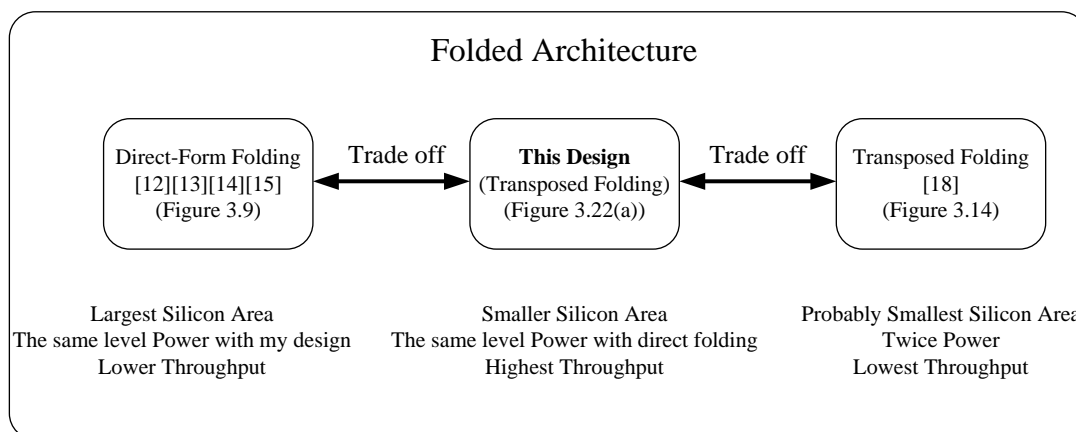


Figure 3.25 Trade-off between the three folded architectures

### 3.8 Implementation Results

The layout of decimator is shown in Figure 3.26, which is fabricated in TSMC 0.18 $\mu\text{m}$  CMOS mixed signal RF general purpose MiM Al 1P6M process. In addition, the information about the number of pad and silicon area of decimator is listed in Table 3.11. In my chip, there are 5 input pads, 27 output pads, 4-pair core power pads and 8-pair IO power pads. The core area (active area) is  $805 \times 805 = 648,025 \text{ um}^2$  (in utilization  $\sim 90\%$ , i.e., cell area =  $583,466 \text{ um}^2$ ). However, the die size is governed largely by IO pads and bonding pads. As a result, the die area is  $1810 \times 1810 = 3,276,100 \text{ um}^2$ .

For IO power, one set IO power pad can provide the power for 3~4 output pads or 6~8 input pads. As a result, eight sets IO power pads are given to provide power for these IO pads (27 output pads and 5 input pads) in my chip. For core power, one set core power pad can provide 40mA current for core cells. From the power simulation of APR tool (SocEncounter), the power dissipations of my chip are 17.58mW and 34.31mW at 51.2 MHz (input sampling-rate) for decimation factor 128 and 64, respectively. However, four sets core power pads are given to prevent IR-drop and electron-migration.

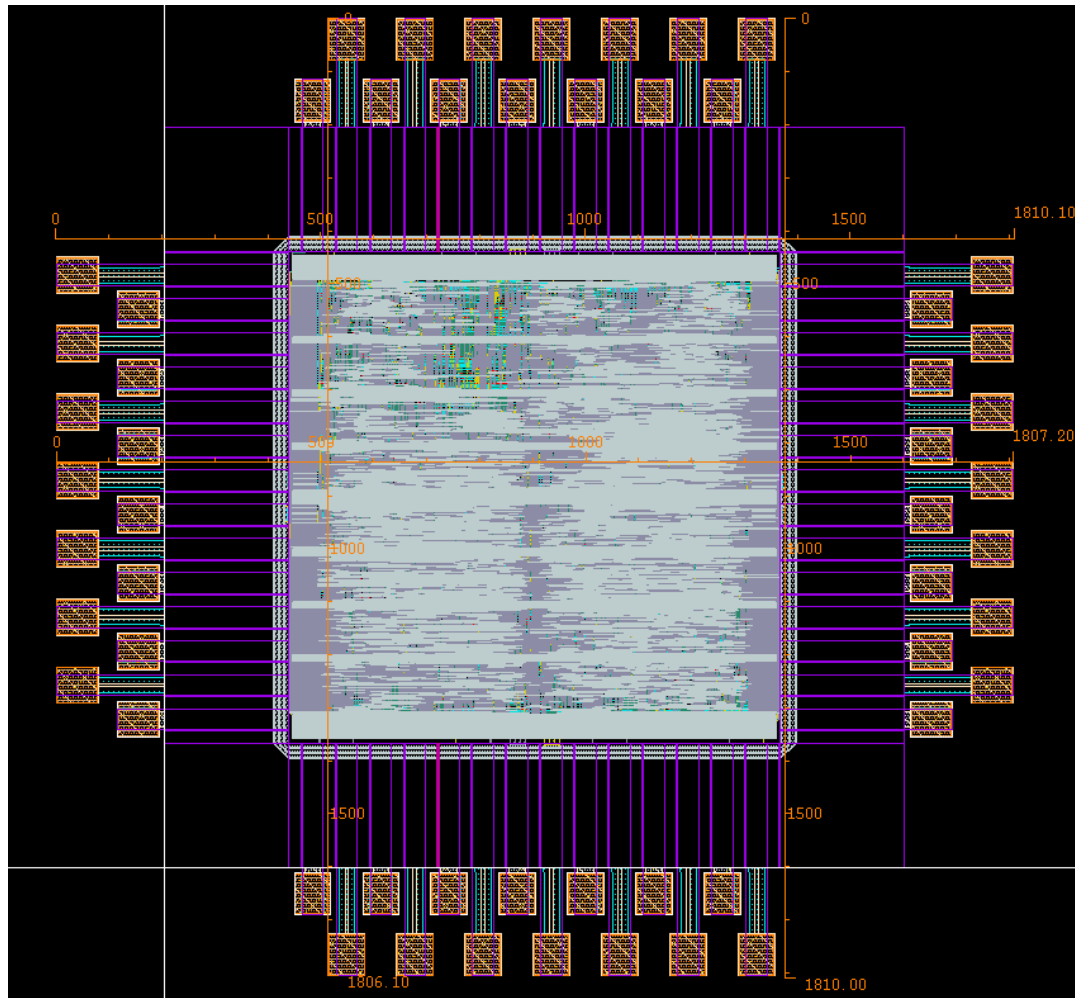


Figure 3.26 Layout of decimator

Table 3.11 Pad and silicon area of the chip (decimator)

<b>Input pad</b>	5
<b>Output Pad</b>	27
<b>Core Power</b>	4 pairs
<b>IO Power</b>	8 pairs
<b>Total Pad</b>	56
<b>Active area</b>	=805x805 $\mu\text{m}^2$
<b>Die area</b>	=1810x1810 $\mu\text{m}^2$

The place and route of cells are finished by APR tool (SocEncounter). And the finished layout is stream-in by Virtuoso and shown in above figure. The IO cells and logic cells are shown as black boxes because these cells are virtual cells (confidential to student users), which must be replaced in CIC.

### 3.8.1 Pad Assignment

The pad assignment of the chip is shown in Figure 3.27.

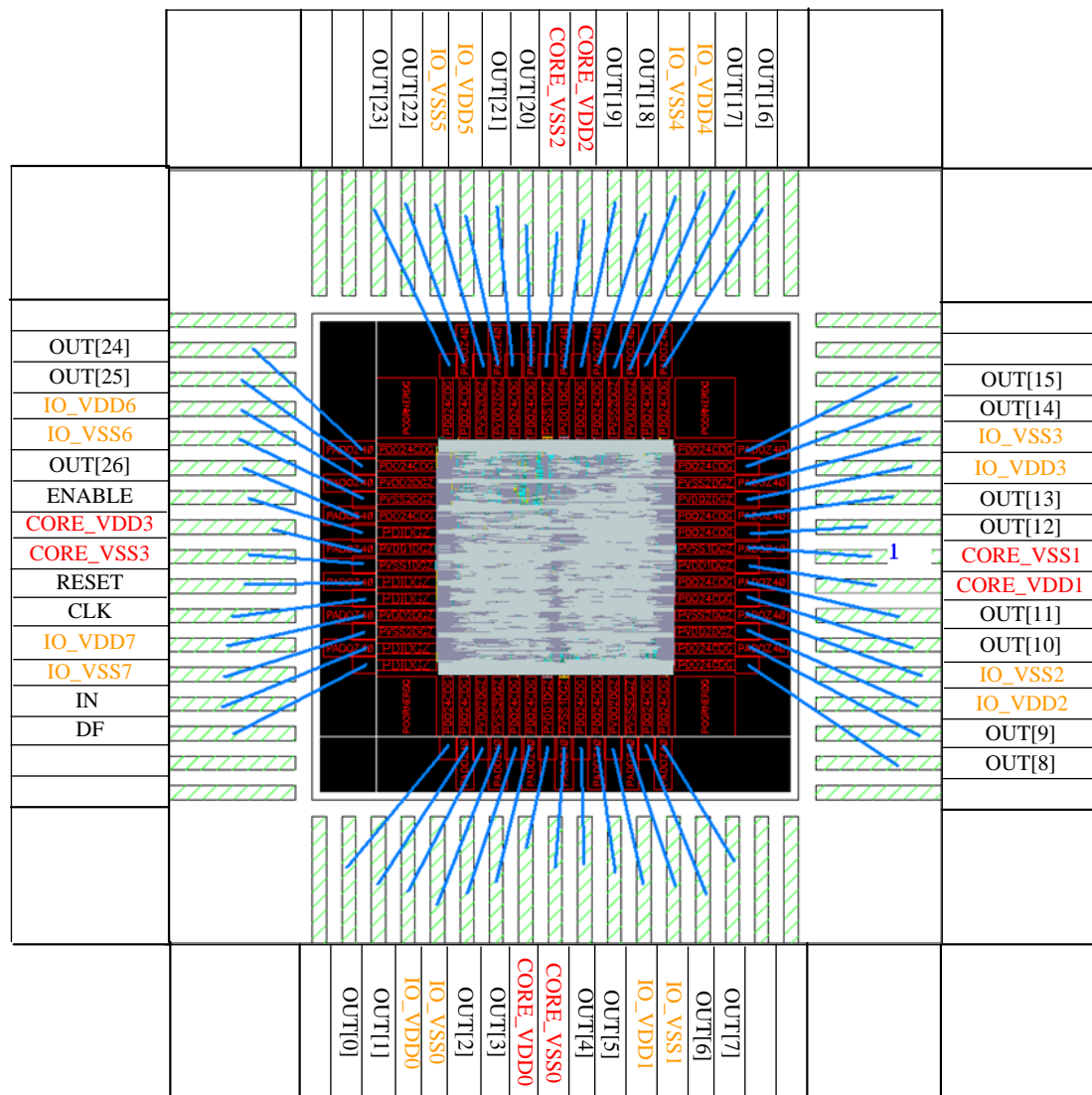


Figure 3.27 Pad assignment

The IO power pads and core power pads are assigned and distributed symmetrically around four-side of chip to obtain probably minimum IR-drop.

The above figure also shows the connections between package and die, i.e. bonding information for chip. The chosen package type is 68LCC (68 pins) and total pins (pads) of my design are 56. Thus, few pins of the package near corners are not assigned.

### 3.9 Decimator Simulation Result

The simulation and verification flow is shown in Figure 3.28 below.

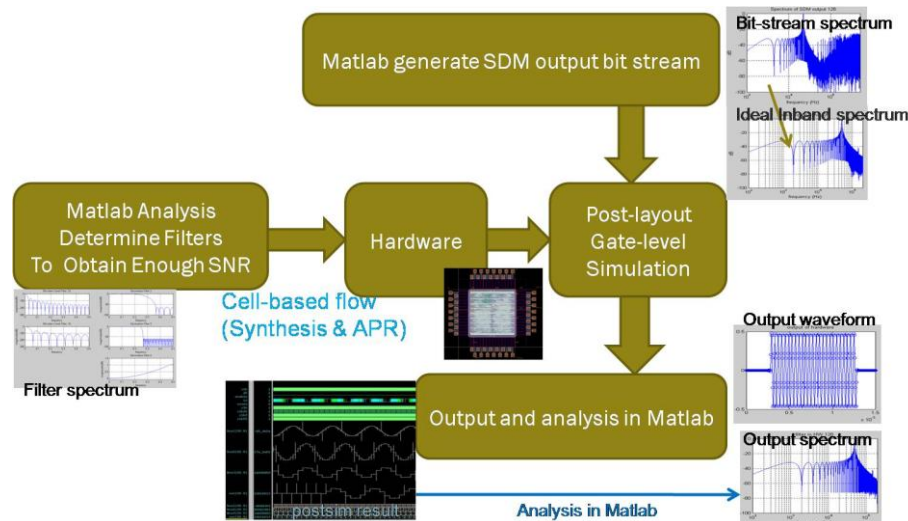


Figure 3.28 The post-layout gate-level simulation and verification flow for decimator

After the decimator is designed (in Chapter 2) and implemented (in Chapter 3) to the layout level, the function of the circuit (decimator) in layout level must be verified. In order to verify the function of the circuit (layout-level), a SDM output bit-stream is used as input to stimulate the circuit (decimator). The simulated output logic values can be obtained and then can be fetched and analyzed further in Matlab. The decimator's output in time-domain could roughly judge the function of decimator is incorrect or not because the expected decimator output is the sampled version of SDM's input. For further and precise verification, to compare the spectra of SDM bit-stream and fetched output (output of decimator) can confirm the behavior of the circuit (decimator) is correct or not. The function of decimator is to preserve the in-band spectrum of decimator's input (SDM bit-stream). Thus, if the spectrum of decimator's output is equal to the in-band spectrum of decimator's input, the behavior of circuit (decimator) is correct.

#### 3.9.1 OSR=128

The post-layout gate-level simulation result of decimator with decimation ratio 128 is shown in Figure 3.29 below and then verified in frequency domain using Matlab seen in Figure 3.30. The input of SDM is a 50 kHz sinusoidal signal (continuous-time signal, namely, analog signal) and the output of SDM (SDM bit-stream, discrete-time signal, namely digital signal) is the input of decimator. Thus, the expected decimator's output is a sampled-version 50 kHz sinusoidal signal. The sampling-rate of input and output of decimator are  $BW*2*OSR=51.2$  MHz and  $BW*2 = 400$  kHz, respectively.

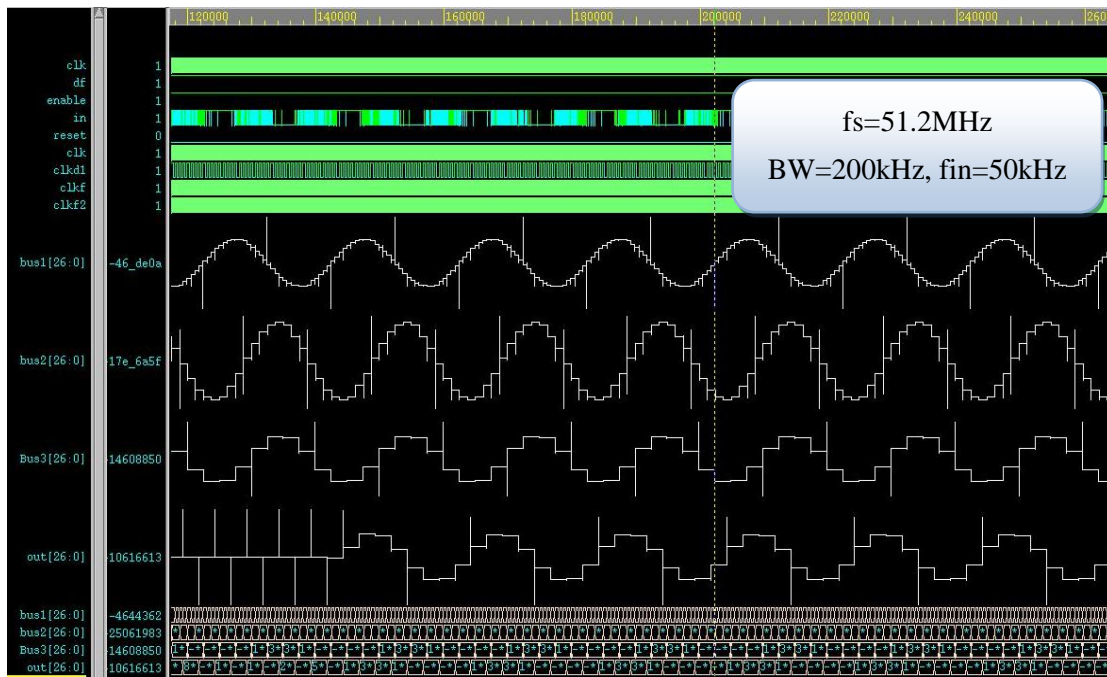


Figure 3.29 Post-layout gate-level simulation result with decimation factor=128 at nWave of workstation

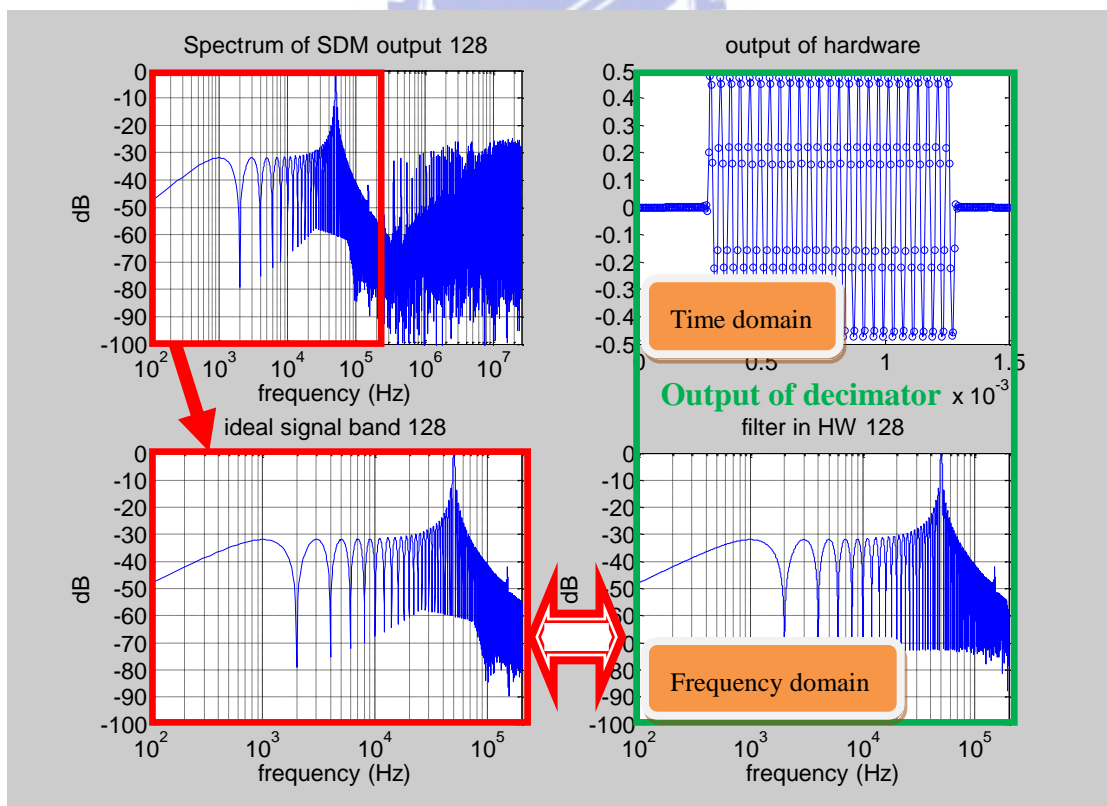


Figure 3.30 Verification in time domain and frequency domain for decimation ratio 128 using Matlab

In Figure 3.29, the digital signals bus1[26:0], bus2[26:0] and bus3[26:0] are output of stage1, stage2 and stage3 of decimator, respectively. The digital signal out[26:0] is the decimator's output. For convenience, these digital signals are also shown as analog waveform to represent the magnitude of logic values. Thus, these digital signals include two waveforms (digital waveform and analog waveform) in above post-layout-simulation figure. Note that all signals in my design are digital signal. When the digital signals are shown as analog waveform, the transition of digital signal (logic values changing) would result in glitch in analog waveform. It is a normal phenomenon to depict digital signals as analog waveform.

From the above simulation figure, a sampled 50kHz sinusoidal signal (BW=200kHz implies Nyquist rate or output sampling rate=400kHz, 400kHz / 8 samples per period=50kHz) appears in the decimator's output (out[26:0]). Basically, the function of decimator is correct because the output of decimator (digital signal) is a sampled-version of SDM's input (analog signal).

To verify further, the output of decimator is fetched to personal computer and analyzed in Matlab. The time-domain and frequency-domain of decimator's output are shown in right side of Figure 3.30. The left-top of Figure 3.30 is the (entire-band) spectrum of decimator's input (SDM output bit-stream). The spectrum of decimator's output shown in right-bottom of Figure 3.30 is equivalent to the in-band spectrum of decimator's input (SDM output bit-stream) shown in left-bottom of Figure 3.30. Thus, the behavior of decimator is verified as correct. (The notches of two spectrums are a little different due to its distinct spectrum resolution. The points of two spectrums shown in left-top and right-bottom of Figure 3.30 are the same  $2^{20}$ . The spectrum shown in left-bottom of Figure 3.30 is the part (in-band) of spectrum, so the point of the spectrum is  $2^{20}/128=2^{13}$ . In addition, the point  $2^{27}$  is out of memory in Matlab.)

### 3.9.2 OSR=64

The post-layout gate-level simulation result of decimator with decimation ratio 64 is shown in Figure 3.31 below and then verified in frequency domain using Matlab seen in Figure 3.32. As described above, the input of SDM is still a 50 kHz sinusoidal signal (continuous-time signal, namely, analog signal) and the output of SDM (SDM bit-stream, discrete-time signal, namely digital signal) is the input of decimator. Thus, the expected decimator's output is a sampled-version 50 kHz sinusoidal signal. The sampling-rate of input and output of decimator are  $BW*2*OSR=25.6$  MHz and  $BW*2 = 400$  kHz, respectively.

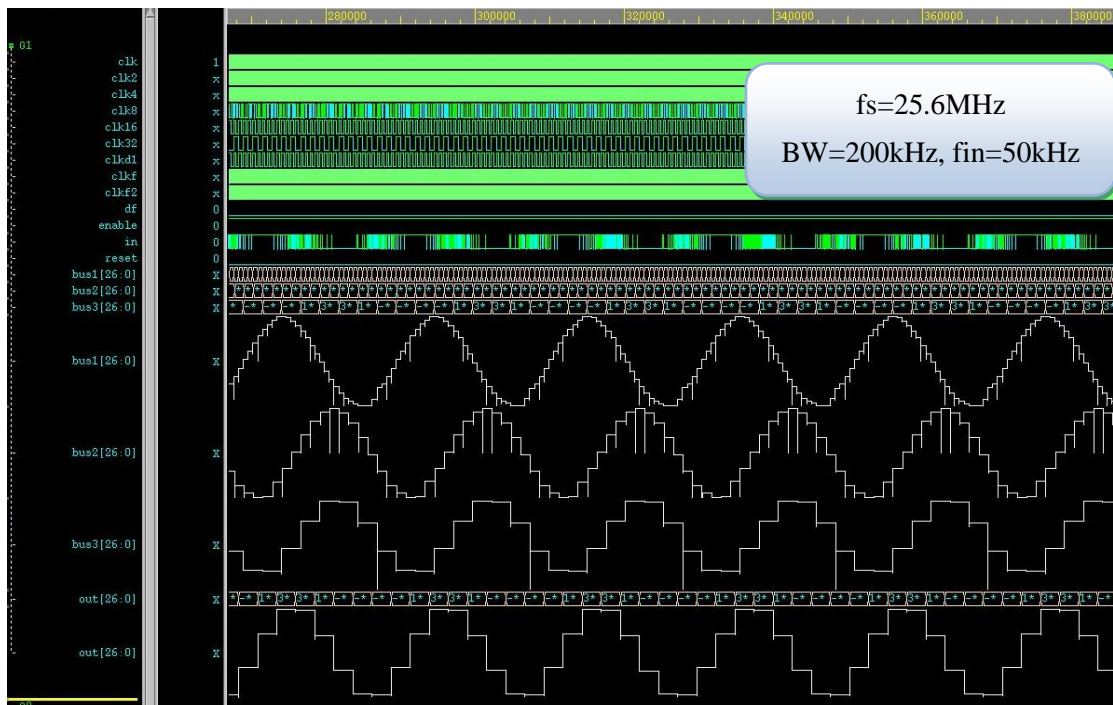


Figure 3.31 Post-layout gate-level simulation result with decimation factor=64 at nWave of workstation

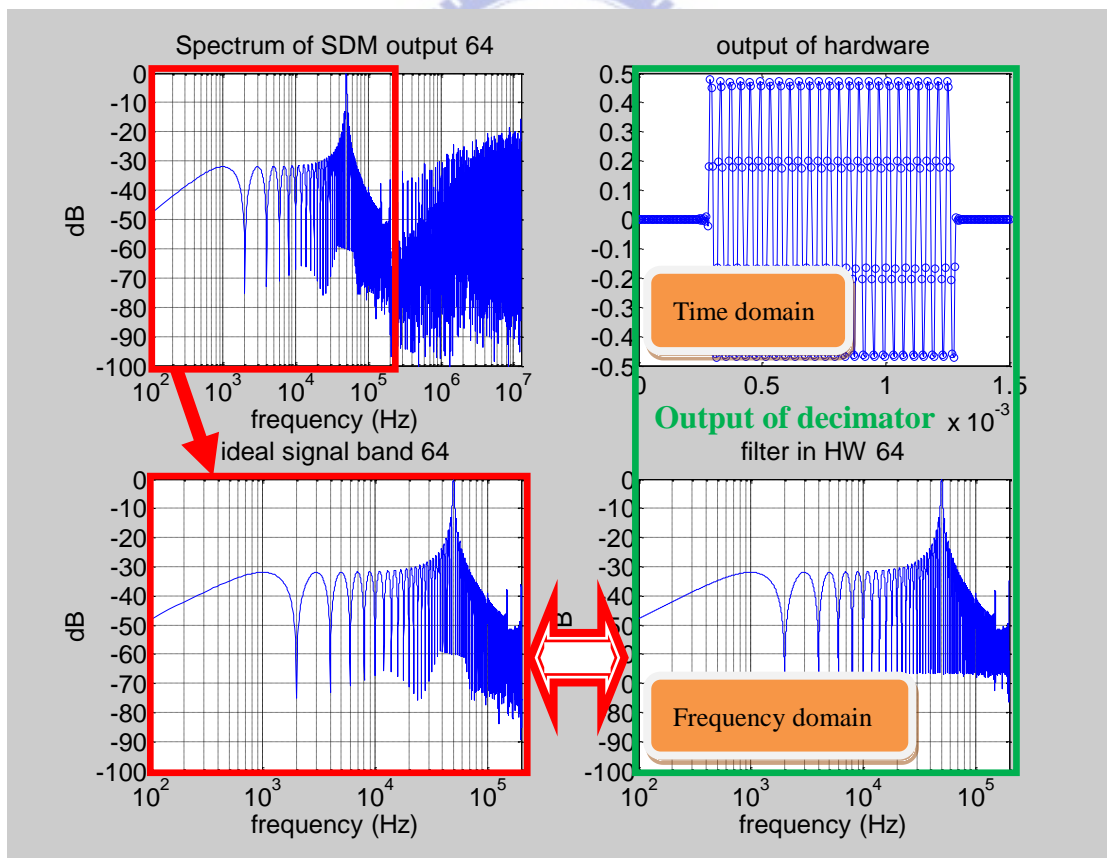


Figure 3.32 Verification in time domain and frequency domain for decimation ratio 64 using Matlab

Similarly, these digital signals are also shown as analog waveform to represent the magnitude of logic values. Thus, these digital signals include two waveforms (digital waveform and analog waveform) in above post-layout-simulation figure. And the glitch is normal phenomenon to depict digital signals as analog waveform. The spectrum of decimator's output shown in right-bottom of Figure 3.32 is equivalent to the in-band spectrum of decimator's input (SDM output bit-stream) shown in left-bottom of Figure 3.32. Thus, the behavior of decimator is verified as correct.

### 3.10 Specification Table

The specification table of chip (decimator) is summarized in Table 3.12.

Table 3.12 Specification

Process	TSMC 0.18um		
Package	LCC68 (68pin)		
Num of Pads	56 Pads		
Die Area	1.81 x 1.81 mm <sup>2</sup> (core size=0.8x0.8mm <sup>2</sup> , utilization=90% )		
Operating Frequency	55MHz		
Power Consumption (By SocEncounter)	17.58mW @51.2MHz for decimation factor=128		
	34.31mW @51.2MHz for decimation factor=64		
Power Consumption (By Nanosim)	40mW (1.8V x 22mA[rms])	20mW (1.8V x 11.1mA[avg])	@51.2MHz,df=128
	108mW (1.8V x 60mA[rms])	37.1mW (1.8V x 20.6mA[avg])	@51.2MHz,df=64

The operating frequency (input sampling-rate) in my design is 51.2 MHz ( $BW*2*OSR=200\text{ kHz} *2 * 128 = 51.2\text{ MHz}$ ) which is overdesigned to 55MHz. And the power dissipations simulated by SocEncounter are 17.58mW and 34.31mW at operating frequency 51.2 MHz for decimation factor 128 and 64, respectively. Besides, the power dissipations simulated by Nanosim in average mode are 20mW and 37.1 mW at operating frequency 51.2 MHz for decimation factor 128 and 64, respectively. The power dissipations simulated by Nanosim in RMS mode are 40 mW and 108 mW at operating frequency 51.2 MHz for decimation factor 128 and 64, respectively.



### 3.11 Paper Comparison

Finally, paper comparison of decimator is listed in Table 3.13.

Table 3.13 Paper comparison

Ref	Process	Die Area (mm <sup>2</sup> )	Operating Frequency (Signal BW)	Bit	Num of Stage	Max Filter taps	Power	Year	Comment
[12]	FPGA		16MHz (125kHz) OSR=64	16	3	127	-	2005	Direct Folding
[13]	FPGA		44.8/12.8MHz (700k/100kHz) DECT/GSM OSR=32/64		3	49	-	2002	Direct Folding
[17]	0.6um	21 (~10)	32MHz (250kHz) OSR=64	24	-	127	490mW	2000	CSD,A/D
[15]	0.18um	1.96	6MHz (47kHz) OSR=64	20	3	63	-	2006	Direct Folding
This Design	0.18um	3.27 (0.64)	55MHz (200kHz) (400kHz only OSR64) OSR=128/64	27	4	127	16.7mW (df=128) 30.6mW (df=64)		Transposed Folding

In die area column of Table 3.13, the number in parentheses is the core area.

Because the process, operating frequency, the word-length of a sample, the number of stages, and the number of taps of FIR filters are quite distinct for these decimator papers, it is difficult to compare the FIR filters' silicon area of each design from above information. From above table, the used folded architectures are all in direct-form except my design. Basically, the above table reveals that my decimator is good enough in each column (operating frequency and so on). And reference paper [13] and my decimator have two decimation ratios.

---

# CHAPTER

# 4

---

## Testing and Measurement Results

---

In this chapter, the testing environment and result of decimator fabricated in TSMC 0.18 $\mu$ m process would be illustrated. In addition, the package type of the chip is LCC68. For convenience, it could be tested and measured by auto test equipment (ATE), the Agilent 93000 Soc Series in CIC where the device under test board (DUT) is provided. Thus, the time required to setup the testing environment is short since the printed circuit board (PCB) is unnecessary to be prepared.

### **4.1 Introduction to Digital IC Testing using Agilent 93000 in CIC**

The Agilent 93000 test system seen in Figure 4.1 is composed of test-head, DUT (device-under-test) board and DUT interface, etc [19] [20].



Figure 4.1 P600 test system of Agilent 93000 SoC Series

The specifications of Agilent 93000 are summarized in Table 4.1 below.

Table 4.1 Specification of Agilent 93000

<b>Digital channels</b>	320pins
<b>Data Rate</b>	660Mbps
<b>Vector Memory (per channel)</b>	28MVectors
<b>Scan Memory (per channel)</b>	84MVectors
<b>DPS channels</b>	8 pairs (7V,6A)
<b>High Resolution AWG</b>	16 bits, 30Msps sampling rate
<b>High Speed AWG</b>	12 bits, 500Msps sampling rate
<b>High Resolution Digitizer</b>	16 bits, 3MHz bandwidth, 2Msps
<b>High Speed Digitizer</b>	12 bits, 100MHz bandwidth, 41Msps

In Table 4.1, the DPS and AWG denote device-power supplies and arbitrary-waveform-generator. Besides, the test development flow is shown in Figure 4.2 to understand the testing steps.

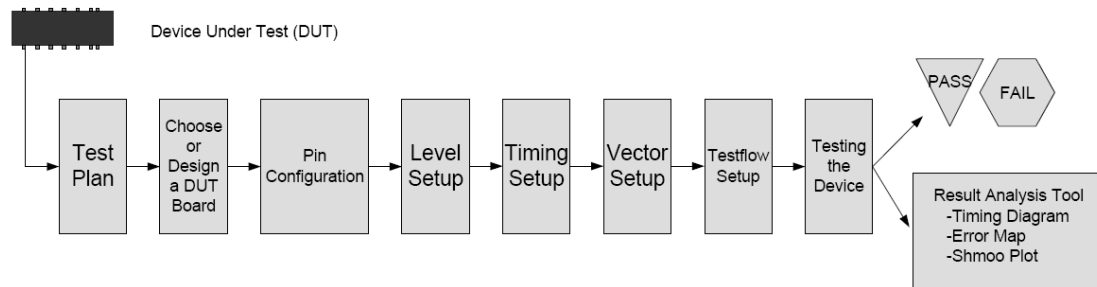


Figure 4.2 Test development flow

The meanings of testing steps are shown as follow: (1) test plan is to determine what kind of test will be performed; (2) according to the package type (DIP48, PLCC68, PLCC84, CQFP100, CQFP128, CQFP144, CQFP160, and CQFP208 supported by CIC), choose the DUT board where the chip is put on the socket of DUT board; (3) pin configuration is to set the input, output and power pins of chip to the test channel; (4) level setup is to set the voltage and current limit of power supply, drive voltage ( $V_{IL}$ ,  $V_{IH}$ ) and compared voltage threshold ( $V_{OL}$ ,  $V_{OH}$ ); (5) timing setup is to set the clock cycle time and waveform of each symbol; (6) vector setup is to describe the testing waveform by vector-format according to the predefine waveform; (7) test-flow setup is to load and set the related file; (8) test device and the result shown is pass or fail.

Pictures of testing environment are shown in Figure 4.3 below



(a)

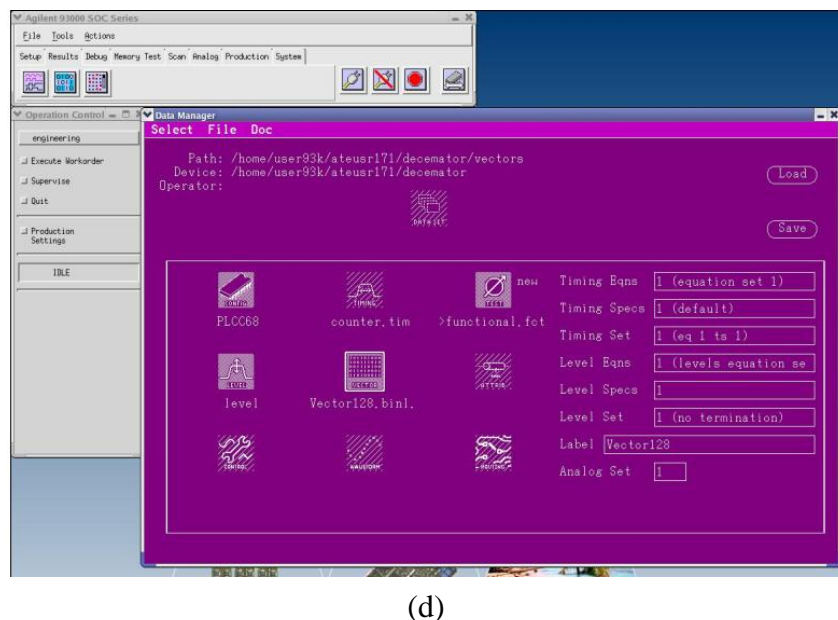
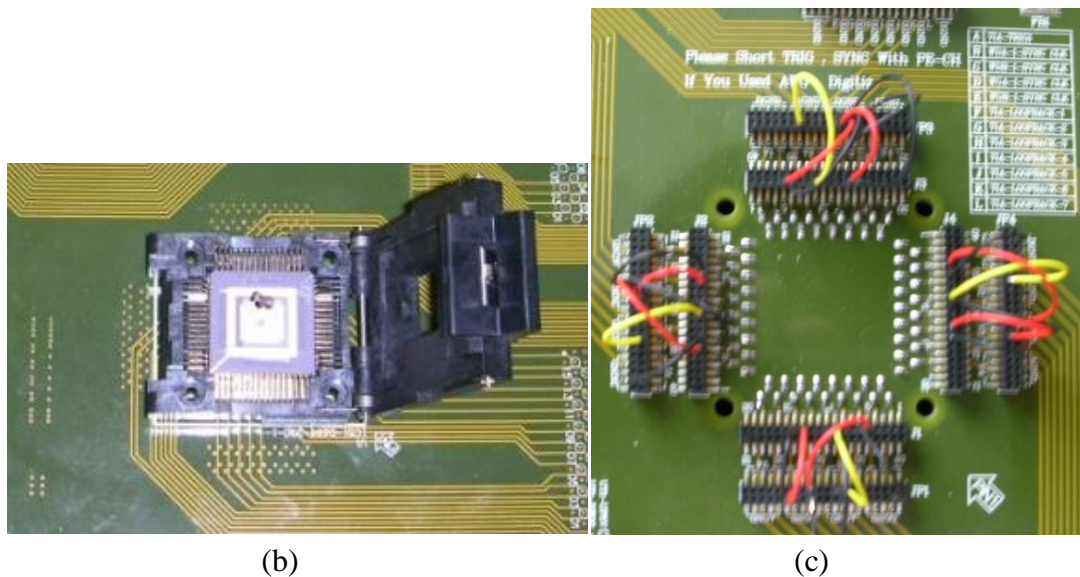


Figure 4.3 (a) DUT board on test-head of Agilent 93000 (b) chip in socket of DUT board (c) the reverse-side of DUT board wired the core-power and io-power of the chip to power-supplies pins (d) Software (SmarTest) used to manipulate the Agilent 93000 in workstation (unix-system)

The test-pattern for Agilent 93000 is composed of drive vector (input of DUT) and expected vector (expected output of DUT) shown in Figure 4.4. The drive vectors are used to stimulate the DUT (chip) and the expected vectors, which are the same logics value at post-layout simulation, are used to compare with the outputs of DUT (chip) measured by Agilent 93000. The detail test patterns for Agilent 93000 are illustrated in Appendix B.

Test Pattern for Agilent 93000					
Drive Vector					Expected Vector
IN	ENABLE	RESET	CLK	DF	OUT[26:0]
0	0	1	1	1	00000000000000000000000000000000
0	0	0	1	1	110001100000001011111010000
0	1	0	1	1	111010111101000011110101001
1	1	0	1	1	000111000000101001011010001
0	1	0	1	1	001111001000010110100000000
.....					
.....					

Figure 4.4 Test-pattern for Agilent 93000 composed of drive vector (input of DUT) and expected vector (expected output of DUT)

The function test is illustrated as Figure 4.5. The drive vectors and expected vectors are described in the test-patterns for Agilent 93000 seen in Appendix B. The response of function test is pass or fail. Pass means the all measured output of DUT is identical to the expected vectors (post-simulation output). If any measured output is different from expected vector, the response of function test is fail.

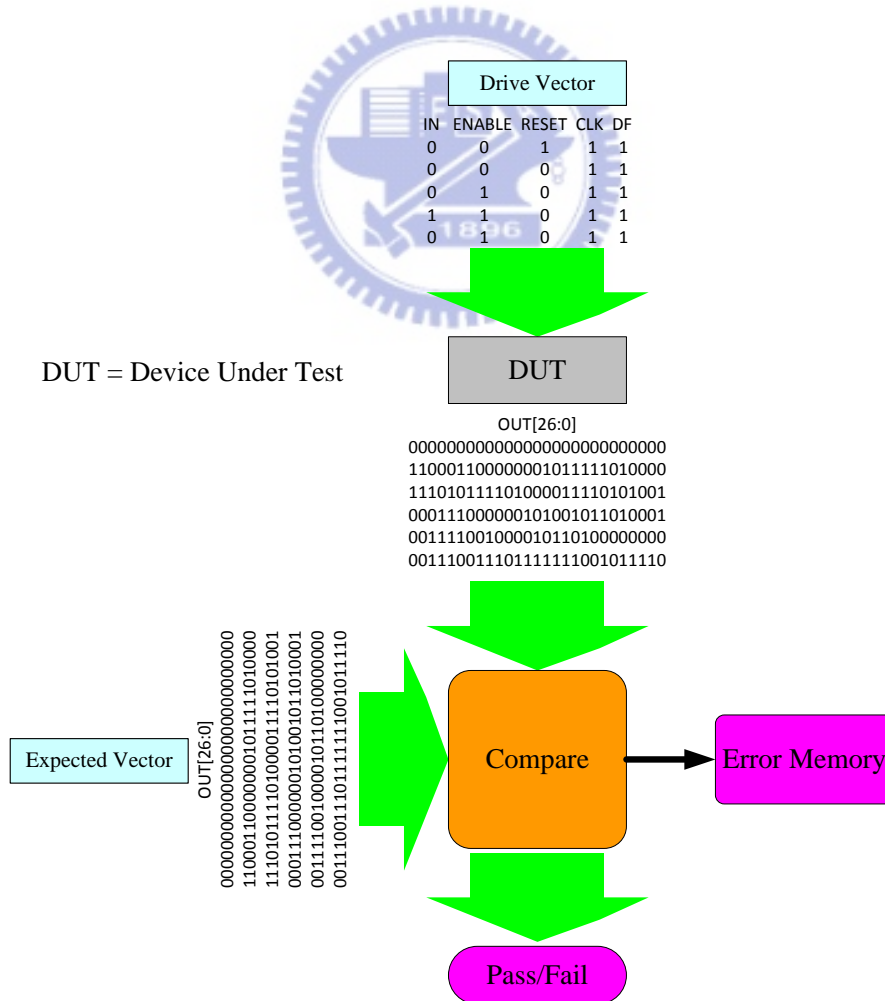


Figure 4.5 Flow of function test

## 4.2 Shmoo Plot

A shmoo plot is a graphical display of the response of a chip varying over a range of conditions and inputs. The voltage, temperature and operating frequency could be the conditions as well as the testing result (pass or fail) could be the response for the chip.

The conditions of shmoo plot in my testing are the voltage (core power, VDD) and the operating frequency, which are ranged from 1.62V to 1.98V (x-axis) and 1MHz to 100MHz (y-axis), respectively. The response of chip is pass or fail. The shmoo plot reveals that the function of the chip is correct or not at certain voltage (core power, VDD) and certain operating frequency.

### 4.2.1 OSR=128

The shmoo plot of the chip (decimator) using bits-stream of SDM with OSR 128 as stimulus is shown in Figure 4.6.

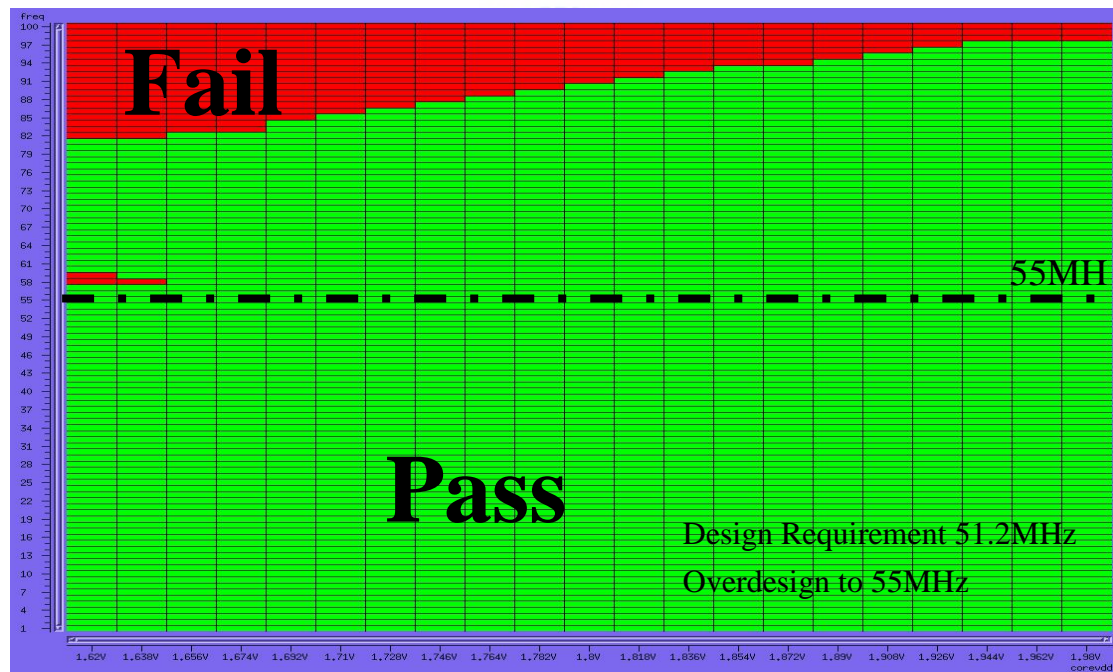


Figure 4.6 Shmoo plot (128)

From the above Shmoo plot, it is obvious that the requirement of operating frequency 51.2 MHz is met even at worst case (VDD=1.62V). Besides, the circuit could operate correctly at higher operating frequency when the core supply voltage is increased. In the implementation process, the circuit (decimator) is designed and guaranteed to operate correctly below 55MHz clock rate. It is consistent with the above Shmoo plot. Furthermore, the fails around core-VDD 1.62V and operating

frequency 58MHz are because the logic values of few cycles of few nodes correlated with reset signal are uncertain and only decide to wrong logic around these VDD and operating frequency. The circuit could operate correctly much higher than 55MHz (90MHz at VDD=1.8V) at decimation ratio 128 except the above condition because over designed operating frequency (input sampling-rate) is 55MHz no matter the decimation ratio is 128 or 64. In other words, there are timing slacks for the 2<sup>nd</sup>, 3<sup>rd</sup> and 4<sup>th</sup> stages at decimation ratio 128 because it lowers the sampling rate more compared with decimation ratio 64.

The response of pass means that expected vector (OUT) is equivalent to measured vector (OUT). The drive vector (IN) and expected vector (OUT) are already known and their spectrums could be depicted and shown in Figure 4.7.

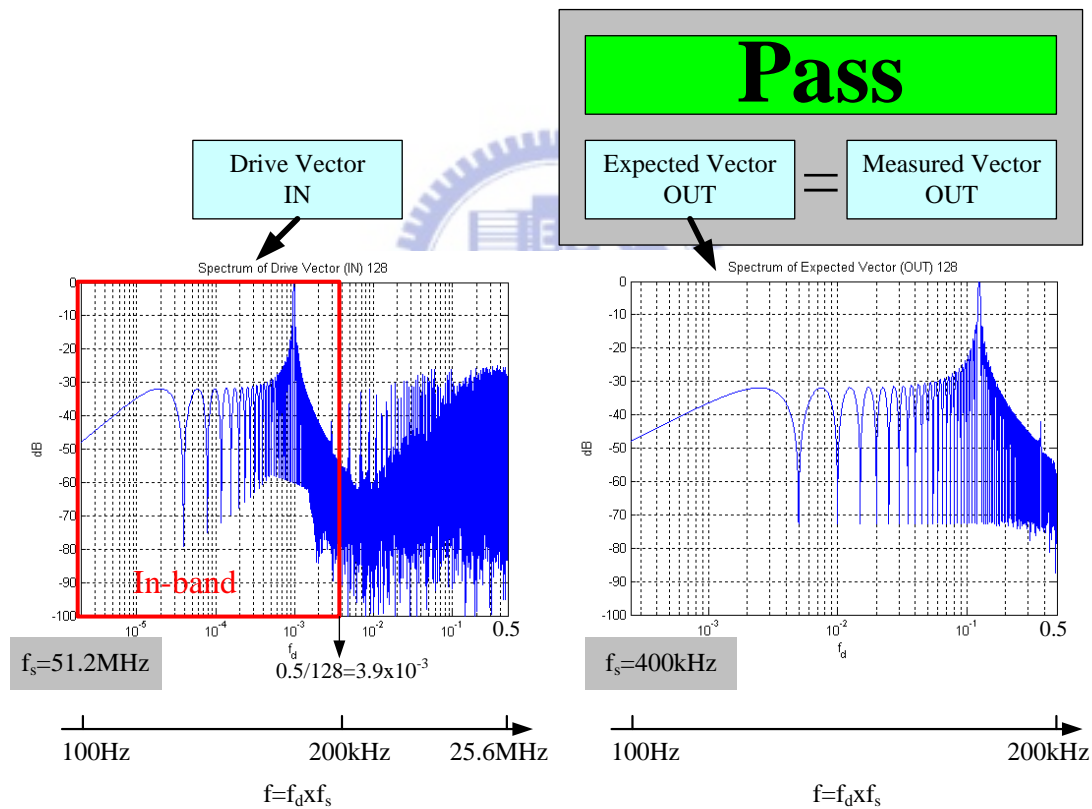


Figure 4.7 The spectrums for drive vector (IN) and expected vector (OUT), decimation factor 128

The spectrum of expected vector (output of decimator) is the in-band spectrum of drive vector (input of decimator). Thus, the behavior of the circuit (decimator) is correct. (The points of FFT for these two figures are the same; however, the spectrum resolutions for these two figures over the frequency range (f=f<sub>d</sub> × f<sub>s</sub>) [100 200k] in Hz



are different, which result in that the notches over the frequency range ( $f=f_d \times f_s$ ) [10k 100k] in Hz are a little different.)

The spectrums of decimator input and output over the frequency range [100 25.6M] are shown in Figure 4.8.

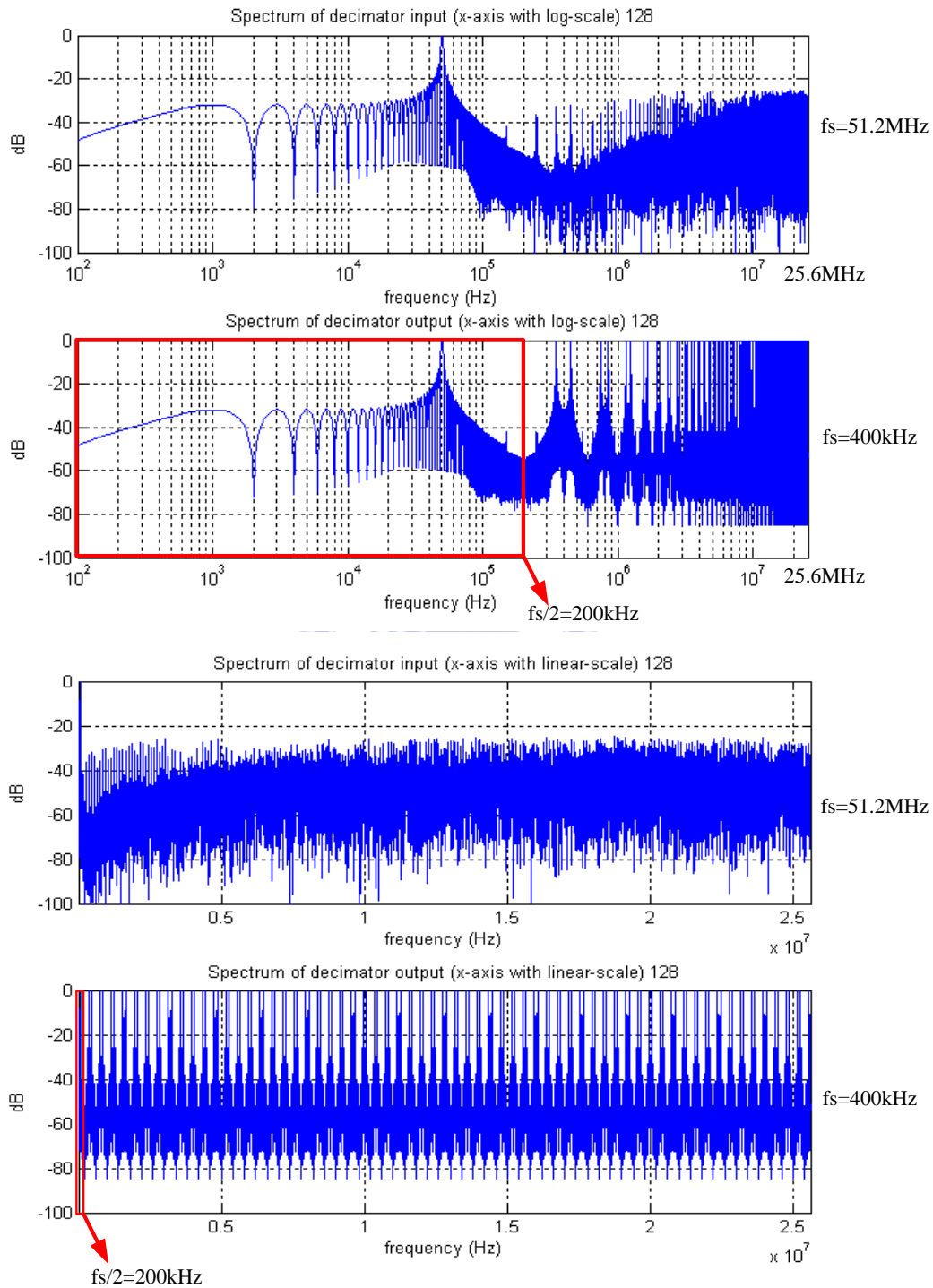


Figure 4.8 Spectrums of decimator input and output over the frequency range [100Hz 25.6MHz] (128)

According to the sampling theorem, the spectrum of the signal would repeat every sampling-frequency ( $f_s$ ). Thus, the spectrum of decimator output, where the  $f_s$  is 400 kHz ( $BW=200$  kHz), is repeated 128-times during that frequency range [0 25.6M] (Hz).

#### 4.2.2 OSR=64

The shmoo plot of the chip (decimator) using bits-stream of SDM with OSR 64 as stimulus is shown in Figure 4.9.

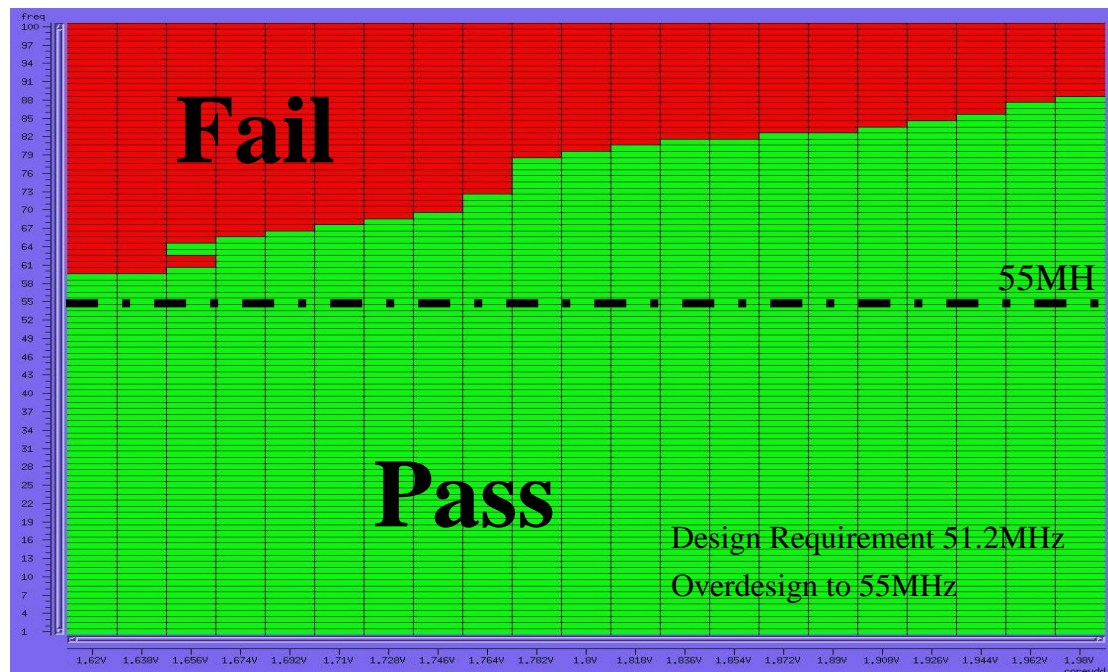


Figure 4.9 Shmoo plot (64)

Similarly, the above Shmoo plot reveals that the requirement of operating frequency 51.2 MHz is met. Besides, the circuit could operate correctly at higher operating frequency when the core supply voltage is increased. In my implementation process, the circuit (decimator) is designed and guaranteed to operate correctly below 55MHz clock rate. It is consistent with the above Shmoo plot.

Likewise, the response of pass means that expected vector (OUT) is equivalent to measured vector (OUT). The drive vector (IN) and expected vector (OUT) are already known and their spectrums could be depicted and shown in Figure 4.10.

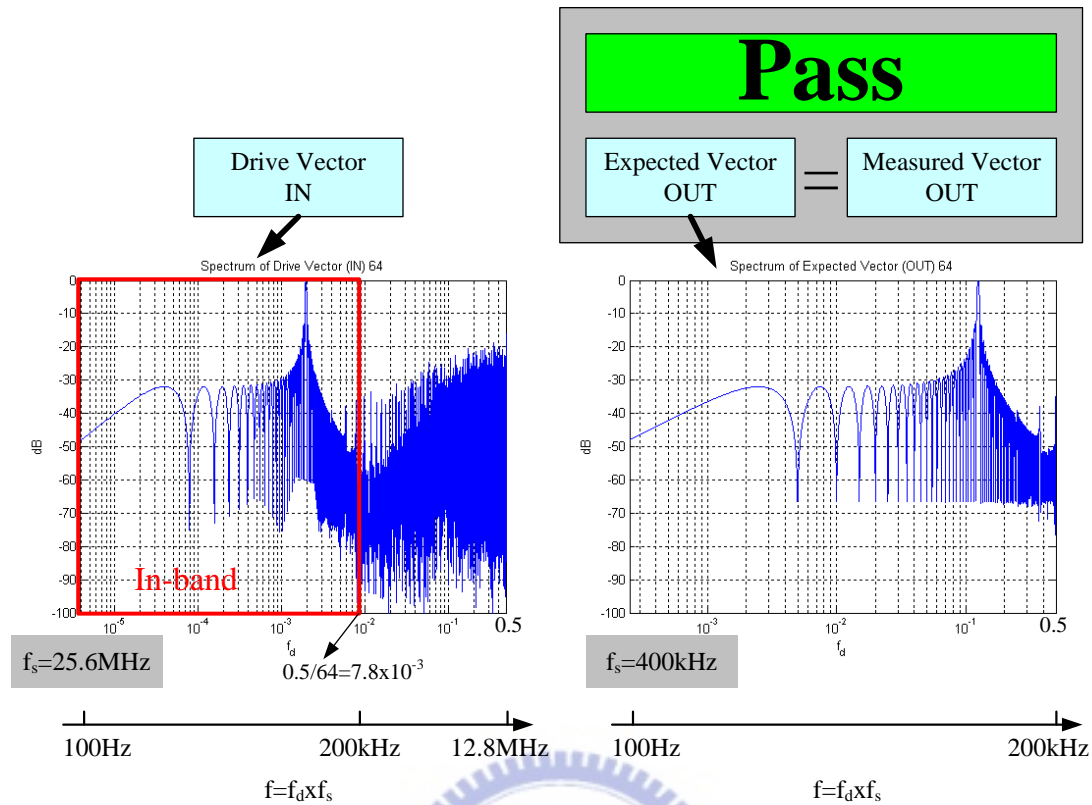


Figure 4.10 The spectrums for drive vector (IN) and expected vector (OUT), decimation factor 64

The spectrum of expected vector (output of decimator) is the in-band spectrum of drive vector (input of decimator). Thus, the behavior of the circuit (decimator) for decimation factor 64 is correct.

The spectrums of decimator input and output over the frequency range [100 12.8M] are shown in Figure 4.11.

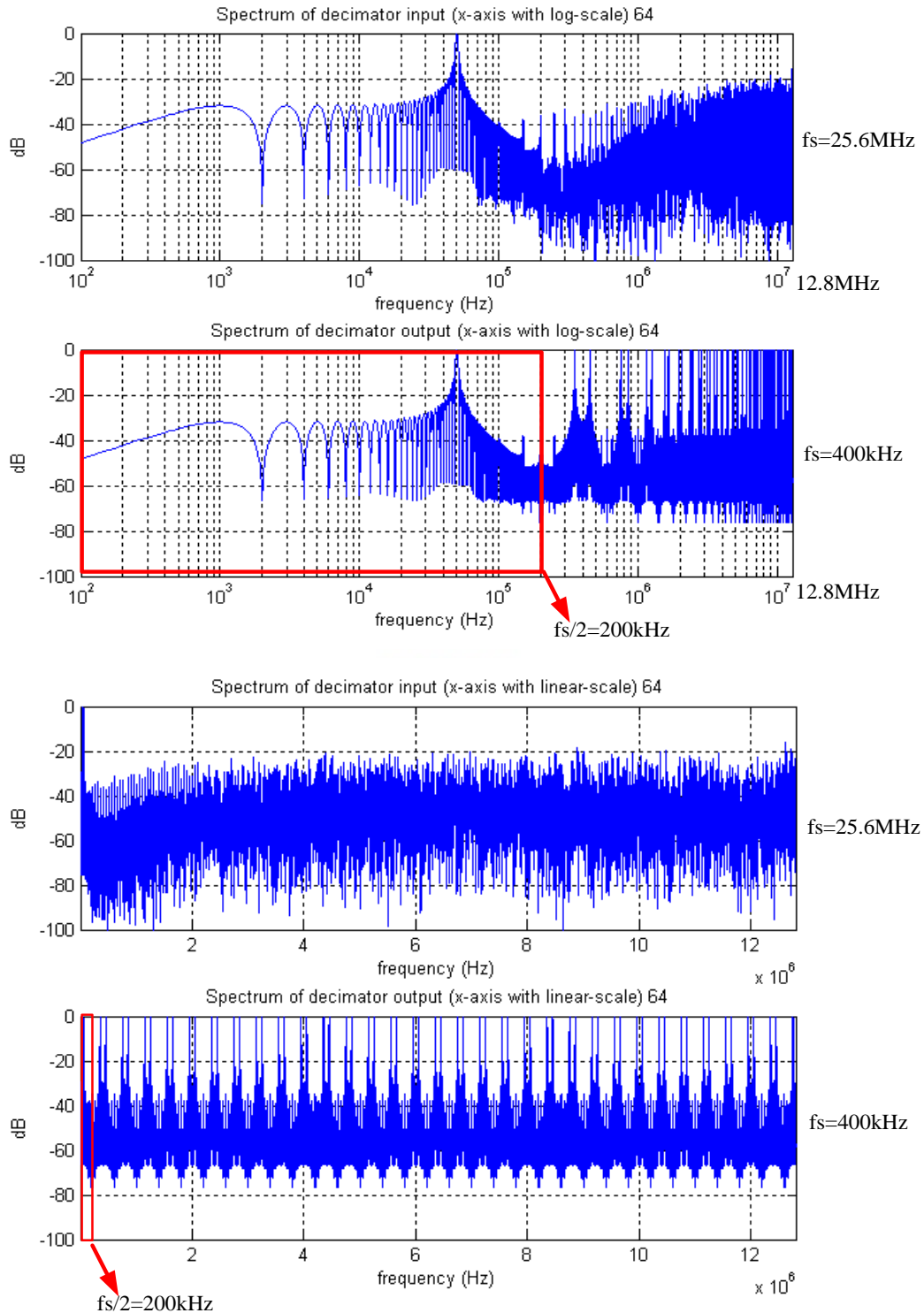


Figure 4.11 Spectrums of decimator input and output over the frequency range [100Hz 12.8MHz] (64)

Likewise, the spectrum of the signal would repeat every sampling-frequency ( $fs$ ). Thus, the spectrum of decimator output, where the  $fs$  is 400 kHz ( $BW=200$  kHz), is repeated 64-times during that frequency range [0 12.8M] (Hz).

### 4.3 Timing diagram

The timing diagram measured by Agilent 93000 for decimation ratio 128 and 64 are shown in Figure 4.12 and Figure 4.13, respectively. Only parts of signals and certain periods are shown in timing-diagram because the signals are shown as time-domain waveforms (i.e., output signals not group into bus to represent as logic values) and the maximum cycles which could be shown in the timing diagram window is 400.

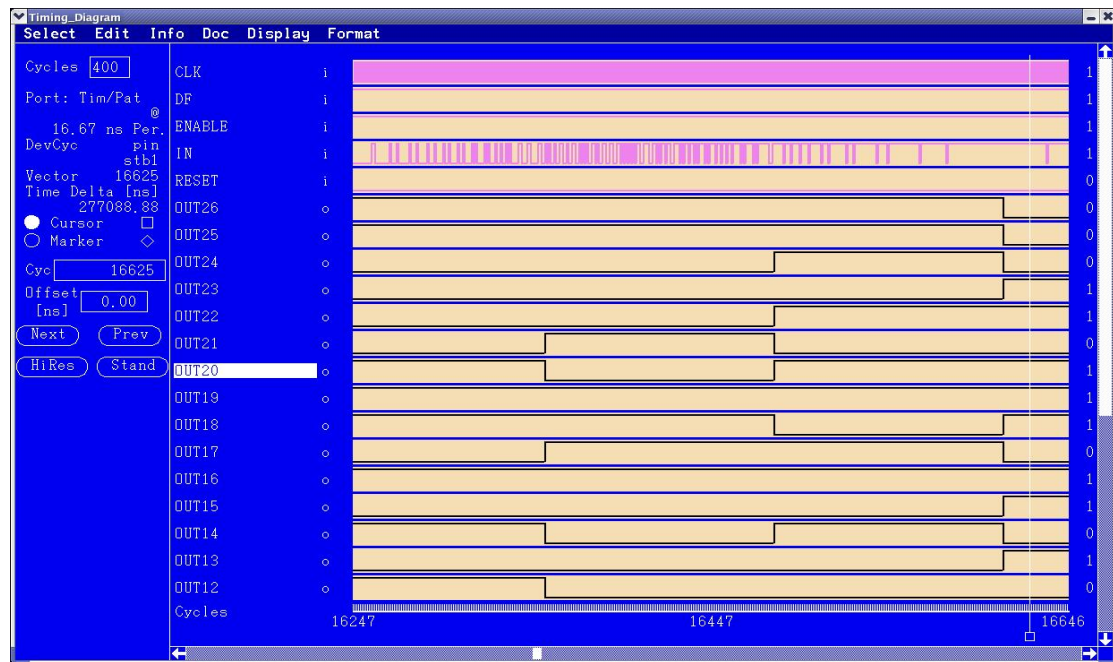


Figure 4.12 Timing diagram (decimation ratio 128) plotted by Agilent 93000

From the above figure, the outputs remain constant for 128 clock cycles due to decimation ratio 128. The measured outputs are compared with expected outputs each cycle; however, the outputs are fetched one time every 128 clock cycles (i.e., one sample output every 128 clock cycles due to decimation ratio 128).



Figure 4.13 Timing diagram (decimation ratio 64) plotted by Agilent 93000

Likewise, the outputs remain constant for 64 clock cycles due to decimation ratio 64. The measured outputs are compared with expected outputs each cycle; however, the outputs are fetched one time every 64 clock cycles (i.e., one sample output every 64 clock cycles due to decimation ratio 64).

#### 4.4 Measured Power

The measured operating current and corresponding power consumption are summarized in Table 4.2 below.

Table 4.2 Measured power consumption

	Decimation ratio=128	Decimation ratio=64
Operating frequency=50MHz		
Minimum operating current Core VDD (1.8V)	8.71mA (15.678mW)	16.6mA (29.88mW)
Maximum operating current Core VDD (1.8V)	9.34mA (16.812mW)	17.2mA (30.96mW)
Average operating current Core VDD (1.8V)	9.28mA ( <b>16.7mW</b> )	17.0mA ( <b>30.6mW</b> )
Operating current IO VDD (3.3V)	924uA (3.05mW)	1.63mA (5.4mW)

The average operating currents are 9.28mA and 17.0mA at operating frequency 50MHz and core-VDD 1.8V for decimation ratio 128 and 64, respectively. Thus, the average power is 16.7mW and 30.6mW at operating frequency 50MHz and core-VDD 1.8V for decimation ratio 128 and 64, respectively. The measured power approximates to the simulation power. (The measured power is slightly smaller than the simulation power because the measured operating current is instant operating current and the simulation current is the average instant operating current which includes normal operating current and peak operating current.)

From the above table, the operating currents of IO power supply are 0.924mA and 1.63mA at operating frequency 50MHz and IO-VDD 3.3V for decimation ratio 128 and 64, respectively. Thus, the IO power is 3.05mW and 5.4mW for decimation ratio 128 and 64, respectively.

#### 4.5 CHIP Summary

The information of chip is summarized in Table 4.3 below.

Table 4.3 Chip summary

<b>Decimator</b>	
<b>Process</b>	TSMC 0.18um
<b>Package</b>	LCC68 (only used 56 pads)
<b>Input pad</b>	5
<b>Output Pad</b>	27
<b>Core Power</b>	4 pairs
<b>IO Power</b>	8 pairs
<b>Die area</b>	1810x1810 $\mu\text{m}^2$
<b>Core area</b>	805x805 $\mu\text{m}^2$
<b>CoreVDD</b>	1.8V
<b>IOVDD</b>	3.3V
<b>Decimation ratio</b>	128 and 64
<b>Designed operating frequency</b>	51.2MHz
<b>Signal band-width</b>	200kHz (400kHz only for df=64)
<b>Power consumption @50MHz (df=128)</b>	16.7mW
<b>Power consumption @50MHz (df=64)</b>	30.6mW

The decimator is fabricated in TSMC 0.18 $\mu$ m process. The package type of the chip (decimator) is LCC68 and only 56 pads (pins) with 4-pair core power and 8-pair IO power are used. The silicon area of the entire die is 1810x1810  $\mu\text{m}^2$  and only 805x805  $\mu\text{m}^2$  is the active area (core area). The core power and IO power of the process are 1.8V and 3.3V, respectively. The decimator is designed to operate at 51.2 MHz input sampling-rate with decimation ratios 128 and 64 so as to preserve the signal bandwidth 200 kHz. The measured power consumption is 16.7mW and 30.6mW at input sampling-rate 50MHz for decimation ratio 128 and 64, respectively.





# CHAPTER

# 5

---

# Conclusions

---

The decimator for programmable oversampling ratio sigma-delta A/D converters is designed and implemented. The critical component of decimator is the high order FIR filter, which govern the most silicon area and power. The proposed folded architecture of decimation FIR filter based on transposed-form using polyphase decomposition is suitable for high order FIR filter to reduce silicon area because the high order folded FIR filter requires more register (the number of adders and multipliers are still invariant for higher order folded FIR filters).

The main advantage of the proposed architecture is smaller area (half registers; -24% for 126<sup>th</sup>-order FIR filter) compared with folded architecture in direct-form (Figure 3.9) [12][13][14][15]. In addition, my architecture reveals the smaller peak power, short critical path and latency. Furthermore, the area reduction percentage of my folded architecture will increase compared with folded architecture in direct-form when the scan chain is inserted to test the circuit due to the half register requirement.

Besides, the advantage of my proposed folded architecture is much less than half power of folded architecture in transposed-form (Figure 3.14) [18]. My design overheads compared with circuit mentioned in [18] are one adder, multiplexers and control circuits. However the silicon area required by transposed-form folding [18] is

not definitely smaller than silicon area required by folded architecture in direct-form and my folded architecture because of the twice operating frequency requirement for transposed-form folding [18] to maintain the throughput (output sampling-rate) (as a result of trading silicon area for speed).

For power concern, the implementation of FIR filters must focus on the unfolded structures, which is a correct direction to lower the power consumption of FIR filters; however, for cost (area) concern, it must focus on folded architecture, which could reduce the functional units most especially for high order FIR filter. In the folded architectures of decimation FIR filters, my folded architecture is a good choice to implement the decimation FIR filters and to obtain area reduction (compared with direct-folding [12][13][14][15]) without much power overhead (transposed-folding) [18].



## Appendix A: Filter Coefficients

For the 2<sup>nd</sup> and 3<sup>rd</sup> stage:

1-bit	19-bit
-------	--------

Table A.1 Coefficients of the 2<sup>nd</sup>-stage FIR filter

coefficients	decimal value	binary value (2's complement)
h0, h18	0.00035667419434	00000000000010111011
h1, h17	0.00147438049316	0000000001100000101
h2, h16	-0.00535202026367	1111111010100001010
h3, h15	-0.00497627258301	1111111010111001111
h4, h14	0.02363014221191	000001100001100101
h5, h13	0.01024436950684	0000001010011111011
h6, h12	-0.07559013366699	11110110010100110001
h7, h11	-0.01518249511719	11111110000011101000
h8, h10	0.30679893493652	00100111010001010011
h9	0.51721763610840	01000010001101000011

Table A.2 Coefficients of the 3<sup>rd</sup>-stage FIR filter

coefficients	decimal value	binary value
h0, h126	0.00014495849609	0000000000001001100
h1, h125	-0.00031471252441	11111111111101011011
h2, h124	0.00013351440430	0000000000001000110
h3, h123	0.00019645690918	0000000000001100111
h4, h122	-0.00015640258789	11111111111110101110
h5, h121	-0.00025367736816	11111111111101111011
h6, h120	0.00020980834961	0000000000001101110
h7, h119	0.00031471252441	0000000000010100101
h8, h118	-0.00030517578125	11111111111101100000
h9, h117	-0.00039291381836	11111111111100110010
h10, h116	0.00041961669922	0000000000011011100
h11, h115	0.00047683715820	0000000000011111010
h12, h114	-0.00056838989258	11111111111011010110
h13, h113	-0.00057411193848	11111111111011010011
h14, h112	0.00074768066406	0000000000110001000
h15, h111	0.00067710876465	0000000000101100011
h16, h110	-0.00096511840820	11111111111000000110

h17, h109	-0.00079154968262	1111111111001100001
h18, h108	0.00122451782227	0000000001010000010
h19, h107	0.00091361999512	0000000000111011111
h20, h106	-0.00153350830078	11111111110011011100
h21, h105	-0.00104522705078	11111111110111011100
h22, h104	0.00189590454102	0000000001111100010
h23, h103	0.00118446350098	0000000001001101101
h24, h102	-0.00231933593750	11111111101101000000
h25, h101	-0.00132942199707	11111111110101000111
h26, h100	0.00281143188477	0000000010111000010
h27, h99	0.00148200988770	0000000001100001001
h28, h98	-0.00337982177734	11111111100100010100
h29, h97	-0.00163650512695	11111111110010100110
h30, h96	0.00403785705566	0000000100001000101
h31, h95	0.00179672241211	0000000001110101110
h32, h94	-0.00479698181152	11111111011000101101
h33, h93	-0.00195884704590	1111111110111111101
h34, h92	0.00567054748535	0000000101110011101
h35, h91	0.00211906433105	0000000010001010111
h36, h90	-0.00667953491211	11111111001001010010
h37, h89	-0.00227928161621	11111111101101010101
h38, h88	0.00785064697266	00000001000000010100
h39, h87	0.00243759155273	0000000010011111110
h40, h86	-0.00921440124512	11111110110100100001
h41, h85	-0.00259017944336	11111111101010110010
h42, h84	0.01082038879395	00000001011000101001
h43, h83	0.00273704528809	0000000010110011011
h44, h82	-0.01273155212402	11111110010111101101
h45, h81	-0.00287628173828	11111111101000011100
h46, h80	0.01504516601563	00000001111011010000
h47, h79	0.00300407409668	0000000011000100111
h48, h78	-0.01791000366211	11111101101101010010
h49, h77	-0.00312232971191	11111111100110011011
h50, h76	0.02156448364258	00000010110000101010
h51, h75	0.00322723388672	0000000011010011100
h52, h74	-0.02642822265625	11111100100111100000
h53, h73	-0.00331878662109	11111111100100110100

h54, h72	0.03329086303711	00000100010000101110
h55, h71	0.00339508056641	00000000011011110100
h56, h70	-0.04384040832520	11111010011000110111
h57, h69	-0.00345611572266	11111111100011101100
h58, h68	0.06248664855957	00000111111111111001
h59, h67	0.00349998474121	00000000011100101011
h60, h66	-0.10539436340332	11110010100000100111
h61, h65	-0.00352478027344	11111111100011001000
h62, h64	0.31807327270508	00101000101101101010
h63	0.50353431701660	01000000011100111101

For the 4<sup>th</sup> stage:

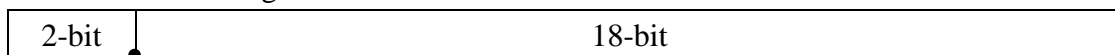


Table A.3 Coefficients of the 4<sup>th</sup>-stage compensation FIR filter

coefficients	decimal value	binary value
h0, h40	0.00000762939453	00000000000000000010
h1, h39	-0.00000762939453	11111111111111111110
h2, h38	0.00000762939453	00000000000000000010
h3, h37	-0.00001144409180	11111111111111111101
h4, h36	0.00001907348633	00000000000000000101
h5, h35	-0.00003051757813	11111111111111111000
h6, h34	0.00004196166992	00000000000000001011
h7, h33	-0.00005722045898	11111111111111110001
h8, h32	0.00008392333984	00000000000000010110
h9, h31	-0.00011444091797	1111111111111100010
h10, h30	0.00016021728516	0000000000000101010
h11, h29	-0.00022506713867	1111111111111000101
h12, h28	0.00031661987305	0000000000001010011
h13, h27	-0.00045394897461	1111111111110001001
h14, h26	0.00066757202148	0000000000010101111
h15, h25	-0.00102996826172	1111111111011110010
h16, h24	0.00168609619141	0000000000110111010
h17, h23	-0.00311279296875	1111111110011010000
h18, h22	0.00709533691406	0000000011101000100
h19, h21	-0.02724456787109	1111110010000011010
h20	1.04457473754883	01000010110110100101

## Appendix B: Test-Patterns for Agilent 93000

The driving signals (input of chip) generated by Agilent 93000 are defined by the vectors which consist of the state characters. Each state character of a signal corresponds to a specific waveform. For example, state character 1 of pin CLK represents the waveform: forces logic 0 at the beginning of the cycle, forces logic 1 at the 1/4 cycle delay from the beginning of the cycle and forces logic 0 at the 3/4 cycle delay from the beginning of the cycle. Corresponding waveforms of state characters of all signals in my design are shown in Figure B.1. For output pins, the time to fetch output could also be defined by Agilent 93000. In my case, the outputs are fetched and compared at the edge of 3/4 cycle.

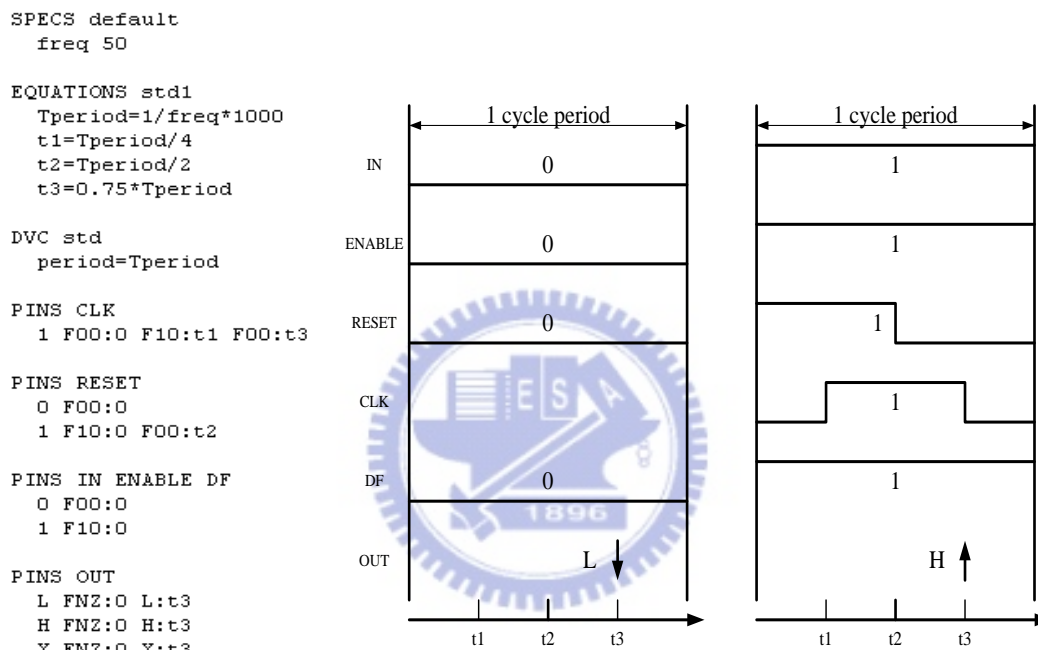


Figure B.1 Corresponding waveforms of state characters of signals in my design

The meanings of the above figure are described below:

PINS Pin\_Name

State character      Action: Timing (delay from beginning)      .....

Drive Action in Pins

F00      force logic 0

F10      force logic 1

Compare Action in Pins

L      compare to low (logic 0)

H      compare to high (logic 1)

X      don't care

Parts of test vectors for SDM with OSR 128 and corresponding post-layout-simulation are shown in Figure B.2 and Figure B.3.

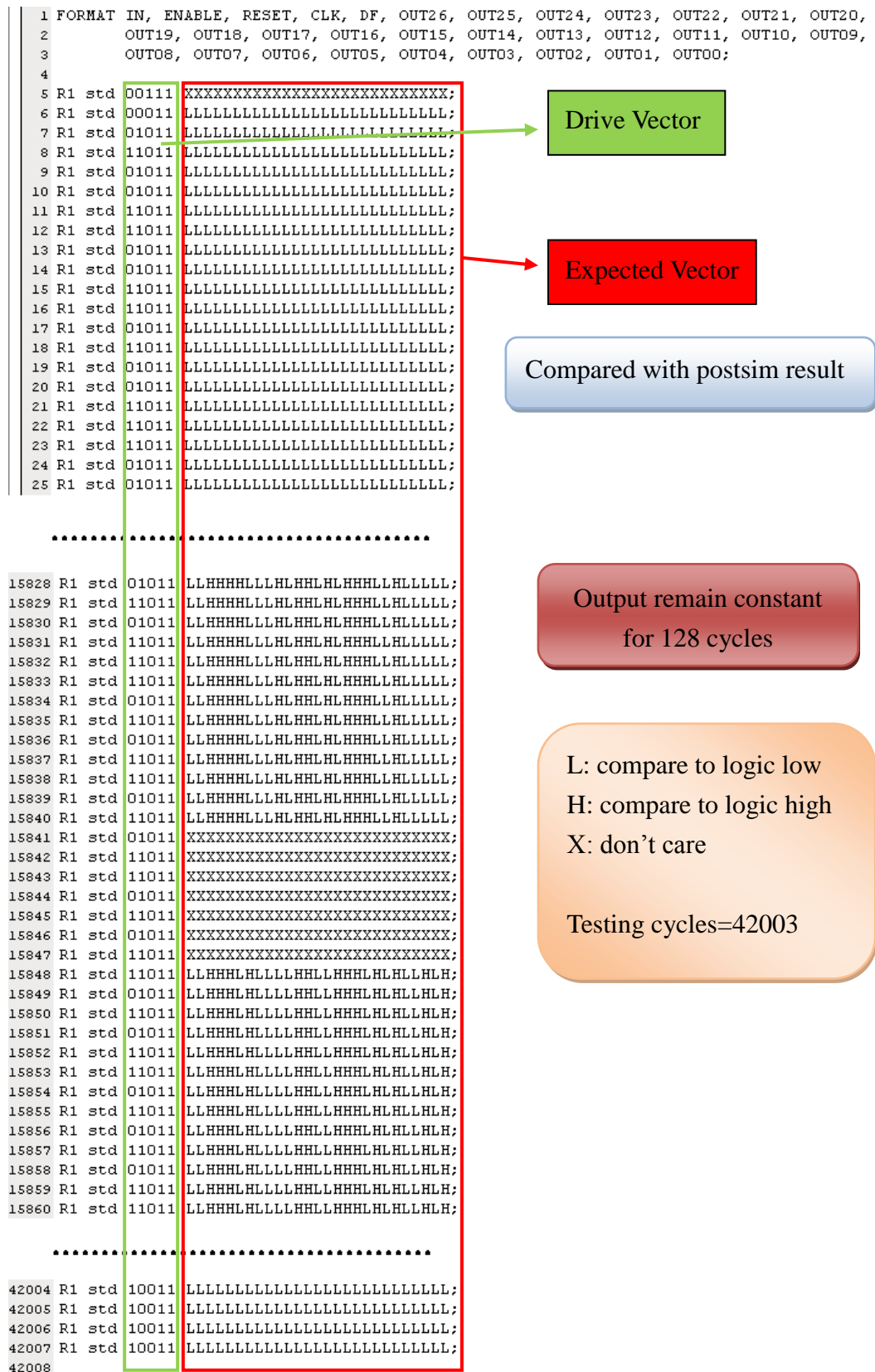
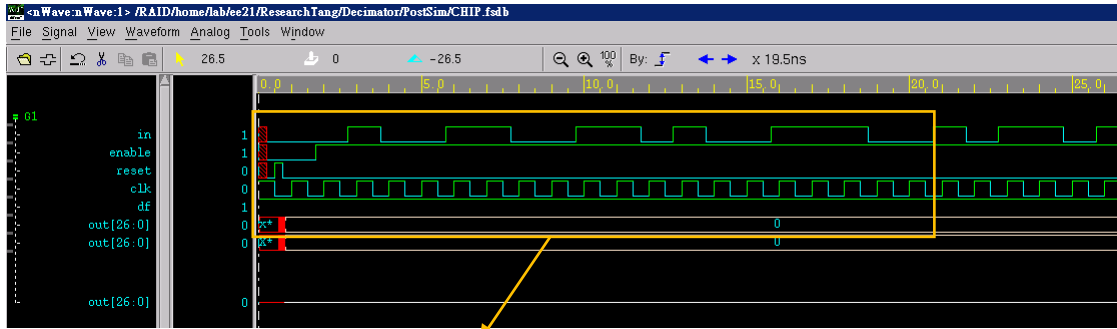


Figure B.2 Parts of test vectors for SDM with OSR 128



in: 0001001100110100111100...

enable: 0011111111111111111111...

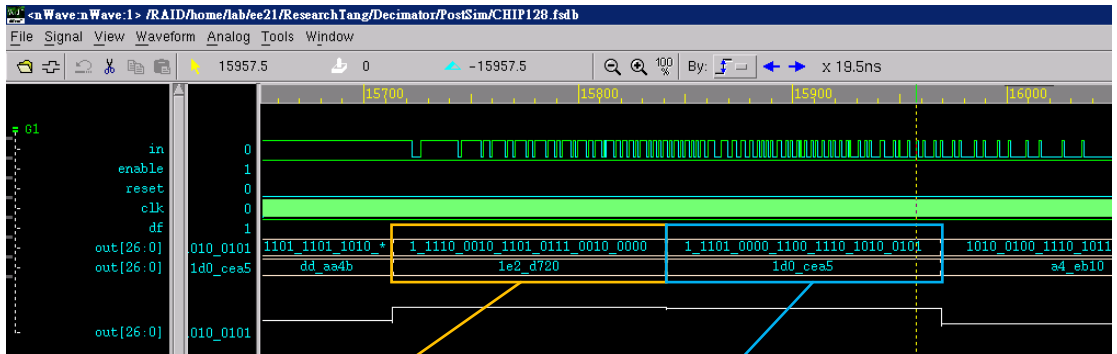
reset: 1000000000000000000000...

clk: 1111111111111111111111...

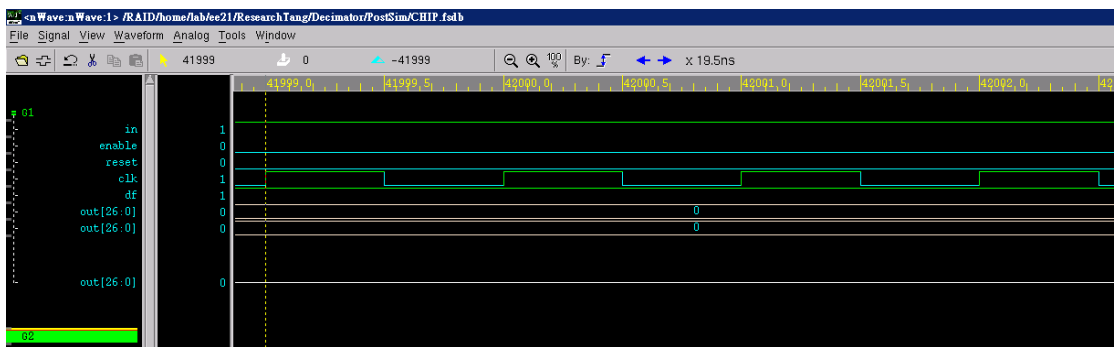
df: 1111111111111111111111...

out[26:0]: 0000000000000000000000...

(i.e., out[26], out[25]..... are all LLLLLLLLLLLLLLLLLLLLLLLLLLLLL during the period)



LLH\_HHHL\_LLHL\_HHLH\_LHHH\_LLHL\_LLLL  
 LLH\_HHLH\_LLLL\_HHLL\_HHHL\_HLHL\_LHLH



in, enable, reset, clk and df are 1, 0, 0, 1, 1 during the period.

out[26:0] are all LLLLLLLLLLLLLLLLLLLLLLLLLLLLL during the period shown in above figure.

Figure B.3 Corresponding post-layout-simulation of parts' test vectors for SDM with OSR 128



From Figure B.2 and Figure B.3, the parts of cycles reveal that the test patterns for Agilent 93000 correspond to the post-layout-simulation. In this case, the expected vectors (OUT) are LLH\_HHHL\_LLHL\_HHLH\_LHHH\_LLHL\_LLLL at the 15835<sup>th</sup> cycle and LLH\_HHLH\_LLLL\_HHLL\_HHHL\_HLHL\_LHLH at the 15850<sup>th</sup> cycle, which correspond to (00)1\_1110\_0010\_1101\_0111\_0010\_0000 and (00)1\_1101\_0000\_1100\_1110\_1010\_0101 of simulation output (out[26:0]). The corresponding timing diagram of the chip under test is shown in Figure B.4. Also the corresponding post-layout-simulation diagram is shown in Figure B.5.

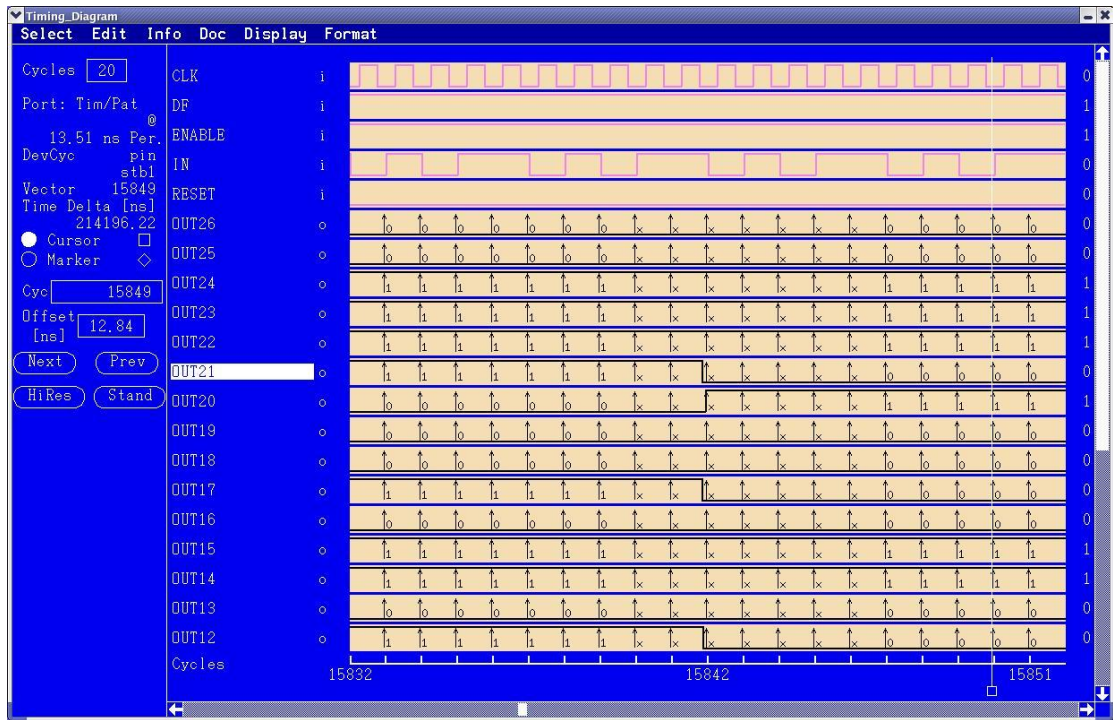


Figure B.4 The corresponding timing diagram measured by Agilent 93000 (128)

Note that the measured outputs are equal to the simulation outputs.

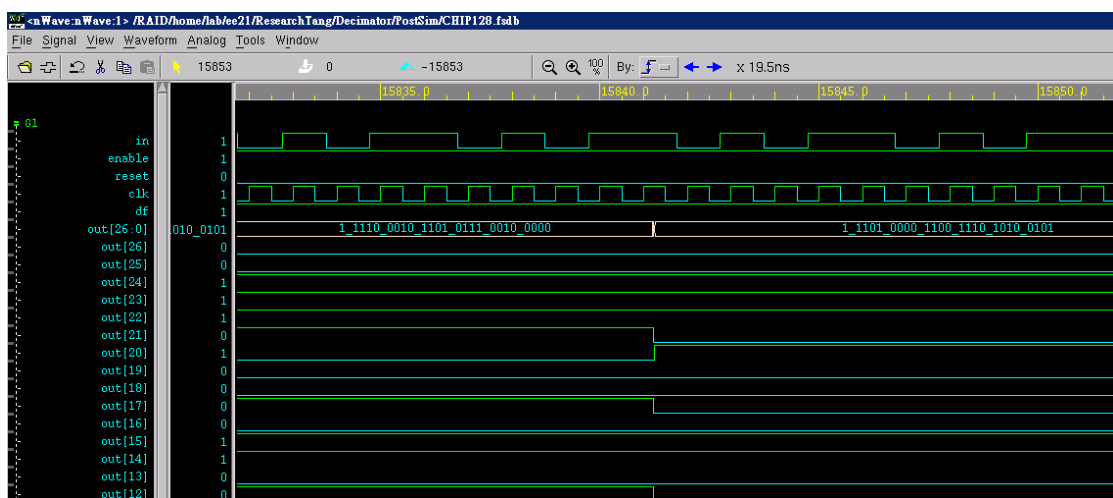
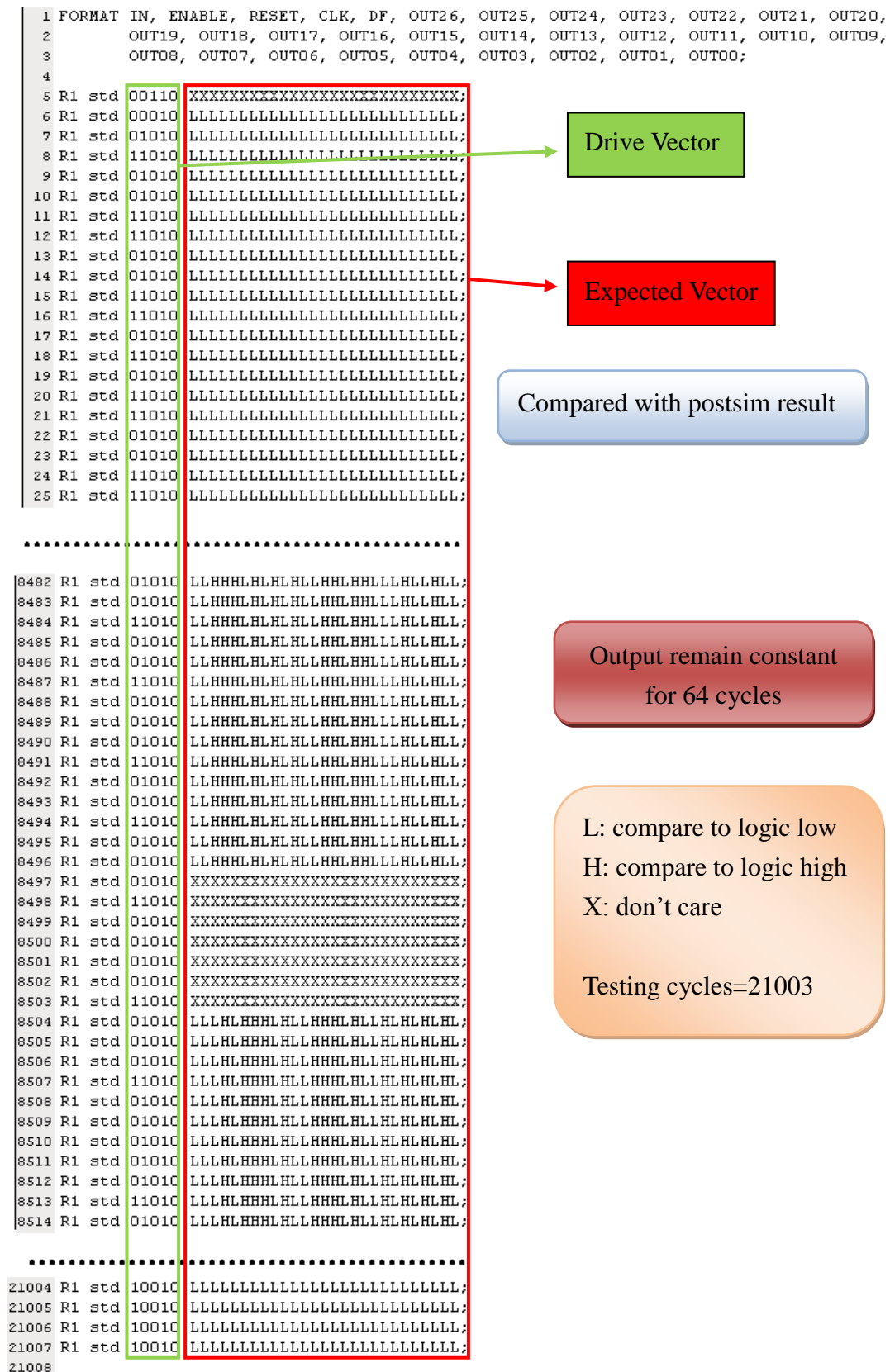


Figure B.5 The corresponding post-layout-simulation (128)

Parts of test vectors for SDM with OSR 64 and corresponding post-layout-simulation are shown in Figure B.6 and Figure B.7.



Drive Vector

Expected Vector

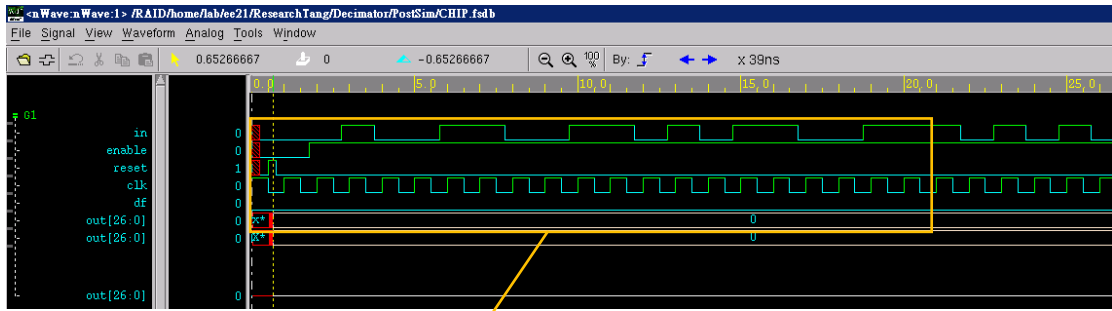
Compared with postsim result

Output remain constant for 64 cycles

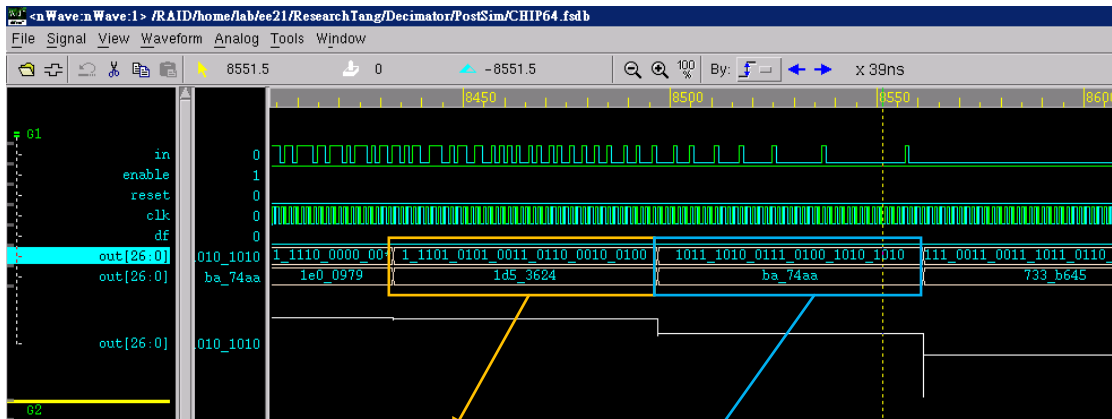
L: compare to logic low  
H: compare to logic high  
X: don't care

Testing cycles=21003

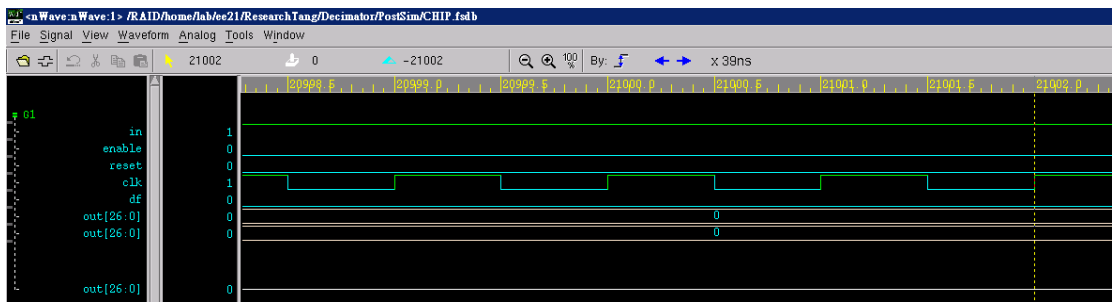
Figure B.6 Parts of test vectors for SDM with OSR 64



in: 000100110011010110011...  
 enable: 001111111111111111111...  
 reset: 100000000000000000000...  
 clk: 111111111111111111111...  
 df: 000000000000000000000...  
 out[26:0]: 000000000000000000000...  
 (i.e., out[26], out[25]..... are all LLLLLLLLLLLLLLLLLLLLLLLLLL during the period)



LLH\_HHLH\_LHLH\_LLHH\_LHHL\_LLHL\_LHLL  
 LLL\_HLHH\_HLHL\_LHHH\_LHLL\_HLHL\_HLHL



in, enable, reset, clk and df are 1, 0, 0, 1, 0 during the period.  
 out[26:0] are all LLLLLLLLLLLLLLLLLLLLLLLLLL during the period shown in above figure.  
 Figure B.7 Corresponding post-layout-simulation of parts' test vectors for SDM with OSR 64

From Figure B.6 and Figure B.7, the parts of cycles reveal that the test patterns for Agilent 93000 correspond to the post-layout-simulation. In this case, the expected vectors (OUT) are LLH\_HHLH\_LHLH\_LLHH\_LHHL\_LLHL\_LHLL at the 8490<sup>th</sup> cycle and LLL\_HLHH\_HLHL\_LHHH\_LHLL\_HLHL\_HLHL at the 8505<sup>th</sup> cycle, which correspond to (00)1\_1101\_0101\_0011\_0110\_0010\_0100 and (000)\_1011\_1010\_0111\_0100\_1010\_1010 of simulation output (out[26:0]). The corresponding timing diagram of the chip under test is shown in Figure B.8. Also the corresponding post-layout-simulation diagram is shown in Figure B.9.

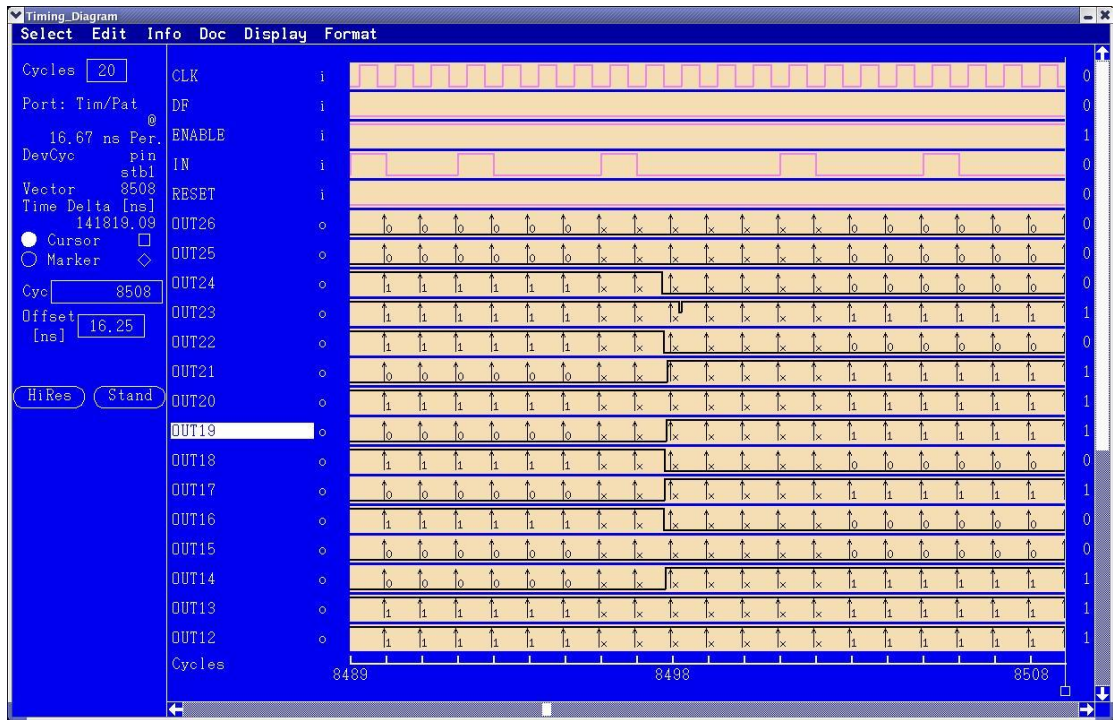


Figure B.8 The corresponding timing diagram measured by Agilent 93000 (64)

Note that the measured outputs are equal to the simulation outputs.

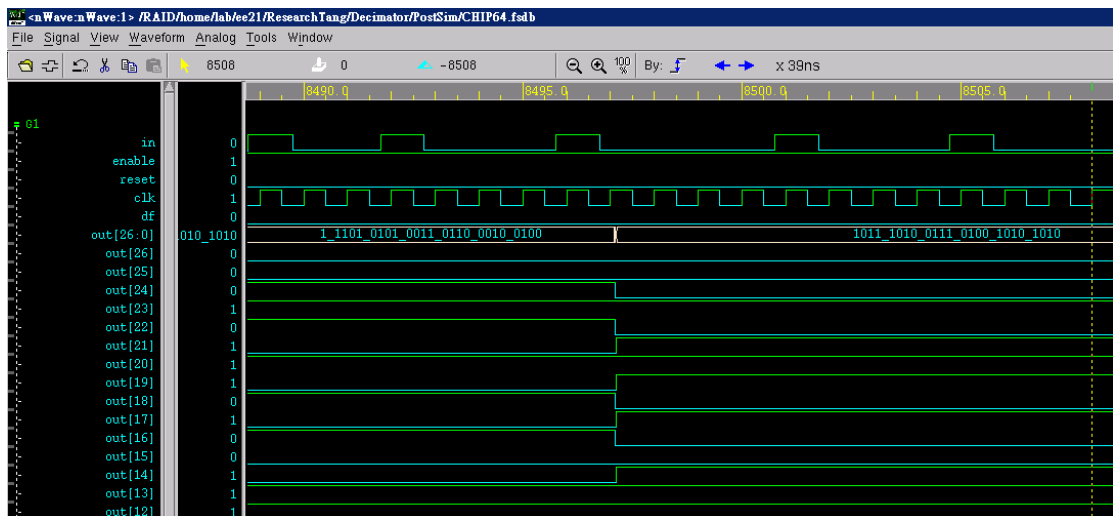


Figure B.9 The corresponding post-layout-simulation (64)

## References

- [1] Pervez M. Aziz, Henrik V. Sorensen and Jan Van der Spiegel, “An overview of sigma-delta converters,” *IEEE Signal Processing Magazine*, vol. 13, Issue 1, pp.61-84, Jan 1996.
- [2] Angelo Nagari, Alessandro Mecchia, Ermes Viani, Sergio Pernici, Pierangelo Confalonieri, and Germano Nicollini, ”A 2.7-V 11.8-mW Baseband ADC With 72-dB Dynamic Range for GSM Applications,” *IEEE Journal of Solid-State Circuits*, vol. 35, no. 6 June 2000.
- [3] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck, “Discrete-Time Signal Processing,” 2<sup>nd</sup> Edition, Prentice Hall, Upper Saddle River, NJ, 1999.
- [4] Salas, Hille and Etgen, “Calculus: one and several variables,” 8<sup>th</sup> Edition, Wiley, NY, 1999.
- [5] R. E. Crochiere and L. R. Rabiner, “Optimum FIR digital filter implementations for decimation, interpolation, and narrowband filtering,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-23, pp. 444–456, 1975.
- [6] Eugene B. Hogenauer, “An economical class of digital filters for decimation and interpolation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. assp-29, no.2, pp. 155-162, April 1981.
- [7] J. C. Candy, “Decimation for sigma delta modulation,” *IEEE Transactions on Communications*, vol, com-34, pp. 72–76, Jan. 1986.
- [8] Letizia Lo Presti, “Efficient modified-sinc filters for sigma-delta A/D converters,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 11, pp. 1204–1213, Nov. 2000.
- [9] Massimiliano Laddomada, “Comb-Based Decimation Filters for  $\Sigma\Delta$  A/D Converters: Novel Schemes and Comparisons,” *IEEE Transactions on Signal Processing*, vol. 55, Issue 5, Part 1, pp.1769 – 1779, May 2007.
- [10] Massimiliano Laddomada, “Generalized Comb Decimation Filters for  $\Sigma\Delta$  A/D Converters: Analysis and Design,” *IEEE Transactions on Circuits and Systems—I: Regular Papers*, vol. 54, no. 5, pp.994-1005, May 2007.
- [11] Yonghong Gao, Lihong Jia, Jouni Isoaho and Hannu Tenhunen, “A Comparison Design of Comb Decimators for Sigma-Delta Analog-to-Digital Converters,” *Analog Integrated Circuits and Signal Processing*, 22, pp.51-60, 1999.
- [12] L. Fucik, A. S. Kuncheva, T. Mougel, and R. Vrba, “New VHDL Design of Decimation Filter for Sigma-Delta Modulator,” *IEEE Asian Conference on Sensors and the International Conference on new Techniques in Pharmaceutical and Biomedical Research*, pp. 204-207, Sept. 2005.
- [13] Adel Ghazel, Lirida Naviner and Khaled Grati, “On Design and Implementation

- of a Decimation Filter for Multistandard Wireless Transceivers,” IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 558-562, October 2002.
- [14] Lirida A. de B. Naviner and Jean-Francois Naviner, “On Customized Decimation Filter Implementation,” IEEE International Conference on Industrial Technology, vol. 1, pp. 304-309, Dec. 8-10, 2004.
- [15] Xin He, Yihe Sun, “VLSI Implementation of a Decimation Filter for Sigma-Delta AD Converters,” IEEE Region 10 Conference TENCN, pp.1-4, Nov. 2006.
- [16] K. Lin et al., “Digital filters for high performance audio delta–sigma analog-to-digital and digital-to-analog conversions,” in Proc. Int. Conf. Signal Processing, Oct. 1996.
- [17] Prabir C. Maulik, Mandeep S. Chadha, Member, Wai L. Lee, and Philip J. Crawley, “A 16-Bit 250-kHz Delta–Sigma Modulator and Decimation Filter,” IEEE Journal of Solid-State Circuits, vol. 35, no. 4, pp. 458-467, April 2000.
- [18] Paul Bougas, Paraskevas Kalivas, Andreas Tsirikos, and Kiamal Z. Pekmestzi, "Pipelined Array-Based FIR Filter Folding," IEEE Transactions On Circuits and Systems—i: Regular Papers, vol. 52, no. 1, pp.108-118, January 2005.
- [19] <http://www.cic.org.tw>
- [20] Agilent Technoliges, “Agilent 93000 Soc Series User Traing Part I,” October 2004.

