

# 國立交通大學

電機資訊國際學位學程碩士班

碩士論文

整合於可攜式腦心監護系統之  
DOT/ECG/EEG 多處理器設計

Integrated DOT/ECG/EEG Multiprocessor Design for Portable Brain-Heart Monitoring Systems

研究生：李鴻溝

指導教授：方偉騏 教授

中華民國一〇〇年九月

整合於可攜式腦心監護系統之  
DOT/ECG/EEG 多處理器設計

Integrated DOT/ECG/EEG Multiprocessor Design for Portable Brain-Heart  
Monitoring Systems

研究生： 李鴻溝      Student： Ericson Go Chua  
指導教授： 方偉騏      Advisor： Dr. Wai-Chi Fang

國立交通大學

電機資訊國際學位學程碩士班



Submitted to the EECS International Graduate Program

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Master

September 2011

Hsinchu, Taiwan, Republic of China

中華民國一〇〇年九月

# 整合於可攜式腦心監護系統之

## DOT/ECG/EEG 多處理器設計

研究生：李鴻溝

指導教授：方偉騏 教授

國立交通大學 電機資訊國際學位學程碩士班

### 中文摘要

近十年隨著老年人口快速膨脹，醫療費用的增加儼然成為全球關注的共同議題。人手的短缺使得醫療照顧體系無法應接數量激增的老年病人，昂貴的診金和治療費更讓許多人無法享用到優質的醫療服務。因此近年來，生物醫學工程已成為至關重要的研究和發展主題。

為了應對在特定緊急護理、長期的觀察、認知科學在醫療監控應用的需求，我們提出發展一個綜合的腦心臟監測系統芯片，並提供一個示範平台，以證明此領域之研究是可行的，也為了今後相關的工作和發展。這項工作的動機有三：第一是以便攜式的生物醫學工具使病人在治療期間更舒適，行動更方便；二是降低整體系統成本，當中包括在家庭和醫院裡相關的設備、操作過程、物流和管理；三，為腦心監測應用的新研究方向鋪路。

本論文旨在發展一種能夠整合多種生物醫學訊號的系統處理器，包括擴散光學腦部影像重建 (DOT) 處理器、可除去腦電訊號 (EEG) 雜訊的獨立成分分析 (ICA) 器、用來分析心電訊號 (ECG) 的心率變異性處理器 (HRV)。同時，為了降低功率消耗與延長工作時間，此種多功能處理器內含一種無失真資料的壓縮處理器，可用來降低從可攜式裝置 (portable device) 傳送生物醫學訊號到生醫資訊工作站 (science station) 時的帶寬要求。此多重處理器設計使用聯華電子 65 奈米 CMOS 製程下線晶片，此外，也實現於 AHB 相容 IP 之 Xilinx FPGA 做系統單晶片設計。

為了展示此多重處理器設計的功能性和即時處理的廣泛應用，此篇論文提出一個

完整的、端至端且實現於 SoC FPGA 開發平台的腦心監測系統。由前端訊號擷取模組所得到的生醫訊號被傳送至相應的即時運算引擎進行分析處理後，處理完的結果再與原始訊號由一個無損失性生醫信號壓縮模組進行資料壓縮，最後再經由一個商業藍牙模組傳至臨近的生醫資訊工作站進行 3D 顯像及遠端觀察與診斷。

關鍵字：獨立成分分析、心率變異率分析、擴散式光學影像重建、整合型生醫系統、藍牙傳輸、無失真資料壓縮、可攜式系統、數位訊號處理、系統晶片





# **Integrated DOT/ECG/EEG Multiprocessor Design for Portable Brain-Heart Monitoring Systems**

**Student: Ericson Go Chua    Advisor: Dr. Wai-Chi Fang**

**EECS International Graduate Program**

**National Chiao Tung University**

## **Abstract**

In the recent decade, the accelerated emergence of an aged population alongside increased medical costs has been recognized as a worldwide problem. Whereas a shortage in medical personnel will leave the healthcare system unable to meet the requirements of a growing number of elderly patients, even more will be deprived of access to quality healthcare due to the high costs of diagnosis and treatment. As a result, in recent years, the field of biomedical engineering has emerged as a top priority research and development topic.

In response to the needs of healthcare monitoring applications in particular emergency care, long-term observation and cognitive science, we propose the development of an integrated brain-heart monitoring system and provide a demonstration platform as a proof of concept for future works and development along this topic. The motivation of this work is threefold; first is to improve patient experience by means of a portable biomedical device; second, to reduce overall system costs associated with the equipment, operations, logistics and management in both hospital and home care settings; and third, to pave the way for new research directions relating to brain-heart monitoring applications.

In this thesis, we present the development of a biomedical signal multiprocessor comprising a novel diffuse optical tomography (DOT) processor for brain imaging, an independent component analysis (ICA) processor for removing electroencephalogram (EEG)

signal artifacts, and a heart rate variability (HRV) analysis processor for monitoring electrocardiogram (ECG) signals. Furthermore, in order to reduce power consumption and prolong operating time, a lossless data compressor is employed to reduce bandwidth requirements during wireless transmission of biomedical data. The multiprocessor design is implemented both as an AHB-compatible IP for ARM-based SOCs on a Xilinx FPGA and as an IC fabricated using UMC 65nm CMOS technology.

To demonstrate the functionality and real-time application of the developed multiprocessor design, a complete, end-to-end brain-heart monitoring system platform employing the SoC-based implementation is presented. EEG, ECG and/or functional near infrared (fNIR) signals acquired by an analog front-end IC are processed or bypassed by the biomedical multiprocessor depending on configuration commands sent wirelessly from a remote science station. Processed or raw biomedical data optionally compressed by a lossless data compressor are packaged according to a fixed output data format and finally sent back to the remote science station for real-time LCD display, data storage, or further off-line processing and analysis.

**Keywords:** Independent Component Analysis, Heart Rate Variability, Diffuse Optical Tomography, Integrated Healthcare System, Bluetooth Wireless Communication, Lossless Data Compression, Portable Healthcare, Digital Signal Processing, System-on-Chip

## Acknowledgements

First and foremost I would like to thank my thesis advisor Professor Wai-Chi Fang for providing an excellent research environment complete with state-of-the-art equipment, facilities and resources for me and fellow classmates to conduct our research work. Moreover, I'm deeply grateful for the NSC research projects he fought dearly for us to exercise our skills, prove our capabilities and develop something useful in a meaningful setting. For his tokens of concern for our technical advancement, from little gestures such as books from the U.S. to more formal occasions such as training arrangements with the industry. For his generous support allowing me to participate in numerous local and international seminar camps and conferences. And for the research collaboration experience at UCSD which proved to be immensely inspiring for me. May his students learn to realize, appreciate and value these doors of opportunity to achievement and walk through them as I have. The many lessons I have learned from him, directly or indirectly, will undoubtedly prove invaluable in the future chapters of my life.

Next, I would like to express my gratitude to Professor Sheng-Jyh Wang and Professor Kea-Tiong Tang for their interest in my work and especially taking their precious time to participate as committee members in my master's degree oral examinations.

Furthermore, I would like to thank my senior and fellow laboratory mates for their painstaking support, advice, guidance and assistance, be it about research work at the lab or about life in general. To my juniors, for the friendship, I wish you luck and hope that you may find fulfillment and accomplishment in your research and overcome whatever great challenges that may lie ahead, with patience, integrity and teamwork.

My sincerest gratitude to the Taiwan government for the Taiwan Scholarship awarded to me through their good offices, the Ministry of Education (MOE), R.O.C. and Taipei Economic and Cultural Office (TECO) of the Philippines. My heartfelt thanks as well to the National Chiao Tung University (NCTU) for the Silver Bamboo NCTU Scholarship. Their

generous financial support has allowed me to focus on my studies and has helped make my stay here in Taiwan a more comfortable one.

Last but not the least, I would like to thank my parents, my sister, relatives and friends, who, each time I go home, have so nourished me with much needed love and emotional support that had been hard for me to find. I would not be where I am today if not for your persevering guidance and support.

Ericson G. Chua  
*Hsinchu, Taiwan (R.O.C.)*  
September 24, 2011



# Table of Contents

中文摘要 .....	i
Abstract.....	iii
Acknowledgements .....	v
Table of Contents.....	vii
List of Figures.....	x
List of Tables .....	xiii
Chapter 1 Introduction .....	1
1.1 Today’s Burdened Healthcare System.....	1
1.2 Technology as a Solution.....	1
1.3 Common Diseases of the Elderly .....	2
1.4 Biosignal Modalities for Detection of Brain and Heart Disease .....	2
1.5 Motivation for an Integrated Brain-Heart Monitoring Solution.....	3
1.6 Application Scenarios.....	5
1.6.1 Long-Term Observation.....	5
1.6.2 Emergency Care.....	7
1.6.3 Research on Brain and Cognitive Science.....	7
1.7 Organization of the Thesis.....	8
Chapter 2 Background on DOT, ECG and EEG .....	10
2.1 Diffuse Optical Tomography (DOT) of Human Tissue .....	10
2.2 Electrocardiogram (ECG).....	11
2.3 Electroencephalogram (EEG).....	14
Chapter 3 Integrated DOT/ECG/EEG Biomedical Multiprocessor for Portable Brain-Heart Monitoring Systems.....	17
3.1 System Overview and Application .....	17
3.2 Algorithm Discussion .....	19
3.2.1 Independent Component Analysis (ICA).....	19
3.2.1.1 Background and Motivation.....	19
3.2.1.2 Discussion.....	21
3.2.2 Heart Rate Variability (HRV) Analysis.....	23
3.2.2.1 Background.....	23
3.2.2.2 Motivation .....	24

3.2.2.3	Discussion.....	24
3.2.3	Diffuse Optical Tomography (DOT).....	28
3.2.4	Lossless Biomedical Data Compression.....	29
3.2.4.1	Background.....	29
3.2.4.2	Compression for Low Power.....	29
3.2.4.3	Survey of Biomedical Data Compression Algorithms .....	31
3.2.4.4	Lossless Data Compression for Low-Power Biomedical Monitoring Systems .....	35
3.3	Hardware Design and Implementation.....	39
3.3.1	Top Level Design.....	39
3.3.1.1	Chip Architecture.....	39
3.3.1.2	System Operation .....	40
3.3.1.3	Interface to the Analog Front-End (AFE) Circuitry .....	42
3.3.1.4	Output Data Format.....	45
3.3.1.5	Wireless Data Communication Interface and UART .....	48
3.3.2	Core Module Design.....	50
3.3.2.1	Priority Data Selector (PDS) .....	50
3.3.2.2	Inter-module Handshaking Mechanism .....	51
3.3.2.3	Diffuse Optical Tomography (DOT) Engine.....	52
3.3.2.4	Independent Component Analysis (ICA) Engine.....	54
3.3.2.5	Heart Rate Variability (HRV) Analysis Engine .....	73
3.3.2.6	Lossless Data Compression (COMP) Engine.....	75
3.3.3	Functional Verification.....	78
3.3.4	Chip Implementation in UMC 65nm CMOS Technology .....	81
Chapter 4	Integrated DOT/ECG/EEG Multiprocessor IP for SoC Implementation .....	83
4.1	Motivation for SoC Implementation.....	83
4.2	SoC-based Design of the Biomedical Multiprocessor System.....	87
4.2.1	SoC Architecture.....	87
4.2.2	Integrated DOT/ECG/EEG Multiprocessor DSP IP Core.....	91
4.2.3	System Operation Details .....	93
4.3	Results of Implementation Using the SOCLE CDK .....	96
4.4	Real-Time Application.....	97

4.4.1	Real-Time Integrated Brain-Heart Monitoring .....	98
4.4.2	SSVEP-based Brain-Computer Interface.....	101
4.4.2.1	Introduction to UCSD’s SSVEP-based Brain Dialer System.....	101
4.4.2.2	Integration with Proposed Biomedical Multiprocessor System ..	102
4.4.2.3	Results of Integration .....	105
Chapter 5	Conclusion and Future Work .....	112
5.1	Conclusion .....	112
5.2	Future Work .....	113
References	.....	115



## List of Figures

Figure 1-1 (a) Traditional EEG measuring equipment (b) Traditional EEG System enlargement at head part (c) Frequency Domain DOT System (d) Frequency domain DOT system enlargement at head part.....	6
Figure 1-2 Application scenario for the proposed wearable brain monitoring system .....	7
Figure 2-1 Foundation of DOT imaging based on NIR physics and technology .....	10
Figure 2-2 A typical ECG waveform and its parts .....	12
Figure 3-1 System overview and architecture of biomedical multiprocessor .....	18
Figure 3-2 Application of ICA on 4-channel EEG.....	20
Figure 3-3 EEG artifact removal using ICA (reproduced from [32]).....	20
Figure 3-4 Flowchart for R-peak detection .....	25
Figure 3-5 Illustration of R-peak detection .....	26
Figure 3-6 R-peak to R-peak intervals of the ECG.....	26
Figure 3-7 Energy savings $(1 - (CR^{-1} * E_{tx} + E_{comp}) / E_{tx})$ in percent as a function of $E_{comp}$ and CR using (a) Zigbee and (b) Bluetooth transceiver ICs. Solid and dotted red lines correspond to the CRs of [43] and [44] respectively.....	31
Figure 3-8 Basic DPCM prediction followed by Golomb-Rice entropy coding described as a block diagram (left) and pseudo code (right) .....	35
Figure 3-9 (a) prediction error distribution and its mapping to (b) Golomb-Rice encoding table with $K = 2$ ; (c) encoded output stream .....	36
Figure 3-10 Context model based on five past sample differences, where the context of $x_k$ is a function of $d_1$ to $d_5$ .....	37
Figure 3-11 Top architecture of the integrated EEG/ECG/DOT multiprocessor IC .....	40
Figure 3-12 Flow chart of system control by SCU.....	41
Figure 3-13 Analog front-end (AFE) interface.....	42
Figure 3-14 Model diagram of the DOT sensor board .....	44
Figure 3-15 40-bit biomedical data packet.....	45
Figure 3-16 Raw EEG biomedical data packets composed of EEG samples.....	46
Figure 3-17 ICA-processed EEG biomedical data packets composed of ICA samples .....	46
Figure 3-18 Raw ECG biomedical data packets composed of ECG samples .....	46
Figure 3-19 HRV data packets composed of frequency spectrum coefficients.....	47
Figure 3-20 DOT biomedical data packets composed of image pixels.....	47



Figure 3-21	DOT image pixel transmission sequence.....	47
Figure 3-22	(a) Bluetooth transceiver module from Hotlife; (b) Specifications summary of employed Bluetooth transceiver module .....	48
Figure 3-23	Architecture of the main biomedical DSP IP core within the chip.....	50
Figure 3-24	Architecture of PDS and priority assignment.....	51
Figure 3-25	Three-stage handshaking mechanism.....	52
Figure 3-26	DOT sensor array board.....	53
Figure 3-27	Hardware architecture of the DOT engine.....	53
Figure 3-28	Architecture and sequenced operation of the ICA engine .....	54
Figure 3-29	Illustration of the data windowing concept .....	57
Figure 3-30	Data windowing scheme (50% overlap) and operation schedule.....	57
Figure 3-31	Illustration of matrix <b>A</b> update using two-sided Jacobi plane rotations.....	60
Figure 3-32	Illustration of parallel Jacobi rotation set sequence.....	63
Figure 3-33	Rotation of vector in a Cartesian plane.....	64
Figure 3-34	Hardware architecture of the CORDIC engine.....	67
Figure 3-35	Hardware architecture of the Jacobi EVD engine .....	69
Figure 3-36	Parallel right rotations for rotation set $\{(1,2),(3,4)\}$ .....	69
Figure 3-37	Non-linear function implemented in the ROM lookup table.....	70
Figure 3-38	Hardware architecture and operation flow of the ICA training unit.....	72
Figure 3-39	Actual hardware states used by the main controller FSM .....	73
Figure 3-40	Hardware architecture of the HRV analysis engine.....	74
Figure 3-41	The sliding window scheme. RR intervals are divided into one minute frames. A data window consists of two frames with one frame overlapping with the previous window.	74
Figure 3-42	Hardware architecture of the lossless data compression engine.....	76
Figure 3-43	IC development flow methodology .....	78
Figure 3-44	System test bench for functional verification of biomedical multiprocessor IC .	80
Figure 3-45	Die micrograph and chip layout of the integrated biomedical DSP IC .....	81
Figure 3-46	Chip bonding map.....	82
Figure 3-47	Chip packaged in 28mm 128-pin CQFP.....	82
Figure 4-1	Atmel AT91SAM9263 SoC architecture .....	84
Figure 4-2	Platform-based system design .....	86

Figure 4-3	Architecture of the Cheetah ARM SoC .....	88
Figure 4-4	SOCLE ARM SoC Cheetah Development Kit (CDK) .....	89
Figure 4-5	Connections between the Cheetah SoC and Xilinx FPGA.....	89
Figure 4-6	Architecture view of SoC-based integrated biomedical multiprocessor system .	90
Figure 4-7	Architecture of the AHB-compatible biomedical multiprocessor DSP IP core...	92
Figure 4-8	Modified ICA data packets; UW matrix transmitted row-wise .....	92
Figure 4-9	Typical system operation sequence; †refer to Table 4-5, ‡refer to 3.3.1.4.....	95
Figure 4-10	Verification setup for the SoC-based biomedical multiprocessor system .....	97
Figure 4-11	Block diagram of the real-time integrated brain-heart monitoring application...	98
Figure 4-12	EEG and ECG electrode placement on human test subject.....	99
Figure 4-13	Analog front-end (AFE) modules used.....	99
Figure 4-14	Configuration wizard user interface on science station .....	100
Figure 4-15	System status monitor on patient-side sensing device.....	100
Figure 4-16	Real-time biomedical data display on LCD.....	101
Figure 4-17	Original SSVEP-based brain dialer system of UCSD .....	102
Figure 4-18	System integration development and evaluation setup.....	103
Figure 4-19	UCSD's VisualStim SSVEP stimulus generation software.....	104
Figure 4-20	Software extension to support de-artifact function.....	105
Figure 4-21	Actual system integration and evaluation setup, showing real-time CCA plot of 9Hz SSVEP without ICA.....	106
Figure 4-22	Real-time CCA plot of 9Hz SSVEP with ICA enabled .....	107
Figure 4-23	MATLAB CCA plot for 9Hz stimulus.....	108
Figure 4-24	MATLAB CCA plot for 10Hz stimulus.....	108
Figure 4-25	MATLAB CCA plot for 11Hz stimulus .....	108
Figure 4-26	Histogram of detected SSVEPs (1-second CCA analysis) .....	110
Figure 4-27	Histogram of detected SSVEPs (2-second CCA analysis) .....	111

## List of Tables

Table 2-1	Characteristics of the three main types of diffuse optical measurements.....	11
Table 2-2	Different peaks in a typical ECG waveform.....	13
Table 2-3	Different types of peak intervals that can be used to evaluate heart health.....	13
Table 2-4	Classification of continuous rhythmic sinusoidal EEG.....	14
Table 3-1	Top-level specifications of proposed system.....	18
Table 3-2	Energy consumption of commercial transceiver ICs.....	30
Table 3-3	Survey of lossless biomedical data compression algorithms.....	32
Table 3-4	Pseudo code of lossless biomedical data compression algorithm.....	38
Table 3-5	Chip I/O pin descriptions.....	40
Table 3-6	Bit field description of system mode byte command.....	41
Table 3-7	Biomedical signal selection using AIC_CHSEL, DOT_CHSEL and LED_SEL.....	43
Table 3-8	Summary of AFE specifications.....	44
Table 3-9	Summary of the output data format.....	45
Table 3-10	Comparison between Bluetooth and Zigbee.....	48
Table 3-11	Specifications of the on-chip UART module.....	49
Table 3-12	Calculation of maximum UART bandwidth usage.....	49
Table 3-13	ICA correlation performance vs. window size and window overlap.....	57
Table 3-14	Constant parameters in the CORDIC algorithm.....	65
Table 3-15	Summary of CORDIC algorithm and interfacing with Jacobi EVD algorithm;..	66
Table 3-16	ICA performance using various configurations of the ROM lookup table.....	71
Table 3-17	State, operation and data control in the ICA training unit.....	73
Table 3-18	Control signal parameters of the lossless data compression engine.....	77
Table 3-19	Verification of the various biomedical signal processing algorithms.....	79
Table 3-20	Chip specifications summary.....	81
Table 3-21	Area and power breakdown across different modules.....	82
Table 4-1	Typical hardware components comprising an SoC.....	83
Table 4-2	Select features of the SOCLE Cheetah ARM SoC platform.....	87
Table 4-3	Remapping of original functional blocks to SoC implementation;.....	90
Table 4-4	Description of AHB registers supported by the biomedical DSP IP core.....	92

Table 4-5	Summary of hardware devices and associated software interrupt handling.....	94
Table 4-6	Xilinx XC3S4000 FPGA device utilization.....	96
Table 4-7	Detected SSVEP frequencies in Figure 4-23 to Figure 4-25 .....	109
Table 4-8	Improvement in SSVEP detection using ICA, 1s CCA vs 2s CCA.....	109



# Chapter 1 Introduction

## 1.1 Today's Burdened Healthcare System

The population of people over the age of 65 worldwide has been predicted to more than double from 375 million in 1990 to 761 million by 2025 [1]. In Taiwan, the average age of the population has risen from 26 to 36 within the past two decades and the number of people below the age of 25 is rapidly decreasing. With later marriages and record-low birthrates, this trend is expected to worsen into the future. Such decline in the working age population is sure to result in a shortage of medical personnel, which, alongside rising costs will leave hospitals unable to meet the medical requirements of the growing number of elderly patients. Therefore, whilst life expectancy is seen to be rising, inadequacies in the provision of healthcare pose a major threat to the quality of life of the aged.

## 1.2 Technology as a Solution

Proper system design and integration of carefully selected technologies can help improve society's healthcare infrastructure and thereby alleviate the burden experienced by the medical community. With such systems, in combination with small, inexpensive, smart and easy-to-use healthcare products, not only can the hospital save on equipment cost and increase its capacity to handle patients, but more importantly, healthcare normally administered in the clinical setting can be pushed towards the home environment allowing for frequent monitoring, early diagnosis, and prevention. With early detection, disease conditions that would have otherwise deteriorated to be life-threatening, costly and time-consuming to treat, can be managed with higher success rates, at a reduced amount of time, cost and burden to both patient and the medical community. Such technologies can encourage elderly patients to become more independent regarding their own health, while also maximizing their quality of life at the convenience and comfort of their own homes. In order to achieve the greatest benefit, the most common diseases of the elderly are targeted and addressed.

### **1.3 Common Diseases of the Elderly**

Studies have shown that the most common ailment affecting the elderly is cardiovascular disease (CVD) followed by central nervous system (CNS) conditions. Deterioration of the circulatory system, in particular atherosclerosis, plays a major role in the progression of CVDs leading to blood clots, stroke and ischemic heart attacks, which are associated with high risk of sudden death. On the other hand, neurodegeneration caused by oxidative stress accelerates with age, causing typical neurological disorders in the elderly such as mental dementia or Alzheimer's disease, Parkinson's disease and epilepsy. Patients afflicted with these neurological conditions experience various degrees of deterioration and loss in cognitive, psychomotor and somatosensory functions. As the elderly population increases, healthcare technologies enabling the early detection of these heart and brain illnesses have become more important than ever before. Through early detection, diagnosis and treatment, the quality of life of the elderly patient can be maintained and even improved.

### **1.4 Biosignal Modalities for Detection of Brain and Heart Disease**

For detection of brain-related illnesses, the electroencephalogram or EEG recording is the most common method and provides important information regarding the state of the central nervous system. Applications of EEG in diagnostic neurology include detection of encephalopathy such as epilepsy, seizures, coma, stroke and tumors. In addition, diffuse optical tomography (DOT), a non-invasive imaging method popularly used to detect cancers in the breast, can also be used to render images of the brain, allowing early localization of dangerous brain tumors, cerebral hemorrhage and blood clot formations that can cause stroke [2]. Other uses of DOT are the measurement of blood flow, blood volume, oxygen saturation and cardiopulmonary function.

On the other hand, the electrocardiogram or ECG recording allows detection of a wide range of heart conditions. In clinical practice, it is possible to make accurate diagnosis of many diseases from the ECG. In myocardial infarction, commonly known as heart attack, the

ECG can be used to determine if heart muscles have been damaged after the attack. It can also serve as major predictor of mortality after myocardial infarction [3]. Other main applications of diagnostic ECG include detection of arrhythmias, disorders of the heart activation sequence, heart enlargement, myocardial ischemia and infarction, electrolyte imbalances and carditis [4].

In other indications, however, combined monitoring of physiological indicators may be required. For cases of sudden death in epilepsy, joint analysis of EEG and ECG can provide a vital indicator for prevention of occurrence [5]. Furthermore, recent studies have shown that analysis of EEG together with heart rate variability (HRV) or brain functional near infrared spectroscopy (fNIRS) can aid in better diagnosis and treatment. For example, EEG and HRV data were jointly analyzed for the automatic detection of seizures in newborns [6] and sleep apnea in hospital patients [7], while the advantage of combined analysis of EEG and fNIRS data for cognitive rehabilitation and post-traumatic stress syndrome was presented in [8] and indicated for the measurement of cerebral blood volume during seizures in [9].

## **1.5 Motivation for an Integrated Brain-Heart Monitoring Solution**

Despite these studies indicating the need for both individual and joint monitoring of brain fNIRS, EEG, and ECG, an integrated brain-heart monitoring solution has not been developed. Furthermore, most biomedical monitoring systems today are very expensive, bulky and heavy, lacking wireless capability and the ability to simultaneously monitor and process multiple types of biomedical signals. The purpose of this work is to address these deficiencies and make available a feasible solution for integrated and portable brain-heart monitoring.

There are many benefits in having an integrated biomedical monitoring solution instead of multiple separate systems. Most of the time, multiple types of biomedical signals need to be recorded and synchronized in time. For example, patients suffering from insomnia

go to the hospital to have their sleep quality evaluated. In this process, the patients are required to be monitored by EEG, ECG, EMG, fNIR, respiration, posture and sound. Without an integrated system, the technician must operate a plurality of medical equipment, prolonging administration, setup, logistics and examination time. With multiple devices, there is more room for error, especially during data and records handling, which is one of the biggest dangers in hospital management. With an integrated portable system, the operation is greatly simplified such that the technician only needs to put attention on a single equipment user interface, and all measured data can be stored safely, reliably and synchronized in one place. With the additional feature of wireless capability, even more convenience can be offered to the patient, since not only is the examination time shortened, but the patient can move around more freely as well. The following list summarizes the benefits of the proposed integrated portable brain-heart monitoring system:

- Lower product/device/system cost
- Improved system design and operation
  - Lower power consumption
  - Efficient use of wireless communication bandwidth (only one channel, and more fully utilized)
  - Multiple types of biomedical signals are synchronized in time
  - Short wiring for feeble physiological electrical signals
  - Decreased chance of sensor fall off
  - Reduced system size and volume allowing wider range of applications
- Lower healthcare operation costs and improved service
  - Lower administration costs
  - Allows experiment to be conducted more quickly
  - Savings in equipment logistics and management for hospital
  - Savings in technician training costs for multiple equipment



- Simpler and more efficient integrated system reduces chances for administration errors
- More convenient and comfortable for patient
  - Less time needed to perform the experiment
  - Single, light-weight, wireless, clutter-free portable and wearable device
  - Allows patient to carry on normal daily lifestyle
  - Can be used in the home setting

With the abovementioned advantages, the integrated portable health-care device will soon become an inevitable trend. In the next sections, three major target application scenarios will be pointed out, and in Chapter 3, a complete architecture for the portable brain-heart monitoring system will be proposed.

## 1.6 Application Scenarios

Before presenting the detailed design of the proposed system, three major target application scenarios are first discussed below. The proposed system targets applications in long-term medical observation, emergency care and potential researches on brain function and cognitive science.

### 1.6.1 Long-Term Observation

Traditional electroencephalogram acquisition systems and DOT systems are very bulky and very heavy. As a result, these expensive equipment installations are only possible in the hospital setting. For patients requiring long-term observation such as seizure, epileptic and degenerative brain disease patients, they have no choice but to stay in the hospital for long periods of time under stringent observation protocols, degrading severely their quality of life. Furthermore, because these equipment lack wireless communication features, many connecting wires (one for each channel) come between the subject and the monitoring

equipment, thus restricting free movement and bringing tremendous inconvenience to the patient. Figure 1-1 shows some traditional EEG and DOT acquisition instruments.

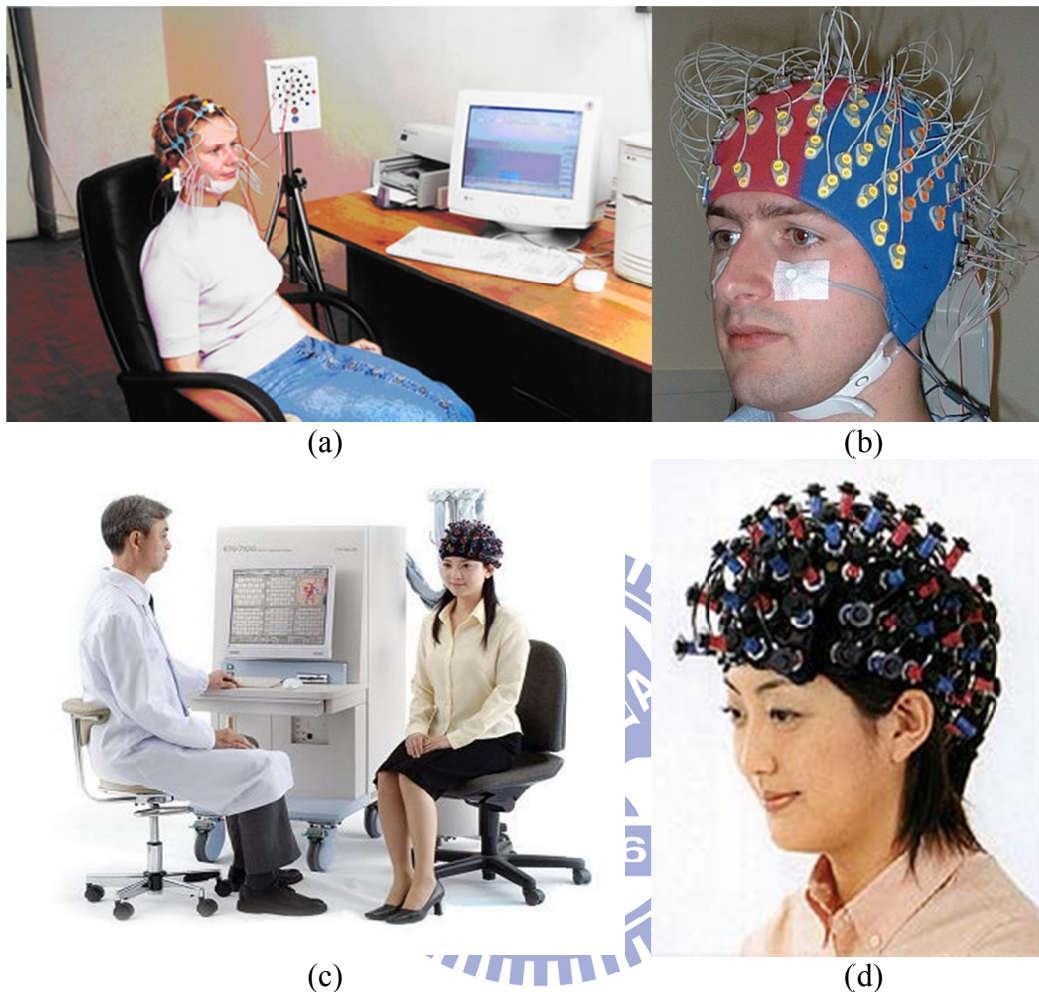


Figure 1-1 (a) Traditional EEG measuring equipment (b) Traditional EEG System enlargement at head part (c) Frequency Domain DOT System (d) Frequency domain DOT system enlargement at head part

The proposed system, shown in Figure 1-2, addresses this problem by offering a portable and inexpensive solution such that the subject is freed from the abovementioned restrictions. With wireless capability, the patient needs only wear a light-weight comfortable headgear for EEG and DOT, coupled with a few electrodes for ECG, thus granting the patient unrestricted movement and significantly improving his or her healthcare experience. Biomedical data received at the base station can be displayed in real-time, and stored into non-volatile storage media for further off-line processing, analysis and diagnosis. In addition,

the data can be sent from the base station to a remote workstation in a hospital for online monitoring and diagnosis by a medical expert.

With an inexpensive and portable implementation, the patient can even bring home the bio-signal acquisition device, improving significantly his or her quality of life, and allowing a more realistic biomedical monitoring more closely in accordance to the patient's daily lifestyle.

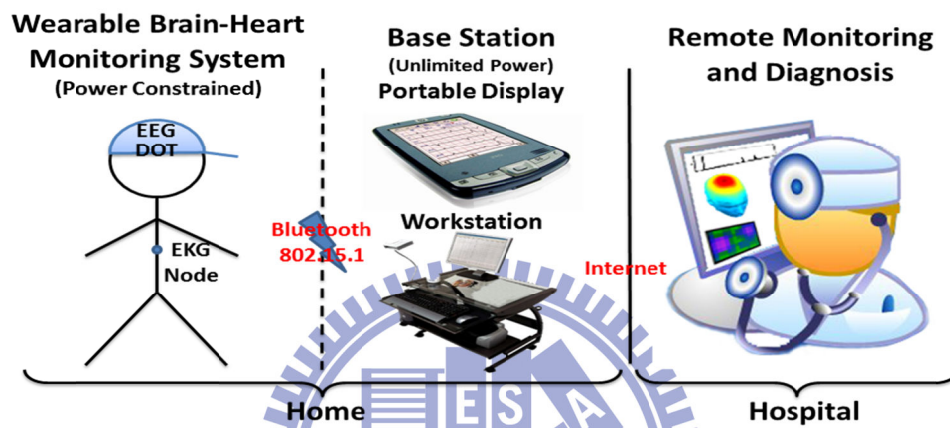


Figure 1-2 Application scenario for the proposed wearable brain monitoring system

### 1.6.2 Emergency Care

Current brain imaging technologies used in hospitals are not useful in emergency situations. For example, due to its massive size, a CT scanning machine cannot be equipped in an ambulance, even though cerebral hemorrhage is a very common case for car accident victims. If the ambulance can be equipped with a portable CW DOT device on board, occurrence of cerebral hemorrhage can be quickly determined on site, while requests for necessary preparations can be relayed to the hospital in advance even when the ambulance is still en route.

### 1.6.3 Research on Brain and Cognitive Science

Past researches have shown that the electroencephalogram (EEG) contains important information about the human cognitive process. As a result, related topics such as brain computer interface [10], artificial intelligence, electronic prosthesis and artificial neural

tissues have become an active and challenging research area in recent years. Even more recently, the research community is seeking more innovative ways of understanding the human brain, with the functional near-infrared (fNIR) technology showing great promise. Through fNIR, the cortical hemodynamic response can be localized thereby showing which area of the brain is active at a given point in time [11]. The flexible nature of DOT, which uses a wearable imaging cap shown in Figure 1-1d, makes it well-suited to human brain studies in enriched environments and for a wide range of behavioral paradigms and activations [12], including visual [11], motor tasks [13], somatosensory system [14], auditory [15], speech [16], and language [17]. Although technologies like magnetic resonance imaging (MRI) and positron emission tomography (PET) together with multiple-channel EEG can provide significantly higher brain activity resolution, their high cost and huge size result in low availability for academic research.

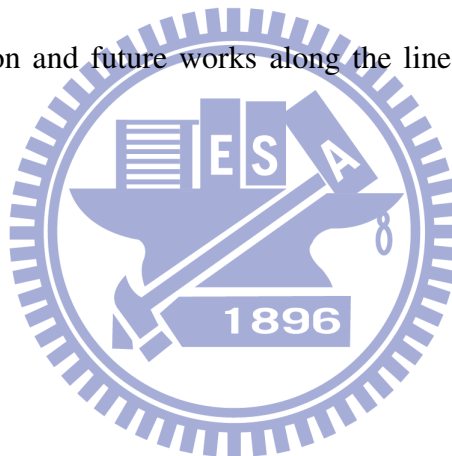
## **1.7 Organization of the Thesis**

The organization of this thesis is as follows. In Chapter 2, a brief background on the mechanics of the three important health indicators namely DOT, ECG and EEG is provided, including the significance of the associated signal processing algorithms independent component analysis (ICA) and HRV for EEG and ECG respectively.

In Chapter 3, the proposed integrated biomedical monitoring system is presented in detail, starting with the overall system overview and application, and followed by a quick review of the algorithms implemented – independent component analysis (ICA) for EEG, heart rate variability (HRV) analysis for ECG, and brain image reconstruction for DOT. The chapter continues with a comprehensive implementation-level description of the system, the hardware architecture, the individual sub modules and their interoperation. Functional verification of the entire system is discussed and finally, the chapter concludes with the results of chip implementation using UMC 65nm CMOS technology.

In Chapter 4, a special integrated DOT/ECG/EEG biomedical SoC IP design based on the previously taped-out chip is presented. The purpose of this SoC IP is to allow practical integration of the developed biomedical solution into today's standard ARM-based SoCs. Modifications from the chip implementation such as hardware-software repartitioning of the original design into an SoC architecture, wrapping of the original biomedical multiprocessor core into an AHB-compatible IP, support for additional functionality, and overall SoC system operation are described in detail. Results of system implementation and verification using SOCLE's ARM-based SoC Cheetah Development Kit are presented, and the chapter concludes with a demonstration of the developed system's functionality in various real-time biomedical applications.

Finally, the conclusion and future works along the line of this thesis are outlined in Chapter 5.



## Chapter 2 Background on DOT, ECG and EEG

### 2.1 Diffuse Optical Tomography (DOT) of Human Tissue

Due to its non-invasive and real-time characteristics as a radiography tool, DOT (Diffuse Optical Tomography) technology has been widely used to detect tumors in the breast and render images of the brain in recent years. Infrared and near-infrared rays (NIR) are transmitted into human tissue, and depending on the internal biological composition and structure, varying degrees of photon diffusion and signal attenuation are detected at the light sensor end. Using an array of NIR source and detector pairs, a map of received light intensities can be established from which the DOT image can be calculated and displayed for medical analysis and diagnosis. Figure 2-1 illustrates the foundation of DOT imaging based on NIR physics and technology.

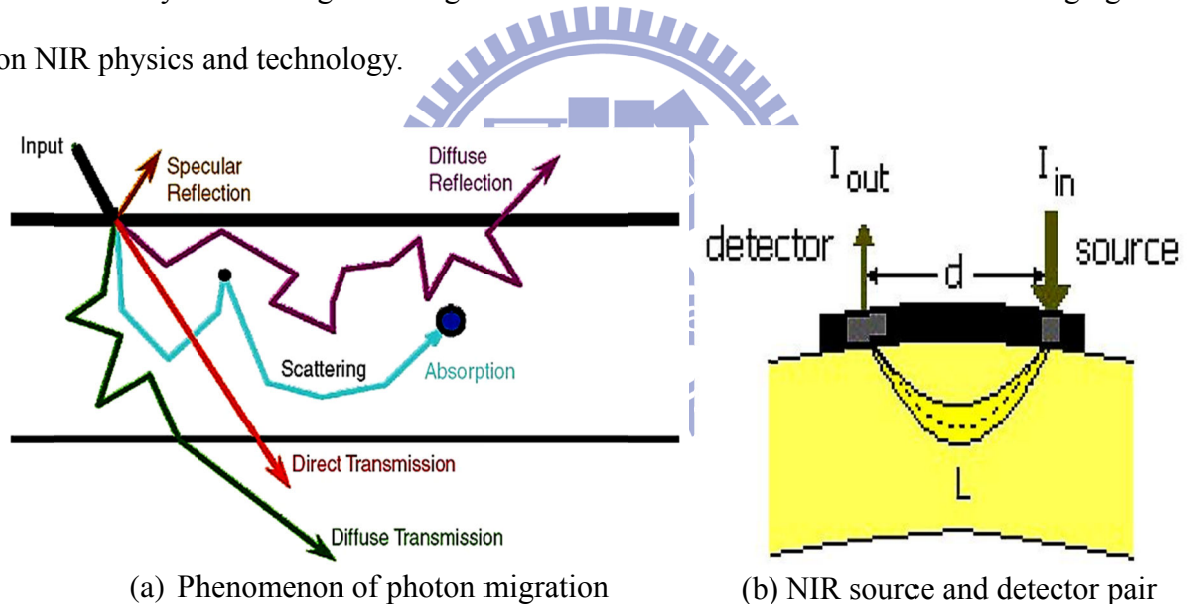


Figure 2-1 Foundation of DOT imaging based on NIR physics and technology

Many researches pertaining to DOT technology have made rapid progress and development in recent years. In particular, DOT can be used to detect oxygenated hemoglobin (HbO) and deoxygenated hemoglobin (Hb) concentration and volume using bi-wavelength near-infrared. Therefore, in clinical application, the primary uses of DOT are monitoring blood flow, blood volume, oxygen saturation, tumors within the brain, and detecting breast cancer [2]. Depending on the method of measurement of the diffused near-infrared light, DOT



can be generally divided into three main categories: the Continuous Wave (CW), Frequency Domain and Time Domain.

Table 2-1 Characteristics of the three main types of diffuse optical measurements

Type	Advantages	Disadvantages
Time Domain (TD)	<ol style="list-style-type: none"> <li>1. Spatial resolution</li> <li>2. Penetration depth</li> <li>3. Most accurate separation of absorption and scattering coefficients</li> </ol>	<ol style="list-style-type: none"> <li>1. High sampling rate</li> <li>2. Instrument size and weight</li> <li>3. Stabilization and cooling</li> <li>4. Cost</li> </ol>
	Example Uses: Imaging cerebral oxygenation and breast imaging	
Frequency Domain (FD)	<ol style="list-style-type: none"> <li>1. Relatively low sampling rate</li> <li>2. Relatively accurate separation of absorption and scattering coefficients</li> </ol>	<ol style="list-style-type: none"> <li>1. Penetration depth</li> <li>2. Instrument size and weight</li> <li>3. Cost</li> </ol>
	Example Uses: Cerebral and muscle oximetry, breast imaging	
Continuous Wave (CW)	<ol style="list-style-type: none"> <li>1. Low sampling rate</li> <li>2. Instrument size, weight and simplicity</li> <li>3. Low cost</li> </ol>	<ol style="list-style-type: none"> <li>1. Penetration depth</li> <li>2. Difficult to separate absorption and scattering coefficients</li> </ol>
	Example Uses: Finger pulse oximeter, functional brain experiments, cerebral hemorrhage	

Table 2-1 shows the characteristics of different DOT systems. The CW system provides advantages such as low cost, high portability, low power consumption and computation overhead, despite lack of depth information [13]. The volume of the CW-DOT system can be miniaturized which is the biggest advantage compared to other algorithms. Therefore, the CW-DOT system appears to be the most feasible candidate for hardware implementation. However, little literature has been published on such implementation of CW-DOT signal processing. Most CW-DOT systems post-process the signal offline by means of a computer such as [18] and [19]. This immediately eliminates the feature of portability, and therefore highlighting the advantage of a VLSI hardware implementation.

## 2.2 Electrocardiogram (ECG)

Electrocardiography (ECG) is an interpretation of the electrical activity of the heart over time captured and externally recorded by skin electrodes [20]. It is a noninvasive

recording produced by an electrocardiographic device. The ECG is an essential tool for health professionals in diagnosing heart conditions such as abnormal heart rhythms or arrhythmia when one is suspected.

The ECG works mostly by detecting and amplifying the tiny electrical changes on the skin that are caused when the heart muscle depolarizes during each heartbeat. Usually more than two electrodes are used and they can be combined into a number of pairs. The output from each pair is known as a “lead”. Different types of ECG measurements can be referred to by the number of leads that are recorded, for example 3-lead, 5-lead or 12-lead ECGs. A 12-lead ECG is one in which 12 different electrical signals are recorded at approximately the same time and will often be used as a one-off recording of an ECG, typically printed out as a paper copy. 3- and 5-lead ECGs tend to be monitored continuously and viewed only on the screen of an appropriate monitoring device, for example during an operation or whilst being transported in an ambulance.

A typical ECG waveform, shown in Figure 2-2, is composed of a P peak, a complex of QRS peaks and a T peak. How these peaks in the ECG are originated is explained in Table 2-2. Additionally, intervals between each peak can indicate the health of the heart. The three most commonly used intervals are listed in Table 2-3 along with their usages and descriptions.

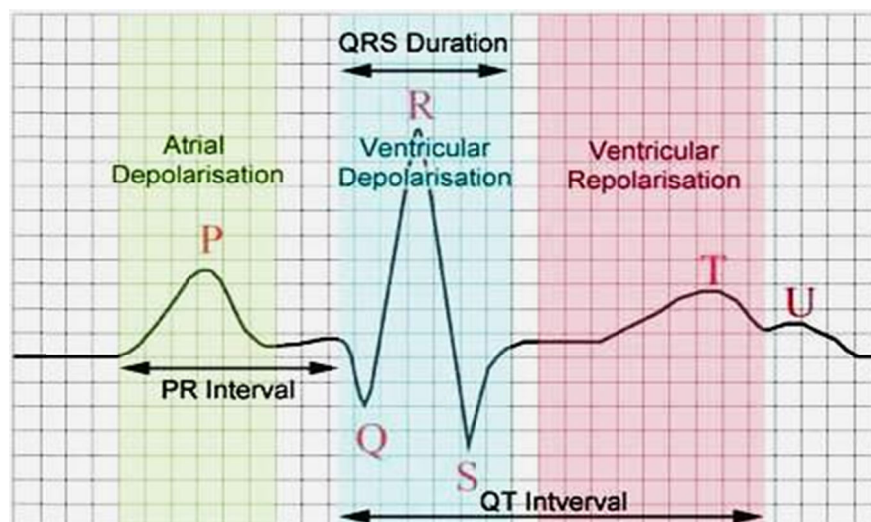


Figure 2-2 A typical ECG waveform and its parts



Table 2-2 Different peaks in a typical ECG waveform

Peak	Origination and Description
P	Systole (depolarization) of the atrium.
QRS	Systole (depolarization) of the ventricle. The amplitudes of QRS peaks are usually larger than P and T peaks, because of the stronger ventricular muscle.
T	Repolarization of the ventricle.

Table 2-3 Different types of peak intervals that can be used to evaluate heart health

Interval	Description
RR	Two adjacent R peaks can represent for the heart rate. The normal heart rate is between 50 bpm to 100 bpm (beat per minute).
PR	It is usually 120 to 200 ms long. The PR interval reflects the time the electrical impulse takes to travel from the sinus node through the AV node and entering the ventricles. The PR interval is therefore a good estimate of AV node function. <ul style="list-style-type: none"> <li>• A long PR interval (of over 200 ms) may indicate a first degree heart block. Prolongation can be associated with hyperkalemia or acute rheumatic fever.</li> <li>• A short PR interval may indicate a pre-excitation syndrome via an accessory pathway that leads to early activation of the ventricles, such as seen in Wolff-Parkinson-White syndrome.</li> <li>• A variable PR interval may indicate other types of heart block.</li> </ul>
QT	The QT interval generally represents electrical depolarization and repolarization of the left and right ventricles. A prolonged QT interval is a risk factor for ventricular tachyarrhythmias and sudden death.

The heart rate (HR) is a non-stationary value; it can vary as the body's need to absorb oxygen and excrete carbon dioxide changes, such as during exercise or sleep. The measurement of heart rate is used by medical professionals to assist in the diagnosis and tracking of medical conditions. It is also used by individuals, such as athletes, who are interested in monitoring their heart rate to gain maximum efficiency from their training. Heart rate variability (HRV) is measured as the variation in the beat-to-beat interval.

Heart rate variation (HRV) may contain indicators of current disease, or warnings about impending cardiac diseases [21]; it has proved to be a valuable tool to investigate the sympathetic and parasympathetic function of the ANS, especially in diabetic and post-infarction patients [21]. Sympathetic activity is associated with the low frequency range (0.04–0.15 Hz) while parasympathetic activity is associated with the higher frequency range (0.15–0.4 Hz) of modulation frequencies of the HR. This difference in frequency ranges allows HRV analysis to distinguish sympathetic from parasympathetic contributions evidently [21].

On the other hand, time-frequency parameters calculated using wavelet transform and extracted from the nocturnal heart period analysis appeared as powerful tools for obstructive sleep apnea syndrome diagnosis. Time-frequency domain analysis of the nocturnal HRV using wavelet decomposition could represent an efficient marker of obstructive sleep apnea syndrome [22].


### 2.3 Electroencephalogram (EEG)




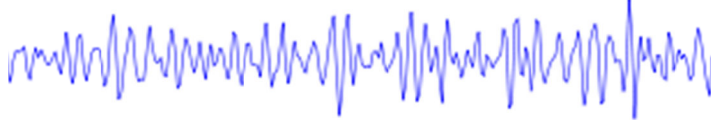
The electroencephalogram (EEG) is a non-invasive tool for recording electrical activity along the scalp produced by the firing of neurons within the brain. EEG measurements of different locations, frequency ranges, amplitudes, waveforms and periodicities can be used to distinguish different type of EEG. The EEG provides important information about the health of the central nervous system (CNS), especially in the newborn [23]. In the medical application of neurology, it is common to use EEG to diagnose conditions such as epilepsy, coma, encephalopathy and brain death.

The typical voltage range of EEG signals is about 10 microvolts to 100 microvolts, and the frequency domain is less than 100 Hz. In addition, there are five major bands of continuous rhythmic sinusoidal EEG activity. They are recognized as Delta (below 4Hz), Theta (4-8Hz), Alpha (8-12Hz), Beta (12-30Hz) and Gamma (above 30Hz) waves, and their characteristics are listed in Table 2-4. Signal components beyond this range are taken to be artifactual noise, under standard clinical recording techniques.

Table 2-4 Classification of continuous rhythmic sinusoidal EEG

Type	Frequency Range (Hz)	Common Amplitude Range (V)	Description
Delta ( $\delta$ )	0 ~ 4	-	Delta is often associated with the very young and certain encephalopathies and underlying lesions. It is seen in stage 3 and 4 sleep.



Theta ( $\theta$ )	4 ~ 7	Below 20 $\mu$	Theta is associated with drowsiness, childhood, adolescence and young adulthood. This EEG frequency can sometimes be produced by hyperventilation. Theta waves can be seen during hypnagogic states such as trances, hypnosis, deep day dreams, lucid dreaming and light sleep and the preconscious state just upon waking, and just before falling asleep.
			
Alpha ( $\alpha$ )	8 ~ 12	20 $\mu$ ~ 80 $\mu$	Alpha is characteristic of a relaxed, alert state of consciousness. For alpha rhythms to arise, usually the eyes need to be closed. Alpha attenuates with drowsiness and open eyes, and typically come from the occipital (visual) cortex. An alpha-like normal variant called mu is sometimes seen over the motor cortex (central scalp) and attenuates with movement, or rather with the intention to move.
			
Beta ( $\beta$ )	12 ~ 30	Below 20 $\mu$	Beta rhythms with low amplitude or multiple and varying frequencies is often associated with active, busy or anxious thinking and active concentration. Rhythmic beta with a dominant set of frequencies is associated with various pathologies and drug effects, especially benzodiazepines.
			
Gamma ( $\gamma$ )	30 ~ 100	-	Gamma rhythms may be involved in higher mental activity, including perception, problem solving, fear, and consciousness.
			

In clinical experiments, EEG signals are displayed based on the location of the electrode, which affects the amplitude, phase and frequency of the signal. EEG measurements can be divided into monopolar derivation and bipolar derivation. The monopolar derivation uses a probe electrode and a reference electrode fixed on the scalp surface, and it measures the relative potential between the probe electrode and reference electrode. On the other hand,

bipolar derivation uses two probe electrodes and a reference electrode. The potential difference between the two probe electrodes is taken as the EEG signal and relatively smaller signal amplitudes can be detected using this method.

Today, there are many proposed identification systems and human brain wave techniques used for medical diagnosis and treatment. For example, Fuzzy C-means (FCM) algorithm can be used to identify epileptic seizures and cerebral palsy [24]. However EEG signals are very weak, and thus often contaminated by various noise such as eye movement, EMG and electrical noise from nearby instruments [25].

Fortunately, this problem can be alleviated by algorithms such as independent component analysis (ICA) [26], which separates artifacts and noise from the measured EEG signals. Wavelet [27] and Spatially-Constrained [28] techniques can be used to identify artifact source channels thus making possible the automatic removal of such artifacts. As a result, clean EEG signals can be derived after the noise channel is eliminated and the remaining source channels remixed. However, the computation complexity is so intense that real-time ICA analysis is not feasible for a software implementation. Therefore, in recent years, research on the VLSI hardware implementation of ICA such as on FPGA and ASIC has become a hot topic.

## **Chapter 3 Integrated DOT/ECG/EEG Biomedical Multiprocessor for Portable Brain-Heart Monitoring Systems**

In this chapter, a highly-integrated multiprocessor chip design enabling the real-time processing of biomedical signals in portable brain-heart monitoring systems is presented. The design comprises a novel diffuse optical tomography (DOT) processor for taking brain imaging, an independent component analysis (ICA) processor for removing artifacts of brain electroencephalogram (EEG) signals, and a heart rate variability (HRV) analysis processor for monitoring heart electrocardiogram (ECG) signals. In the following subsections, a detailed discussion on the chip development starting from the top-level system overview and application, to algorithm flow, hardware design and leading down to the final chip implementation using UMC 65nm CMOS technology is presented.

### **3.1 System Overview and Application**

The proposed system, shown in Figure 3-1, comprises an analog front-end (AFE) circuit, the developed integrated biomedical DSP chip, and a commercial Bluetooth module supporting the UART protocol. The AFE circuit acquires, digitizes and sends multi-channel biomedical data such as EEG, ECG, and DOT to the front-end control unit (FICU) upon request by the DSP chip. The different biomedical signals are relayed to their respective engines and time-multiplexed according to a priority scheme for minimizing data latency at the output side. To reduce the bandwidth requirement and thereby save power, the multi-channel biomedical data is losslessly compressed and packetized prior to UART and wireless transmission. Finally, biomedical data packets received by the science station over the Bluetooth channel are decoded, displayed in real-time, or stored in non-volatile media for further processing and analysis. Operation modes for bypassing a particular engine or disabling compression are received as control packets from the base station and decoded by the system control unit. A summary of the system specifications is presented in Table 3-1.

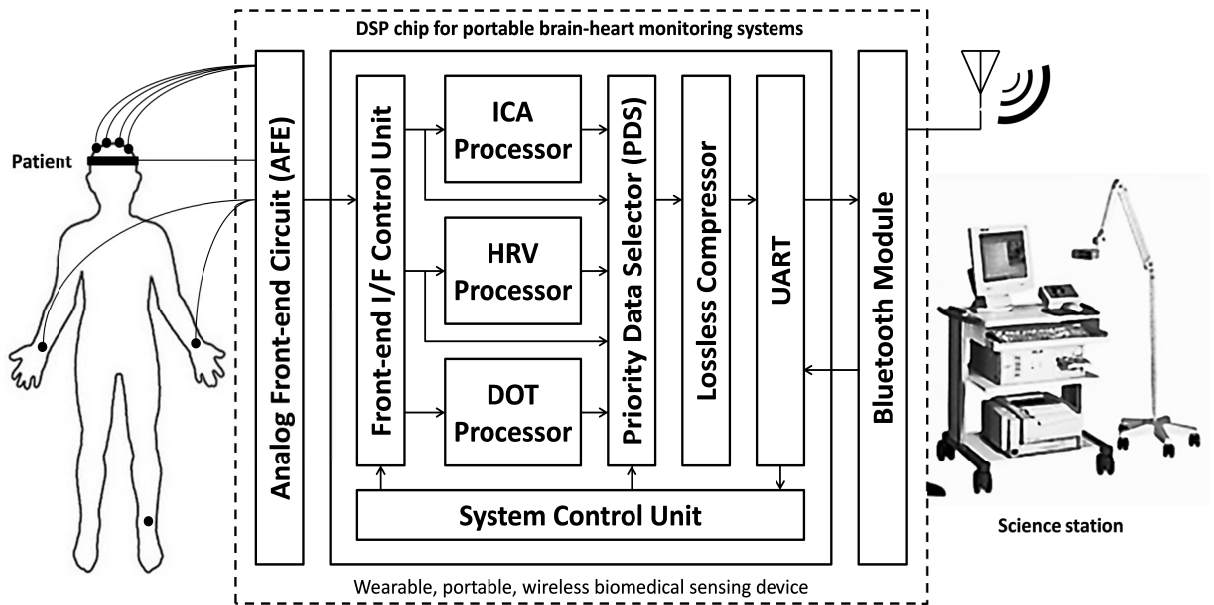


Figure 3-1 System overview and architecture of biomedical multiprocessor

Table 3-1 Top-level specifications of proposed system

Parameter	DOT Sub-System	EEG Sub-System	ECG Sub-System
Primary Function	DOT Image Reconstruction	Independent Component Analysis	Heart Rate Variability
Signal Source	Change in absorption coefficients influenced by hemoglobin	Electrical activity on the scalp caused by firing neurons in the brain	Electrical changes on the skin caused by heartbeat
Sample Rate(Hz)	1 frame (24 sensor values)	128	256
#Sensor (Channel)	12	4	3
#NIR LED	6	-	-
ADC Resolution	10-bit	10-bit	10-bit
Configuration Options	DOT/ Off	EEG/ ICA/ Off	ECG/ ECG+HRV/ Off
Compression Mode	On / Off	On / Off	ECG: On / Off HRV: not supported
UART Configuration	115200 baud, 8-N-1		
Supported Output Communication	RS232, Bluetooth or Zigbee		

## 3.2 Algorithm Discussion

In this section, a brief discussion on the various digital signal processing algorithms employed in the integrated biomedical multiprocessor is presented. As indicated in Table 3-1, the primary functions of the EEG, ECG and DOT subsystems are independent component analysis (ICA), heart rate variability (HRV) analysis, and DOT image reconstruction respectively. In addition, the theoretical basis of the specially developed lossless data compression algorithm is also discussed in this section as well.

### 3.2.1 Independent Component Analysis (ICA)

#### 3.2.1.1 Background and Motivation

Recently, blind source separation by Independent Component Analysis (ICA) has received attention because of its potential applications in signal processing such as in speech recognition systems, telecommunications and medical signal processing [29]. The goal of ICA is to recover hidden independent sources given only sensor observations that are unknown linear mixtures of the unobserved independent source signals.

In physiological electrical signal measurement such as the EEG, the observed signals are always the superposition of independent source signals, as shown in Figure 3-2. However, EEG signals are especially vulnerable and easily contaminated by artifacts such as eye movement, eye blink, power line noise and muscle (EMG) noise due to its weak signal strength at the microvolt range, thus posing a serious problem in the analysis and interpretation of EEG recordings. Fortunately, a particular flavor of ICA called the Infomax ICA [30] has been demonstrated to be an effective, powerful and feasible method for EEG denoising, which is able to identify both EEG components and artifact components and separate them into different channels [31]. By zeroing out the artifactual channels and performing a remixing of the remaining EEG channels of interest, artifact-free EEG can be obtained. Figure 3-3 summarizes the process of EEG artifact removal using the ICA technique.



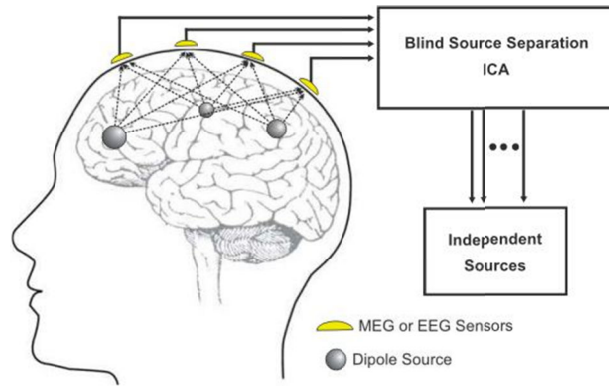
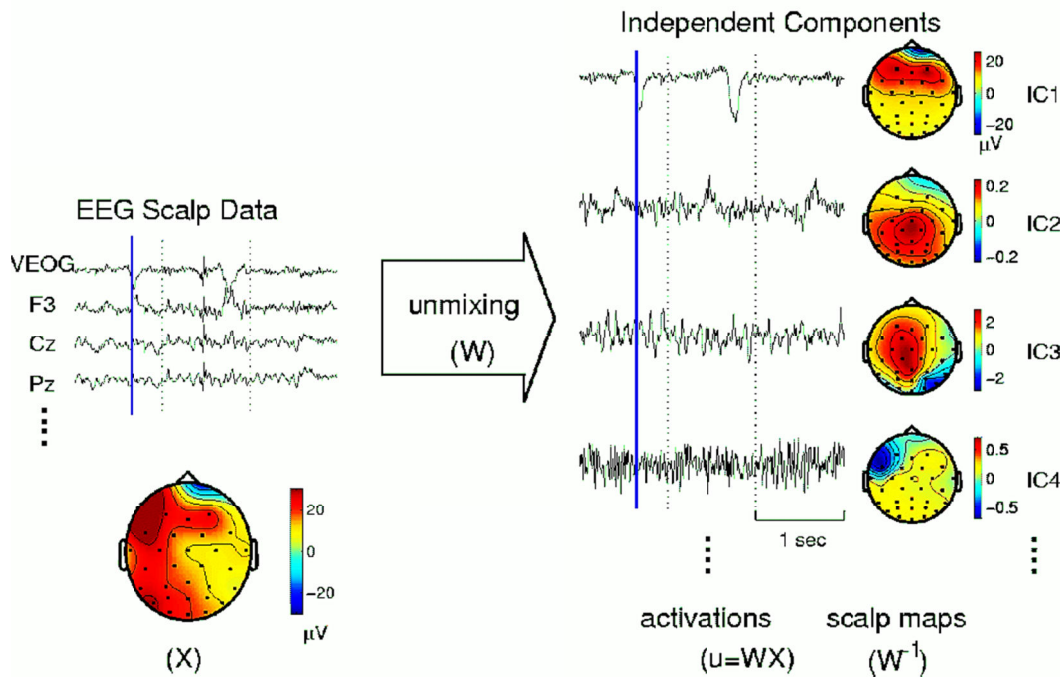


Figure 3-2 Application of ICA on 4-channel EEG

(a) ICA decomposition into independent components



(b) Removal of artifacts by remixing EEG signal components only

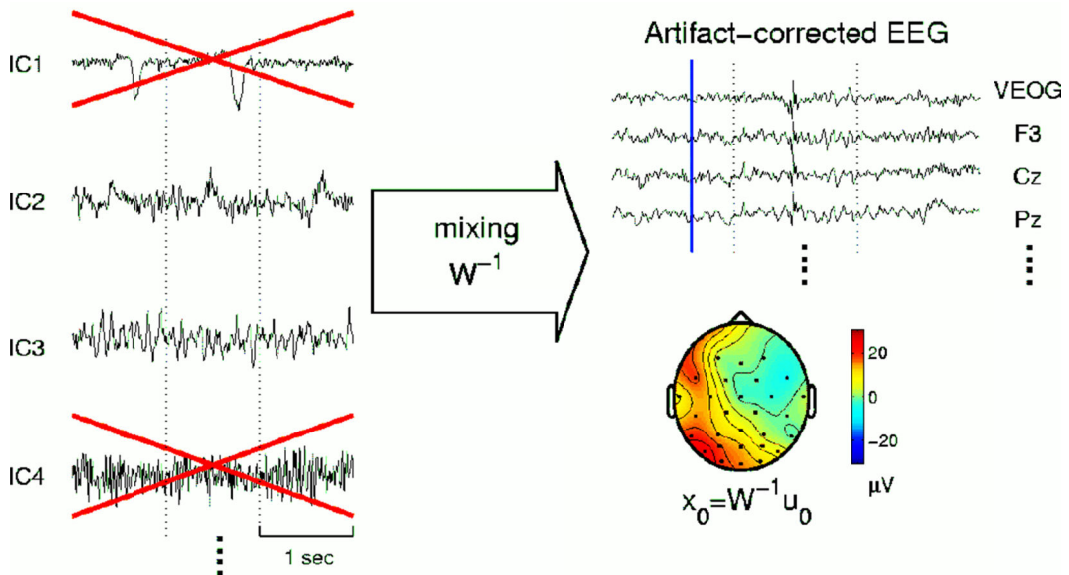


Figure 3-3 EEG artifact removal using ICA (reproduced from [32])



### 3.2.1.2 Discussion

Moving on from the theoretical background and application of ICA on EEG, we proceed to discuss the mathematical details of the implemented signal processing algorithm for the EEG subsystem. The algorithm implemented comprises three major steps: 1) pre-processing through centering and whitening, 2) determination of the unmixing weight using Infomax ICA and 3) computation of the ICA components. Note that the final stage of artifact channel selection and removal is left out from this section since it is implemented off-chip.

Because the Infomax ICA algorithm takes many training iterations to achieve convergence, a pre-processing step called whitening (a.k.a. principal component analysis or PCA) is employed to accelerate the training process. The whitening transformation is a de-correlation method that converts the covariance matrix of a set of samples into an identity matrix. This effectively creates new random variables that are uncorrelated and have the same variances as the original random variables. After whitening, the number of training iterations needed to achieve convergence is largely decreased. Prior to whitening, the EEG data is first centered to obtain zero mean data

$$X(i, j) = X_{\text{raw}}(i, j) - E\{X_{\text{raw}}(j)\} = X_{\text{raw}}(i, j) - \frac{1}{N} \sum_{i=1}^N X_{\text{raw}}(i, j) \quad (3-1)$$

where  $j = 1$  to 4 indicating the channel, and  $i = 1..N$  where  $N$  is the window size. Next, the actual whitening step is performed to obtain uncorrelated data. The covariance matrix of the centered EEG data is calculated first

$$\mathbf{X}_{\text{cov}} = E\{\mathbf{X}\mathbf{X}^T\} \quad (3-2)$$

resulting in a 4x4  $\mathbf{X}_{\text{cov}}$  covariance matrix for four channel ICA. The next step is to determine the whitening matrix  $\mathbf{P}$  such that the resulting transformation  $\mathbf{Z} = \mathbf{P}\mathbf{X}$  yields a  $\mathbf{Z}_{\text{cov}} = \mathbf{I}$ . In order to find  $\mathbf{P}$ , the eigenvalue decomposition (EVD) of  $\mathbf{X}_{\text{cov}}$  is calculated such that

$$\mathbf{X}_{\text{cov}} = E\{\mathbf{X}\mathbf{X}^T\} = \mathbf{E} \mathbf{D} \mathbf{E}^T \quad (3-3)$$

where  $\mathbf{E}$  is the orthogonal matrix of eigenvectors of  $\mathbf{X}_{\text{cov}}$ , and  $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  is the diagonal matrix of eigenvalues of  $\mathbf{X}_{\text{cov}}$ . Since  $\mathbf{X}_{\text{cov}}$  is a positive semi-definite matrix, the resulting eigenvalues in  $\mathbf{D}$  are all positive. With these observations, we choose

$$\mathbf{P} = \mathbf{E} \mathbf{D}^{-\frac{1}{2}} \mathbf{E}^T \quad (3-4)$$

as the whitening transformation matrix. Because  $\mathbf{E}^T = \mathbf{E}^{-1}$  and  $\mathbf{D}^{-\frac{1}{2}} = (\mathbf{D}^{-\frac{1}{2}})^T$ , we can show that

$$\mathbf{Z}_{\text{cov}} = \mathbf{E}\{\mathbf{Z}\mathbf{Z}^T\} = \mathbf{E}\{\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T\} = \mathbf{P}\mathbf{E}\{\mathbf{X}\mathbf{X}^T\}\mathbf{P}^T = \mathbf{E} \mathbf{D}^{-\frac{1}{2}} \mathbf{E}^T \mathbf{E} \mathbf{D} \mathbf{E}^T \mathbf{E} (\mathbf{D}^{-\frac{1}{2}})^T \mathbf{E}^T = \mathbf{I} \quad (3-5)$$

satisfying the original requirement for  $\mathbf{Z}$ . After whitening, the next step is to find the unmixing weight matrix  $\mathbf{W}$  by performing Infomax ICA training on the new uncorrelated random variables  $\mathbf{Z}$ .

The procedure of Infomax ICA training is described as follows:

1. Initialize  $\mathbf{W}(1)$  to  $\mathbf{I}$ , iteration number  $i = 1$
2. Calculate independent source estimates

$$\mathbf{U}(i) = \mathbf{W}(i) \mathbf{Z}$$

3. Apply sigmoid contrast function to source estimates

$$\mathbf{Y}(i) = \frac{1}{1 + e^{-\mathbf{U}(i)}}$$

4. Apply the gradient ascent learning rule to improve unmixing weight estimate  $\mathbf{W}$

$$\Delta \mathbf{W} = L_{\text{rate}}(\mathbf{I} + (\mathbf{1} - 2\mathbf{Y})\mathbf{U}^T)\mathbf{W}$$

$$\mathbf{W}(i + 1) = \mathbf{W}(i) + \Delta \mathbf{W}$$

5. Check for convergence

$$\text{If } \|\Delta \mathbf{W}\|^2 < T_{\text{conv}} \text{ or } i = I_{\text{limit}}$$

Output  $\mathbf{W}$

else

$i = i + 1$ , go back to step 2

Finally, the estimates for the independent source components are calculated as  $\text{ICA\_OUT} = \mathbf{W}' \mathbf{P}' \mathbf{X}$ , where  $\mathbf{W}'$  and  $\mathbf{P}'$  are the converged unmixing weight and whitening

transformation matrices respectively of the previous sample window. Thus, it should be clear that ICA\_OUT is not the same as the final  $U$  calculated in the Infomax ICA training procedure. Based on MATLAB simulations, the parameters  $L_{rate}$ ,  $T_{conv}$  and  $I_{limit}$  have been chosen as  $7.4768 \times 10^{-4}$ ,  $1.0012 \times 10^{-8}$  and 512 respectively to achieve a balance among convergence speed, convergence stability, memory size and calculation time.

### **3.2.2 Heart Rate Variability (HRV) Analysis**

#### *3.2.2.1 Background*

Heart rate variability (HRV) is a normal physiological phenomenon where the interval between successive heart beats of an individual varies over time. The term “heart rate variability” has widely become the adopted term to describe the variations of both instantaneous heart rate and RR interval [33]. To understand the implications of HRV, the origins of the heart rate and HRV are first discussed.

In the human body, visceral functions are controlled by the autonomic nervous system (ANS). These functions include heart rate, digestion, perspiration, and respiration. While some actions such as breathing may be controlled through conscious thought, visceral functions are generally involuntary. This is in contrast to voluntary motor functions controlled by the somatic nervous system (SNS), which together with the ANS formulates the peripheral nervous system (PNS).

The ANS classically consists of two main systems: the parasympathetic nervous system and the sympathetic nervous system. The parasympathetic and sympathetic nervous systems can be seen as two opposing branches exerting opposite effects on various internal organs. The parasympathetic nervous system regulates a ‘resting’ mechanism and causes heart rate and blood pressure to decrease. The sympathetic nervous system, on the other hand, provides a ‘fighting’ mechanism which increases heart rate and blood flow to muscles. The complementary nature of the two nervous systems allows humans to rest when possible and to

react to mentally or physically stressful situations when required. The resulting state of the autonomic system due to influences of the sympathetic and parasympathetic nervous system has become known as the sympathovagal balance.

#### 3.2.2.2 *Motivation*

As the heart rate is largely under the control of the ANS, variations in heart rhythm can reflect the influences of the parasympathetic and sympathetic nervous systems. Under resting conditions the level of activity of the parasympathetic nervous system, or vagal tone, prevails [34], and contributes to the high frequencies (HF) of HRV. Low frequencies (LF), on the other hand, can be associated with sympathetic activity which occurs in response to stress, exercise, and heart disease [21].

HRV has been shown to be an important indicator of cardiovascular health [33]. As the regulation mechanism of the heart is closely governed by the sympathetic and parasympathetic nervous systems, HRV is often used as a quantitative marker of the autonomic nervous system. Studies have shown that the HRV is an important indicator in many diseases and may contribute to a better treatment [21]. Applications of HRV have been applied to many forms of medical researches including studies in sleep apnea, patient monitoring after cardiac arrest [35], and use in intensive care units [36].

#### 3.2.2.3 *Discussion*

In order to provide insight to the intrinsic periodicities of HRV, the use of spectral analysis is indicated since HRV is the result of super-imposed components relating to the ANS. Through spectral analysis the contributions of sympathetic and parasympathetic activity can be viewed in a much clearer perspective than time-domain analysis or geometrical methods. In this section, the algorithm employed to perform the HRV is presented in detail.

The HRV algorithm takes in raw ECG samples and outputs a time-frequency spectrum representing the heart rate variability, and comprises the following steps: 1) R-peak detection

based on Pan and Tompkins [37], 2) R-peak to R-peak (RR) interval calculation and 3) spectral analysis of the RR intervals using the Lomb periodogram [38].

### 3.2.2.3.1 R-peak detection

In consideration of architecture simplicity and real-time properties, a classical derivative-based QRS detection algorithm based on Pan and Tompkins [37] is employed. Figure 3-4 summarizes the R-peak detection algorithm employed, comprising 1) differentiation, 2) squaring, 3) threshold detection and finally 4) peak detection. Differentiation is performed to identify the slope of the R wave in the QRS complex, of which the transfer function is

$$H(z) = \frac{1}{8}(-2z^{-2} - z^{-1} + z^1 + 2z^2) \quad (3-6)$$

with the corresponding difference equation

$$d[n] = \frac{1}{8}(2x[n] + x[n-1] - x[n-3] - 2x[n-4]) \quad (3-7)$$

After the derivative is calculated a squaring is performed to enhance the characteristics of the signal. Then a threshold is applied to the squared signal to detect the start of the QRS complex. The peak of the QRS complex is identified as the R peak of the ECG data segment.

Figure 3-5 shows the progression of the algorithm towards detecting the R peak location.

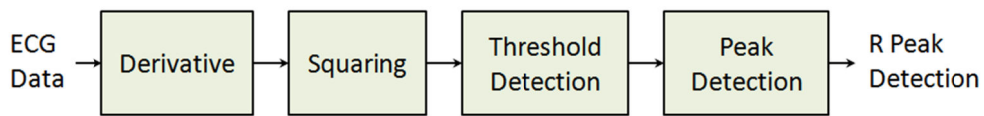


Figure 3-4 Flowchart for R-peak detection

### 3.2.2.3.2 RR interval calculation

After the R-peaks have been identified, the next step is to record the time intervals in between them. Figure 3-6 illustrates the time intervals between R-peaks, denoted by  $t_i$ , such that the RR interval time series is formed as

$$\mathbf{RR} = \{t_1, t_2, t_3, t_4, t_5, t_6, \dots, t_i\} \quad (3-8)$$

Finding the variability of these values with respect to time in terms of frequency is our main objective in HRV analysis. These values are passed on to the Lomb periodogram algorithm for spectral analysis.

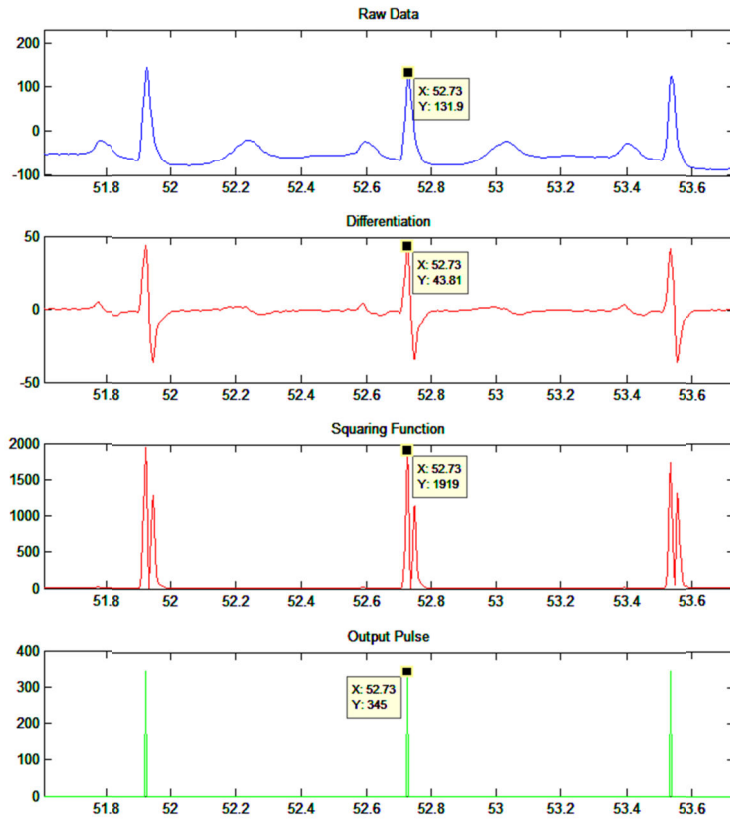


Figure 3-5 Illustration of R-peak detection

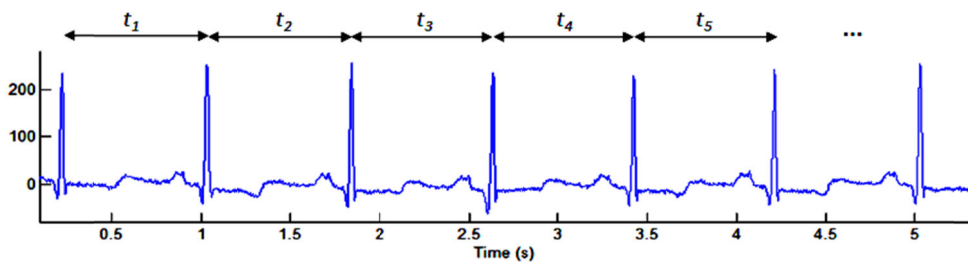


Figure 3-6 R-peak to R-peak intervals of the ECG

### 3.2.2.3.3 Spectral analysis of RR intervals using the Lomb periodogram

Spectral analysis of time series signals is typically performed using transforms such as the Fast Fourier Transform (FFT). However, the RR interval time series is unevenly sampled in time, thus, if FFT is to be used, the data must be resampled into evenly-spaced points in time. Unfortunately, studies have shown that such resampling and the choice of interpolation

scheme introduces inconsistencies in the final HRV analysis [39]. Thus, to address this issue while also maintaining an area-efficient portable solution, the Lomb periodogram [38], a spectral density tool specially designed for unevenly-spaced data sets, is chosen instead.

The Lomb method uses least squares fitting to estimate the amplitude of a given sinusoid with angular frequency  $\omega_j$  over non-uniformly sampled data. In other words, the power of the given sinusoid,  $P_N(\omega_j)$ , for a set of data points of length  $N$  is computed using a least-squares fit to the model

$$x(t_i) = A\cos(\omega_j t_i) + B\sin(\omega_j t_i) + n(t_i) \quad (3-9)$$

for  $i=0,1,\dots,N$ , where  $n(t_i)$  is noise. The Lomb transform is based on the DFT for unevenly sampled signals given as

$$X_N(\omega) \equiv \sum_j x_j e^{-i\omega t_j} \quad (3-10)$$

where  $t_j$  corresponds to the time when  $x_j$  is sampled. As the Lomb method weights the data on a “per point” basis rather than a “per time interval” basis, it is suitable for the analysis of non-uniform data.

Since the frequency range of interest in the analysis of HRV is between 0 to 0.4 Hz, the frequency range is normalized to 0 to 1Hz using  $N=256$  points, yielding a frequency resolution of approximately 0.004 Hz per point. Setting  $\omega$  to  $2\pi k/N$  and substituting into (3-10), the Lomb transform is given as

$$\begin{aligned} X_N\left(\frac{2\pi k}{N}\right) &= \sum_j x_j e^{-i\frac{2\pi k}{N}t_j} \\ &= \sum_j x_j \cos\frac{2\pi k}{N}t_j - i \sum_j x_j \sin\frac{2\pi k}{N}t_j \end{aligned} \quad (3-11)$$

where  $k = 0, 1, \dots, N-1$ , and  $x_j$  are zero-mean data. The pseudo code of the final algorithm is given as

*for n = 1 to (Number of RR intervals)*

```

for k = 0 to 255
    t = t + x[n];
    x = x[n] - μ;
    X(k) = X(k) + x * [cos(k/N * t) - i * sin(k/N * t)] ;
end
end

```

### 3.2.3 Diffuse Optical Tomography (DOT)

The scattering behavior of near-infrared photon projected through biological tissue has been derived based on the radiative transfer equation (RTE) since the 1990s. Since the scattering probability is much greater than the absorption probability in turbid media, the diffusion approximation to the transport equation can be used. Under the assumption of a continuous wave optical source, the diffusion equation is reduced to

$$-D\nabla^2\Phi(r) + v\mu_a\Phi(r) = vS(r) \quad (3-12)$$

where  $\Phi(r)$  and  $S(r)$  are the photon fluence rate and light source density with respect to location  $r$ , while  $D$ ,  $\mu_a$  and  $v$  are constant properties of a homogenous medium referred to as scattering factor, absorption coefficient and speed of light, respectively.

In the presence of impurities in the homogenous medium, changes in the absorption coefficient can be manifested. Using the Rytov approximation, a linearized equation of the form  $\mathbf{b} = \mathbf{A}\mathbf{x}$  can be derived from (3-12), and extending to the case of  $n$  voxels and  $m$  source-detector pairs,

$$\begin{bmatrix} \Phi_1(r_{s1}, r_{d1}) \\ \Phi_2(r_{s2}, r_{d2}) \\ \vdots \\ \Phi_m(r_{sm}, r_{dm}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} \Delta u_a(r_{v1}) \\ \Delta u_a(r_{v2}) \\ \vdots \\ \Delta u_a(r_{vn}) \end{bmatrix} \quad (3-13)$$

where each row of matrix  $\mathbf{a}$  transforms the changes in the absorption coefficient  $\Delta\mathbf{u}_a$  at various voxels  $v_1, v_2, \dots, v_n$  into light densities  $\Phi$  transmitted between particular source-detector pairs. Solving for the pseudoinverse of  $\mathbf{a}$ , the inverse solution is established, from which the DOT image manifested as  $\Delta\mathbf{u}_a$  can be reconstructed based on the light intensities  $\Phi$  received



at the detector side. Note that any two  $\Phi_i(r_{si}, r_{di})$ ,  $\Phi_j(r_{sj}, r_{dj})$  are considered distinct if at least  $r_{si} \neq r_{sj}$  or  $r_{di} \neq r_{dj}$ .

In our actual wearable DOT headband application, the target depth, locations of the LED sources and detectors, and medium characteristics are fixed, and as a result, the forward model matrix  $\mathbf{a}$  and its inverse becomes a fixed characteristic of the DOT system. With this observation, the inverse solution matrix is implemented as a constant array. By multiplying it with the detected light intensities, the changes in absorption coefficient at various locations at the target depth can be derived and interpreted as the DOT image.

### 3.2.4 Lossless Biomedical Data Compression

#### 3.2.4.1 Background

Today, portability and thus wireless transmission capability in patient monitoring systems is highly desired in order to enhance the patient's comfort and convenience. Strict restrictions on the size, weight and construction of portable devices have greatly limited the available onboard battery capacity, whereas wireless transmission of multi-channel biomedical data only aggravates the energy problem in these inherently power-isolated devices. Since most of the power is dissipated during wireless transmission, minimizing the amount of data through compression is essential to reduce the total system energy consumption, thereby allowing prolonged device autonomy and battery operating time.

#### 3.2.4.2 Compression for Low Power

It is well-known that wireless data communication takes up a large share of the total power consumption in most portable wireless devices or systems, with power dissipation proportional to the amount of data transferred. By compressing the data prior to wireless transmission, power can be saved provided that the compression operation itself does not consume too much power. A power tradeoff analysis for wireless EEG systems was presented in [40], showing the relationships among compression ratio (CR), power required to perform

compression ( $P_{\text{comp}}$ ), and power required for wireless transmission ( $P_{\text{tx}}$ ). If  $P_{\text{comp}} + CR^{-1} \cdot P_{\text{tx}} < P_{\text{tx}}$ , then total power consumption can be reduced.

For short-range, low data bandwidth applications such as brain-heart signal monitoring, the Bluetooth and Zigbee wireless protocols are recommended over ultra-wideband and Wi-Fi [41]. Table 3-2 shows a comparison of various commercial wireless transceiver ICs in terms of their power-related characteristics assuming nominal operating usage. The lower operating current draw of the Bluetooth and Zigbee ICs is particularly attractive, especially in applications with low-power requirements such as portable biomedical devices.

Table 3-2 Energy consumption of commercial transceiver ICs

<b>Protocol</b>	<b>Bluetooth</b>	<b>Zigbee</b>	<b>UWB</b>	<b>Wi-Fi</b>
Chipset	BlueCore2	CC2430	XS110	CX53111
VDD (V)	1.8	3.0	3.3	3.3
TX (mA)	57	24.7	227.3	219
RX (mA)	47	27	227.3	215
Bit rate (Mb/s)	0.72	0.25	114	54
<i>Energy consumption (nJ/bit)</i>	<i>143</i>	<i>296</i>	<i>6.5</i>	<i>13.4</i>

In order to save energy, a common practice is to turn on the transceiver only when data is available for transmission. Thus, when data bandwidth utilization is low, the transceiver spends most of its time in “sleep” mode, minimizing unnecessary energy consumption. Since power-up and power-down overheads are minimal, duty cycling can result in considerable energy savings. From an operational point of view, data compression can reduce the energy consumption even further by effecting a reduction in the amount of transmission data and essentially the resulting duty cycle. Figure 3-7 illustrates the possible energy savings  $E_{\text{saved}}$  as a function of CR and  $E_{\text{comp}}$  for cases employing a commercial Bluetooth or Zigbee transceiver IC. Energy savings can be maximized if a good compression

ratio can be realized at a minimal energy consumption cost, but may become negative when CR is too low or when  $E_{\text{comp}}$  is too high.

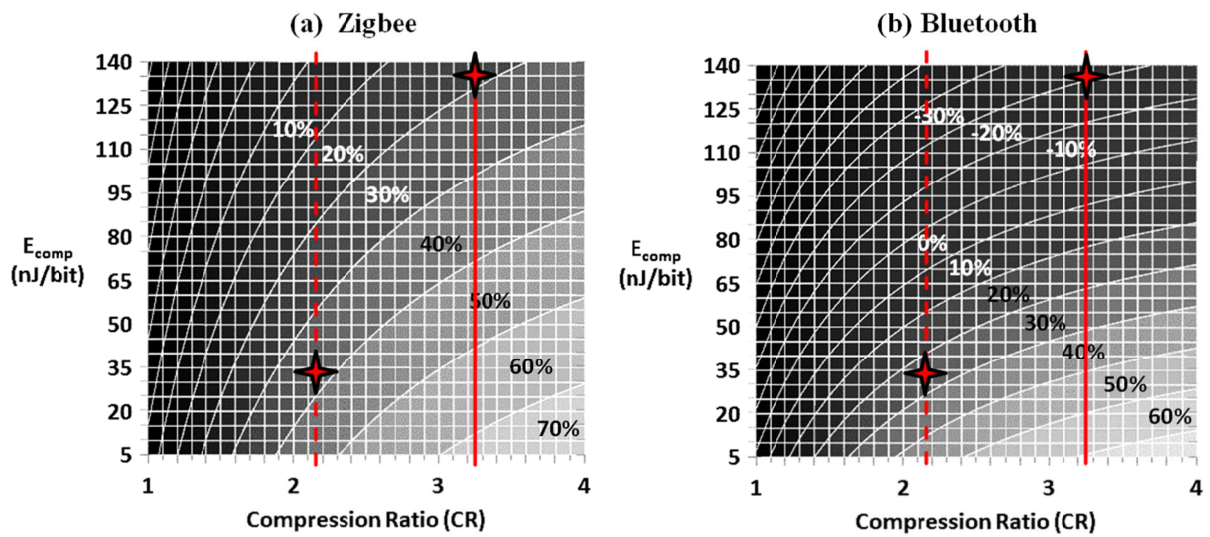


Figure 3-7 Energy savings  $(1 - (CR^{-1} * E_{\text{tx}} + E_{\text{comp}}) / E_{\text{tx}})$  in percent as a function of  $E_{\text{comp}}$  and CR using (a) Zigbee and (b) Bluetooth transceiver ICs. Solid and dotted red lines correspond to the CRs of [43] and [44] respectively.

### 3.2.4.3 Survey of Biomedical Data Compression Algorithms

Many biomedical data compression algorithms have already been developed in the past, mostly for EEG and ECG signals, and can be classified as either lossy or lossless. For brain-heart monitoring systems, we only consider lossless compression techniques in order to avoid the possibility of losing biomedical signal artifacts of potential diagnostic value. Table 3-3 shows a survey of various lossless biomedical data compression algorithms and their reported compression ratio performance. Since the compression ratio is highly sensitive to the characteristics of the input data (e.g. type of biomedical signal, sample precision, sampling frequency, slew rate, etc.), and most works were done independently using their own input data sets, the reported figures above only suggest expected compression performance possible within the bounds of standard clinical practice, and should not be objectively compared against each other. In summary, published lossless compression techniques report average compression ratio figures of 2.16 to 3.23 for EEG and 2.18 to 3.49 for ECG signals.

Table 3-3 Survey of lossless biomedical data compression algorithms

Ref.	Biomedical Signal	Algorithm			Compression Ratio (CR)
		Prediction	Transform	Entropy Coding	
[42]	EEG (16-bit)	DPCM	Integer Karhunen-Loève Transform, Stereo Integer DCT	Huffman	2.80
[43]	EEG (16-bit)	SLP Neural Network, Adaptive Error Modeling, Context-Based Bias Cancellation	--	Arithmetic	3.23
[44]	EEG (16-bit)	Auto-Regression Filter, Context-Based Bias Cancellation	--	Conditional Coding, Huffman	2.16 (approx.)
[45]	ECG (11-bit)	Auto-Regression Filter	Burrows-Wheeler Transform, Inversion Ranks	Arithmetic	3.41 (approx.)
[46]	ECG (11-bit)	Short-Term Prediction (Context-Based Bias Cancellation), Long-Term Prediction (R-R Interval)	--	Golomb-Rice	3.49 (approx.)
[47]	ECG (13-bit)	Lempel-Ziv '77/Complex Extract	--	Exp-Golomb/Huffman	2.23/2.18

However, unlike traditional data compression applications where storage space reduction and hence CR is the primary, and usually only, figure of merit, compression for low overall system power requires the algorithm's space-time complexity to be considered as well. For example, [43] is exceedingly superior over [44] as far as the compression ratio is concerned, but then requires large buffers and numerous computational iterations for training and accurate error modeling. In a hardware implementation point of view, the former is expected to require significantly more memory and computational elements or iterations, resulting in higher leakage and switching power consumption. As an illustration, Figure 3-7 shows how an algorithm with modest compression performance can be more suitable in a particular application when power consumption is also considered. Note that the power consumption points in the figure are indicated only for discussion purposes and do not represent actual results.

#### 3.2.4.3.1 Algorithm complexity considerations

In order to select a suitable candidate for a baseline hardware implementation, an assessment of algorithm complexity in the literature is presented. Most algorithms comprise a prediction step followed by entropy coding. More powerful algorithms [42], [45] additionally

employ a lossless reversible transform in between, typically resulting in improved compression ratios. However, aside from introducing considerable computational overheads, transforms cause data dependencies that require sample buffering, negatively impacting power consumption as well as latency. With respect to low power compression, the inclusion of transforms is generally not recommended.

Entropy coding is an essential step in compression algorithms, where frequently occurring values or symbols are mapped to shorter binary sequences and less frequent ones to longer sequences. In the literature, the entropy coding step is well-represented by Huffman, arithmetic and variations of Golomb coding. Although Huffman and arithmetic codes can closely follow source entropies, they require the upkeep of large memory structures for modeling source symbol probabilities. Alternatively, Golomb coding only requires the storage and estimation of a single code scaling parameter, since it assumes a particular shape of symbol probability distribution. Because predictive coding of biomedical signals roughly satisfies the statistical assumptions of Golomb coding, the resulting entropy coding performance is only slightly inferior from optimal (by around 5%), but the hardware complexity can be significantly much lower. Hence, for entropy coding, the power of two variant of Golomb coding, Golomb-Rice, is suggested.

Another important problem distinct from entropy coding is prediction. Prediction attempts to model the source signal such that an estimate of the current sample can be derived from previous samples. If the prediction can be recreated at the decoder side, then only the difference between the original and prediction values, called the prediction error, needs to be transmitted. An accurate predictor yields very small prediction errors, resulting in a low entropy signal that encodes efficiently into a shorter binary sequence after entropy coding.

Prediction techniques for biomedical signals range from the very simple like discrete pulse code modulation (DPCM) (where the previous sample is taken as the expected value for the current sample) to memory-intensive and computationally involved ones like neural

networks [43], auto-regression (AR) filters [44]-[46], Lempel-Ziv and complex extraction [47].

As mentioned earlier, most works focus only on the compression ratio, and hence take full advantage of more sophisticated mathematical techniques in order to achieve better prediction. For example, AR modeling and neural networks require multiple training iterations to be run against long sequences of samples in order to find precise floating-point model parameters that yield good prediction. Similarly, the Lempel-Ziv and complex extraction methods perform pattern or template matching on blocks of samples, exploiting the periodicity of ECG signals [47]. For the same reasons, the above methods are generally considered unsuitable for real-time, low power hardware implementation.

#### 3.2.4.3.2 Context-based bias cancellation

In [44], although the best reported compression ratio was associated with a 6<sup>th</sup> order AR predictor, it was shown that a very simple DPCM predictor, coupled with a computationally simple context-based bias cancellation, can attain compression performance close to that of more complicated techniques such as the AR model. In this scheme, contexts are defined based on past samples, and the typical DPCM prediction error in each context is estimated. By subtracting this estimate from the original prediction, an improved prediction can be achieved. Due to its simplicity, the method is seen to be very suitable for low power compression.

From the foregoing discussions in the previous sections, the prediction technique based on a DPCM predictor with context-based bias cancellation [44] shows most promise, while an entropy coding method based on Golomb-Rice is recommended due to its low complexity and good entropy coding performance. In the next section, the chosen lossless data compression algorithm is described in more detail.

### 3.2.4.4 Lossless Data Compression for Low-Power Biomedical Monitoring Systems

#### 3.2.4.4.1 Basic DPCM prediction with Golomb-Rice entropy coding

The proposed lossless data compression algorithm is largely based on a basic discrete pulse code modulation (DPCM) predictor followed by Golomb-Rice entropy coding, whose block diagram and algorithm listing is shown in Figure 3-8. The previous sample is taken as the prediction for the current sample, and the prediction error is obtained by subtracting the two. From a window size  $WS$  of prediction errors, the Golomb-Rice code scaling  $K$  parameter is estimated and the prediction errors are Golomb-Rice entropy coded using this parameter. Finally, the encoded stream is packed into data chunks of fixed width for output.

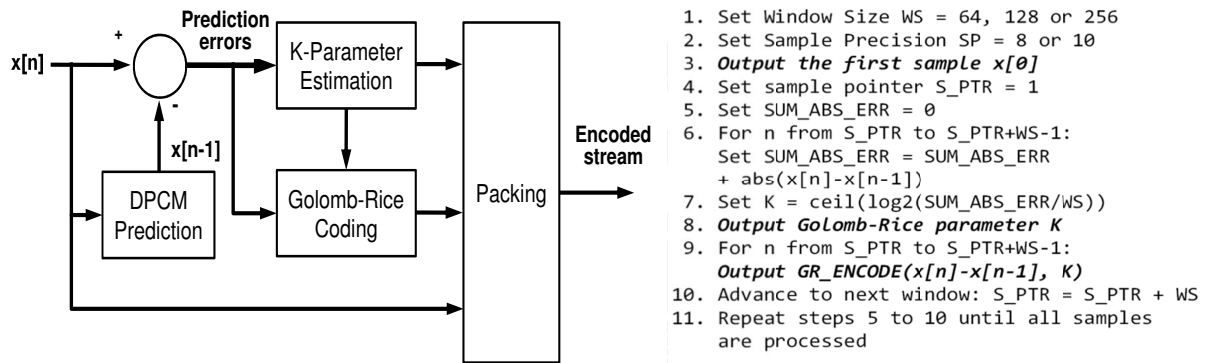


Figure 3-8 Basic DPCM prediction followed by Golomb-Rice entropy coding described as a block diagram (left) and pseudo code (right)

DPCM prediction is partly able to model the redundancies between consecutive samples, such that the resulting frequency distribution of prediction errors tends to center and peak near zero, as shown in Figure 3-9a. From this distribution, an appropriate  $K$  parameter can be estimated, such that distributions with smaller variances result in smaller  $K$  and distributions with larger variances result in larger  $K$ . A Golomb-Rice encoding table with  $K$  parameter equal to 2 is shown as an example in Figure 3-9b. Since smaller symbol (the prediction error is taken as the symbol) values occur more frequently, shorter code lengths dominate, resulting in overall compression. Finally, the encoded output stream, illustrated in Figure 3-9c, contains all the necessary information for the decoder to reconstruct the original signal.



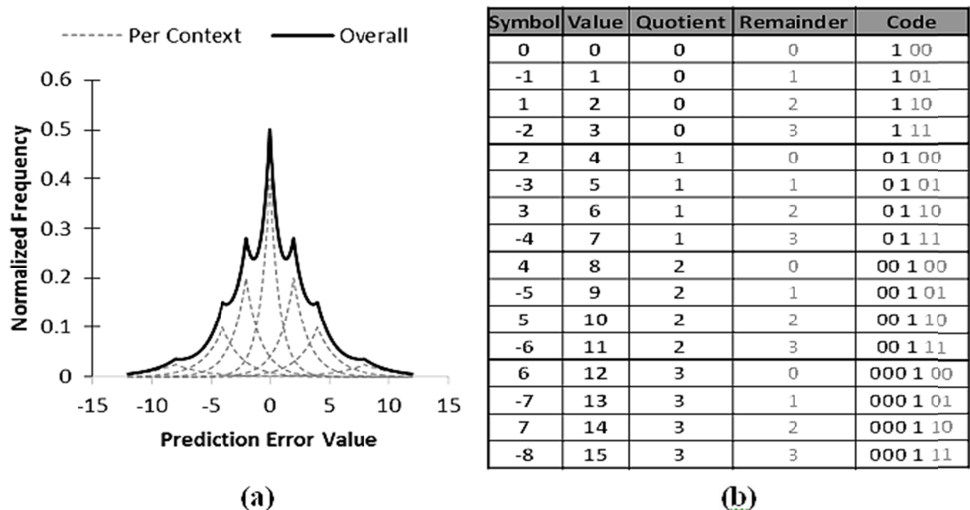


Figure 3-9 (a) prediction error distribution and its mapping to (b) Golomb-Rice encoding table with  $K = 2$ ; (c) encoded output stream

### 3.2.4.4.2 Context modeling

A typical overall distribution of prediction errors after DPCM is shown in Figure 3-9a, and is typically zero-mean with an overall variance to which an optimal  $K$  parameter is associated. However, it has been demonstrated in earlier literature [44], [48] that the overall Laplacian distribution is actually a composition of a plurality of Laplacian distributions, each having its own variance and mean shift away from zero (Figure 3-9a). By appropriately defining a context model, these individual structures can be extracted, upon which 1) bias cancellation can be performed to re-center each distribution back to zero, resulting in smaller prediction errors; and 2) more optimal Golomb-Rice  $K$  parameters can be estimated for each distribution.

A simple yet effective context model definition based on the past five sample differences was suggested in [44] for EEG signals, and is depicted in Figure 3-10. The current sample is  $x_k$ , and the sample differences  $d_i$  are defined as

$$d_i = x_{k-i} - x_{k-i-1} \quad (3-14)$$

where  $i$  is an integer between 1 and 5, inclusive, indexing the past five samples. To limit the number of contexts, the sample differences are quantized according to (3-15)

$$Q(d_i) = \begin{cases} 0, & \text{if } d_i < T \\ 1, & \text{if } d_i \geq T \end{cases} \quad (3-15)$$

where  $T = 0$ , resulting in a total of 32 contexts as defined according to (3-16).

$$\text{context}(x_k) = \{Q(d_1), Q(d_2), Q(d_3), Q(d_4), Q(d_5)\} \quad (3-16)$$

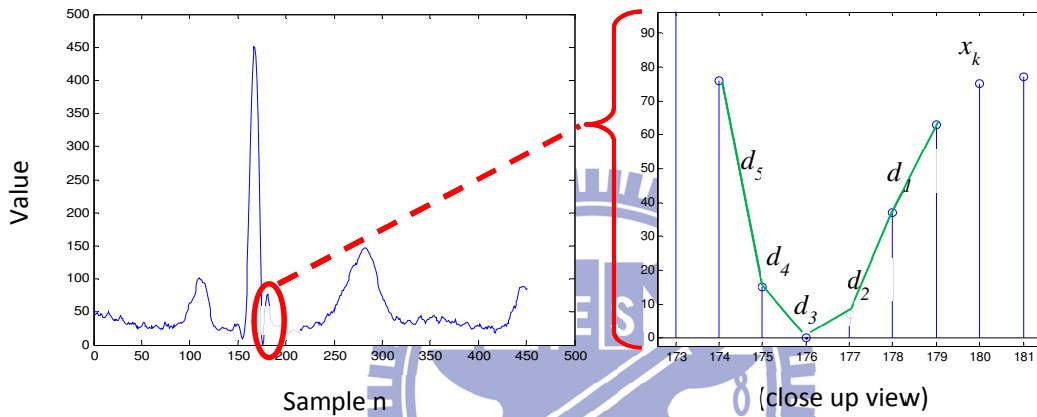


Figure 3-10 Context model based on five past sample differences, where the context of  $x_k$  is a function of  $d_1$  to  $d_5$

#### 3.2.4.4.3 Bias estimation and cancellation

For a particular context, the bias cancellation estimate is taken as the average prediction error occurring in that context. For example, the simple DPCM prediction for a current sample in an “always increasing” context (i.e.  $\text{context}(x_k) = \{1,1,1,1,1\}$ ) tends to be short by some positive value. This bias is added back to the DPCM prediction, and the result is a shorter binary code sequence after Golomb-Rice coding.

#### 3.2.4.4.4 Latency reduction

From a system point of view, a major disadvantage of the algorithm described thus far is its high output latency. Ideally, once a biomedical data sample (e.g. EEG, ECG) is sensed, it can be immediately shown on the display device. Since the Golomb-Rice encoder can only start outputting when the  $K$  parameter or bias cancellation value  $C$  has been calculated, the

latency is seen to be the window size WS. From simulation results, the window size WS must be large enough (typically a quarter to half the period) to achieve reasonable compression performance. In many clinical situations, a latency of half a second is already unacceptable.

To solve this problem, the estimation loop for the K parameter and bias cancellation value C is opened and is instead performed on a per sample basis according to [49]. As an additional benefit of this scheme, since at any time, the estimates are based on past samples only, the estimation procedure can be performed in exactly the same manner at the decoder, and hence both the K parameter and bias cancellation value C need not be transmitted as part of the output coded stream anymore.

#### 3.2.4.4.5 Final algorithm

Unfortunately, in the final algorithm, the bias cancellation mechanism had to be removed due to restrictions in chip real estate; resulting in around 7% performance loss in CR. Context-based estimation of the Golomb-Rice K parameter was retained. Table 3-4 summarizes all the above considerations to arrive at the final chosen lossless data compression algorithm for low power biomedical monitoring systems.

Table 3-4 Pseudo code of lossless biomedical data compression algorithm

```

1 Initialize N = ones(1,32) % N is the number of occurrences of a particular context
2 Initialize A = 8*ones(1,32) % A is the accumulated absolute prediction error in a context
3 Initialize prev_sample = 0, dpcm_pred = 0
4 Initialize context = 1 % 1 (00000) increasing ~ 32 (11111) decreasing
5 Initialize idx = 1
6 For each sample in signal do
7   context = 1 + (dpcm_pred - prev_sample >= 0) + ((context-1)%16) << 1
8   dpcm_pred = prev_sample
9   pred_resd = signal(idx) - dpcm_pred
10  % compute Golomb-Rice parameter k
11  K = 0
12  While (N(context)*2K < A(context))
13    K = K + 1
14  End while
15  % do Rice mapping
16  If (pred_resd < 0)
17    resd_mapped = -2 * pred_resd - 1
18  Else
19    resd_mapped = 2 * pred_resd
20  End
21  % do Golomb-Rice encoding using K on resd_mapped, and output

```

```

22 For unary = 1 to resd_mapped/2K
23     OUTPUT(1)
24 End for
25 OUTPUT(0) % output delimiter
26 OUTPUT(resd_mapped2K)
27 % update context counters, resetting (by halving) when N = 255
28 A(context) = A(context) + abs(pred_resd)
29 If (N(context)==255)
30     A(context) = A(context)/2
31     N(context) = 128
32 Else
33     N(context) = N(context) + 1
34 End if
35 % setup next iteration
36 prev_sample = signal(idx)
37 idx = idx + 1
38 End for

```

### 3.3 Hardware Design and Implementation

In this section, the hardware design and implementation of the proposed integrated EEG/ECG/DOT multiprocessor chip is described in detail. A top to bottom approach in the discussions is adopted, beginning with the system or chip top level design, architecture, operation and I/O interfacing, followed by the presentation of the main signal processing engines namely the ICA engine, HRV engine, DOT engine and the lossless data compression engine. Design considerations such as data buffering, inter-module handshaking and data bandwidth calculations are then discussed, and finally the results of chip implementation using UMC 65nm CMOS process technology is presented as a conclusion to this section.

#### 3.3.1 Top Level Design

##### 3.3.1.1 Chip Architecture

The chip top architecture, shown in Figure 3-11, comprises 1) a UART communication interface for receiving control and configuration commands and transmitting mixed multichannel raw or processed biomedical data; 2) a system control unit (SCU); 3) a front-end control interface unit (FICU) for controlling the analog front-end (AFE) circuitry and 4) a main biomedical DSP IP core containing the ICA, HRV, DOT and lossless data compression engines. A detailed description of the chip's I/O pins is given in Table 3-5.

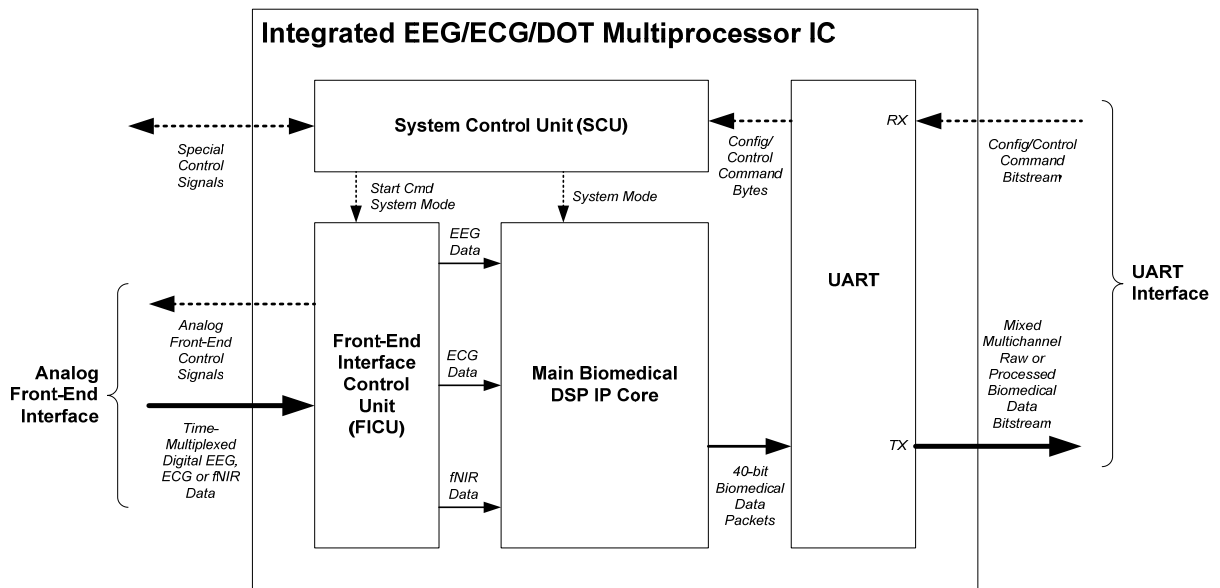


Figure 3-11 Top architecture of the integrated EEG/ECG/DOT multiprocessor IC

Table 3-5 Chip I/O pin descriptions

Pin Name	Direction	Width	Description
<b>Clock and Reset</b>			
CLK	Input	1	24MHz master clock
RESET	Input	1	Master reset signal
<b>Analog Front-End Interface</b>			
AIC_CLK10K	Output	1	10KHz clock for CHDDA and LPF of AIC
AIC_CLK_SLOW	Output	1	1.2MHz master clock for ADC of AIC
AIC_RESET	Output	1	Synchronous reset for ADC
AIC_CH_SEL	Output	3	Bio-signal channel select for ADC conversion
LED_SEL	Output	3	
DOT_CHSEL	Output	4	
AIC_START	Output	1	Start of conversion command to ADC
AIC_VALID_CONVERSION	Output	1	Debug pin (no need to connect)
AIC_EOC	Input	1	End of conversion flag from ADC
AIC_DATA	Input	10	Converted digital bio-signal output by ADC
<b>UART Interface</b>			
RX	Input	1	UART RX port for receiving system commands
TX	Output	1	UART TX port for transmitting biomedical data
<b>Special Control Signals</b>			
OVERFLOW	Output	1	Indicates that buffer overflow has occurred inside the chip
MANUAL_START	Input	1	Manually initialize chip to default mode w/o using UART

### 3.3.1.2 System Operation

Upon external reset, the chip enters into an inactive state. For chip operation to start, a configuration/control command must be sent to the UART RX interface. The command byte, whose fields are shown in Table 3-6, tells the chip which biomedical signal processing engines to enable. Note that if the HRV or ICA engines are to be used, their corresponding biomedical signal sampling must be enabled as well. So, commands like 8'bxxxxx10 (ICA on, but EEG sampling off) and 8'bxxxx10xx (HRV on, but ECG sampling off) are illegal.

Table 3-6 Bit field description of system mode byte command

Bit	Meaning
0	Enable 4-ch EEG data sampling
1	Enable 4-ch EEG ICA processing
2	Enable 3-ch ECG data sampling
3	Enable HRV analysis on ECG channel 0
4	Enable fNIR data sampling and DOT
5	Bypass EEG compression
6	Bypass ECG compression
7	Bypass DOT compression

Upon receiving the system mode command bit stream, the UART module decodes it into byte-register format and sends it to the SCU along with a start trigger signal. Figure 3-12 illustrates the operation of the SCU. After receiving the trigger from the UART, the SCU issues a chip-wide internal reset and broadcasts the system mode (by way of register pins) to relevant modules. The main DSP IP core requires 96 clock cycles to fully initialize. After initialization is done, the SCU sends a trigger signal to the FICU to start periodic requesting of biomedical data from the AFE. Depending on the system mode configuration, EEG, ECG and/or fNIR samples are received and processed by the main DSP IP core. Finally, raw or processed biomedical data is output through the UART TX interface according to the output data format described in 3.3.1.4. The whole chip continues to operate in an infinite loop of data sampling, signal processing and data transmission as described above until a stop command (8'bxxx00000) or new configuration command is received.

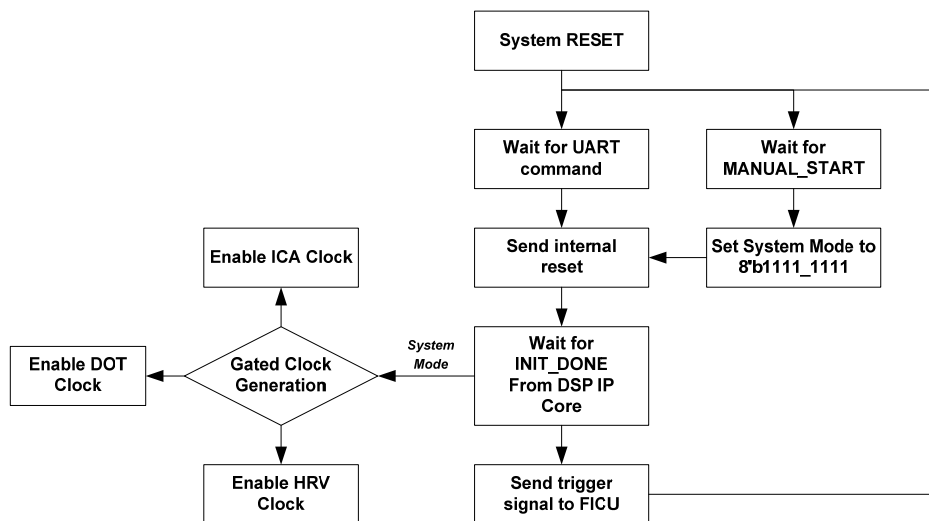


Figure 3-12 Flow chart of system control by SCU

Alternatively, the chip can be triggered to operate using the MANUAL\_START special control signal. In this case, the system mode defaults internally to 8'b1111\_1111 such that the biomedical signal processing engines ICA, HRV and DOT are all turned on and lossless data compression of any sort is turned off. From this point forward, the chip operates as described in the normal mode, and can be turned off or reconfigured through new commands received at the UART RX interface.

### 3.3.1.3 Interface to the Analog Front-End (AFE) Circuitry

The interconnections between the analog front-end circuit and the FICU form the interface from which mixed multichannel biomedical data (i.e. EEG, ECG, fNIR) are acquired into the system. Figure 3-13 shows the schematic of the AFE circuit interface. A 10 KHz clock generated by the FICU is provided to the chopper-stabilized differential difference (CHDDA) instrumentation amplifiers and low-pass switched-capacitor filters (SC LPF) of the AFE circuit, while a 1200 KHz clock is provided as master clock to the AD converter.

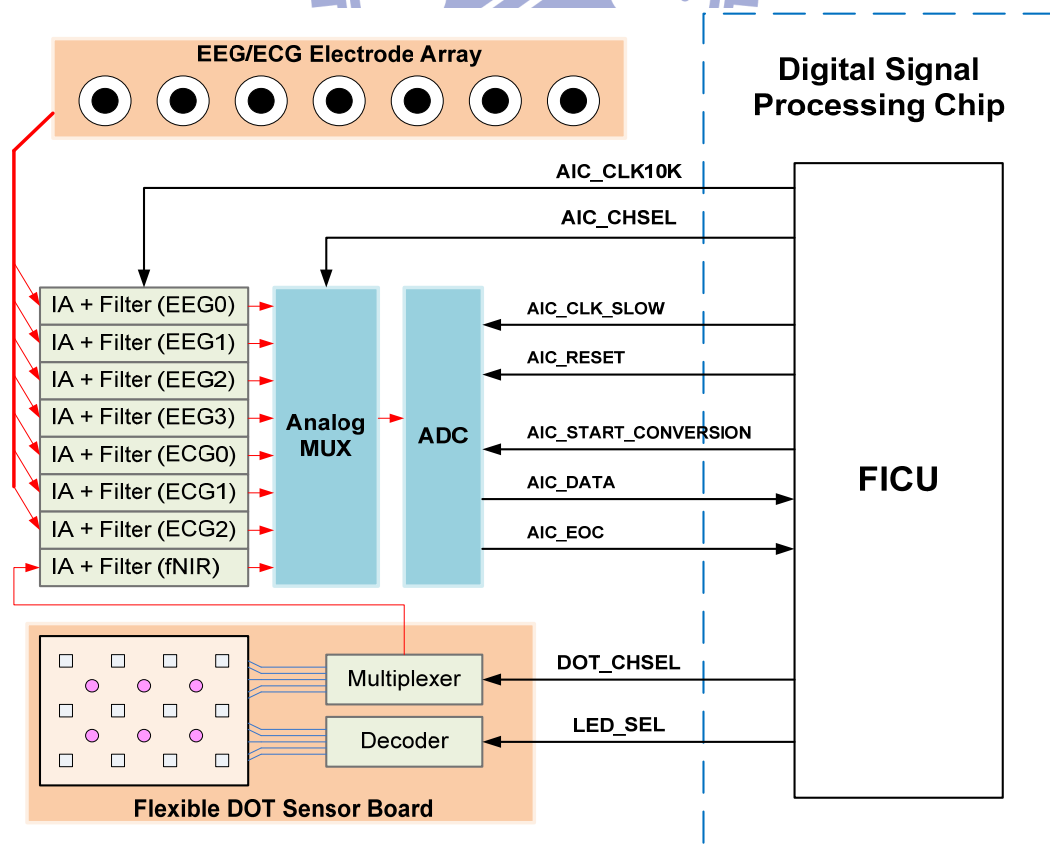


Figure 3-13 Analog front-end (AFE) interface



Since there are a total of 31 analog signal channels involved (7 for EEG/ECG and 24 for fNIR), the different signals are time-multiplexed through a single AD converter in order to reduce the number of I/O pins at the chip interface. Prior to starting AD conversion, the FICU first selects the desired biomedical signal using AIC\_CHSEL, LED\_SEL and DOT\_CHSEL. A description of these signals is provided in Table 3-7.

Table 3-7 Biomedical signal selection using AIC\_CHSEL, DOT\_CHSEL and LED\_SEL

AIC_CHSEL	Selected Channel		
0	EEG Channel 0		
1	EEG Channel 1		
2	EEG Channel 2		
3	EEG Channel 3		
4	EKG Channel 0		
5	EKG Channel 1		
6	EKG Channel 2		
7	LED_SEL	DOT_CHSEL	DOT Channel
	0 (LED 1)	0	0
		4	1
		1	2
		5	3
	1(LED 2)	1	4
		5	5
		2	6
	2 (LED 3)	6	7
		2	8
		6	9
	3 (LED 4)	3	10
		7	11
		4	12
	4 (LED 5)	8	13
		5	14
		9	15
		5	16
	5 (LED 6)	9	17
		6	18
		10	19
		6	20
		10	21
7		22	
11		23	

Once the selected biomedical signal is set up at the AD converter’s analog input, the FICU asserts AIC\_START\_CONVERSION for one clock cycle, to which the ADC responds with the converted digital data 12 cycles later together with a single-cycle assertion of AIC\_EOC. Internally, latching of AIC\_DATA depends only on the assertion of AIC\_EOC, and so the *end-of-conversion* (EOC) signal can be directly used as an INPUT\_VALID signal for AIC\_DATA.

The FICU has 128 Hz, 256 Hz and 24 Hz internal counters to periodically generate the acquisition requests for EEG, ECG and fNIR samples respectively, as needed depending on the system mode. Since the requests are asserted independently and can occur at the same time, actual access to the ADC interface is arbitrated according to a simple ADC priority scheme to ensure proper operation.

The fNIR sampling requires a 24 Hz counter since there are a total of 24 conversions that need to be performed within 1 second. Figure 3-14 shows a model diagram of the DOT sensor board with the 6 fNIR LED sources and 12 detectors labeled for reference. For each source, the four nearest detectors are enabled in sequence for AD conversion. For example, DOT conversions 1 to 4 (in black font) requires LED 1 (orange circle) to be turned on while sampling detectors (yellow squares) 0, 4, 1 then 5 in succession. The complete sequence of LED sensor control and fNIR data acquisition is ordered from top to bottom in Table 3-7. A summary of the specifications of the AFE circuitry and interface is shown in Table 3-8.



Figure 3-14 Model diagram of the DOT sensor board

Table 3-8 Summary of AFE specifications

Parameter	DOT Sub-System	EEG Sub-System	ECG Sub-System
Sampling rate (Hz)	1 frame (24 sensor values)	128	256
Sensors (channels)	12	4	3
fNIR LEDs	6	-	-
LPF cut-off freq. (Hz)	10	50	100
Gain (dB)	-	250	5000
Output range (V)	0 ~ 2.2 (external sensor)	0 ~ 2.5 (built-in IA)	0 ~ 2.5 (built-in IA)
ADC sample precision	10-bit	10-bit	10-bit
Digitized data range	0 ~ 900	0 ~ 1023	0 ~ 1023
ADC priority	3	1	2
ADC order	One sample each time	Ch0-Ch1-Ch2-Ch3	Ch0-Ch1-Ch2

### 3.3.1.4 Output Data Format

After the biomedical data has been sampled and processed, the results need to be output. To allow the easy adoption of the proposed design in third party systems, a standard output data format is specified and documented.

The output data format is defined on top of the UART byte layer, wherein the data unit is a 40-bit biomedical data packet as shown in Figure 3-15. The unit biomedical data packet comprises a 38-bit data payload and a 2-bit data type header specifying whether the packet carries DOT, HRV, ECG or EEG/ICA biomedical data. The presence of a particular type of biomedical data depends on the configured system mode (received from the science station) and is described in Table 3-9. Note that because the lower bytes are transmitted first before the upper bytes, only until the last byte is received (BYTE4) will determination of the data type be possible. The rest of the output data format specifications is summarized in Table 3-9. Note that although the EEG and ICA data share the same data type header, the science station can distinguish between the two based on the system mode it sent to the chip.

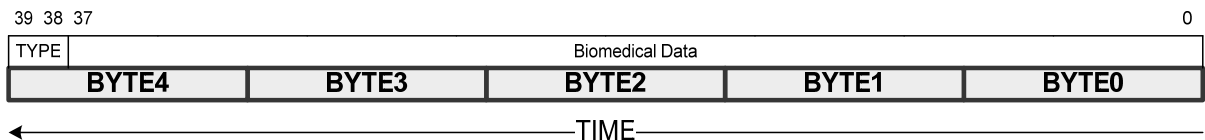


Figure 3-15 40-bit biomedical data packet

Table 3-9 Summary of the output data format

	DOT	HRV	ECG	EEG / ICA	
System Mode Bit	4	3	2	1	0
<b>Action</b>	Enable NIR Sampling and DOT Engine	Enable HRV Engine (operates on CH0 only)	Enable 3-ch EKG Sampling	Enable ICA Engine	Enable 4-ch EEG Sampling
<b>Output Description</b>	8x12 pixel DOT image (735nm)	256-point frequency spectrum	3-ch EKG signal	4-ch ICA signal	4-ch EEG signal
<b>Output Data Rate</b>	1 image / sec	1 spectrum / min	3-ch x 256 samples / sec	4-ch x 128 samples / sec	4-ch x 128 samples / sec
<b>Output Unit</b>	20-bit signed pixel intensities	32-bit signed coefficient pairs {16-bit REAL, 16-bit IMAG}	10-bit signed EKG samples	16-bit signed independent component samples	10-bit signed EEG samples
<b>Output Sequence</b>	Sub-frame, raster order	coeff0, coeff1, ..., coeff255	CH0 -> CH1 -> CH2	CH0 -> CH1 -> CH2 -> CH3	CH0 -> CH1 -> CH2 -> CH3
<b>Data Type Header</b>	2'b11	2'b10	2'b01	2'b00 (if ICA is enabled, raw EEG samples are not output)	

In order to achieve low latency real-time performance, processed biomedical data are immediately packetized and output, such that consecutive packets can be of different data types during actual operation. Since the time lapse between the last pixel of a DOT image to the first pixel of the next takes one second, data flush pad bits are appended to the last pixel to form a complete 40-bit packet for immediate transmission. The same is done for HRV data, wherein the frequency spectrum is output only once every minute. For the times series ECG and EEG/ICA data, the worst case stall is just a minimal sampling period and so data padding is not employed. An explicit illustration of the different types of biomedical data packets (lumped together for clean layout) is shown in Figure 3-16 through Figure 3-21.

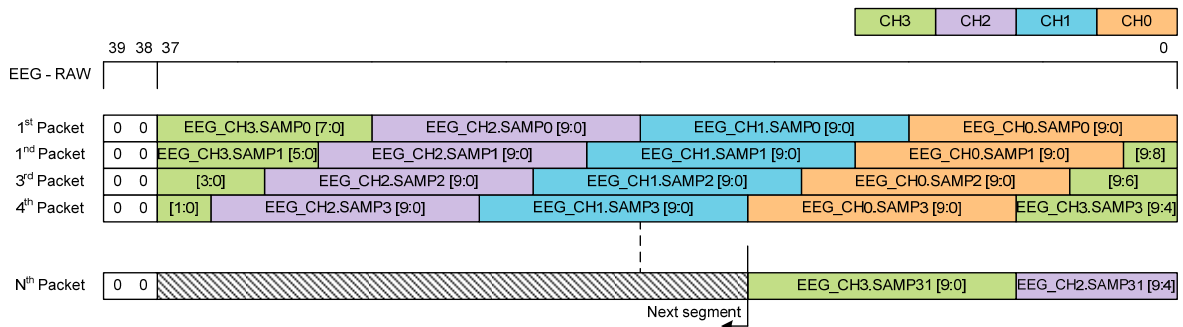


Figure 3-16 Raw EEG biomedical data packets composed of EEG samples

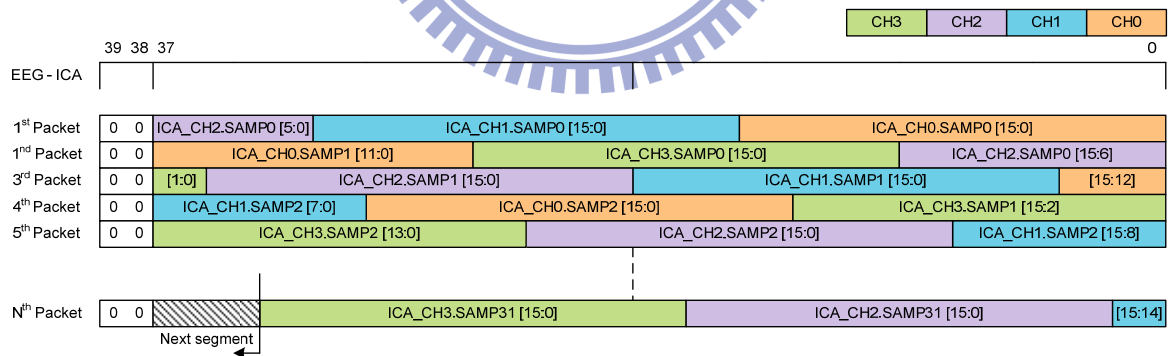


Figure 3-17 ICA-processed EEG biomedical data packets composed of ICA samples

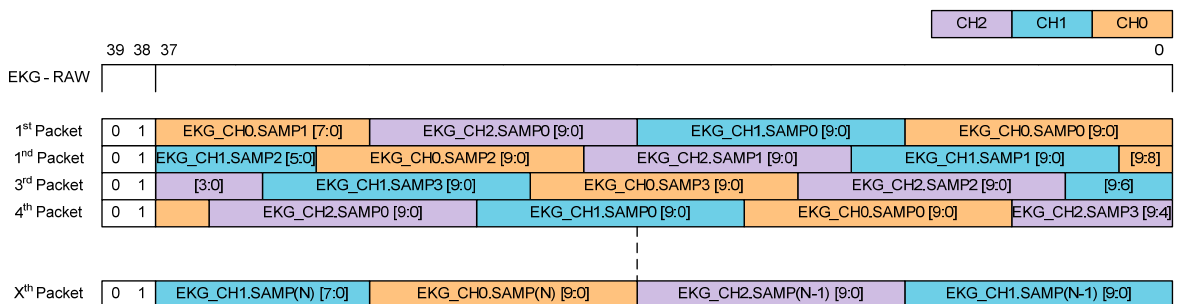


Figure 3-18 Raw ECG biomedical data packets composed of ECG samples

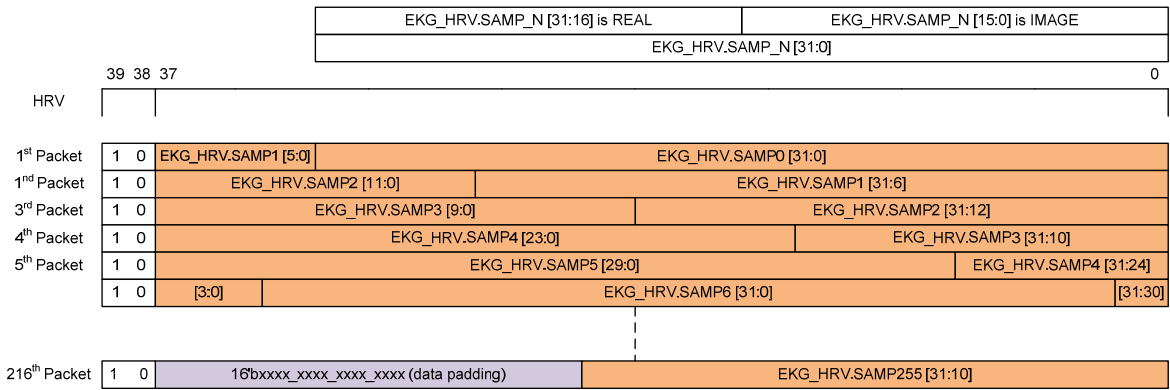


Figure 3-19 HRV data packets composed of frequency spectrum coefficients

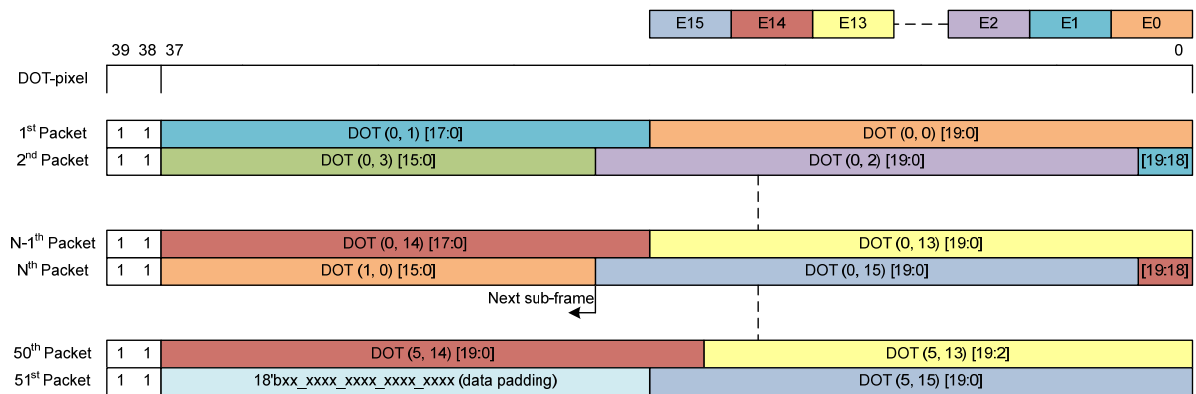


Figure 3-20 DOT biomedical data packets composed of image pixels

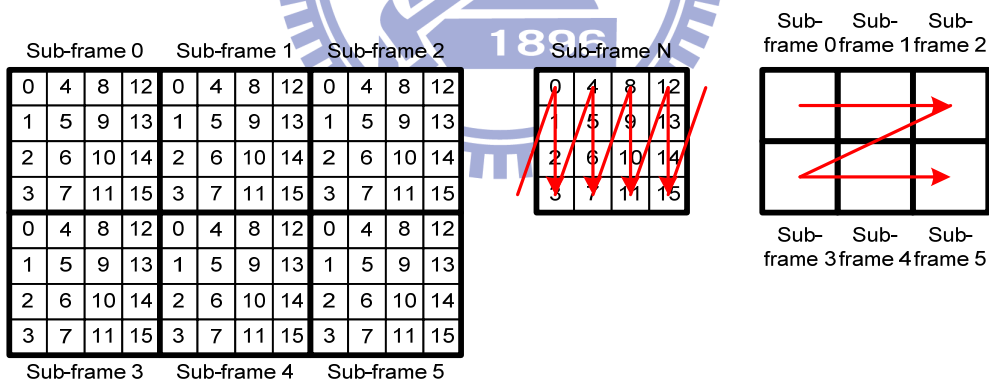


Figure 3-21 DOT image pixel transmission sequence

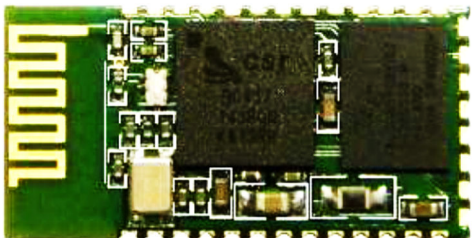
In the case when lossless data compression is enabled for EEG/ICA, ECG or DOT, the output data format is the same, except that the fixed-precision output sample units become variable length comprising Golomb-Rice coded prediction errors (in the form of *unary*(quotient), *delimiter*, *binary*(remainder)) as previously illustrated in Figure 3-9c. In this case, the amount of data padding for DOT and HRV data becomes conditional to whether or not the last pixel or coefficient pair completely fills up the 40-bit data packet, respectively.

### 3.3.1.5 Wireless Data Communication Interface and UART

Among the wireless data communication protocols commercially available today, Bluetooth and Zigbee are two protocols specially designed for portable applications, featuring short communication ranges and low power consumption at reasonable data rates [41]. Table 3-10 shows a comparison between the Bluetooth and Zigbee protocols. Given that the intended application consists of only a few nodes communicating at a low data bandwidth, Zigbee appears to be the more attractive solution, since the nominal power consumption is much lower. However, none of today's consumer devices, i.e. smart phones, laptops, PDAs or tablets, support the Zigbee protocol. Since connectivity and accessibility are primary requirements, Bluetooth was the protocol chosen to serve as the wireless communication interface in our system. Figure 3-22a shows the commercial Bluetooth IC module employed for wireless data communication in the proposed system and its specifications are summarized in Figure 3-22b.

Table 3-10 Comparison between Bluetooth and Zigbee

Parameter	Bluetooth	ZigBee
Operating Frequency	2.4 GHz	2.4 GHz
Modulation Technique	Frequency Hopping Spread Spectrum	Direct Sequence Spread Spectrum
Transmission rate (kbit/s)	1000	250
Nominal Range	10 meters	10-100 meters
Typical network join time	3 seconds	30 milliseconds
Nominal TX Power	0 to 10 dBm	-25 to 0 dBm



(a)

<b>Standard</b>	Bluetooth specification version 2.1+EDR
<b>Profile</b>	Serial port profile (SPP)
<b>Frequency</b>	2.402GHz ~ 2.480GHz unlicensed ISM band
<b>Modulation</b>	GFSK (1Mbps); $\Pi/4$ -DQPSK (2Mbps); 8-DPSK (3Mbps)
<b>UART I/F</b>	TxD, RxD, CTS, and RTS 1.2 to 921.6kbps, 8 - N/E/O - 1/2
<b>RF Spec.</b>	TX: Class 2, Max $4 \pm 1$ dBm RX: -80 dBm typical sensitivity
<b>Range</b>	Up to 10m
<b>Power</b>	3.3V, up to 60mA
<b>Dimensions</b>	27mm x 13mm x 2mm

(b)

Figure 3-22 (a) Bluetooth transceiver module from Hotlife; (b) Specifications summary of employed Bluetooth transceiver module

In order for the proposed chip to connect to the Bluetooth module, a UART communication interface is implemented within the chip. Aside from the advantage of having a reduced number of I/O pins at the chip interface, by employing a UART interface, the usability of the chip is further enhanced such that its connection to many other devices that support UART (e.g. PC through RS232 COM port, Zigbee module w/ UART interface, and so on) becomes possible. Table 3-11 summarizes the specifications of the UART module implemented in the chip while Table 3-12 shows the maximum bandwidth usage calculations in the case of full mode (system mode = 8'b1111\_1111) operation. Note that the effective baud rate is only 80% of the gross baud rate since the UART start and stop bits take up 20% of the UART frame.

Table 3-11 Specifications of the on-chip UART module

Parameter	Description
Baud rate	115,200
Data bits	8
Parity bits	None
Stop bits	1
Pins	TX, RX
Flow control	None

Table 3-12 Calculation of maximum UART bandwidth usage

Data	Calculation	Bandwidth (bps)
ICA	$16 \text{ bits/sample} \times 128 \text{ samples/s-ch} \times 4 \text{ ch}$	8,192
ECG	$10 \text{ bits/sample} \times 256 \text{ samples/s-ch} \times 3 \text{ ch}$	7,680
HRV	$32 \text{ bits/coefficient} \times 256 \text{ coefficients/60 s}$	136.53
DOT	$20 \text{ bits/pixel} \times 96 \text{ pixels/s}$	1920
Header (approx.)	$(8,192 + 7,680 + 136.53 + 1,920) \times \frac{2}{38}$	943.6
Total Used	--	18,872.13
Total Available	$115,200 \text{ bps} \times \frac{8}{10}$	92,160
<b>Percent Used</b>	--	<b>20.4%</b>

### 3.3.2 Core Module Design

In this section, the design of the main biomedical DSP IP core containing the ICA, HRV, DOT and lossless data compression engines is described in detail. Figure 3-23 shows the architecture of the core biomedical DSP IP and its relation to the rest of the chip. The main biomedical DSP core IP, conditional to the configured system mode, takes in EEG, ECG and/or fNIR data, processes them and outputs various combinations of 1) raw EEG signals; 2) independent component signals; 3) raw ECG signals; 4) HRV frequency spectrum coefficients and 5) DOT image pixels, packaged according to the 40-bit output format specified in Section 3.3.1.4. In the following subsections, top-level data flow issues will be discussed first, after which the detailed design of the ICA, HRV, DOT and COMP engines will be presented.

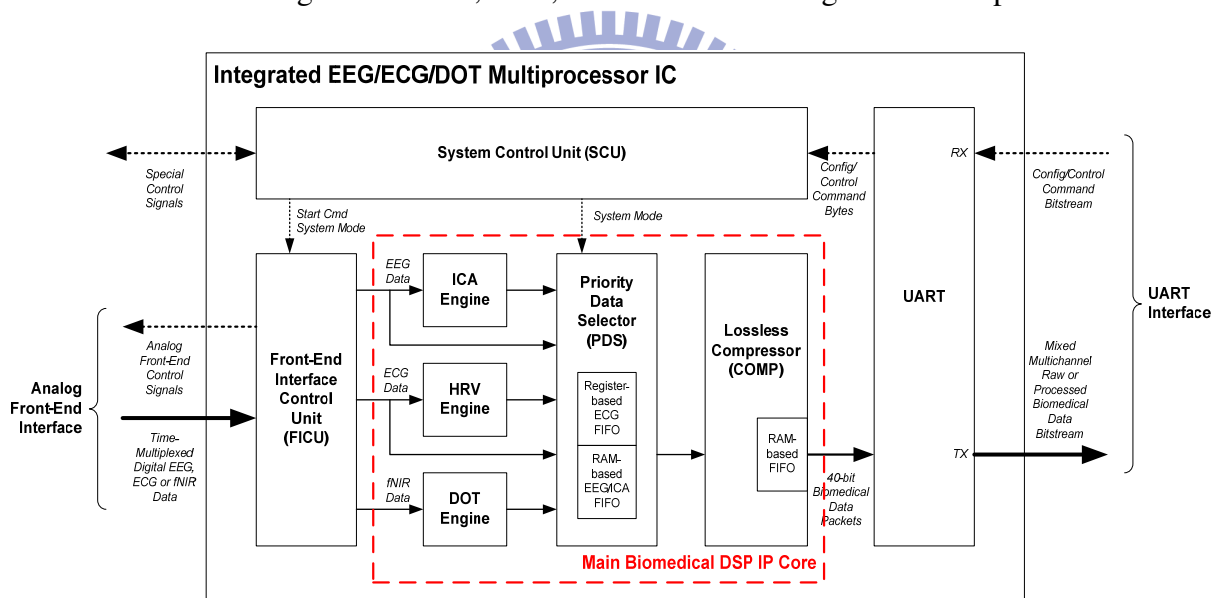


Figure 3-23 Architecture of the main biomedical DSP IP core within the chip

#### 3.3.2.1 Priority Data Selector (PDS)

The ICA, HRV and DOT engines process input EEG, ECG and fNIR data as described in Section 3.2. Although the timing of the input data is guaranteed to be non-simultaneous, the processing delays of the corresponding engines vary causing the possibility of simultaneous data at the output side. Since the lossless compression module accepts only one biomedical data sample at a time, a priority data selector (PDS) module is employed to



perform time-multiplexing of the different types of biomedical data into a single data stream. Although it's possible to schedule the input data timing such that the outputs are ensured to arrive at different times, for robustness of the design we choose to support arbitration based on priority instead. The architecture of the priority data selector module, including the implemented priority assignment, is shown in Figure 3-24.

Each biomedical data sample sent to the lossless data compressor is annotated with COMP info parameters by the PDS. These info parameters indicate the sample precision (COMP\_SP) and type (CH\_SEL) of the biomedical data and also whether the sample will be compressed or not (BYPASS). Given these information, the lossless compression module conditionally compresses and packages the received biomedical data samples accordingly into 40-bit packets as described in Section 3.3.1.4.

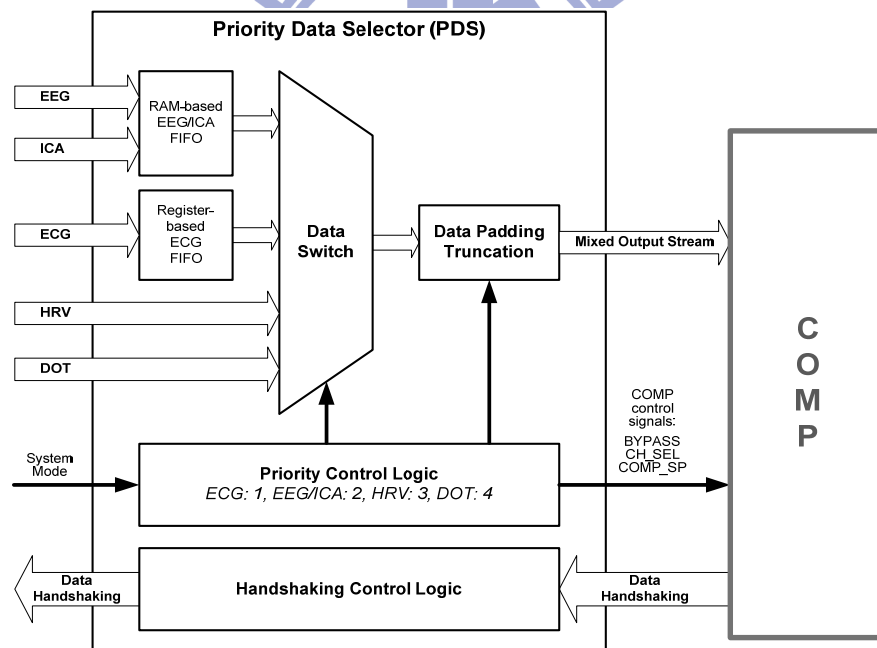


Figure 3-24 Architecture of PDS and priority assignment

### 3.3.2.2 Inter-module Handshaking Mechanism

Although it has been shown in Section 3.3.1.5 that the UART bandwidth is enough to handle the maximum output data rate during full mode operation, the instantaneous data rate at the input of the UART can, during some short segments in time, actually exceed the

maximum UART baud rate. This is in part due to the process of sample windowing employed by the signal processing algorithms which causes bursty data timing at the output side, and is further exacerbated by the faster clock used by the processing engines. Therefore, to address this issue, a three-stage inter-module handshaking mechanism is employed such that upstream modules can be instructed to stall and hold their output until the UART is ready again to receive new data. Figure 3-25 illustrates the three-stage handshaking mechanism wherein the backward READY handshaking signal is highlighted in red. Finally, to prevent hard stalling of the modules and also to serve as allowance over possible data overflows, sample and stream buffers are employed in the PDS and COMP modules respectively.

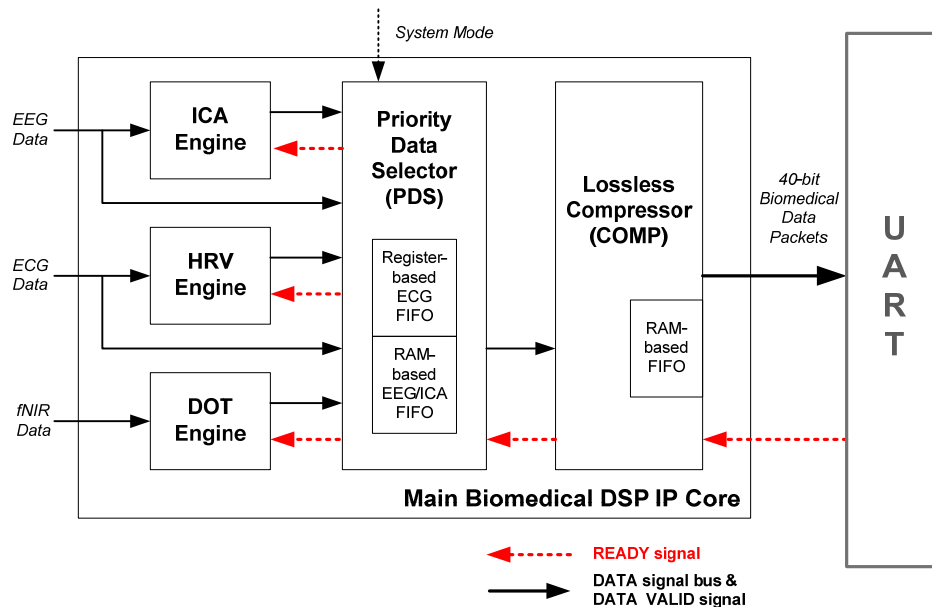


Figure 3-25 Three-stage handshaking mechanism

### 3.3.2.3 Diffuse Optical Tomography (DOT) Engine

The main design problem in the development of the DOT engine is the determination of the inverse solution matrix. As discussed in Section 3.2.3, indeed when the LED source-detector pair geometry, target medium and depth parameters are fixed, the coefficients of the inverse matrix are constant. Therefore, the inverse matrix was derived empirically based on an actual experimental set up and then implemented as a look-up table using a ROM. The distances between the LED source-detector pairs are illustrated in Figure 3-26, whereas the

absorption coefficient, scattering factor and target depth are fixed at  $\mu_a = 0.05\text{cm}^{-1}$ ,  $\nu = 10\text{cm}^{-1}$  and  $d = 0.5\text{cm}$  respectively in the experiments. To reduce the complexity of the forward model, a sub-frame image reconstruction technique [50] was employed such that the  $8 \times 12$  DOT image is broken into smaller  $4 \times 4$  sub images each reconstructed based on its 4 nearby fNIR sensor readings (surrounding an LED source).

Figure 3-27 shows the hardware architecture of the DOT engine. Input data in the form of fNIR light intensity values are received and initially stored in a shift register buffer. After receiving a complete input data unit of 24 fNIR data values, the main state machine shifts to a calculate and save state. During this state, the data access sequencer logic controls the memory accesses and time-sharing of the singular MAC unit to effect a matrix multiplication operation, wherein the intermediate and final results are stored in the output pixel buffer. Finally, the calculated image pixels forming the  $8 \times 12$  DOT image are read from the output pixel buffer RAM and sent to the PDS unit. The pixel output sequence is as shown previously in Figure 3-21.

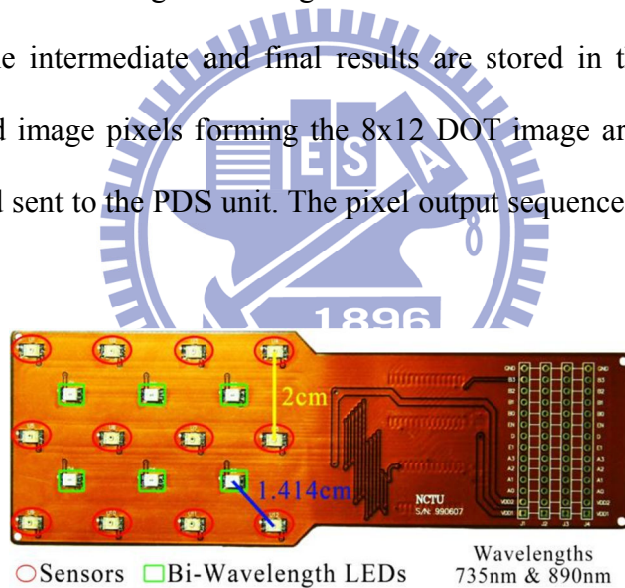


Figure 3-26 DOT sensor array board

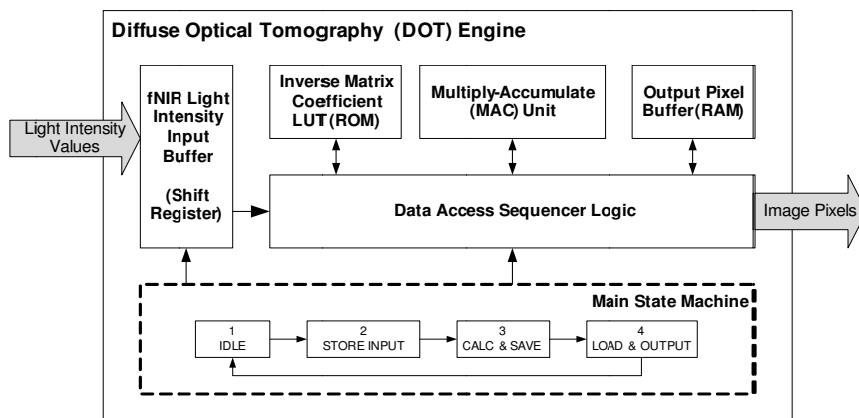


Figure 3-27 Hardware architecture of the DOT engine

### 3.3.2.4 Independent Component Analysis (ICA) Engine

The independent component analysis (ICA) engine is mostly a straightforward implementation of the Infomax ICA algorithm outlined in Section 3.2.1. The hardware architecture of the ICA engine, shown in Figure 3-28, comprises 1) an input buffering and preprocessing unit (STAGE1) that calculates the covariance matrix and centers the input EEG data; 2) a whitening unit (WU) which calculates the whitening transformation matrix; 3) an ICA training unit (TU) which performs the unmixing weight training; and 4) an ICA computation unit (CU) that calculates the whitened unmixing weight matrix and outputs the final estimated independent source signals.

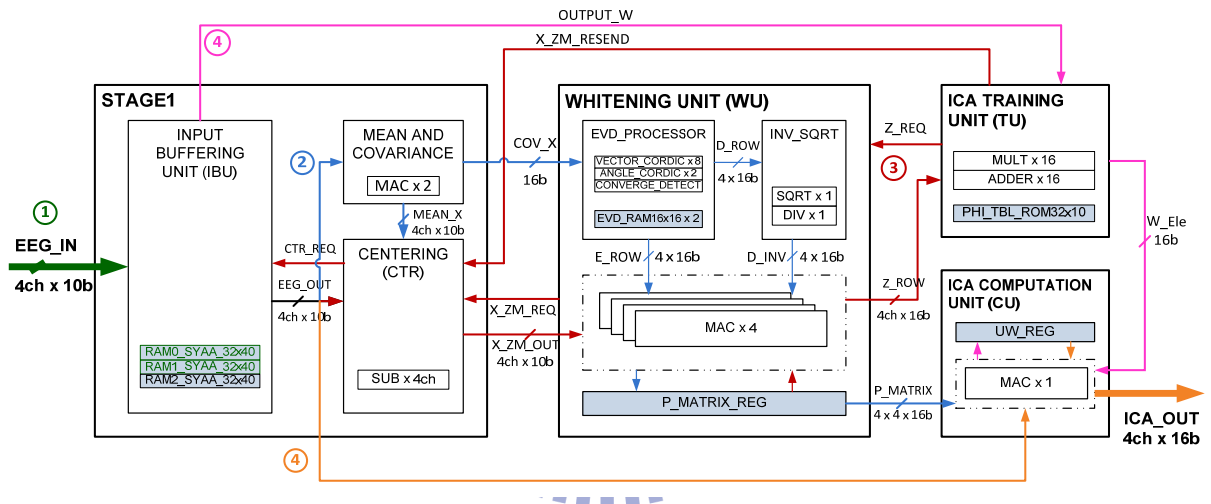


Figure 3-28 Architecture and sequenced operation of the ICA engine

The top operation of the ICA engine follows the described algorithm flow in Section 3.2.1 and is detailed as follows:

1. STAGE1 receives sparsely-timed EEG data and the input buffering unit (IBU) stores them into an interleaved SRAM array for later processing. (sequence in green)
2. Once a complete window of 64 by 4ch EEG samples has been received, the EEG samples are read out from the memory and the channel means are calculated. The mean values are then 1) immediately used to calculate the covariance matrix; and 2) saved in a register array for later use during the centering operation. The eigenvalue value decomposition (EVD) unit of the whitening unit then iteratively diagonalizes the

covariance matrix, and upon convergence outputs the final estimated eigenvalue  $\mathbf{D}$  and eigenvector  $\mathbf{E}$  matrices. The inverse square root of  $\mathbf{D}$  calculated by the INV\_SQRT unit is then multiplied together with  $\mathbf{E}$  and its transpose, and the resulting matrix  $\mathbf{P}$  is stored in a register array for use later on during Infomax ICA training and calculation. (sequence in light blue)

3. All this time, the ICA training unit (TU) had been waiting for whitened data to train on, asserting the request signal Z\_REQ. Once the whitening matrix  $\mathbf{P}$  has been calculated, the whitening unit issues a request (X\_ZM\_REQ) and STAGE1 starts outputting centered data (channel means are subtracted from the raw data read from IBU's SRAM). The whitening unit then transforms the centered data  $\mathbf{X}_{4 \times 1}$  to uncorrelated whitened data  $\mathbf{Z}_{4 \times 1} = \mathbf{P}_{4 \times 4} \mathbf{X}_{4 \times 1}$ , and forwards the result to the ICA training unit. The ICA TU calculates an estimate of the unmixing weight matrix  $\mathbf{W}$  based on a window of  $\mathbf{Z}_{4 \times 1}$ , if the change in  $\mathbf{W}$  is small enough, convergence is reached and training concludes; otherwise, the ICA TU will keep on requesting for the same window of  $\mathbf{Z}_{4 \times 64}$  data (through X\_ZM\_RESEND and X\_ZM\_REQ) up to 512 times or until  $\mathbf{W}$  convergence is achieved, whichever comes first. The final value of  $\mathbf{W}$  in this process serves as the best estimate of  $\mathbf{W}$  and is stored for 1) training the next window; and 2) calculating ICA\_OUT. (sequence in red; note how all of IBU, CTR, WU and TU are active concurrently)
4. In the beginning of the next window cycle when the  $\mathbf{P}$  and  $\mathbf{W}$  have not yet been calculated and overwritten, STAGE1 triggers ICA output calculation by issuing an OUTPUT\_W command for ICA TU to send to ICA CU the previously calculated best estimate of the unmixing weight matrix  $\mathbf{W}$ . Upon receiving  $\mathbf{W}$ , ICA CU is triggered to calculate  $\mathbf{UW} = \mathbf{WP}$  which is stored in UW\_REG upon completion (sequence in pink). Finally, STAGE1 sends  $\mathbf{X}$  and the ICA CU calculates the final  $\mathbf{ICA\_OUT} = \mathbf{WPX}$  (sequence in orange).

The above discussion only focuses on the top operation of the ICA engine, omitting the details on the data windowing scheme, eigenvalue and eigenvector calculation in the whitening unit, and the iterative operation of the Infomax ICA training unit. These mechanisms actually form bulk of the complexity of the ICA engine; therefore, in the following subsections, more details will be provided regarding these topics, and special highlight will be shed on the hardware implementation issues associated with each of them.

#### 3.3.2.4.1 Data windowing scheme

A basic but important parameter to consider in the implementation of real-time ICA is the input data length (number of multi-channel sampling events) on which one full run of the algorithm is operated upon. This length of data or “window size (WS)” is illustrated in Figure 3-29. The input data is broken down into data segments each of length  $N$ , denoted as  $x_1, x_2, \dots, x_n$ , and data windows are specified by combining a number of consecutive data segments, denoted as  $w_1, w_2, \dots, w_n$ . In order to achieve tractable performance and complexity in the proposed design, only fixed window sizes are considered (the window size is not allowed to vary during operation) while overlapped windows are introduced in order to improve ICA training and to facilitate pipelined operation.

A window size of 512 with 50% overlap ( $N=256, WS=2N, OV=50\%$ ) was initially chosen since it was able to achieve a good ICA performance of 0.9208 correlation coefficient using super-gaussian random pattern sets. However, because the tape out shuttle area budget was severely limited, this design was not feasible to be released due to its large area attributed to high memory requirements. Therefore with the intention of minimizing memory size at acceptable performance loss, a performance trade-off analysis of the ICA algorithm against various combinations of window size (WS) and overlap (OV) parameters was performed using MATLAB. Based on the results shown in Table 3-13, a window size of 64 with 50% window overlap was chosen for the final design since it was able to achieve an acceptable 0.84 correlation using only a minimal 3.84Kb of memory.

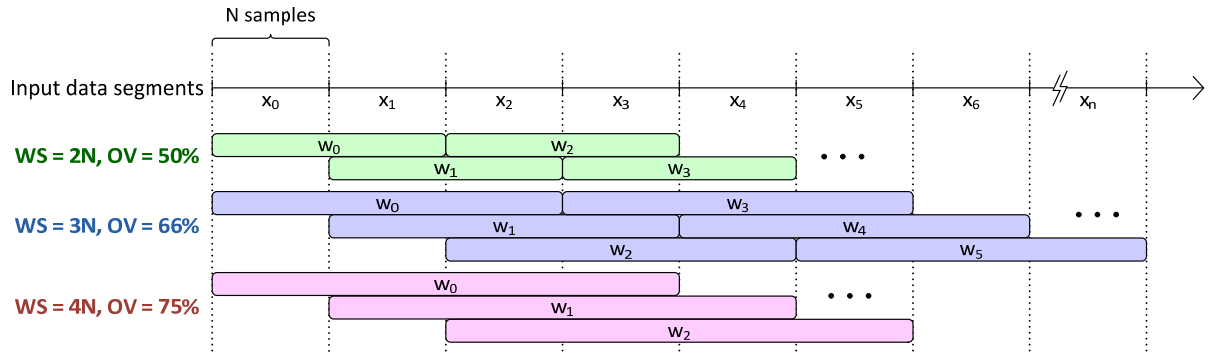


Figure 3-29 Illustration of the data windowing concept

Table 3-13 ICA correlation performance vs. window size and window overlap

Window Size (WS)	Window Overlap (OV)	N	Average Correlation	Memory Size(Kb)
512	256 (50%)	256	0.9208	30.72
128	112 (87.5%)	16	0.8307	5.76
	96 (75%)	32	0.8336	6.4
	64 (50%)	64	0.8334	7.68
64	32 (50%)	32	0.8401	3.84

Figure 3-30 illustrates the implications of the chosen design parameters ( $N=32$ ,  $WS=2N$ ,  $OV=50\%$ ) on the pipelined operation schedule of the ICA engine. Three separate interleaved SRAMs each of size  $N$  are employed and allocated such that memory writes performed during storing of  $x_2$  (RAM2) do not conflict with the memory reading of  $x_0$  and  $x_1$  (RAM0 and RAM1) during training of window  $w_0$ . Because of the 50% window overlap, a new unmixing weight matrix  $w_n$  is available for every new  $x_n$ . Therefore, under the assumption that the unmixing weight matrix does not change too much from window to window, the unmixing of  $x_n$  (ICA computation) is performed using the unmixing weight matrix  $w_{n-1}$ , which is based on the training of window  $w_{n-1}$  formed by  $x_n$  and  $x_{n-1}$ .

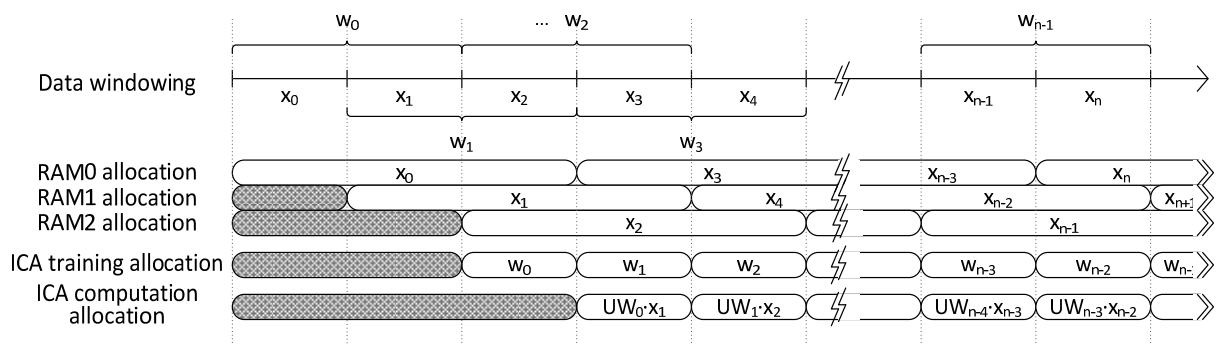


Figure 3-30 Data windowing scheme (50% overlap) and operation schedule

#### 3.3.2.4.2 Eigenvalue and eigenvector calculation in the whitening unit

The primary function of the whitening unit is the determination of the whitening transformation matrix  $\mathbf{P}$  expressed in Equation (3-4). Its calculation requires the matrix factorization of the covariance matrix of  $\mathbf{X}$  into the form  $\mathbf{E} \mathbf{D} \mathbf{E}^T$ , where each of diagonal matrix  $\mathbf{D}$  and orthogonal matrix  $\mathbf{E}$  comprises the covariance matrix's eigenvalues and eigenvectors respectively. The diagonalization of the real, symmetric, positive semi-definite covariance matrix  $\mathbf{X}_{\text{cov}}$  into  $\mathbf{E} \mathbf{D} \mathbf{E}^T$ , known more formally in linear algebra as eigenvalue decomposition or EVD, is specifically performed by the EVD\_PROCESSOR shown in Figure 3-28. It is a non-trivial mathematical problem whose solution takes up most of the computational burden handled by the whitening unit and is arguably the most interesting and algorithmically colorful and complex numerical method in the ICA engine. The details of the EVD computation, in terms of both algorithm and hardware architecture, are described here instead of the ICA algorithm discussion in 3.2.1, as the method's rationale is more of a consequence of hardware implementation rather than ICA itself.

#### *The Jacobi Eigenvalue Decomposition Algorithm - Introduction*

Eigenvalue decomposition is an old mathematical problem and many methods for its solution have been proposed in the literature. Among these methods, the two most popular are the Jacobi eigenvalue algorithm [51] and the QR algorithm [52][53]. Although the QR method has been widely celebrated for its superior computational speed (presumably on general purpose computers), it has proved to be numerically less stable and less accurate than the Jacobi method [54][55][56]. On the other hand, the Jacobi algorithm possesses, in addition, desirable characteristics of simplicity, elegance, and regularity [57] making it extremely well-suited to efficient parallel VLSI implementations [58]. Indeed, while hardware implementation of the QR method has proven to be problematic due to the division, square root and inverse square root operations performed in the algorithm [59][60], rotation matrix multiplications in the parallel Jacobi method [61] can be efficiently handled by CORDIC



(COordinate Rotation DIgital Computer) arithmetic [62] such that even multiplications can be avoided altogether [63][64]. Due to its numerous attractive features, especially ones concerning hardware implementation, the Jacobi method was chosen for solving the EVD problem in the developed ICA engine. In the following subsection the Jacobi eigenvalue and the CORDIC algorithms are described in more detail.

#### *The Jacobi Eigenvalue Decomposition Algorithm – Basic Idea*

The basic idea behind Jacobi’s method is to systematically reduce to zero the quantity

$$\text{off}(\mathbf{A}) = \sqrt{\sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}^2} \quad (3-17)$$

i.e., the “norm” of the off-diagonal elements of the input symmetric n-by-n matrix  $\mathbf{A}$ . The input matrix  $\mathbf{A}$  is iteratively updated according to

$$\mathbf{A}_{k+1} = \mathbf{J}_k^T \mathbf{A}_k \mathbf{J}_k, \quad (3-18)$$

where  $\mathbf{J}$  is orthogonal, such that the new  $\mathbf{A}$  is more diagonal than the previous one in the sense of (3-17). When  $\text{off}(\mathbf{A}_{k+1})$  goes below a predefined threshold, say after  $K$  iterations, the algorithm is said to have converged; and the final

$$\mathbf{A}_K = \mathbf{J}_{K-1}^T \mathbf{J}_{K-2}^T \dots \mathbf{J}_2^T \mathbf{J}_1^T \mathbf{J}_0^T \mathbf{A}_0 \mathbf{J}_0 \mathbf{J}_1 \mathbf{J}_2 \dots \mathbf{J}_{K-2} \mathbf{J}_{K-1} \quad (3-19)$$

is a diagonal matrix  $\mathbf{D}$  containing the eigenvalues of  $\mathbf{A}_0$ . Furthermore, recalling that  $\mathbf{J}$  is orthogonal, then  $\mathbf{J}^T = \mathbf{J}^{-1}$  and  $\mathbf{J}\mathbf{J}^T = \mathbf{J}^T\mathbf{J} = \mathbf{I}$ , and manipulating both sides of (3-19) gives

$$\mathbf{J}_0 \mathbf{J}_1 \dots \mathbf{J}_{K-1} \mathbf{A}_K \mathbf{J}_{K-1}^T \dots \mathbf{J}_1^T \mathbf{J}_0^T = \mathbf{J}_0 \mathbf{J}_1 \dots \mathbf{J}_{K-1} \mathbf{J}_{K-1}^T \dots \mathbf{J}_1^T \mathbf{J}_0^T \mathbf{A}_0 \mathbf{J}_0 \mathbf{J}_1 \dots \mathbf{J}_{K-1} \mathbf{J}_{K-1}^T \dots \mathbf{J}_1^T \mathbf{J}_0^T \quad (3-20)$$

$$\mathbf{E} \mathbf{D} \mathbf{E}^T = \mathbf{A}_0 \quad (3-21)$$

where matrix  $\mathbf{E} = \mathbf{J}_0 \mathbf{J}_1 \dots \mathbf{J}_{K-1}$  is orthogonal and whose rows (or columns) are the eigenvectors of  $\mathbf{A}_0$ , thus solving the original eigenvalue decomposition problem.

By now it should be apparent that the key to Jacobi’s method lies in the proper design of matrix  $\mathbf{J}$  to produce the effects described above. It follows then that the specifications of  $\mathbf{J}$

must at least include the following: 1) it should be orthogonal; 2) for manageability, it should be able to selectively “annihilate” specific off-diagonal elements only; 3) it should be able to cause a net effect of  $\text{off}(\mathbf{A}_{k+1}) < \text{off}(\mathbf{A}_k)$  after each iteration; and 4) it should have the ability to converge.

To satisfy these requirements, Jacobi defined  $\mathbf{J}$  as

$$\mathbf{J}(p, q, \theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos \theta & \cdots & \sin \theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin \theta & \cdots & \cos \theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} p \\ q \end{matrix}, \quad p < q \quad (3-22)$$

called the Jacobi rotation matrix, which is simply an identity matrix whose elements at (p,p), (q,q), (p,q) and (q,p) are replaced with sine and cosine trigonometric functions as shown above. By using trigonometric identities, it can be easily verified that  $\mathbf{J}$  is orthogonal, satisfying the first requirement. To illustrate the second requirement, Figure 3-31 shows an expansion of (3-18) using the Jacobi rotation matrix defined in (3-22). The left (right) rotations in blue (red) simply “remixes” or “rotates” pairs of column (row) elements  $(a_{pi}^k, a_{qi}^k)$  to  $(a_{pi}^{k+1}, a_{qi}^{k+1})$  ( $(a_{ip}^k, a_{iq}^k)$  to  $(a_{ip}^{k+1}, a_{iq}^{k+1})$ ). It becomes obvious then that the only opportunity here is to “rotate” the energy of the off-diagonal elements  $a_{pq}$  and  $a_{qp}$  into the diagonal elements  $a_{pp}$  and  $a_{qq}$ , since the remaining off-diagonal elements (red only or blue only) have no paired diagonal elements to transfer their energy to. It becomes clear then that the rotation angle  $\theta$  must be chosen based on  $a_{pp}$ ,  $a_{qq}$ ,  $a_{pq}$  and  $a_{qp}$  in order to achieve the desired effect.

$$\begin{bmatrix} a_{11}^{k+1} & \cdots & a_{1p}^{k+1} & \cdots & a_{1q}^{k+1} & \cdots & a_{1n}^{k+1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{p1}^{k+1} & \cdots & a_{pp}^{k+1} & \cdots & a_{pq}^{k+1} & \cdots & a_{pn}^{k+1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{q1}^{k+1} & \cdots & a_{qp}^{k+1} & \cdots & a_{qq}^{k+1} & \cdots & a_{qn}^{k+1} \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n1}^{k+1} & \cdots & a_{np}^{k+1} & \cdots & a_{nq}^{k+1} & \cdots & a_{nn}^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos \theta & \cdots & -\sin \theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin \theta & \cdots & \cos \theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_{11}^k & \cdots & a_{1p}^k & \cdots & a_{1q}^k & \cdots & a_{1n}^k \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{p1}^k & \cdots & a_{pp}^k & \cdots & a_{pq}^k & \cdots & a_{pn}^k \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{q1}^k & \cdots & a_{qp}^k & \cdots & a_{qq}^k & \cdots & a_{qn}^k \\ \vdots & & \vdots & & \vdots & & \vdots \\ a_{n1}^k & \cdots & a_{np}^k & \cdots & a_{nq}^k & \cdots & a_{nn}^k \end{bmatrix} \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos \theta & \cdots & \sin \theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin \theta & \cdots & \cos \theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

Figure 3-31 Illustration of matrix  $\mathbf{A}$  update using two-sided Jacobi plane rotations

Evaluating the right hand side of Figure 3-31 for  $a_{pq}^{k+1}$  and  $a_{qp}^{k+1}$  and noting that  $a_{qp}^k = a_{pq}^k$  since  $\mathbf{A}_k$  is symmetric, gives

$$\begin{aligned}
a_{qp}^{k+1} = a_{pq}^{k+1} &= a_{pp}^k \sin\theta \cos\theta + a_{pq}^k \cos\theta \cos\theta - a_{pq}^k \sin\theta \sin\theta - a_{qq}^k \sin\theta \cos\theta \\
&= (a_{pp}^k - a_{qq}^k) \sin\theta \cos\theta + a_{pq}^k \cos^2\theta - a_{pq}^k \sin^2\theta \\
&= (a_{pp}^k - a_{qq}^k) \sin\theta \cos\theta + a_{pq}^k (\cos^2\theta - \sin^2\theta) \\
&= (a_{pp}^k - a_{qq}^k) \tan\theta + a_{pq}^k (1 - \tan^2\theta)
\end{aligned} \tag{3-23}$$

Since the off-diagonal elements in the updated matrix must be 0, setting  $a_{pq}^{k+1} = a_{qp}^{k+1} = 0$  gives

$$\begin{aligned}
\frac{2 \tan\theta}{1 - \tan^2\theta} &= \frac{2a_{pq}^k}{a_{qq}^k - a_{pp}^k} \\
\tan 2\theta &= \frac{2a_{pq}^k}{a_{qq}^k - a_{pp}^k} \\
\theta &= \frac{1}{2} \tan^{-1} \left( \frac{2a_{pq}^k}{a_{qq}^k - a_{pp}^k} \right)
\end{aligned} \tag{3-24}$$

Using the rotation angle  $\theta$  “annihilates”  $a_{pq}^{k+1}$  and  $a_{qp}^{k+1}$  to zero, and their energy is transferred to the diagonal elements. As for the rest of the rotation pairs (in blue only or red only), it can be shown that each pair’s total norm remains constant, i.e.,  $(a_{ij}^{k+1})^2 + (a_{mn}^{k+1})^2 = (a_{ij}^k)^2 + (a_{mn}^k)^2$ , such that on the global perspective,  $\text{off}(\mathbf{A}_{k+1})^2 - \text{off}(\mathbf{A}_k)^2 = 0^2 - 2(a_{pq}^k)^2$  or

$$\text{off}(\mathbf{A}_{k+1})^2 = \text{off}(\mathbf{A}_k)^2 - (a_{pq}^k)^2 \tag{3-25}$$

satisfying the third requirement.

The remaining question now is how to choose the indices (p,q) in each iteration. From the standpoint of maximizing the reduction of  $\text{off}(\mathbf{A}_k)$  in (3-25), it makes sense to choose (p,q) so that  $(a_{pq}^k)^2$  is maximal. Indeed, this is the strategy originally proposed by Jacobi in his work [51], where a proof of convergence is also given. However, the drawback of this method is that sweeping the whole  $n$ -by- $n$  matrix just to search for the largest off-diagonal element at every iteration is a costly operation. This problem is overcome by fixing the sequence of (p,q) locations to be updated in advance. Common schemes are the cyclic-by-row and the cyclic-

by-column procedures where the pair  $(p,q)$  is chosen in a row-by-row, or column-by-column fashion respectively [65][66]. Any such sequence, called a *sweep*, must cover all possible pairs of  $(p,q)$  and requires  $N = n(n-1)/2$  iterations. For example, the row-cyclic strategy uses the following sequence of update locations  $(p,q)$ :

$$\begin{aligned}
 &(1,2) \rightarrow (1,3) \rightarrow (1,4) \rightarrow \dots \rightarrow (1,n-1) \rightarrow (1,n) \rightarrow \\
 &(2,3) \rightarrow (2,4) \rightarrow (2,5) \rightarrow \dots \rightarrow (2,n) \rightarrow \\
 &\dots \\
 &(n-2,n-1) \rightarrow (n-2,n) \rightarrow \\
 &(n-1,n)
 \end{aligned}$$

By foregoing the off-diagonal searching procedure, the cyclic Jacobi executes considerably faster than Jacobi's original method.

While not immediately apparent in Jacobi's original method due to the conditional off-diagonal search, the more regular cyclic version is able to expose more readily the inherent parallelism present in Jacobi's method. From Figure 3-31, it can be seen that each right rotation (matrix multiplication in red) involves only columns  $p$  and  $q$ . Thus, any other right rotation  $(p',q')$  where  $p' \neq p$  and  $q' \neq q$  is a totally separate problem. For example, given  $n = 8$ , a possible "non-conflicting" set of rotation pairs could be  $\{(1,2),(3,4),(5,6),(7,8)\}$  such that all four can be performed in parallel. After all the right rotations are done, the left rotations can all be performed in parallel as well.

The final question would then be how to group together the remaining  $(p,q)$  pairs in the sweep while also maintaining the maximum  $n/2$ - way parallelism as shown above. A practical approach to the problem is to visualize a chess tournament with  $n$  players in which everybody must play everybody else exactly once; and, to minimize the total duration of the tournament (like a sweep), matches are conducted in parallel as much as possible. A simple method for generating rotation sets covering a full sweep is presented in [61] and a mnemonic illustration of this process is given in Figure 3-32.

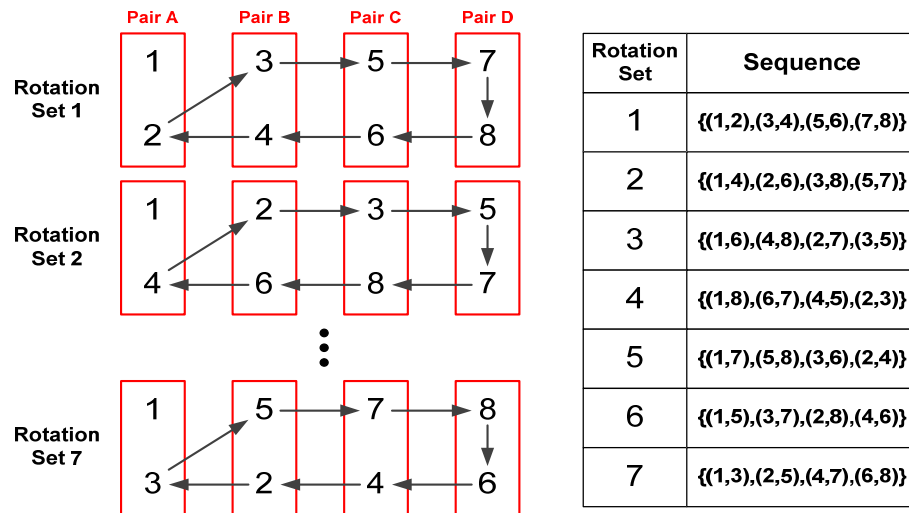


Figure 3-32 Illustration of parallel Jacobi rotation set sequence

### The CORDIC (COordinate Rotation DIgital Computer) Algorithm

Based on the discussion above, hardware implementation of the Jacobi EVD algorithm presents some challenges, particularly in the following sub operations: 1) squaring and square root functions in the calculation of  $\text{off}(A)$  during convergence check, as in (3-17); 2) division and arctangent operations for calculating  $\theta$  in (3-24); and 3) numerous matrix rotation multiplications during iterative updates, involving sine and cosine elements as shown in Figure 3-31. Straightforward evaluation of these functions can prove costly in hardware; therefore alternative approaches are investigated in order to achieve efficient implementation. The square and square root operations in 1) are avoided by simplifying the condition for convergence to  $\max(|a_{ij}|, i \neq j) < \text{threshold}$ . On the other hand, explicit division, arctangent and matrix rotation operations associated with 2) and 3) are avoided altogether through the use of CORDIC arithmetic, a powerful algorithm capable of evaluating these functions using only simple add, shift and table look-up operations.

CORDIC stands for “COordinate Rotation DIgital Computer” and is a simple yet very powerful computational algorithm invented by Jack E. Volder in 1959 [62]. Using only shift-add and look-up operations, it has the ability to evaluate a wide variety of mathematical computational tasks including trigonometric, hyperbolic, exponential and logarithmic

functions, real and complex multiplication, division, square-root, and many others, thus making it indispensable in solving higher level mathematical problems such as linear systems, eigenvalue estimation, singular value decomposition, QR factorization and so on. The CORDIC has been utilized for applications in diverse areas such as signal and image processing, communication systems, robotics and 3D graphics apart from general scientific and technical computation.

The CORDIC algorithm is based on the 2D geometric rotation transformation wherein a vector in the Cartesian plane is rotated counterclockwise by an angle  $\theta$ , as shown in Figure 3-33a. The relationship between the original vector  $(x,y)$  and the rotated vector  $(x',y')$ , as constrained by  $\theta$ , can be derived using elementary trigonometry and is shown in (3-26).

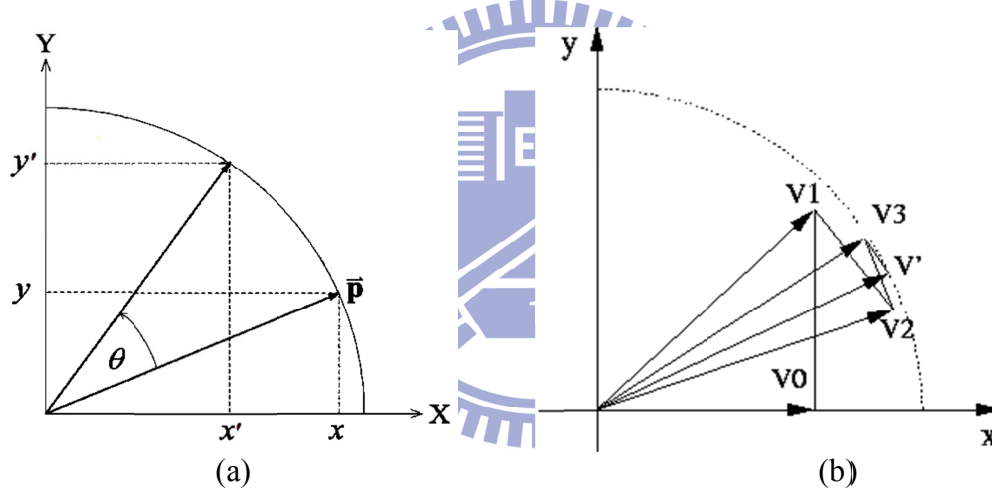


Figure 3-33 Rotation of vector in a Cartesian plane

$$\begin{aligned} x' &= x \cos\theta - y \sin\theta \\ y' &= x \sin\theta + y \cos\theta \end{aligned} \quad (3-26)$$

Instead of finding the solution of (3-26) in one evaluation, the CORDIC algorithm breaks down the problem into a series of smaller rotations, such that each incremental rotation moves the system closer to the correct solution as in Figure 3-33b. The rotation angle  $\theta$  is decomposed into component rotations  $\alpha_i = \tan^{-1}(2^{-i})$  such that  $\theta = \sum_{i=0}^{n-1} d_i \alpha_i$ , where  $d_i = \pm 1$  specifying the direction of rotation. These fixed component rotation angles  $\alpha_i$  are shown in Table 3-14. Any angle  $\theta$  can then be uniquely represented as a sequence of rotation directions  $\{d_0, d_1, d_2, \dots, d_{n-1}\}$  where  $n$  is the total number of iterations.

The advantage of this scheme becomes apparent when the cosine terms in (3-26) are factored out and the equations rewritten in incremental form:

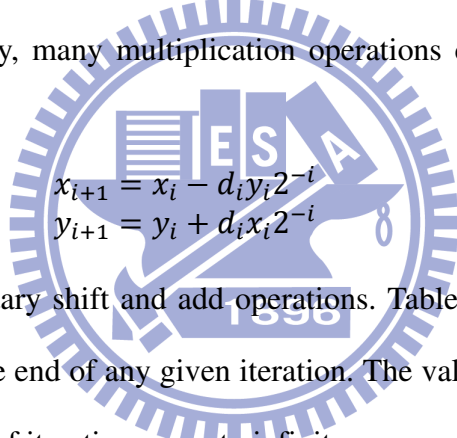
$$\begin{aligned} x_{i+1} &= \cos(d_i \alpha_i) (x_i - y_i \tan(d_i \alpha_i)) \\ y_{i+1} &= \cos(d_i \alpha_i) (y_i + x_i \tan(d_i \alpha_i)) \end{aligned} \quad (3-27)$$

and substituting  $\tan(\alpha_i) = 2^{-i}$ ,

$$\begin{aligned} x_{i+1} &= \cos(\alpha_i) (x_i - d_i y_i 2^{-i}) \\ y_{i+1} &= \cos(\alpha_i) (y_i + d_i x_i 2^{-i}) \end{aligned} \quad (3-28)$$

since  $\alpha_i < \pi/2$  and  $\cos()$  and  $\tan()$  are even and odd functions respectively.

However, based on (3-28),  $\cos(\alpha_{i-1})$  can be factored out from the right product term  $x_i - d_i y_i 2^{-i}$ , and then  $\cos(\alpha_{i-2})$  and so on. Thus, instead of multiplying by  $\cos(\alpha_i)$  in every iteration, an accumulated scaling product  $K = \prod_{i=0}^n \cos(\alpha_i)$  is defined, and applied only once at the final output. This way, many multiplication operations can be saved and the update iterations simplified to



$$\begin{aligned} x_{i+1} &= x_i - d_i y_i 2^{-i} \\ y_{i+1} &= y_i + d_i x_i 2^{-i} \end{aligned} \quad (3-29)$$

which only requires elementary shift and add operations. Table 3-14 shows the value of the accumulated product  $K$  at the end of any given iteration. The value of  $K$  actually converges to 0.607252935 as the number of iterations goes to infinity.

Table 3-14 Constant parameters in the CORDIC algorithm

$i$	$2^{-i}$	$\alpha_i = \tan^{-1}(2^{-i})$	$\cos(\alpha_i)$	$K = \prod_{i=0}^n \cos(\alpha_i)$
0	1.000000000000	<b>0.785398163397</b>	0.70710678118655	0.70710678118655
1	0.500000000000	<b>0.463647609001</b>	0.89442719099992	0.6324553203368
2	0.250000000000	<b>0.244978663127</b>	0.97014250014533	0.61357199107790
3	0.125000000000	<b>0.124354994547</b>	0.99227787671367	0.60883391251775
4	0.062500000000	<b>0.062418809996</b>	0.99805257848289	0.60764825625617
5	0.031250000000	<b>0.031239833430</b>	0.99951207608708	0.60735177014130
6	0.015625000000	<b>0.015623728620</b>	0.99987795203470	0.60727764409353
7	0.007812500000	<b>0.007812341060</b>	0.99996948381879	0.60725911229889
8	0.003906250000	<b>0.003906230132</b>	0.99999237069278	0.60725447933256
9	0.001953125000	<b>0.001953122516</b>	0.99999809265682	0.60725332108988
10	0.000976562500	<b>0.000976562190</b>	0.99999952316318	0.60725303152913
11	0.000488281250	<b>0.000488281211</b>	0.99999988079073	0.60725295913895
12	0.000244140625	<b>0.000244140620</b>	0.99999997019768	0.60725294104140
13	0.000122070313	<b>0.000122070312</b>	0.99999999254942	0.60725293651701
14	0.000061035156	<b>0.000061035156</b>	0.99999999813735	0.60725293538591
15	0.000030517578	<b>0.000030517578</b>	0.99999999953434	0.60725293510314
16	0.000015258789	<b>0.000015258789</b>	0.99999999988359	0.60725293503245
17	0.000007629395	<b>0.000007629395</b>	0.99999999997090	0.60725293501477
18	0.000003814697	<b>0.000003814697</b>	0.99999999999272	0.60725293501035
19	0.000001907349	<b>0.000001907349</b>	0.99999999999818	<b>0.60725293500925</b>

In addition to calculating the Cartesian coordinates in (3-29), the accumulated angle must be taken note of as well:

$$\omega_{i+1} = \omega_i - d_i \alpha_i \quad (3-30)$$

The negative accumulation is just a matter of convention. In the typical vector rotation use case, the rotation angle  $\theta$  is assigned to  $\omega_0$ , and  $\omega_i$  is iteratively decremented until  $\omega_n = 0$ .

The CORDIC iterations in (3-29) and (3-30) can be used in two operating modes, namely *rotation mode* (RM) and *vectoring mode* (VM). In rotation mode, the desired information are the new coordinates  $(x_n, y_n)$  of an input vector  $(x_0, y_0)$  after being rotated by a specified angle  $\omega_0 = \theta$ . In this mode, the direction of each micro rotation  $d_i$  is determined by the sign of  $\omega_i$ ; if  $\omega_i$  is positive, then  $d_i = 1$  (counterclockwise) otherwise  $d_i = -1$  (clockwise). After  $n$  iterations,  $\omega_n$  becomes zero and  $(x_n, y_n)$  is multiplied by  $K$  to get the final output vector  $(x', y')$  as in (3-26). On the other hand, in vectoring mode, an input vector  $(x_0, y_0)$  is rotated onto the x-axis, such that  $y_n$  becomes zero and the magnitude and angle of the vector can be obtained from  $x_n$  and  $\omega_n$  respectively. The direction of each micro rotation is determined by the sign of  $y_i$ ; if  $y_i$  is positive, then  $d_i = -1$  (clockwise) otherwise  $d_i = 1$ . As before,  $x_n$  must be scaled by  $K$  to get the real magnitude. Table 3-15 shows the summary of the CORDIC algorithm, as well as its application in (3-24) and Figure 3-31.

Table 3-15 Summary of CORDIC algorithm and interfacing with Jacobi EVD algorithm;  
\* refers to  $\theta$  in (3-24)

	Rotation Mode (RM)	Vectoring Mode (VM)
Iterative system of equations	$x_{i+1} = x_i - d_i y_i 2^{-i}$ $y_{i+1} = y_i + d_i x_i 2^{-i}$ $\omega_{i+1} = \omega_i - d_i \alpha_i$	
Input	$x_0 = x = a_{pi}^k$ $y_0 = y = a_{qi}^k$ $\omega_0 = \theta^*$	$x_0 = x = a_{qq}^k - a_{pp}^k$ $y_0 = y = 2a_{pq}^k$ $\omega_0 = 0$
Operation description	Counterclockwise rotation of vector $(x_0, y_0)$ by $\omega_0$	Rotation of vector $(x_0, y_0)$ onto x-axis
Direction of micro rotation	$d_i = \begin{cases} -1, & \omega_i \leq 0 \\ 1, & \omega_i > 0 \end{cases}$	$d_i = \begin{cases} 1, & y_i \leq 0 \\ -1, & y_i > 0 \end{cases}$
Output result	$a_{pi}^{k+1} = x' = Kx_n = x_0 \cos \omega_0 - y_0 \sin \omega_0$ $a_{qi}^{k+1} = y' = Ky_n = x_0 \sin \omega_0 + y_0 \cos \omega_0$ $\omega_n = 0$	$Kx_n = \sqrt{x_0^2 + y_0^2}$ $2\theta^* = \omega_n = \tan^{-1}(y_0/x_0)$ $y_n = 0$



### Hardware Architecture of the CORDIC Engine

The hardware architecture of the CORDIC engine supporting both rotation (RM) and vectoring (VM) modes follows naturally from Table 3-15 and is shown in Figure 3-34. The engine is first configured to perform vectoring or rotation operation through the VM/RM pin. Once the inputs  $x_0$ ,  $y_0$  and  $\omega_0$  have all been latched into registers  $D_x$ ,  $D_y$  and  $D_\omega$ , the main FSM sets  $x\_sel$ ,  $y\_sel$  and  $\omega\_sel$  to iteration update mode and starts the CORDIC operation. The CORDIC architecture proceeds with the calculations described in Table 3-15 as the counter  $i$  is incremented in each iteration. After  $n$  iterations, the solution converges and the output logic is asserted. A fixed CORDIC iteration count of  $n = 16$  is used in this implementation. Excluding memory read and write cycles associated with loading of  $a_{ij}^k$  for the inputs and storing of the outputs as  $a_{ij}^{k+1}$ , the total processing time is 17 clock cycles. Multiplication by constant  $K$  is implemented using the canonical signed digit method which comprises shift and add circuits only.

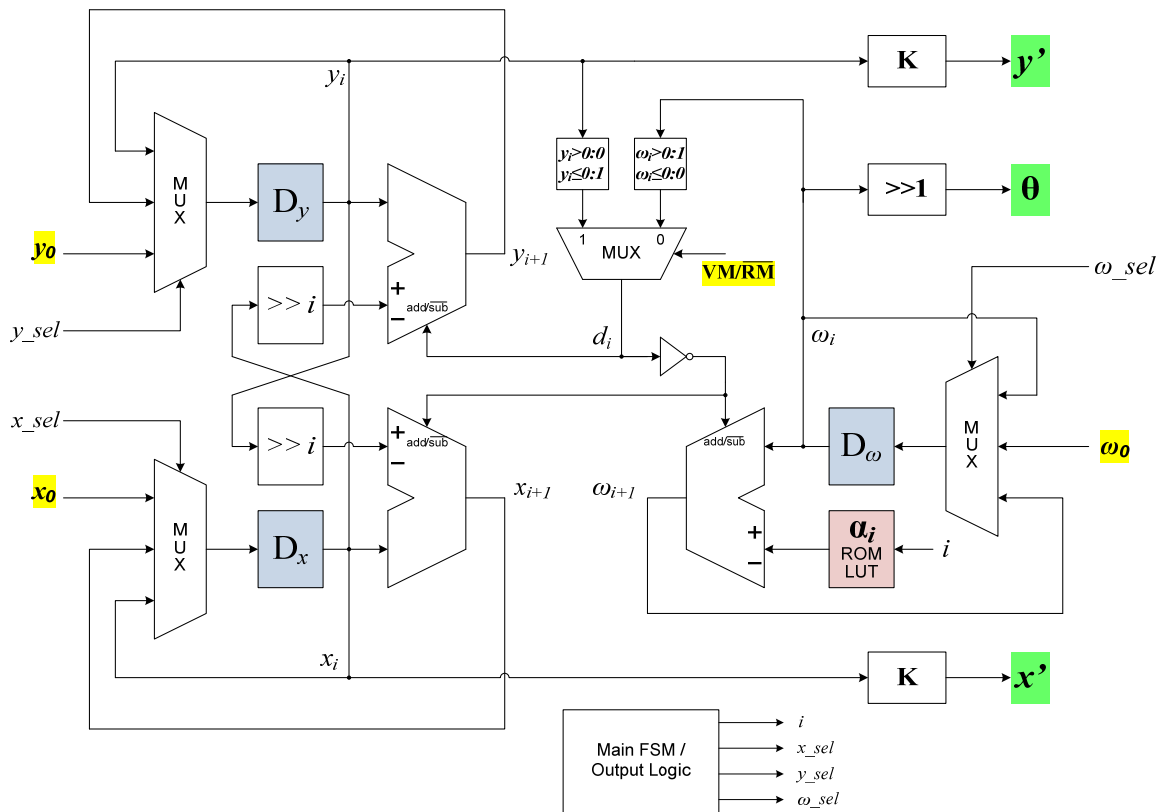


Figure 3-34 Hardware architecture of the CORDIC engine

### Hardware Architecture of the Jacobi EVD Engine

As discussed earlier, by adopting the parallel cyclic Jacobi rotation sequence, significant parallelism can be introduced into the architecture and operation of the Jacobi EVD. The Jacobi EVD engine (EVD\_PROCESSOR in Figure 3-28) features multiple parallel CORDIC engines and is shown in Figure 3-35. The CORDIC engines efficiently handle the calculation of the rotation angle  $\theta$  in (3-24) and the two-sided rotational matrix multiplications in Figure 3-31. For the case of  $n = 4$  (4-by-4 matrix), each *sweep* in the Jacobi algorithm comprises three sets of non-conflicting Jacobi (p,q) rotation pairs:  $\{(1,2),(3,4)\}$ ,  $\{(1,4),(2,3)\}$  and  $\{(1,3),(2,4)\}$ . Memory address translations based on these patterns are automatically managed by the parallel cyclic Jacobi sequencer built into the main controller FSM.

The Jacobi EVD engine operates as follows. The received input covariance matrix is initially stored in a single-port random access memory **D**. The rotation angles for two non-conflicting Jacobi (p,q) rotation pairs (ex.  $J(1,2)$  and  $J(3,4)$ ) are then calculated using two parallel CORDIC vectoring mode (VM) engines. Ideally, the two non-conflicting right-side matrix multiplications can then proceed fully in parallel, and then the left-side multiplications afterwards. However, because the original and rotated matrix elements are read from and written to the same single-port memory **D**, the CORDIC engines are operated in a staggered, pseudo-parallel schedule instead as shown in Figure 3-36. The accumulated product of Jacobi rotation matrices is calculated by the E rotator CORDIC (RM) engines and stored in a separate single-port memory **E**. Since the eigenvector **E** matrix updates involve left rotations only, they are performed together with the **D** left rotations to save on control circuitry.

Finally, at the end of the sweep, the convergence detection unit checks if all the non-diagonal elements of **D** written back to memory are all less than the threshold. If this condition is satisfied, or if the sweep count is already 3, the main controller FSM enables the output I/F to send out the eigenvalues **D** and eigenvectors **E**, and returns to an idle state. Otherwise, the sweep count is incremented and another sweep is performed again.

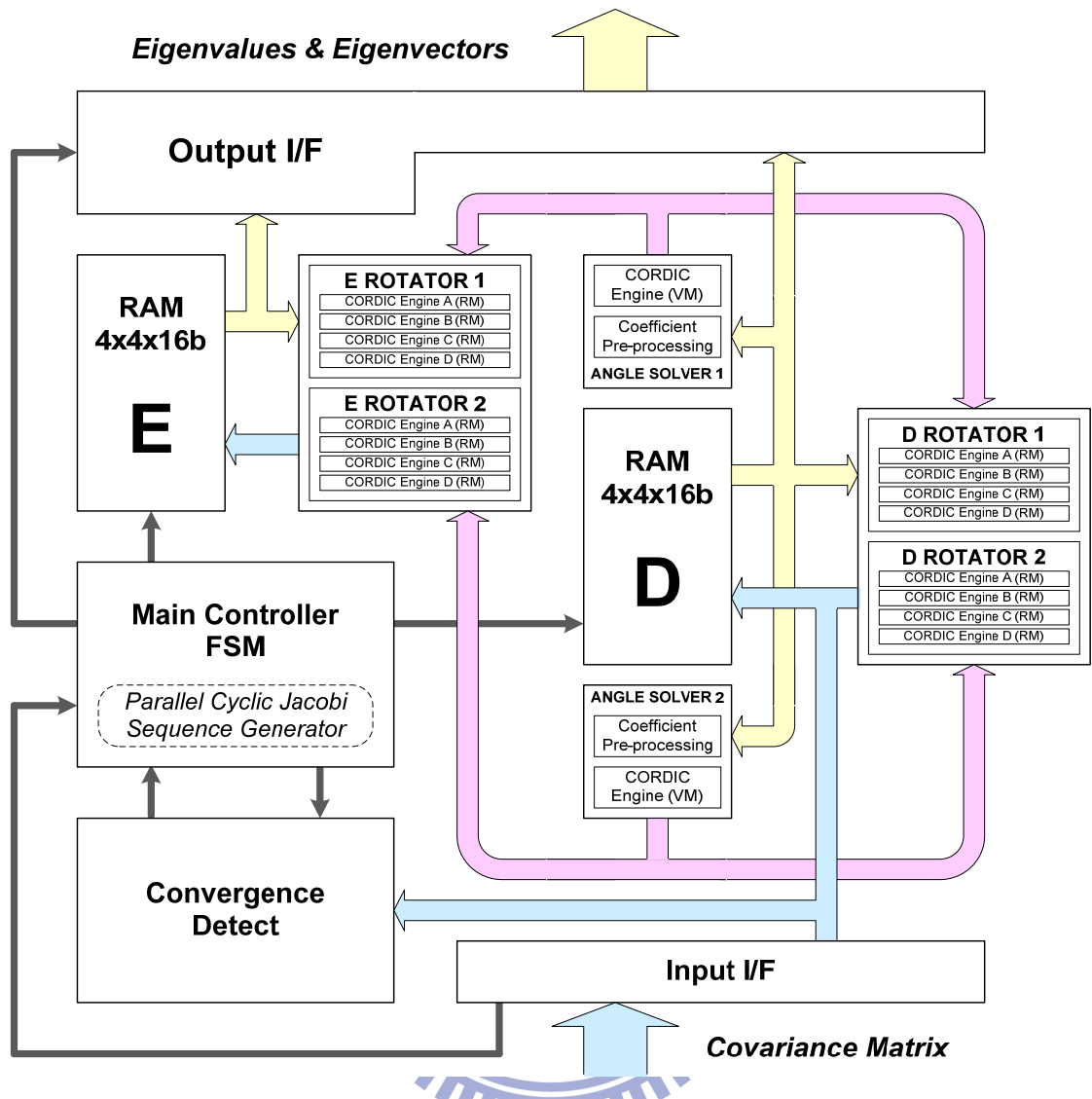


Figure 3-35 Hardware architecture of the Jacobi EVD engine

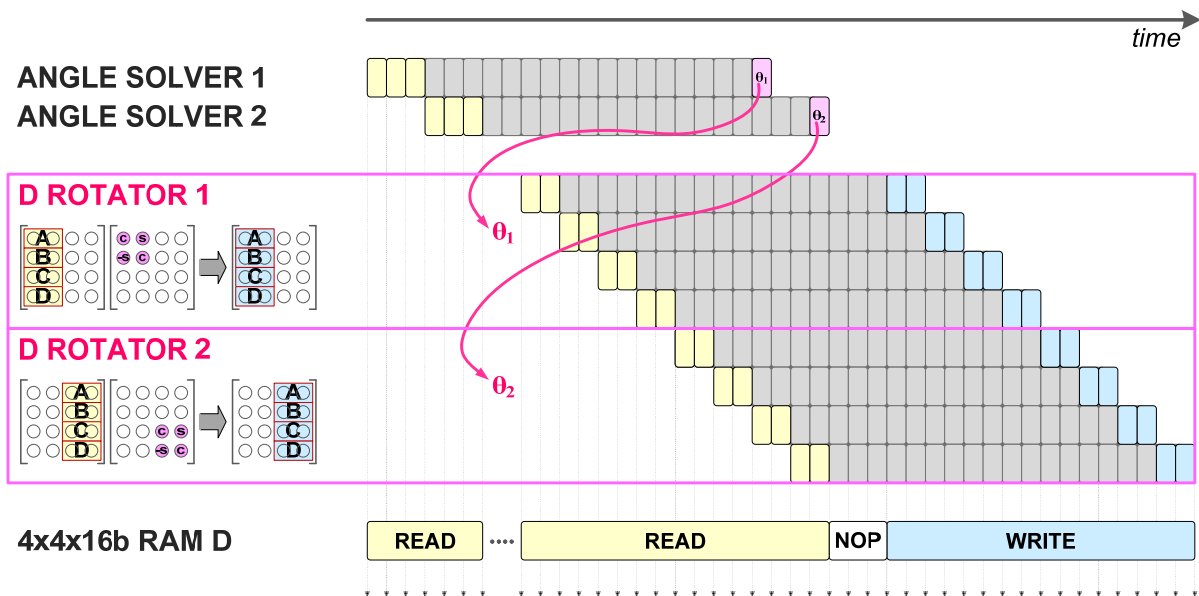


Figure 3-36 Parallel right rotations for rotation set  $\{(1,2),(3,4)\}$

### 3.3.2.4.3 Hardware design of the Infomax ICA training unit

Straightforward implementation of the Infomax ICA algorithm listed in 3.2.1.2 presents two major issues from a hardware point of view. The first concern is the ROM size for the non-linear contrast function lookup table and the second is the amount of temporary variables used throughout the ICA training iterations.

The main purpose of the ROM lookup table is to support the calculation of the exponential function required in the ICA algorithm. However, instead of just simply implementing  $\exp(-u)$  (step 3 in 3.2.1.2) into the ROM lookup table, it makes sense to implement the more complete expression  $g(u) = 1 - 2Y(u)$  (step 4 in 3.2.1.2), such that the division, addition, subtraction and shift operations get built into the lookup operation as well, thus saving a considerable amount of hardware resources. The graphical plot of  $g(u)$  is shown in Figure 3-37.

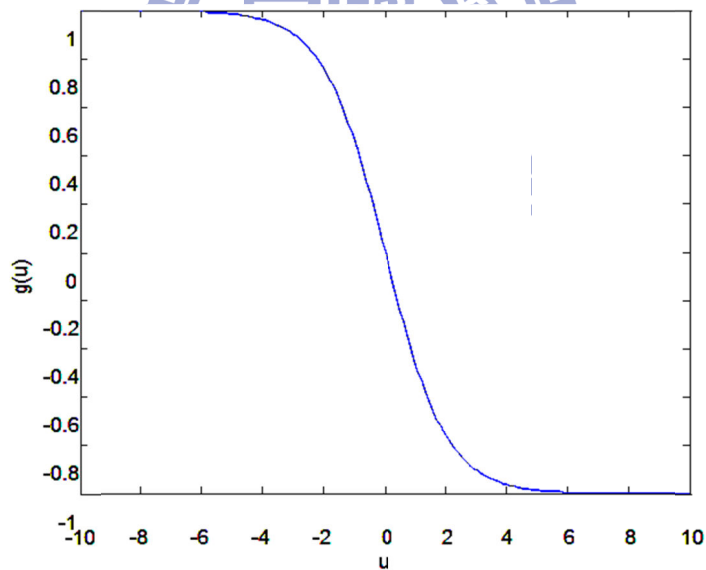


Figure 3-37 Non-linear function implemented in the ROM lookup table

In order to implement the function as a ROM lookup table, the range  $g(u)$  is digitized to 10-bits and the domain  $u$  discretized in steps of  $1/2^N$  within  $[-L, L)$ . If an input is out of the domain's range, a saturated value of  $-1$  or  $1$  is output. Taking into account the anti-symmetric property of  $g(u)$ , the calculated ROM size can be cut in half to  $L \cdot 2^N$ . Common sense suggests that larger values of both  $L$  and  $N$  will lead to better ICA performance. However, this will lead

to an unnecessarily large ROM size. For example, L need not be too large since  $g(u)$  saturates to -1 or 1 when  $|u| > 8$ . To objectively choose a suitable configuration, MATLAB simulations using various combinations of L and N were performed, whose results are summarized in Table 3-16. From these results, we see that by choosing an LUT configuration of N=2 and L=8, the ROM size can be reduced by 87.5% from 256 to 32 entries, with almost no loss in performance. The resulting architecture of the mirrored ROM lookup table unit is shown in Figure 3-38.

Table 3-16 ICA performance using various configurations of the ROM lookup table

L	Step Size	ROM Size	Average Correlation	#Training Iterations
16	(float)	--	0.8612	502
16	1/16	256	0.8581	486
16	1/8	128	0.8588	503
16	1/4	64	0.8675	531
16	1/2	32	0.8704	-
8	1/8	64	0.8588	503
8	1/4	32	0.8655	535
8	1/2	16	0.8704	584

The second design issue concerns the amount of temporary variables needed to support the ICA training iterations. Assuming a 4-channel implementation and a window size of 64 as discussed in Section 3.3.2.4.1, steps 2 to 4 of the ICA algorithm outlined in 3.2.1.2 can be relisted to include the matrix dimensions as follows:

$$\mathbf{U}_{4 \times 64} = \mathbf{W}_{4 \times 4} \mathbf{Z}_{4 \times 64} \quad (3-31)$$

$$\mathbf{Y}_{4 \times 64} = \frac{1}{1 + e^{-\mathbf{U}_{4 \times 64}}} \quad (3-32)$$

$$\Delta \mathbf{W}_{4 \times 4} = L_{\text{rate}} (\mathbf{I}_{4 \times 4} + (\mathbf{1}_{4 \times 64} - 2\mathbf{Y}_{4 \times 64})(\mathbf{U}^T)_{64 \times 4}) \mathbf{W}_{4 \times 4} \quad (3-33)$$

and substituting  $g(u)$  as discussed previously into (3-33),

$$\Delta \mathbf{W}_{4 \times 4} = L_{\text{rate}} (\mathbf{I}_{4 \times 4} + (\mathbf{G}_{4 \times 64})(\mathbf{U}^T)_{64 \times 4}) \mathbf{W}_{4 \times 4} \quad (3-34)$$

It would be extremely inefficient if each equation were to be completely solved before proceeding to the next, as doing so would require the intermediate storage of  $\mathbf{U}_{4 \times 64}$ ,  $\mathbf{Y}_{4 \times 64}$  and  $\mathbf{G}_{4 \times 64}$ . This problem can be partially solved by processing  $\mathbf{Z}$  in a column-wise fashion, since

each column in  $\mathbf{U}_{4 \times 64}$ ,  $\mathbf{Y}_{4 \times 64}$  and  $\mathbf{G}_{4 \times 64}$  depends only on a particular column of  $\mathbf{Z}$ . The problematic bottleneck in the algorithm is actually equation (3-34), where conventional dot product style multiplication of  $\mathbf{G}$  and  $\mathbf{U}^T$  requires the complete storage of  $\mathbf{G}_{4 \times 64}$  and  $\mathbf{U}_{4 \times 64}$  as intermediate variables.

To solve this problem, the multiplication of  $\mathbf{G}$  and  $\mathbf{U}^T$  is performed instead as an accumulation of outer products as follows

$$\mathbf{T}_{4 \times 4} = \mathbf{G}_{4 \times 64}(\mathbf{U}^T)_{64 \times 4} \quad (3-35)$$

$$\mathbf{T}_{4 \times 4} = \mathbf{G}_{4 \times 1}^1(\mathbf{U}^T)_{1 \times 4}^1 + \mathbf{G}_{4 \times 1}^2(\mathbf{U}^T)_{1 \times 4}^2 + \dots + \mathbf{G}_{4 \times 1}^{64}(\mathbf{U}^T)_{1 \times 4}^{64} \quad (3-36)$$

such that  $\mathbf{G}_{4 \times 1}^1$  and  $(\mathbf{U}^T)_{1 \times 4}^1$  depend only on the first column of  $\mathbf{Z}_{4 \times 64}$ ,  $\mathbf{G}_{4 \times 1}^2$  and  $(\mathbf{U}^T)_{1 \times 4}^2$  only on the second column, and so on. As a result, each received column  $\mathbf{Z}_{4 \times 1}$  can be consumed into  $\mathbf{T}_{4 \times 4}$  and discarded immediately following each accumulation.

The resulting hardware architecture and operation flow of the ICA training unit is shown in Figure 3-38, and its main controller state machine and detailed data and operation control logic are shown in Figure 3-39 and Table 3-17 respectively.

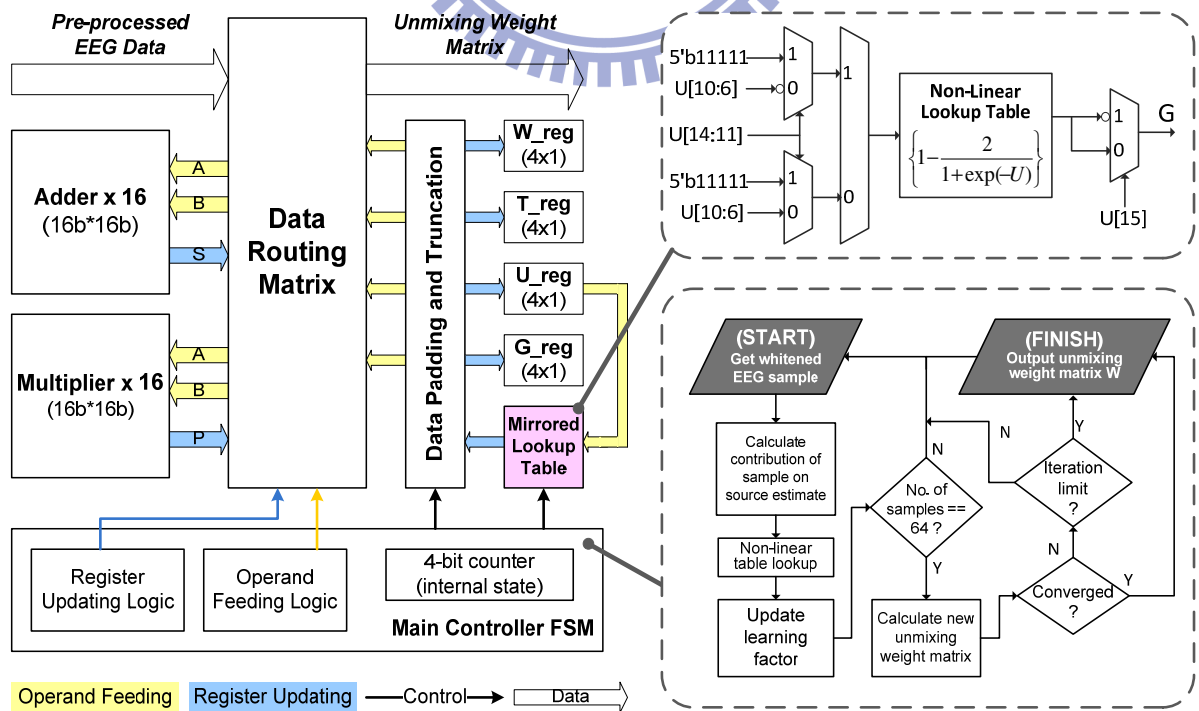


Figure 3-38 Hardware architecture and operation flow of the ICA training unit

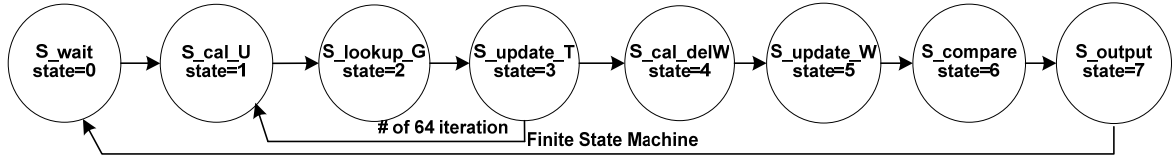


Figure 3-39 Actual hardware states used by the main controller FSM

Table 3-17 State, operation and data control in the ICA training unit

State	Adder Operation	Multiplier Operation [OpA, OpB]	Variable Update	Required No. of Cycles
0 S_wait	-	-	$T=I_{4 \times 4}$ $u=Z_{4 \times 1}$	1
1 S_cal_U	[mul <sub>i</sub> _out,mul <sub>i</sub> _out] [add <sub>i</sub> _out,add <sub>i</sub> _out]	[W,u]	$u_1=$ add3_out $u_2=$ add6_out $u_3=$ add9_out $u_4=$ add12_out	1
2 S_lookup_G	-	-	$g_1=$ Lookup( $u_1$ ) $g_2=$ Lookup( $u_2$ ) $g_3=$ Lookup( $u_3$ ) $g_4=$ Lookup( $u_4$ )	4
3 S_update_T	[T,mul_out]	[g,u]	$u=Z_{4 \times 1}$ $T=$ add_out	1
4 S_cal_delW	[mul <sub>i</sub> _out,mul <sub>i</sub> _out] [add <sub>i</sub> _out,add <sub>i</sub> _out]	For counter=0 [R <sub>learning</sub> ,T] For counter=1~4 [T(1,:),W] [T(2,:),W] [T(3,:),W] [T(4,:),W]	$T=$ mul_out $T(1,:)=$ add3_out $T(2,:)=$ add6_out $T(3,:)=$ add9_out $T(4,:)=$ add12_out	5
5 S_update_W	[W,T]	-	$W=$ add_out	
6 S_compare	[mul <sub>i</sub> _out,mul <sub>i</sub> _out] [add <sub>i</sub> _out,add <sub>i</sub> _out]	[T,T]	$T=I_{4 \times 4}$	
7 S_output	-	-	-	16

### 3.3.2.5 Heart Rate Variability (HRV) Analysis Engine

The hardware architecture of the HRV analysis engine follows from the algorithm discussion in 3.2.2.3 and is shown in Figure 3-40. The RR interval calculation and HRV analysis units are straightforward implementations of the algorithm described previously. Of special interest in this section is the sliding window memory manager, which is the hardware mechanism by which real-time operation of the HRV analysis engine is made possible. Similar to the data windowing scheme employed by the ICA engine (3.3.2.4.1), its operation is transparent to both the upstream and downstream blocks. That is, the RRI calculation and HRV analysis units both need not be aware of the sliding window scheme employed.

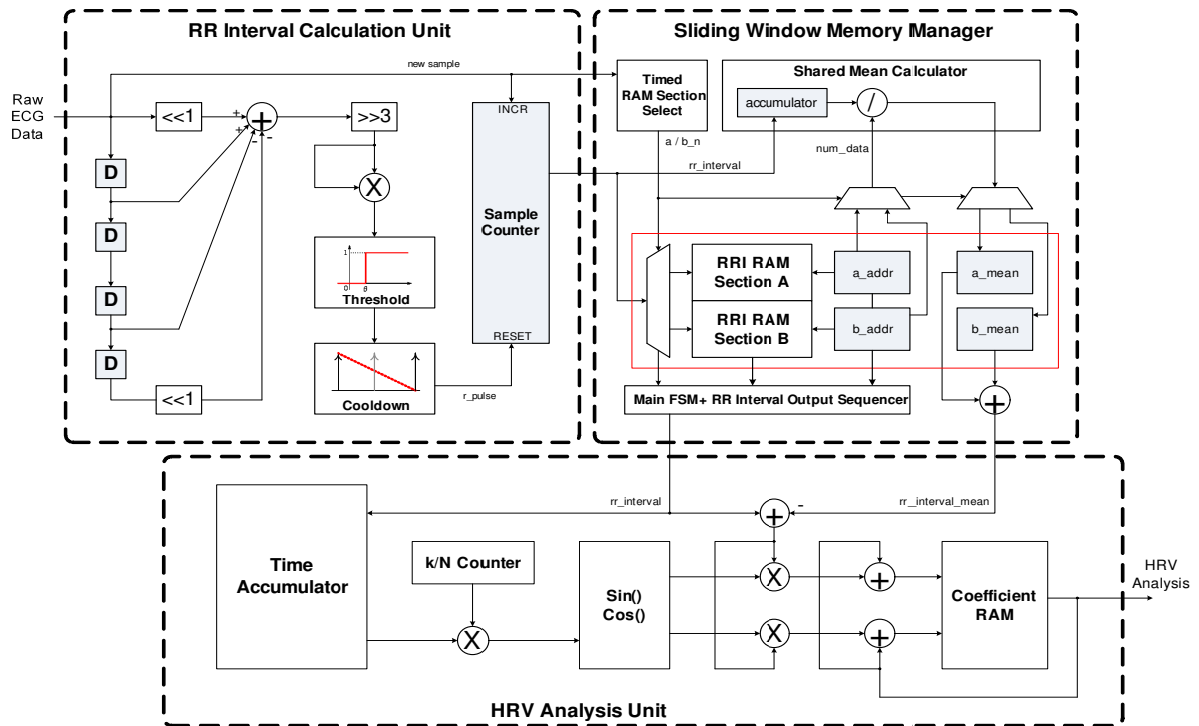


Figure 3-40 Hardware architecture of the HRV analysis engine

RR intervals spanning a period of two to five minutes is suggested for evaluation of short term HRV [33]. In our design, in order to address the HF and LF components of HRV, a two minute window is adopted. Time-frequency analysis of HRV is achieved by performing spectral analysis on the RR intervals using a sliding window configuration. This produces a series of power spectra from sequential windows of data over time. To ensure a smoother transition between windows, and to provide better observability, the sliding window is configured to have fifty percent overlap. This implies that the window of data moves forward by one minute after every analysis of power spectra. The configuration of the sliding window is shown in Figure 3-41.

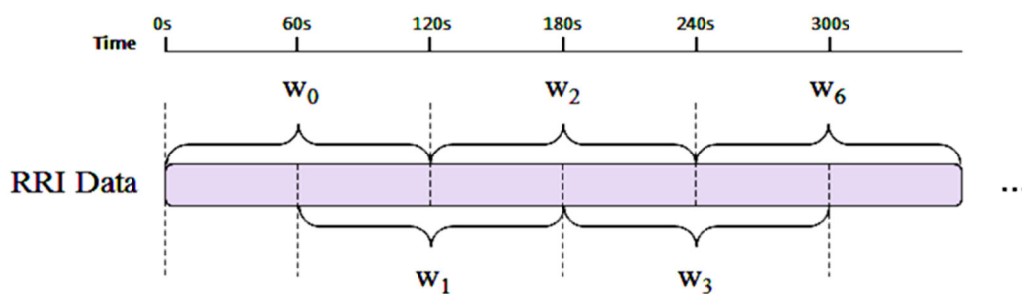


Figure 3-41 The sliding window scheme. RR intervals are divided into one minute frames. A data window consists of two frames with one frame overlapping with the previous window.



The RR interval data are buffered in an SRAM memory prior to HRV analysis. The size of the memory depends on the maximum number of RR intervals (or equivalently, heart beats) that can fit in two minutes. Thus, assuming that the maximum safe heart rate does not exceed 220 bpm [67], a 512 word SRAM memory is chosen. For robustness, a cooldown block (in the RR interval calculation unit) that limits the frequency of RR interval data ensures that this data size requirement is strictly met at the point prior to buffering.

The overlap period between each window is one minute, so each minute of RR intervals, or frames, has to be stored separately. In our design, the SRAM memory is categorized into two sections to store the different frames of RR intervals so 256 words can be allocated for each minute of data. During one minute, data is stored into one section of the memory, after which the number of elements ( $x\_addr$ ) is noted and storage of data is switched to the second section using a path selector. After two minutes, the RR intervals are passed to the HRV analysis unit for spectral analysis.

To output the RR intervals to the HRV analysis unit, a bank selector is employed to choose which section of the SRAM is currently active. After all RR interval data have been read from one section the next section is selected. As spectral analysis requires the signal to be zero-mean, the mean of the data in each section is calculated and stored ( $x\_mean$ ). This ensures that when the next window is encountered the overlapping section from the previous window has the correct mean value. The two mean values are summed to form the final mean of the data. Division by two is simplified to a bit-shift (or re-position of the fixed point). The architecture of the sliding window memory manager is shown in Figure 3-40.

#### 3.3.2.6 *Lossless Data Compression (COMP) Engine*

The hardware architecture of the lossless data compression engine is shown in Figure 3-42. It comprises three pipeline stages including first, a prediction and parameter estimation stage, second, a Golomb-Rice entropy coding stage, and third, an output packaging stage. Since Golomb-Rice codes vary in length, the number of clock cycles needed to completely

pack an encoded stream onto a fixed bus width varies at the final packaging stage. To prevent pipeline overflow, a ready-acknowledge handshaking mechanism is employed throughout every stage in the engine, including the input and output ports that connect to the PDS and UART respectively.

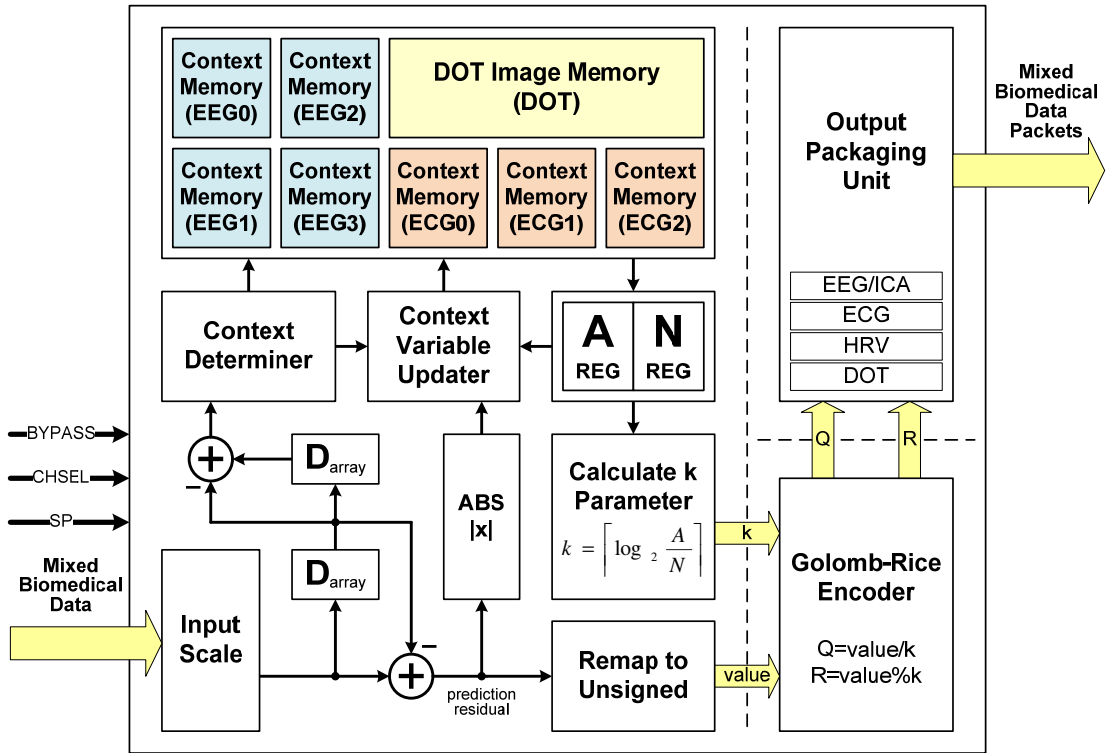


Figure 3-42 Hardware architecture of the lossless data compression engine

To conserve hardware resources, the prediction circuitry is shared among the different biomedical signals. The upstream PDS unit functions as a multiplexer/arbitrator, feeding the otherwise parallel multi-channel biomedical signals serially into the lossless data compression engine. At the same time, the PDS unit also annotates each sample with its signal type, precision and bypass mode through the CHSEL, SP and BYPASS pins respectively. Table 3-18 shows a detailed description of these control signals.

The prediction stage, upon receiving a biomedical sample, determines its context and loads the context statistics from memory. In the following clock cycle, the  $k$  parameter is calculated, the prediction error remapped, and the context variables updated and written back to the memory. The prediction stage maintains the individual memory locations for the

context statistics of the different biomedical data channels. For DOT, DPCM prediction is performed on an inter-frame, per pixel basis, and context-based  $k$  parameter estimation is not performed, due to memory space limitations.

Table 3-18 Control signal parameters of the lossless data compression engine

Port Name	Width	Description			
<b>BYPASS</b>	1	If the input data comes with the BYPASS signal set to high, data compression is bypassed and the input data is directly packed as raw data by the output packaging unit			
<b>CHSEL</b> (channel select)	4	Value	Channel	Value	Channel
		0	EEG/ICA Ch0	5	DOT Pixel
		1	EEG/ICA Ch1		
		2	EEG/ICA Ch2	6	EKG Ch0
		3	EEG/ICA Ch3	7	EKG Ch1
<b>SP</b> (sample precision)	3	4	HRV Coefficients	8	EKG Ch2
		0	8-bit	4	16-bit
		1	10-bit	5	18-bit
		2	12-bit	6	20-bit
		3	14-bit	7	32-bit

The Golomb-Rice entropy coder in Figure 3-42 implements the Golomb-Rice coding table shown in Figure 3-9b for various values of  $k$ . It calculates the quotient  $Q$  and remainder  $R$  based on the estimated Golomb-Rice  $k$  parameter and input remapped prediction errors, and outputs the result to the next stage within a single clock cycle.

The output packaging unit is the final pipeline stage and maintains four separate output buffers for the different biomedical signals, which are filled up as samples are encoded into bit streams. Whenever any of the buffers become full, the buffer value is driven onto the output bus, together with an appended data type ID indicating the type of biomedical data, as described previously in 3.3.1.4. In case two or more buffers are full simultaneously, a priority scheme is enforced such that minimal output latency is achieved.

The first two stages, namely the prediction and Golomb-Rice encoding stages, correspond to the proposed algorithm described in 3.2.4 to which the overhead energy consumption  $E_{\text{comp}}$  (3.2.4.2) is attributed. The PDS and output packaging stages perform mainly data routing and packing, which operate regardless of whether compression is employed or not.

### 3.3.3 Functional Verification

The functionality of the complete system from algorithm down to hardware implementation must be verified thoroughly before taping out the design for chip fabrication. To achieve this, detailed functional verification is performed at key points in the pre-silicon phase of the IC development flow shown in Figure 3-43. In general, the algorithms (i.e. DOT, ICA, HRV and COMP) have been from the very beginning developed with hardware issues considered as much as possible. However, as one cannot foresee all of the problems that may arise in the chip implementation down the road (either due to lack of judgment or experience), redesign at the RTL, architectural or even algorithm level is typically necessary to resolve any area, timing closure or power issues that may block the way to successful physical implementation. In the following discussion, the verification approaches at both individual subsystem and chip integration levels are described further in detail.

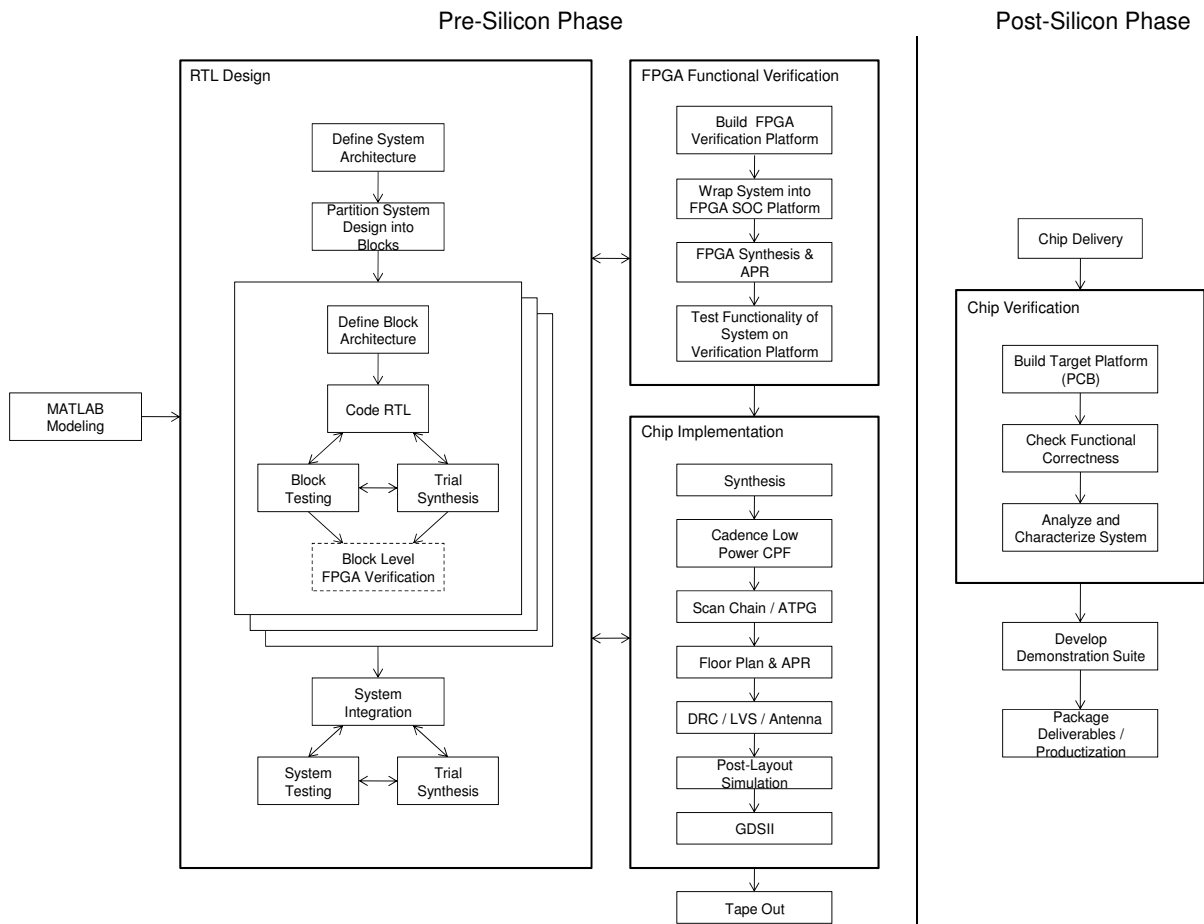


Figure 3-43 IC development flow methodology

The algorithms for the individual biomedical signal processing engines, including lossless data compression, have been previously developed and verified using MATLAB. Simulation results of the developed ICA algorithm demonstrate a verified average correlation coefficient performance of 0.86 using randomly generated mixed super-gaussian sources. Performance of the HRV algorithm was evaluated and verified using both real ECG signals (from the MIT-BIH database) and artificially controlled RRI data [68]. Reconstructed DOT images based on fNIR light intensities acquired from self-developed DOT sensor array board were compared to reference images using MSE [50]. Finally, the lossless data compression algorithm was evaluated using publicly available reference biomedical data such as the MIT-BIH and UCI-KDDI databases as well as outputs of the upstream algorithm blocks [69]. Table 3-19 shows a summary of the function verification and performance evaluation procedures performed for the various biomedical signal processing algorithms as described above.

Table 3-19 Verification of the various biomedical signal processing algorithms

Subsystem	ICA	DOT	HRV	COMP
Algorithm model	MATLAB	MATLAB	MATLAB	MATLAB
Input pattern	1. Random mixed super-gaussian signals 2. Real EEG acquired using Neuroscan EEG acquisition equipment	1. fNIR light intensities acquired from self-made, calibrated DOT sensor array board	1. MIT-BIH ECG Database 2. Artificial RRI	1. MIT-BIH Database 2. UCI KDDI (EEG) 3. EEGLab (EEG) 4. Output data of ICA, DOT and HRV
Third-party model	EEGLab	--	--	--
Function verification/performance evaluation	0.86 correlation coefficient between <i>original</i> and <i>extracted</i> source signals, and comparison with EEGLab output	MSE comparison between reference and reconstructed DOT images	LF/HF ratio of artificially generated RRI vs FFT and Lomb floating point	Exact match between original source and decompressed data; average compression ratio (CR) of 2.05 for mixed biomedical streams

In order to ensure that the hardware implementation behaves the same way, golden test patterns at key points in the algorithms are generated from the MATLAB models, and attached to the RTL, gate-level and post-layout simulation test benches for automated data checking. Individual unit testing of the ICA, DOT, HRV and COMP engines at the RTL level is first performed before moving up to chip integration level testing so as to allow easier

localization of bugs or defects. While the main focus during unit testing is to check the correctness of the individual processing engines against their respective algorithms, system-level verification focuses more on the correctness of data communication and chip integration.

The system-level test bench used to verify the chip design and integration is shown in Figure 3-44. After the test patterns have been attached to their respective models and monitors, chip operation is initiated by the main test control either through a manual pin trigger or a system mode command transmitted through UART. The DUT then begins to periodically request for raw biomedical data samples for processing. In response, the AFE model retrieves the requested data from attached raw input data files and sends them back to the DUT according to the AFE interface protocol. Processed data received by the UART model is unpacked, decoded and finally verified on-line against expected data generated from the MATLAB models.

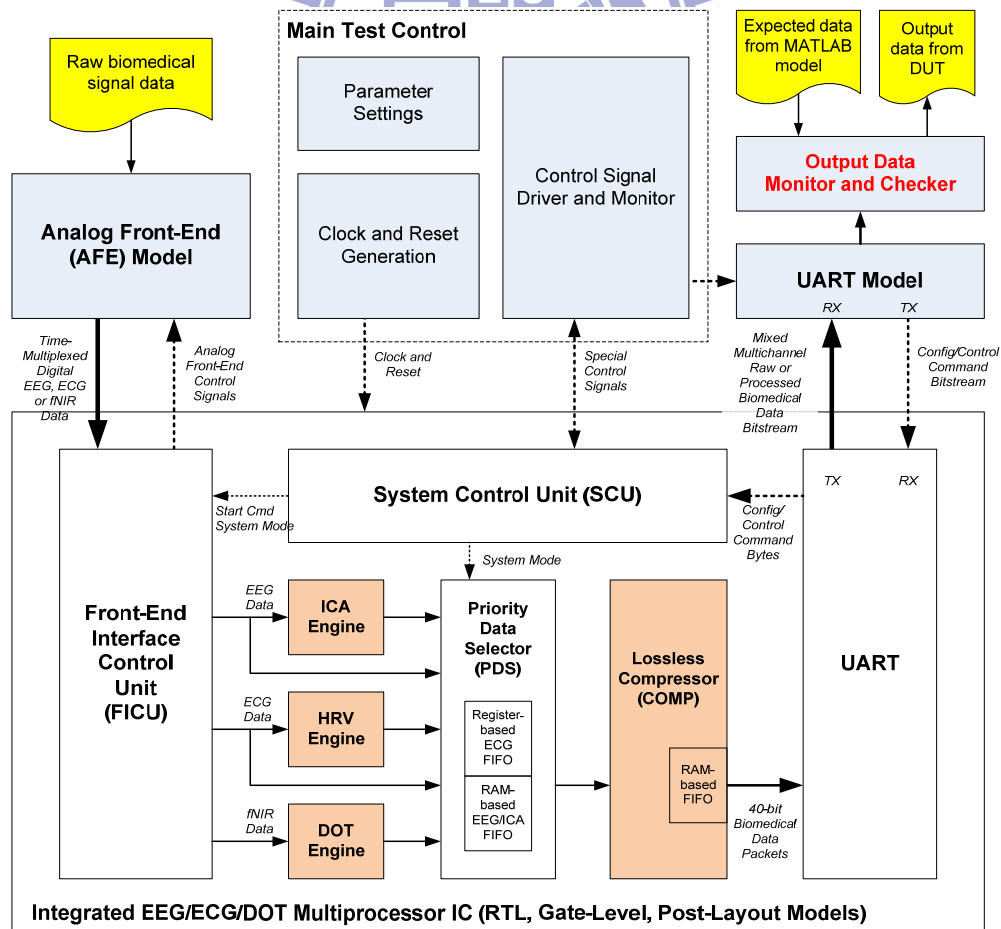


Figure 3-44 System test bench for functional verification of biomedical multiprocessor IC

### 3.3.4 Chip Implementation in UMC 65nm CMOS Technology

The functionally and physically verified chip has been successfully implemented and fabricated using UMC 65nm CMOS 1P10M technology. The die size is 1,317 x 1,317  $\mu\text{m}^2$ , and the core size is 680 x 680  $\mu\text{m}^2$ , comprising a total of 368,314 gates. Power consumption estimated using Synopsys Prime Power reports a total of 3.6mW consumed at an operating condition of 1.0V core voltage and 24MHz clock frequency. The details of the area and power breakdown across the different modules are shown in Table 3-21. The floor plan and die micrograph of the fabricated chip is shown in Figure 3-45 and a summary of the chip specifications is given in Table 3-20. The chip bonding map is shown in Figure 3-46 and the final packaged chip is shown in Figure 3-47.

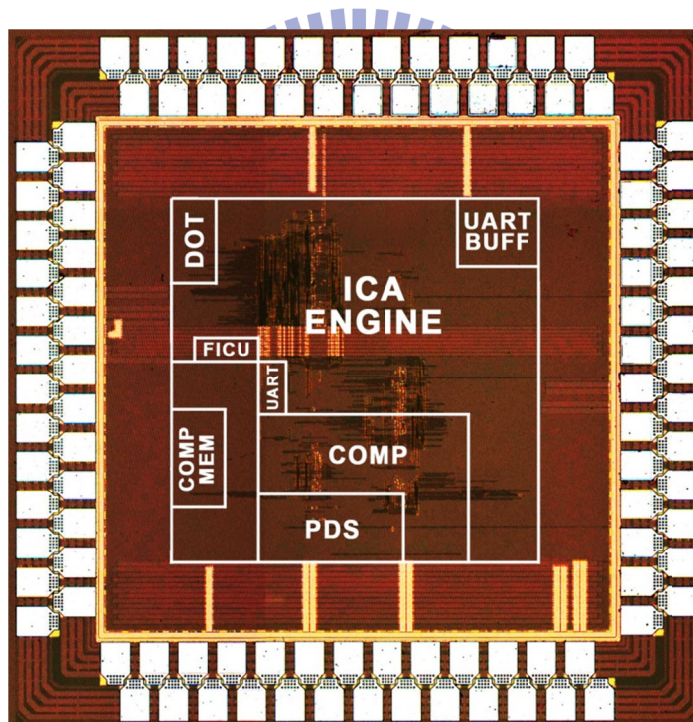


Figure 3-45 Die micrograph and chip layout of the integrated biomedical DSP IC

Table 3-20 Chip specifications summary

<b>Technology</b>	UMC 65nm CMOS 1P10M
<b>Pad/Core Voltage</b>	1.8V / 1.0V
<b>Die/Core Area</b>	1.317 x 1.317 mm <sup>2</sup> / 0.680 x 0.680 mm <sup>2</sup>
<b>Logic Gates</b>	368.3K
<b>Operating Frequency</b>	24 MHz
<b>Power Consumption</b>	3.6 mW (full system mode)
<b>Number of PADS</b>	104 pins (functional / IO power / core power : 32 / 64 / 8)
<b>Test Package</b>	128-pin CQFP
<b>Package Dimensions</b>	28mm x 28mm







## Chapter 4 Integrated DOT/ECG/EEG Multiprocessor IP for SoC Implementation

In the previous chapter, the design and implementation of a highly-integrated DOT/ECG/EEG biomedical signal multiprocessor IC was presented. Although it possesses desirable features of small size, low power and a high degree of functional integration, the designed chip lacks the flexibility and adaptability to be extended in future derivative applications. In this chapter, the extended development of the proposed biomedical multiprocessor as an SoC-compatible IP is presented as a solution.

### 4.1 Motivation for SoC Implementation

SoC stands for system-on-a-chip, and is literally the large scale functional integration of traditional discrete PCB (printed circuit board) components comprising a complete computer system onto a single chip. A typical SoC design is a HW/SW integration of at least one programmable microcontroller, microprocessor or DSP core; on-chip memory blocks such as ROM, RAM, EEPROM and flash; HW accelerators that perform special processing tasks (ex. data encryption, multimedia encoding, 3D graphics rendering, etc.); I/O connectivity modules such as ADC/DAC, UART, USB, SPI, SD, Ethernet, Firewire, Thunderbolt, LCD and GPIO to interface with the outside world; computer architecture peripherals such as direct memory access (DMA), memory and interrupt controllers, bus arbiters and bridges, counter timers and real-time clocks; bus interconnects for data transfer and control; oscillators, phase-locked loops (PLL), voltage regulators, power management circuits and reset generators for providing clock, power and reset infrastructure; and most importantly, the embedded software that controls it. Table 4-1 shows a summary of the above enumeration and Figure 4-1 shows a typical SoC hardware architecture.

Table 4-1 Typical hardware components comprising an SoC

Category/Function	Component(s)
Main processor	Microcontroller, microprocessor or DSP core
Memory	ROM, RAM, EEPROM, Flash

Architecture peripherals	DMA /memory/interrupt controllers, bus arbiters and bridges, counter timer, real-time clock
Data and control lines	Bus interconnects, encoders, decoders and multiplexers
HW accelerator IPs	Data encryption, multimedia encoder/decoder, 3D graphics engine, etc. (examples)
I/O connectivity	ADC/DAC, UART, USB, SPI, SD, Ethernet, LCD, GPIO
Clock, power and reset	Oscillators, PLLs, voltage regulators, power management circuits, reset and clock generators

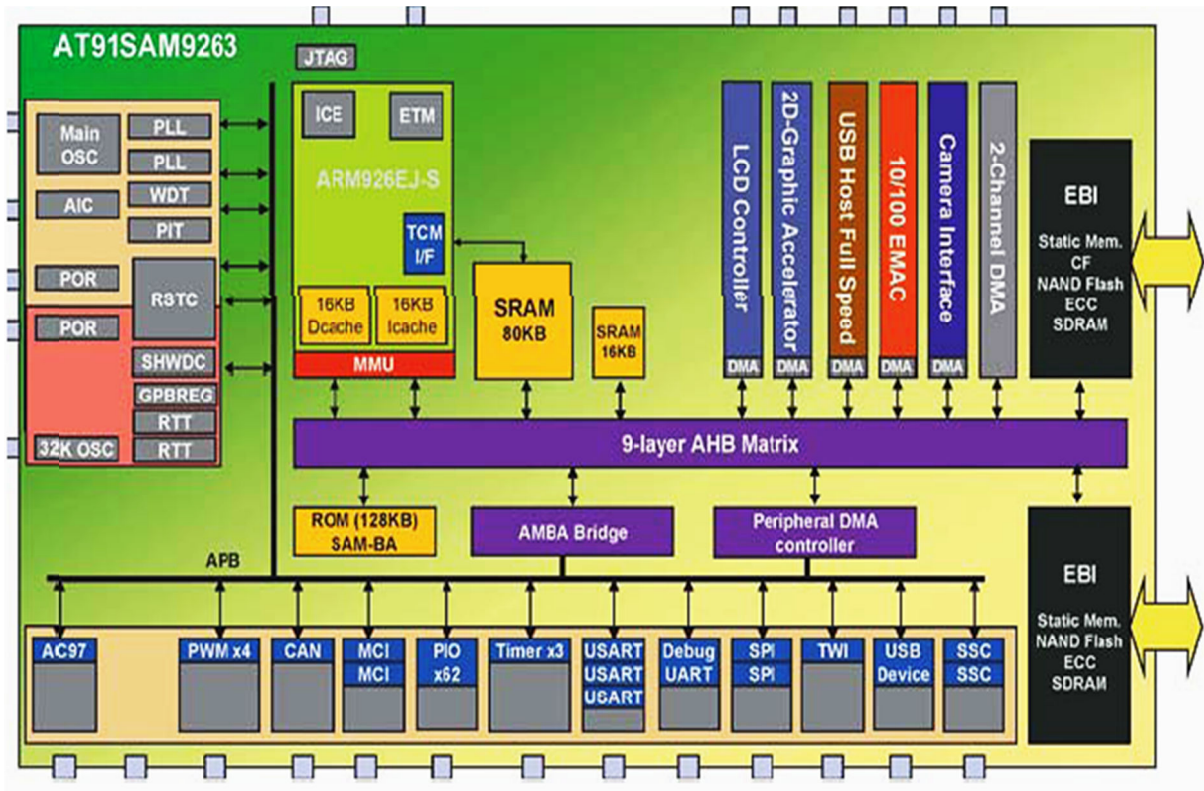


Figure 4-1 Atmel AT91SAM9263 SoC architecture

Such high degree of system integration is possible thanks to the steady advances in semiconductor device fabrication technology wherein transistor sizes have shrunk more than 100,000 fold over the past decades [70]. As more components are moved from the board into the SoC, the system's form factor and power consumption become greatly reduced while reliability and performance are enhanced significantly (i.e. due to faster internal interconnects, on-chip caches/memories). Furthermore, because the SoC is programmable, it is possible to differentiate an SoC-based system to support new functional requirements by simply updating its software components. The useful lifetime of existing silicon is extended and the need to redesign, verify and fabricate new chips from the ground up can be avoided. With cost-

effective silicon that can be shared across different applications, shorter software development schedules, lower component counts and assembly costs, dramatic reductions in the total system cost can be achieved. With these many advantages, SoC-based systems have become extremely popular that they can be found in virtually every modern electronic device today.

SoC development is not without challenges however. As more and more functions are pushed into the SoC, the SoC becomes increasingly complex that the time and effort needed for design and verification becomes a significant issue. To make matters worse, product lifecycles and development schedules demanded by the market have only but shortened over the years. To work around these difficulties, many different approaches have been widely adopted by the industry such as design reuse, orthogonal design partitioning, and higher abstraction levels (ex. transistors, gates, RTL, behavioral) to increase productivity. An SoC design methodology called platform-based design is a culmination of these ideas, whose practice has changed from option to necessity in recent years.

In a typical platform-based design, basic system functionality is provided by a reusable HW/SW platform (Figure 4-2, left side), comprising a reference SoC platform design, a corresponding basic set of device driver software, and optionally an operating system with system function libraries. This pre-verified HW/SW platform serves as a stable foundation upon which application-specific hardware and software (Figure 4-2, right side) can be rapidly and reliably built to allow system customization/differentiation for various target applications. This is in particular supported by the extensive use of well-designed system interfaces such as libraries and APIs for software and standard data transfer protocols, memory maps and interrupt schemes for hardware (Figure 4-2, in red colored font). Such organized structure of hardware and software architectures facilitates easy functional extension, parallel development (both in-house and third-party) and maximal reuse of IP at both platform and component levels; as a result, system complexity becomes manageable and development time, effort and costs become reduced significantly.

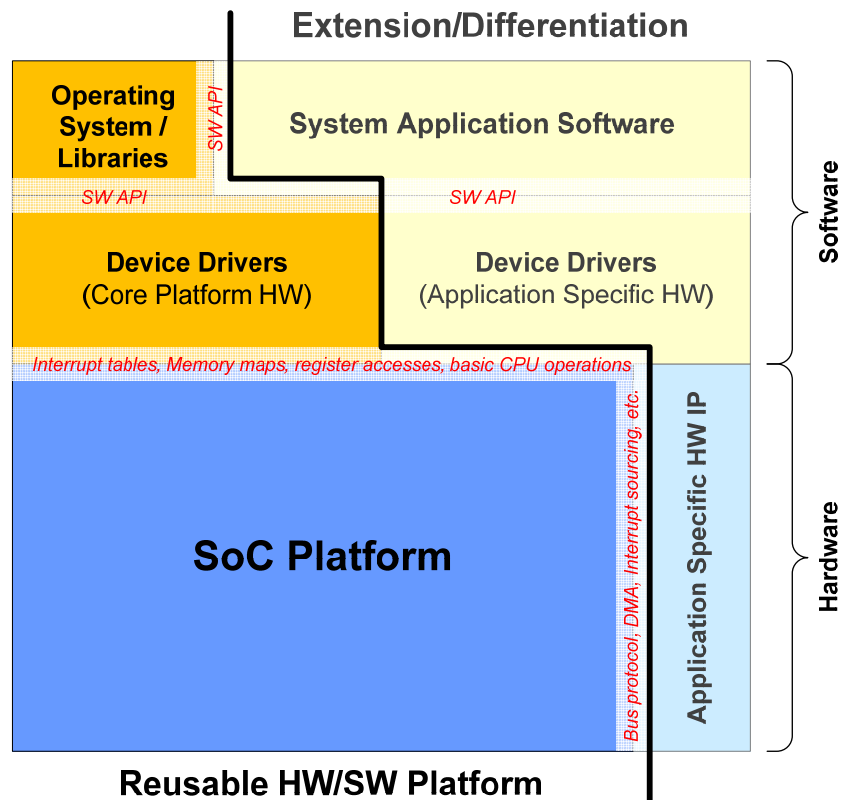


Figure 4-2 Platform-based system design

In recent years, as SoC platforms have become more common, system design is increasingly performed from the perspective of software running on an SoC embedded processor [71]. As much as possible, differentiation is implemented in software avoiding the use of silicon, with the goal of achieving a performance just enough to meet the requirements. Only when really called for based on software profiling are a few select functions elected for specialized hardware implementation. With bulk of the commercial value getting associated more and more with system design and application differentiation nowadays, modeling of customized systems becomes a top priority. Examples of some new methodologies further built on top of the framework of platform-based design are early high-level exploration of function versus architecture in electronic system-level (ESL) design, and separation of computation and communication as manifested in transaction-level modeling (TLM).

In line with these recent trends, the previously developed integrated DOT/ECG/EEG biomedical multiprocessor design is repackaged and delivered as a pre-verified IP core for ready integration in today's ubiquitous platform-based SoCs. Among the many different SoC

platforms out in the market, we choose the platform based on the ARM processor and AMBA bus architecture due to its status as the most widely adopted platform in the industry today. In the remaining sections of this chapter, the development of the proposed biomedical multiprocessor IP as an application extension to SOCLE's Cheetah ARM SoC platform is presented in greater detail.

## 4.2 SoC-based Design of the Biomedical Multiprocessor System

### 4.2.1 SoC Architecture

The Cheetah ARM SoC platform solution offered by SOCLE Technology Corp. (platform vendor) is chosen for the development of the proposed multiprocessor IP, for its rich set of SoC features and friendly technical support. With respect to Figure 4-2, the Cheetah SoC platform solution comprises 1) for hardware, a reusable SoC platform based on the ARM926EJ-S processor and AMBA AHB/APB bus protocols, including many of the peripheral/support hardware components listed in Table 4-1; and 2) a software suite comprising a toolchain set, board support package, boot loader (U-boot), operating system (OpenLinux), root file system tools (BusyBox) and device drivers. A feature list of the Cheetah ARM SoC is shown in Table 4-2 and its architecture is shown in Figure 4-3.

Table 4-2 Select features of the SOCLE Cheetah ARM SoC platform

<b>CPU (ARM926EJ-S)</b>	Clock frequency up to 266MHz
<b>AHB bus</b>	Clock frequency up to 50/133MHz (FPGA expansion/no FPGA exp.) Bus extensions: 2 Master, 2 Slave
<b>Memory Controller</b>	NOR-Flash/NAND-Flash/SRAM/ROM (4 banks) SDRAM (4 banks)
<b>System Controller</b>	Reset control, power mode control (normal/idle/slow) Clock control: CPU/AHB clock ratios 8:1/4:1/3:1/2:1/1:1
<b>DMA Controller</b>	4 channels, mem-to-mem/IO-to-mem/mem-to-IO transfers
<b>Interrupt Controller</b>	31 sources, programmable rise/fall/high/low scheme, 1 FIQ
<b>General Purpose I/O (GPIO)</b>	8 individually programmable input/output pins
<b>UART</b>	3 channels 16-byte RX and 16-byte TX FIFOs per channel Programmable up to 2.7Mbps
<b>PCI Host Bridge</b>	v2.1/v2.2 compliant (2 devices)
<b>USB 2.0 Host / Device</b>	1
<b>10/100 Mbps Ethernet MAC</b>	1
<b>IDE Host (ATA/ATAPI-6)</b>	Up to 2 devices, PIO or Multi-word DMA mode Supports microdrive, CF card, PCMCIA, CD-ROM, HDD (16-bit)
<b>STN/TFT Controller</b>	Up to 1024x768, 1/2/4/8/16/24 bpp, with programmable timing



<b>SPI Interface</b>	2 channels w/ interrupt-based operation
<b>I2C-Bus Interface</b>	v2.1 compliant, programmable clock up to 400Kbps Master/slave/multi-master modes
<b>I2S-Bus Audio Interface</b>	1 channel, DMA or interrupt based operation 8 or 16-bits resolution, 32 to 96KHz sample rate
<b>Secure Digital (SD) Host</b>	1 channel, DMA or interrupt based operation, v1.01 compliant
<b>Timers</b>	32-bit timer w/ PWM (2 channels) 32-bit internal timer (3 channels)
<b>Real Time Clock</b>	24-hr time mode, 1/10s precision, alarm interrupt 32.768KHz operation
<b>A/D Converter (ADC)</b>	8 channels, 10-bit resolution, max. 50kSPS

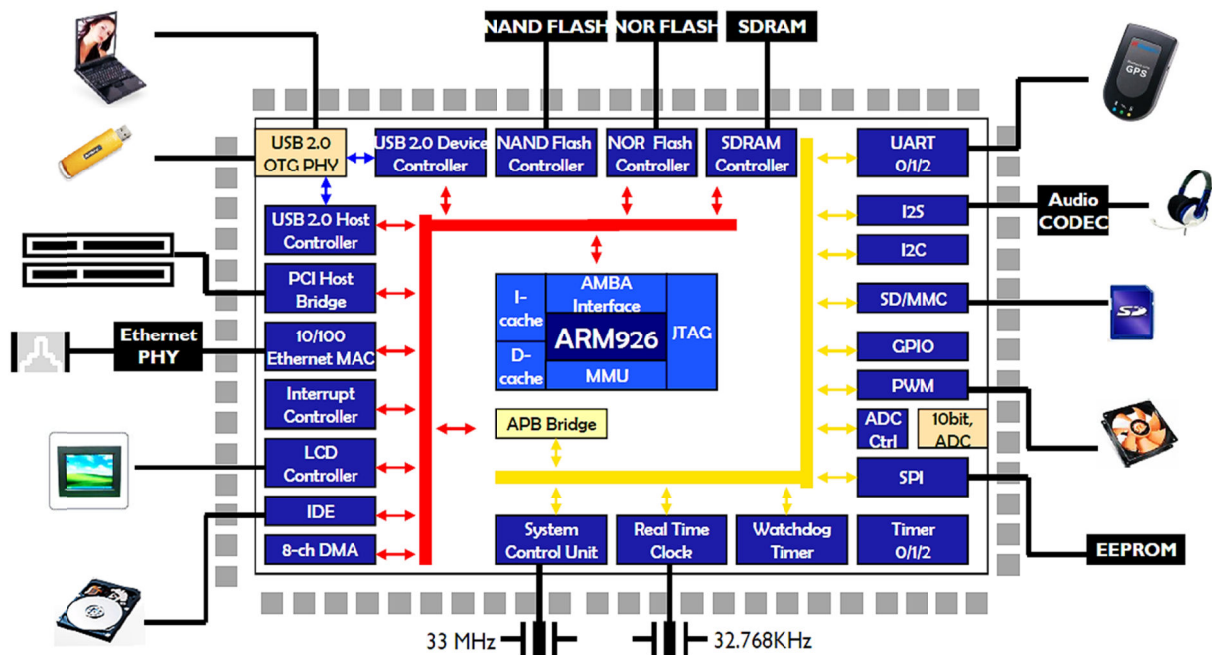


Figure 4-3 Architecture of the Cheetah ARM SoC

To speed up the development process, SOCLE provides a programmable, board-level system integration prototype upon which applications can be rapidly verified on top of the Cheetah SoC platform in real-time. The development board, called the SOCLE Cheetah Development Kit (CDK), exercises the full functionality of the Cheetah ARM SoC implemented with all necessary board-level elements including off-chip memories, I/O connectivity hardware ports (ex. RS232/UART, USB, SPI, IDE, etc.), and other components controlled by the SoC (ex. LCD screen, GPIO ports, audio line out, etc). For development of application-specific hardware IP, a Xilinx Spartan-3 XC3S4000 FPGA with direct connections to the Cheetah SoC's internal AMBA buses, interrupt and direct memory access (DMA) lines is provided. The SOCLE CDK board is shown in Figure 4-4 and the connections between the Cheetah ARM SoC platform and the FPGA are illustrated in Figure 4-5.

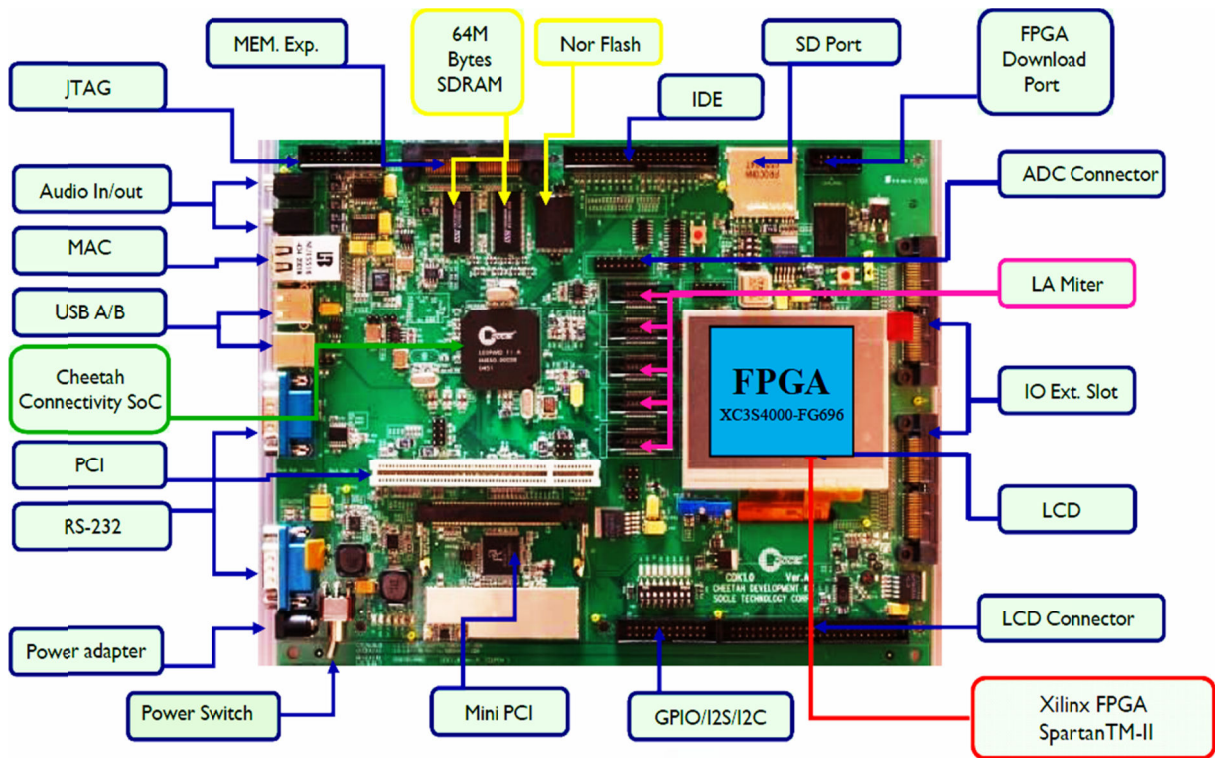


Figure 4-4 SOCLE ARM SoC Cheetah Development Kit (CDK)

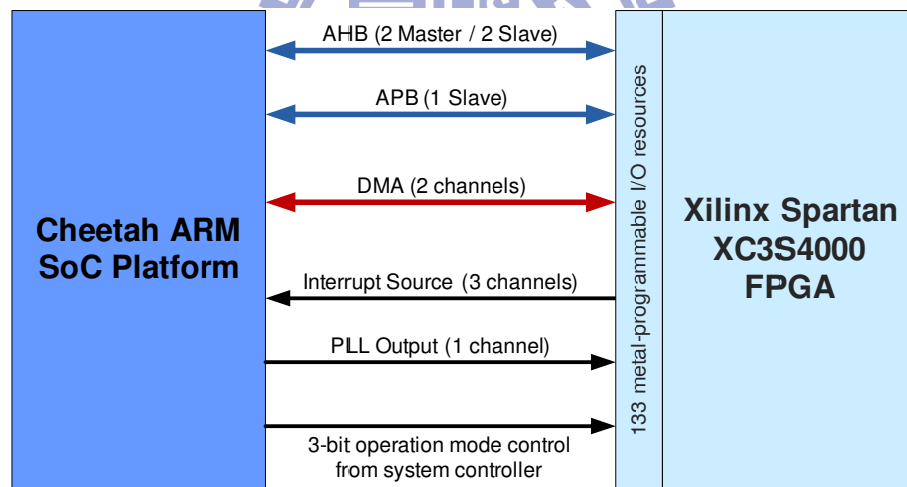


Figure 4-5 Connections between the Cheetah SoC and Xilinx FPGA

The first step in porting the original design to the Cheetah ARM SoC platform is to decide the HW/SW repartitioning of the original functional blocks. The original functional blocks are derived from Figure 3-23, and remapped into an SoC-oriented HW/SW implementation as shown in Table 4-3. The functions of the SCU, FICU and UART can be sufficiently covered by the SoC platform and application-specific software, whereas the computationally intensive data processing of multichannel DOT/ECG/EEG signals is

implemented as an AHB-compatible hardware IP extension. Based on this function mapping, the resulting architecture block diagram of the complete biomedical system is shown in Figure 4-6. In the following sections, the design of the DSP IP core is presented first, after which the full system operation is described in detail.

Table 4-3 Remapping of original functional blocks to SoC implementation;  
 \* modified to support output of ICA UW matrix for downstream deartifact calculation

Function	Original (non-SoC)	Remapped (SoC)			
		Hardware (HW)		Software (SW)	
		Platform	Extension	Platform	Extension
System control	SCU	--	--	--	Reset, clock, initialization and main control
Data sampling	FICU	ADC, TMR, GPIO	--	ADC driver, TMR driver, GPIO driver	ADC ISR TMR ISR
Data processing	DSP IP core	--	DSP IP core* + AHB slave wrapper + interrupt logic	--	Core driver + ISR
Comm. I/F	UART	UART	--	UART driver	UART ISR

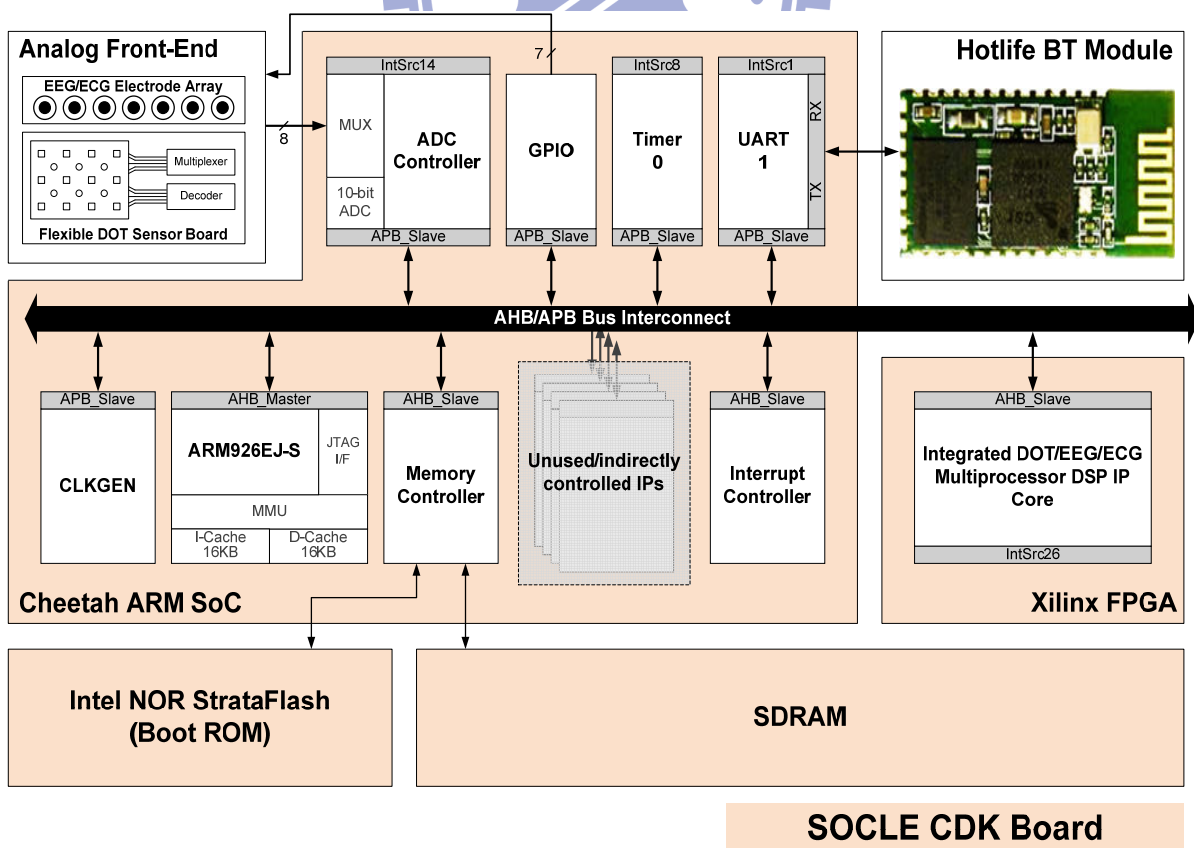


Figure 4-6 Architecture view of SoC-based integrated biomedical multiprocessor system



#### 4.2.2 Integrated DOT/ECG/EEG Multiprocessor DSP IP Core

The integrated biomedical multiprocessor IP connects to the Cheetah ARM SoC platform via an AMBA AHB slave interface and an interrupt pin. Control actions such as reset and configuration of the biomedical IP are performed through software register writes via the AHB interface. Similarly, sending of raw biomedical data and retrieval of processed results to and from the IP core are handled by programmed I/O (PIO) register writes and reads respectively. To avoid wasteful register data polling for processed biomedical data output, an interrupt pin indicating their availability in the output FIFO for reading is provided. The complete list of software-programmable AHB registers supported by the biomedical IP core is shown in Table 4-4 and the hardware architecture is illustrated in Figure 4-7.

The biomedical IP is typically operated as follows. First, the main biomedical IP core is reset by setting the `BIO_IP_RESET` register to 1. Next, `BIO_IP_MODE` is configured with the desired operating mode according to Table 3-6 and the FIFO interrupt threshold value is set by writing to `BIO_IP_THRESHOLD`. Raw biomedical data are input to the three processing engines by writing to `BIO_IP_DOT_IN`, `BIO_IP_ECG_IN` and `BIO_IP_EEG_IN` periodically according to their respective biomedical signal sampling frequencies. After processing, the results are multiplexed by the PDS and packed by the lossless compression engine (COMP) into its output FIFO according to 3.3.1.4. When the number of entries in the output FIFO reaches the value set in `BIO_IP_THRESHOLD`, the interrupt logic issues an interrupt to indicate availability of biomedical data packets for reading. The software, in the form of an interrupt service routine (ISR), finally reads out the data packets from `BIO_IP_FIFO_OUT`.

In this version of the biomedical multiprocessor design, the ICA engine was slightly modified to support outputting of the ICA UW matrix for downstream deartifact calculations. The resulting modified data format for ICA data packets replaces Figure 3-17 and is shown in Figure 4-8.

Table 4-4 Description of AHB registers supported by the biomedical DSP IP core

Address Offset	Name	Access	Description
+0x0000	BIO_IP_RESET	W	Reset and initialization
+0x0004	BIO_IP_MODE	RW	System operation mode
+0x0008	BIO_IP_DOT_IN	W	fNIR sample input (10-bit, LSB-aligned)
+0x000C	BIO_IP_ECG_IN	W	ECG sample input (10-bit, LSB-aligned)
+0x0010	BIO_IP_EEG_IN	W	EEG sample input (10-bit, LSB-aligned)
+0x0014	BIO_IP_FIFO_OUT	R	Mixed biomedical data output (8-bit, LSB-aligned)
+0x001C	BIO_IP_THRESHOLD	RW	FIFO interrupt threshold; minimum number of entries in the output FIFO to cause interrupt assertion

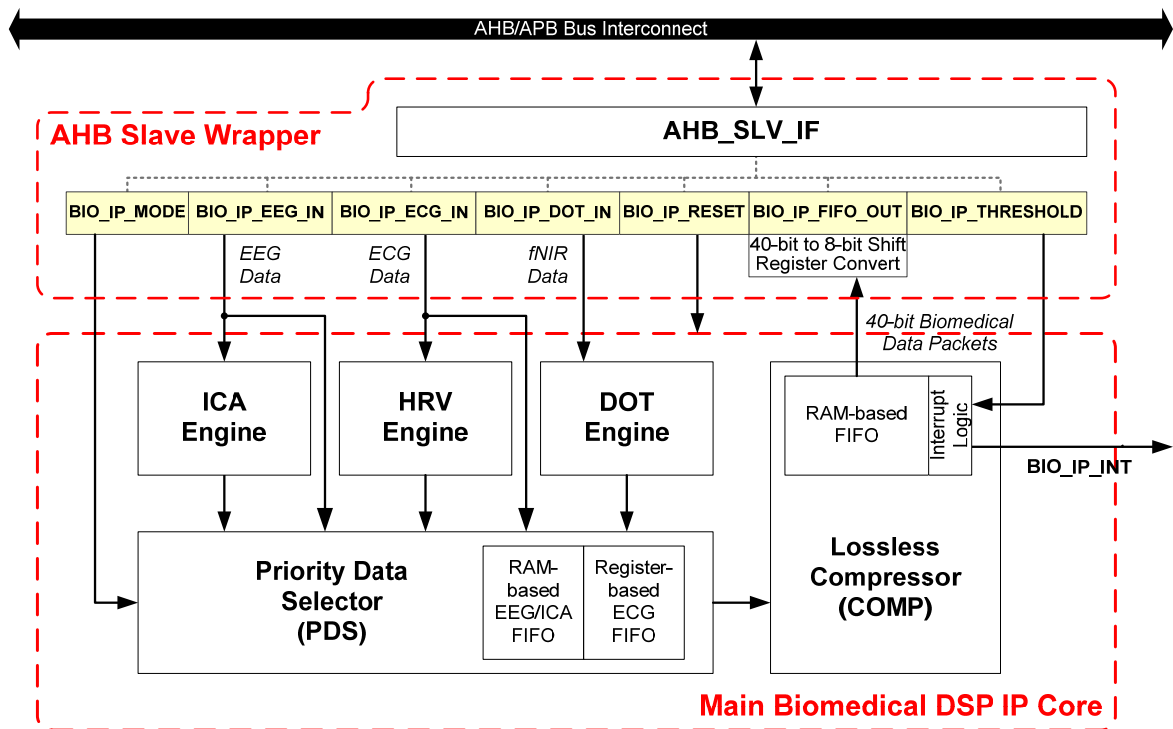


Figure 4-7 Architecture of the AHB-compatible biomedical multiprocessor DSP IP core

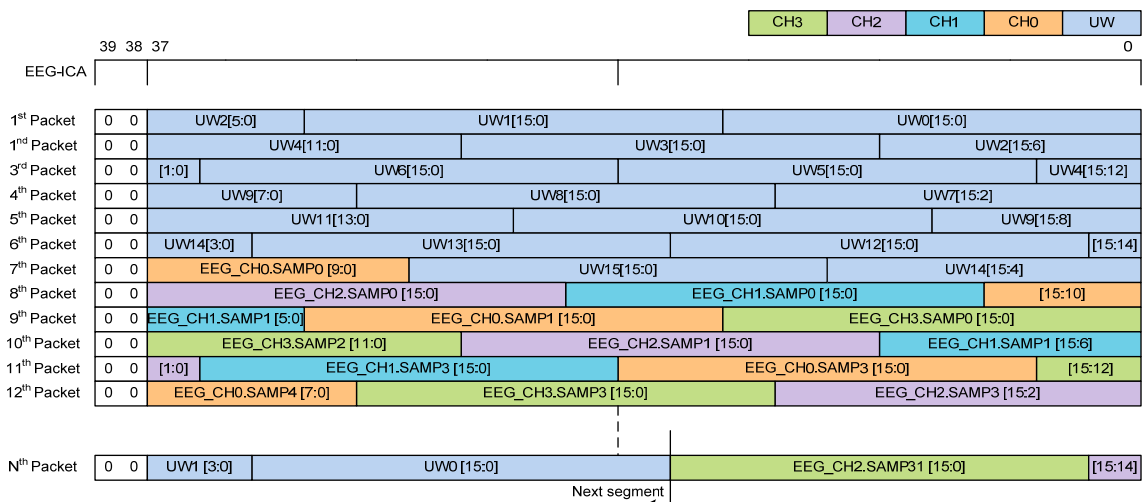


Figure 4-8 Modified ICA data packets; UW matrix transmitted row-wise

### 4.2.3 System Operation Details

Due to the orthogonality of function versus architecture characteristic to SoC-based design, the system operation of the biomedical multiprocessor system is not apparent from the SoC architecture shown in Figure 4-6. Therefore, in this section, a discussion detailing the system operation is provided. The system operation can be generally divided into three distinct threads, namely 1) system mode command reception and consequent operation of the biomedical multiprocessor system; 2) biomedical data sampling and consequent sample input and processing by the DSP IP core; and 3) transmission of processed results from the IP core back to the science station.

The system mode is chosen by the user at the science station and transmitted to the SOCLE CDK's UART device to trigger start of operation. The UART device, upon receiving the system mode, asserts its interrupt and the CPU branches to the `uart_isr()`. The `uart_isr()` clears and disables all other modules and their associated interrupts, and configures the DSP IP according to the system mode retrieved from the UART receive register. The timer is enabled for periodic 256Hz countdown and the ADC is readied for biomedical data sampling. Finally, all the interrupts are enabled to prepare for full system operation.

Each time the timer countdown expires, an interrupt is sent to the CPU and the CPU branches to the `tmr_isr()`. Inside the `tmr_isr()`, an AD conversion chain (sequence of channels to be sampled) is set up according to the biomedical sampling frequencies and the system mode configured earlier. The `tmr_isr()` triggers the AD conversion chain by starting the first AD conversion. Upon (each) successful conversion, the ADC asserts its interrupt and the CPU branches to the `adc_isr()`, which moves the converted sample from the ADC to the DSP IP for signal processing. The `adc_isr()` finally triggers the next AD conversion link in the chain, and the process repeats until all AD conversions in the chain are completed.

As processing of the biomedical samples proceeds, the output FIFO of the DSP IP core gets filled. When the number of entries in the output FIFO reaches a specified threshold, the

DSP IP core asserts its interrupt and the CPU branches to the ip\_isr(). The ip\_isr() simply moves data packets from the DSP IP core to the UART for eventual transmission back to the science station.

The UML (unified modeling language) 2.0-like sequence diagram shown in Figure 4-9 illustrates the typical system operation described above and the driver configuration and software ISR behavior are summarized in Table 4-5. The three threads discussed previously are depicted in Figure 4-9 as yellow, blue and purple action state sequences respectively.

Table 4-5 Summary of hardware devices and associated software interrupt handling

HW Block	Function Driver Configuration	Interrupt		
		Meaning	Source/ (Priority)	Software ISR Action
ADC Controller	Biomedical data sampling	Single ADC conversion finished	Source 14 (1)	1. Read recently converted biomedical sample 2. Send sample to appropriate engine in DSP IP 3. Start next AD conversion in conversion chain (if not final conversion) <i>AD conversion chain: ECGx3 (if available) → EEGx4 (if available) → DOTx1 (if available)</i>
	CH0: DOT      CH4: EEG ch3 CH1: EEG ch0    CH5: ECG ch0 CH2: EEG ch1    CH6: ECG ch1 CH3: EEG ch2    CH7: ECG ch2			
Timer0	Generate periodic interrupt	1/256 sec has elapsed	Source 8 (2)	1. Setup AD conversion chain according to the system mode (Table 3-6) and biomedical signal sampling frequencies 2. Start first AD conversion (head of chain) <i>AD conversion chain: ECGx3 (if available) → EEGx4 (if available) → DOTx1 (if available)</i>
	Countdown frequency: 256Hz Periodic: Yes ECG: 256Hz x 4 channels EEG: 128Hz x 3 channels DOT: 1Hz x 1 channel			
DSP IP Core	Biomedical data processing System mode: from UART RX ISR FIFO interrupt threshold: 8 entries	Processed biomedical data output available	Source 26 (3)	If the UART output buffer is empty, transfer 16 bytes of biomedical data from the DSP IP core to the UART transmit register (one byte at a time)
UART1	Data communication Baud rate: 115200 Data bits: 8 Parity: No Stop bits: 1	System command received	Source 1 (0) (highest)	1. Disable Timer0, ADC and DSP IP core 2. Clear and disable all other interrupts 3. Get system mode from UART receive register 4. Reset DSP IP core 5. Initialize DSP IP core with the received system mode and threshold equal to 8 6. Configure and enable Timer0 and ADC 7. Enable Timer0, ADC and DSP IP interrupts
Interrupt Controller	Interrupt management Interrupt Priority: 0 – UART1 RX (highest) 1 – ADC 2 – Timer0 3 – DSP IP core	--	--	--
GPIO	fNIR LED and sensor select --	--	--	--
CLK GEN	Clock management Clock frequency settings: CPU/AHB/APB – 80/10/5 MHz	--	--	--

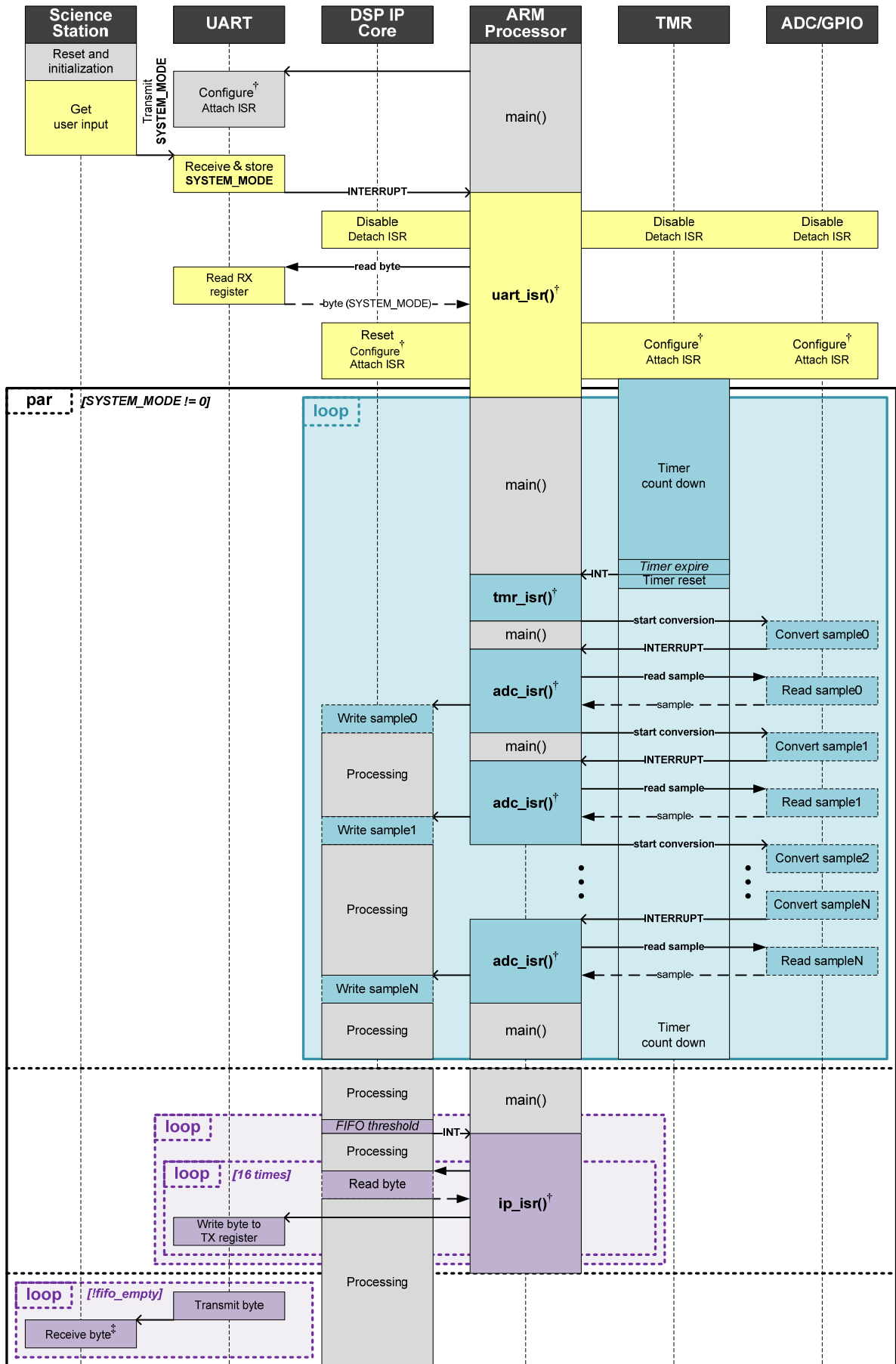


Figure 4-9 Typical system operation sequence; †refer to Table 4-5, ‡refer to 3.3.1.4

### 4.3 Results of Implementation Using the SOCLE CDK

The SoC-based biomedical multiprocessor system based on the previously taped-out design has been successfully ported and implemented using the SOCLE ARM-based SoC Platform Cheetah Development Kit. Application-specific and extension software have been built on top of the reference BSP provided by the platform vendor, resulting in a total software image size of 31kB for total ROM and 51kB for total RAM, using the toolchain configured for an ARM926EJ-S target with RealView debug information. The application-specific biomedical multiprocessor DSP IP implemented on a Xilinx Spartan-3 XC3S4000-5fg676 reports an FPGA device utilization shown in Table 4-6 and can run up to a maximum clock frequency of 15.7MHz. The main DSP IP core is operated using an AHB clock of 10MHz, while the CPU and APB clock frequencies are set to 80MHz and 5MHz respectively.

Table 4-6 Xilinx XC3S4000 FPGA device utilization

Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	10,615	55,296	19%
Number of 4 input LUTs	30,550	55,296	55%
Number of occupied Slices	18,747	27,648	67%
Number of Slices containing only related logic	18,747	18,747	100%
Number of Slices containing unrelated logic	0	18,747	0%
Total Number of 4 input LUTs	30,774	55,296	55%
Number used as logic	30,550		
Number used as a route-thru	224		
Number of bonded IOBs	91	489	18%
Number of RAMB16s	14	96	14%
Number of MULT18X18s	23	96	23%
Number of BUFGMUXs	1	8	12%
Average Fanout of Non-Clock Nets	3.44		

An unavoidable consequence of porting the biomedical multiprocessor IP from an IC to an FPGA implementation is the necessary change in the IP libraries used. Synopsys DesignWare IP blocks used in the IC implementation must be replaced with components available in Xilinx ISE's CORE Generator IP library. At times, the interfaces and functionality are not perfectly the same, and additional logic must be introduced to successfully do the porting. Therefore, to verify the integrity of the new target FPGA implementation after the porting process, golden patterns used during verification of the IC implementation are employed to allow comparison between the old (IC) and new (FPGA) implementations.

Figure 4-10 shows the verification setup for the SoC-based biomedical multiprocessor IP. Golden raw biomedical signal data are introduced into the system by pre-loading them into system memory through an in-circuit emulator (ICE) connected to the ARM processor. In addition, the `adc_isr()` is modified to retrieve these data patterns from memory instead of real-time biomedical data from the ADC. The golden patterns are sent to the respective biomedical DSP engines and the processed results are sent over Bluetooth and saved to file on a remote PC using free commercial serial port monitoring software. Finally, the file contents are decoded by a MATLAB packet decoder and verified to match with golden expected data. The tests are repeatedly performed under various system modes. On a separate test case for verifying the complementary ADC function, the sampling of various fixed-frequency sinusoids has also been verified through frequency spectrum analysis using MATLAB.

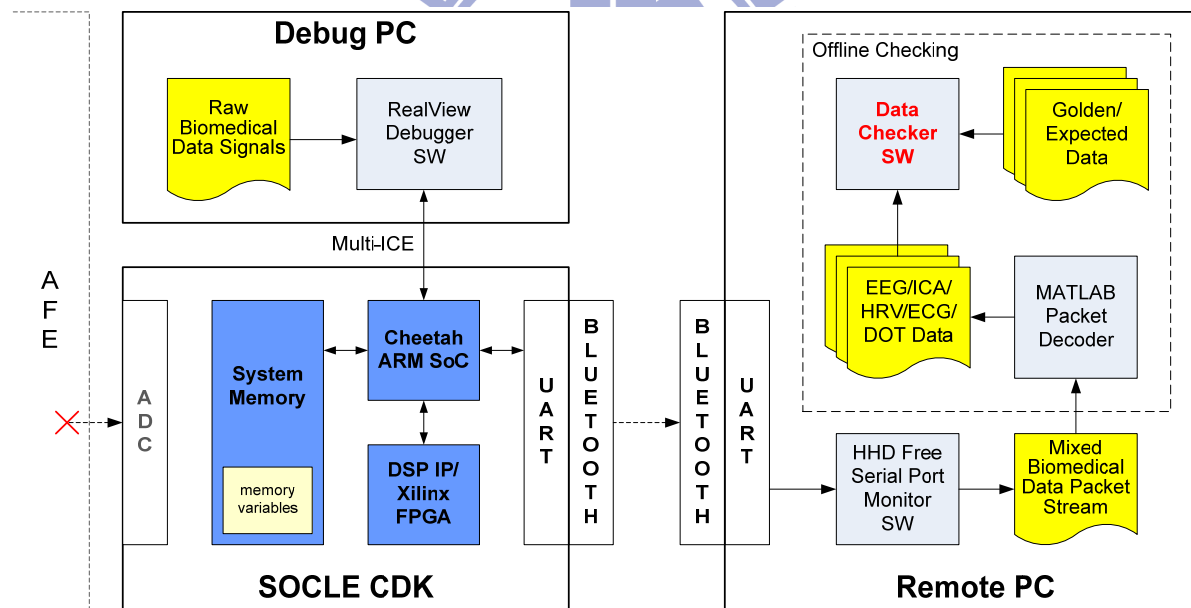


Figure 4-10 Verification setup for the SoC-based biomedical multiprocessor system

#### 4.4 Real-Time Application

With the biomedical multiprocessor system implemented and verified as described in the previous section, we proceed to discuss the employment of the newly developed integrated DOT/ECG/EEG multiprocessor system in various real-time biomedical applications. The general real-time demonstration setup comprises 1) a human test subject, 2)

the proposed system, which represents a portable biomedical sensing device worn by the patient, and 3) a science station model, which represents a remote biomedical application apparatus that delivers real-time services to 4) a medical expert. In this thesis, the prototype development and demonstration of two application models are presented.

#### 4.4.1 Real-Time Integrated Brain-Heart Monitoring

In the first application, the remote biomedical device takes the form of an SoC-based portable wireless science station modeled using the SOCLE CDK. The science station supports user selection and transmission of the type of examination (system mode) to be performed, as well as real-time decoding and display of multi-channel DOT, ECG, EEG, ICA or HRV data received as in Figure 4-9. These application-specific features are developed as an all-software differentiation built on top of the SOCLE Cheetah SoC platform. A configuration wizard user interface (UI) is employed to allow user selection and transmission of the system mode, while decoding and LCD display of the biomedical samples are supported in a UART receive interrupt service routine. The architectural block diagram of the application demo is shown in Figure 4-11 and some photos showing the demonstration setup and operation are shown from Figure 4-12 to Figure 4-16.

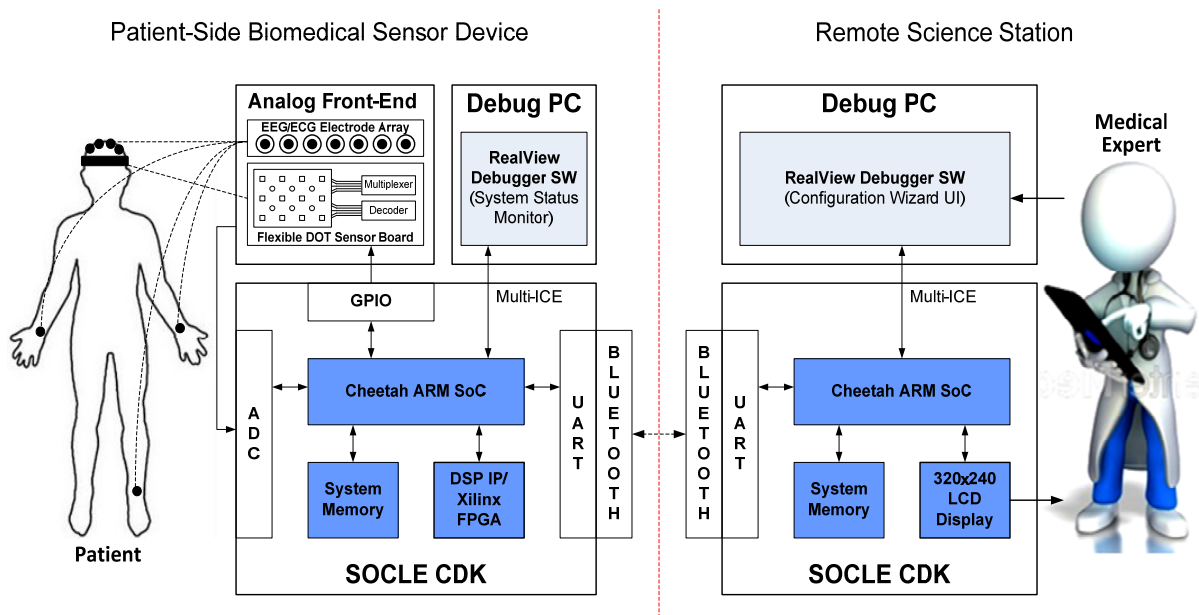


Figure 4-11 Block diagram of the real-time integrated brain-heart monitoring application



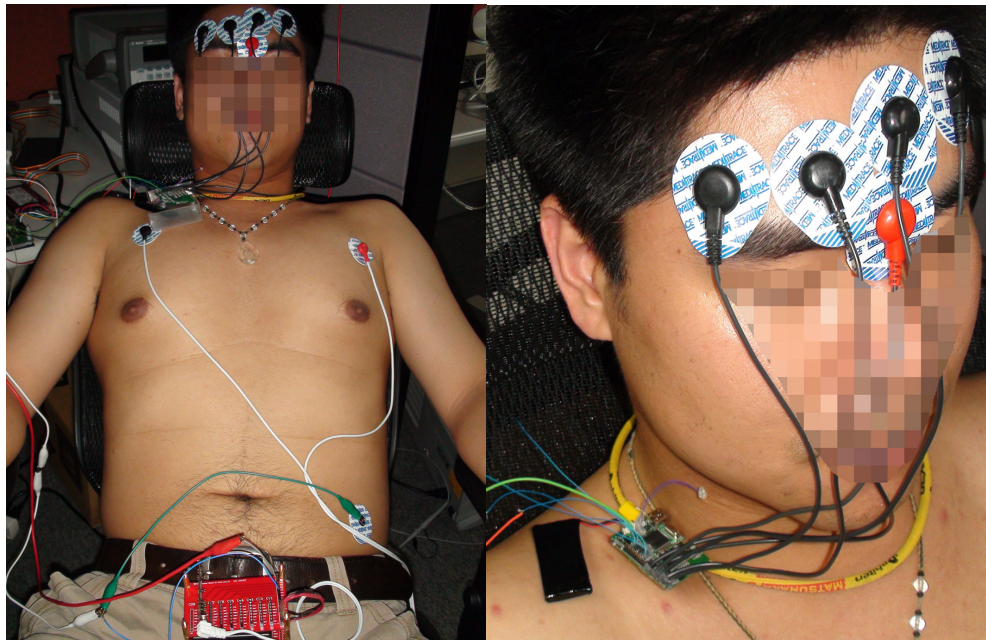
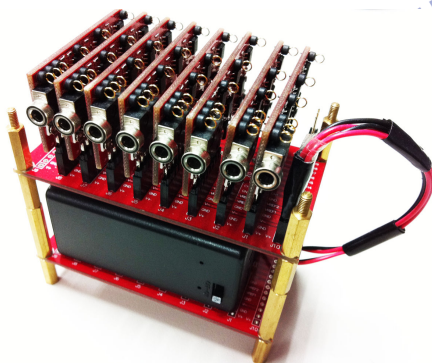
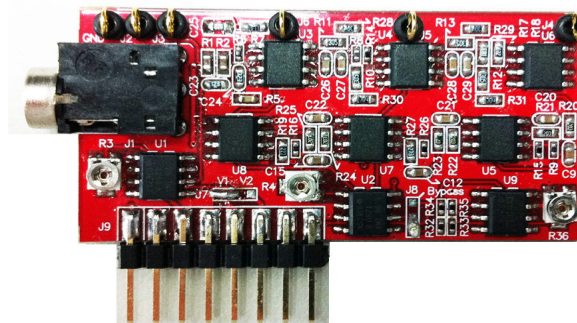


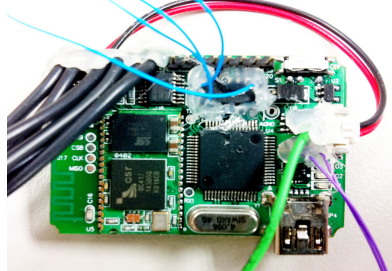
Figure 4-12 EEG and ECG electrode placement on human test subject



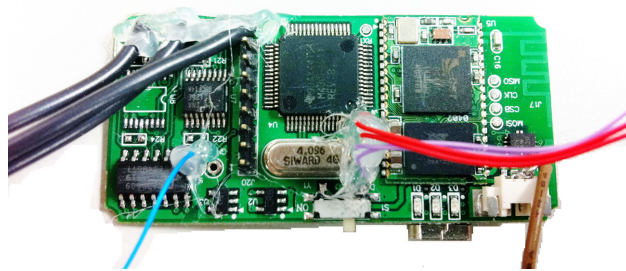
(a) 8-ch max. EEG/ECG AFE



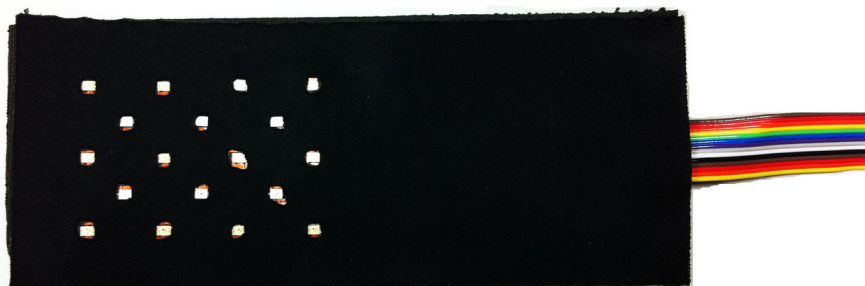
(b) Single channel IA+filter AFE



(c) Alternate 4-ch EEG AFE



(d) Alternate 3-ch ECG AFE



(e) DOT LED source and sensor array

Figure 4-13 Analog front-end (AFE) modules used

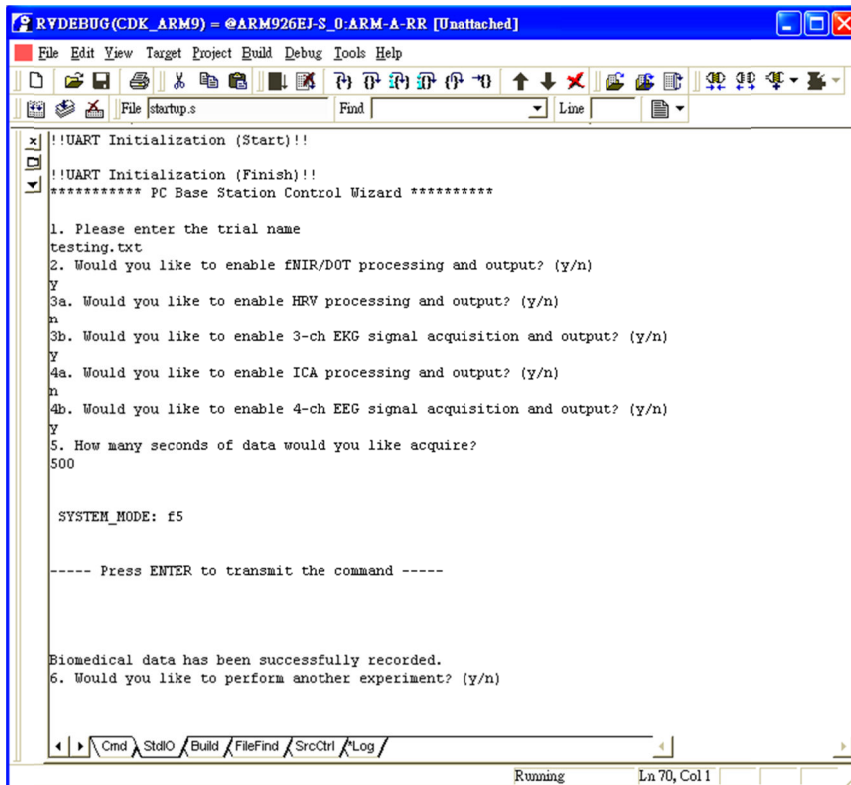


Figure 4-14 Configuration wizard user interface on science station

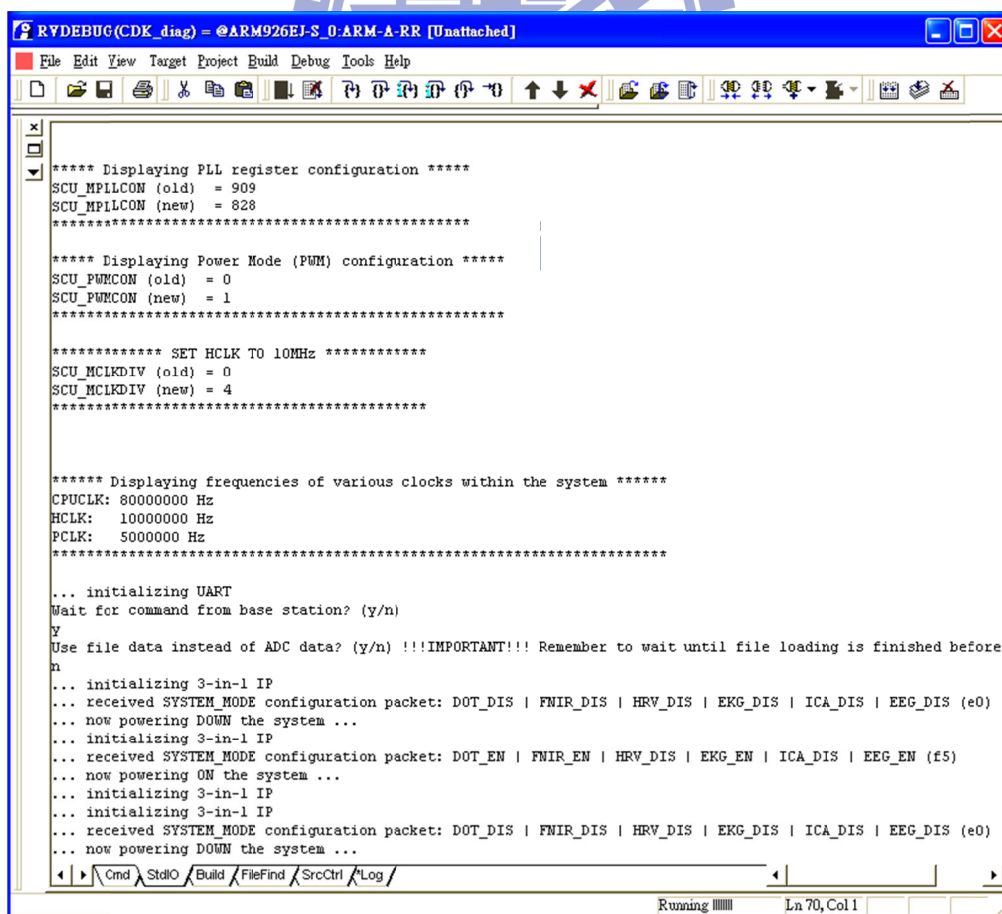


Figure 4-15 System status monitor on patient-side sensing device



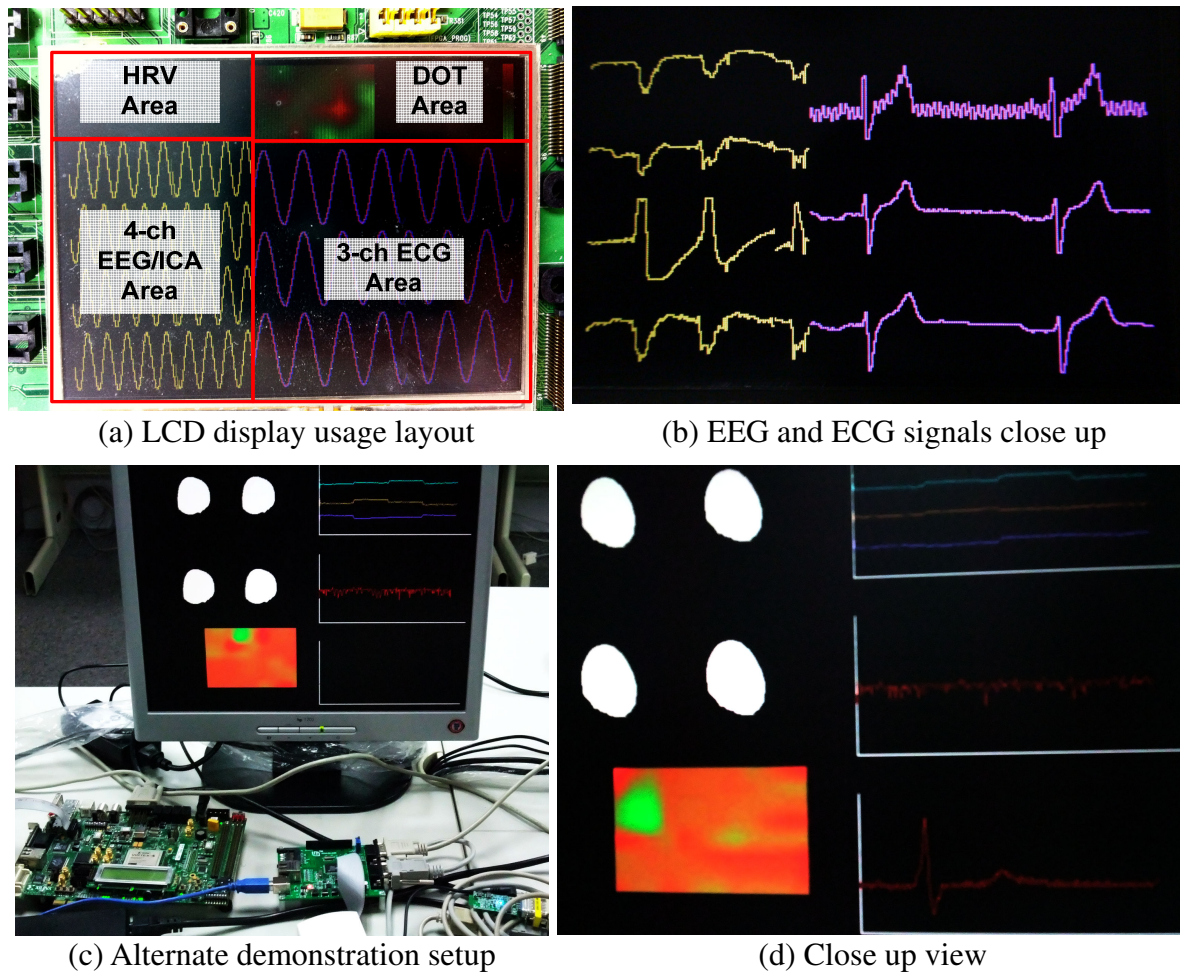


Figure 4-16 Real-time biomedical data display on LCD

## 4.4.2 SSVEP-based Brain-Computer Interface

### 4.4.2.1 Introduction to UCSD's SSVEP-based Brain Dialer System

In the second application, the proposed system is integrated with University of California at San Diego's (UCSD) SSVEP-based brain dialer system [72]. SSVEP, which stands for steady-state visually evoked potential, is the brain's natural response to visual stimulus that flickers at a specific frequency. When the retina is excited by a stimulus ranging from 6 Hz to 75 Hz, the brain generates electrical activity (i.e. EEG) of the same (or multiples of) frequency as the visual stimulus [73][74]. UCSD's brain dialer system takes advantage of this physiological phenomenon to allow a user to dial a phone number using only EEG.

A block diagram of the original system reproduced from [72] is shown in Figure 4-17. Buttons of a virtual telephone keypad comprising digits 0-9, '\*' and '#' are frequency-coded

and displayed onscreen as 12 flickering rectangles ranging from 9 to 11.75Hz with 0.25Hz interval between consecutive keys. To dial a number or symbol, the user focuses his visual field onto a specific rectangle; consequently evoking an SSVEP with energy centered about the target key's associated frequency. The SSVEP-containing EEG is then acquired by a 4-ch EEG headband unit [75] worn by the user, and transmitted to a nearby commercial cellphone via Bluetooth. The cellphone, whose microprocessor is programmed to perform various time- and frequency-domain signal processing algorithms, identifies the dominant frequency in the EEG and determines the user's intended key. Upon successful detection, the cellphone prompts the user with an auditory feedback for the next entry. When the phone number is complete, the cellphone finally dials out the digits and symbols to place the call.

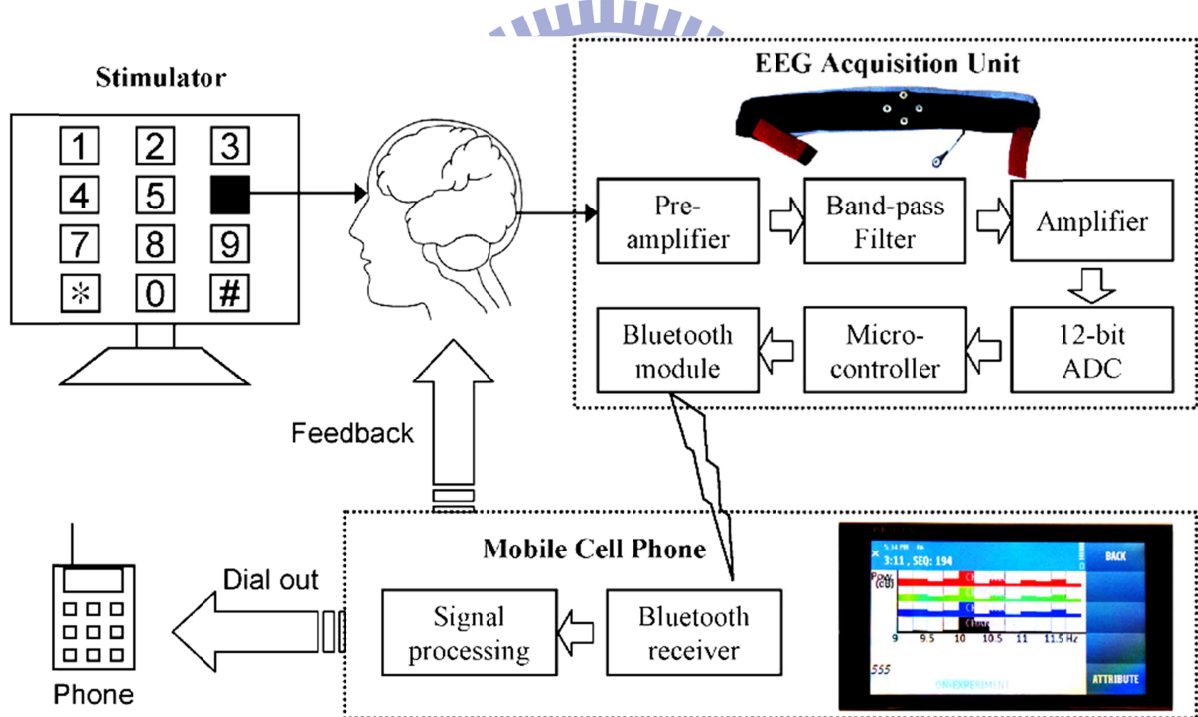


Figure 4-17 Original SSVEP-based brain dialer system of UCSD

#### 4.4.2.2 Integration with Proposed Biomedical Multiprocessor System

Although the published work [72] reports promising results, artifactual noise such as those caused by eye blinks and body movements disrupt the accuracy of the system, as a result limiting its practical application in daily life settings. Therefore, the employment of an ICA preprocessor for the removal of these EEG artifacts is hypothesized. However, neither

the EEG acquisition unit nor the cellphone is powerful (or efficient) enough to take on the heavy computational load of ICA; thus, a hardware-based approach such as our proposed biomedical multiprocessor system is offered as a real-time solution to the problem.

To explore the feasibility of ICA in the improvement of SSVEP detection in EEG signals, a preliminary investigation with UCSD is performed using the collaborative system integration development and evaluation setup shown in Figure 4-18. The setup comprises 1) UCSD's VisualStim software, which can generate visual stimuli of various frequencies, sizes, colors, etc. for operation on different monitors with different refresh rates (Figure 4-19); 2) NCTU Brain Research Center's 4-ch EEG module, tapped at its final analog output (Figure 4-13c); 3) our proposed biomedical multiprocessor implemented on the SOCLE CDK; 4) a PC/Laptop-based real-time EEG display and analysis Java software, which uses the canonical correlation analysis (CCA) method for SSVEP detection [76]; and 5) a set of MATLAB tools used for offline analysis. Since the ICA engine does not perform EEG artifact removal, the application software of the biomedical multiprocessor SoC is extended to support this function. The details of this development are described in the pages that follow.

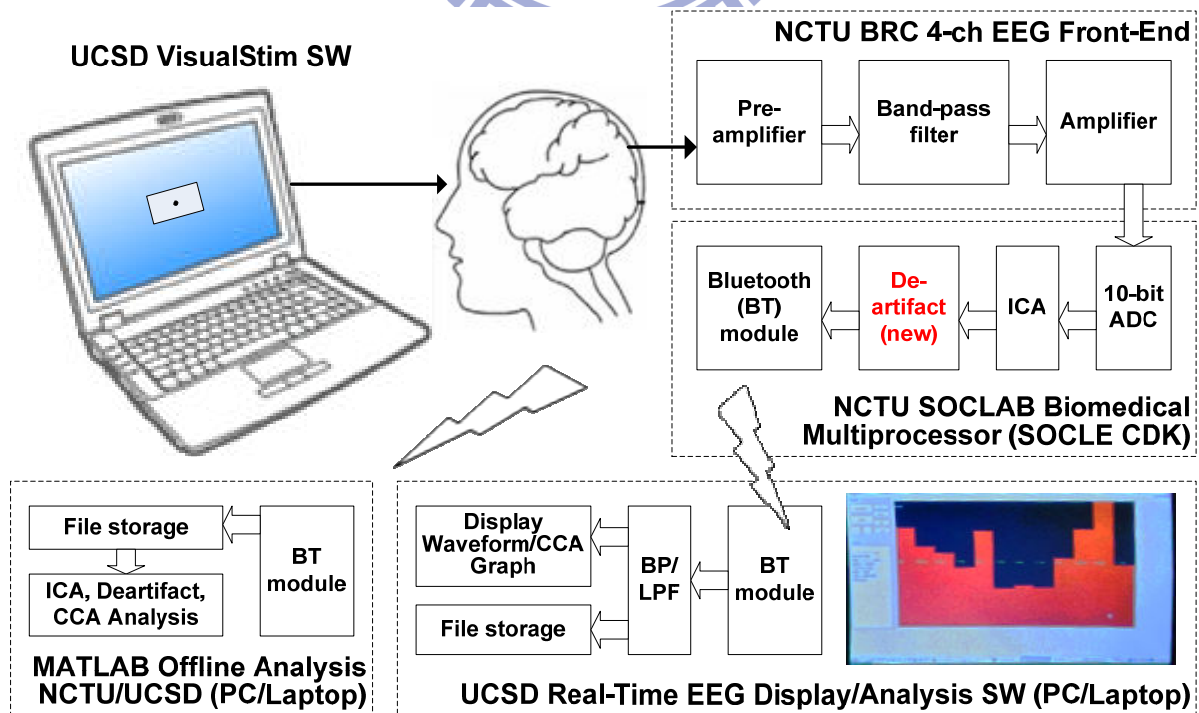


Figure 4-18 System integration development and evaluation setup

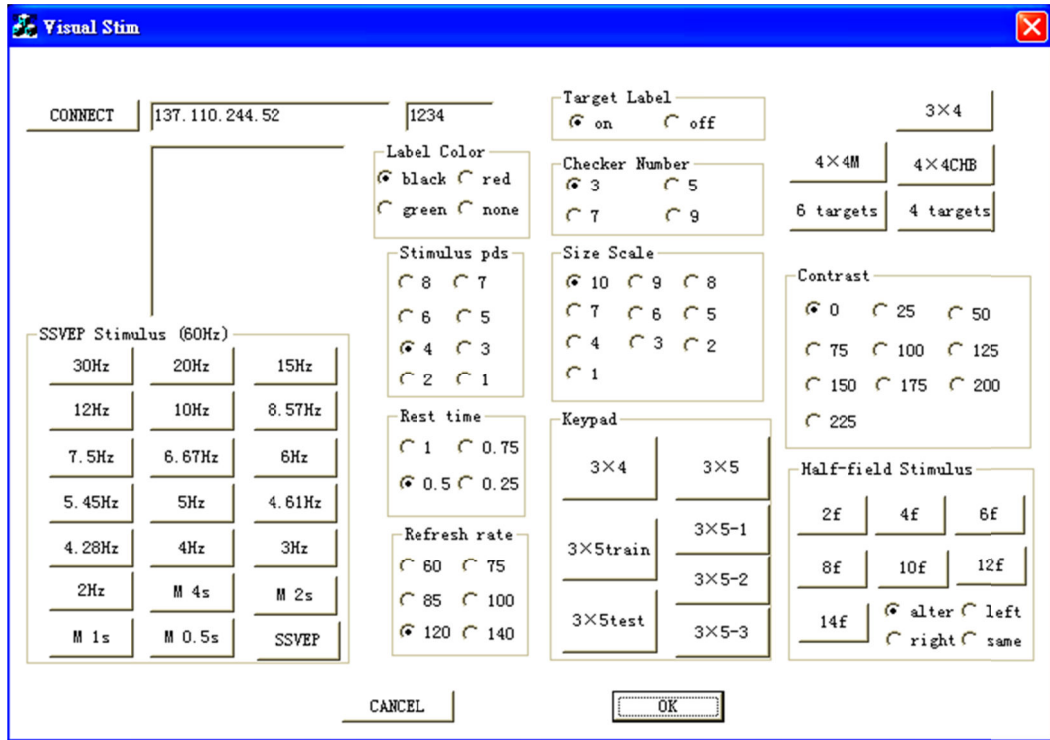


Figure 4-19 UCSD's VisualStim SSVEP stimulus generation software

The functional diagram of the biomedical multiprocessor SoC modified to support the de-artifact function is shown in Figure 4-20. A pure-software block distributed over `ip_isr()` and `main()` is inserted between the biomedical multiprocessor IP and UART to perform the artifact removal function. The system mode is configured to ICA only (0xE3).

When the biomedical multiprocessor IP asserts its interrupt, instead of moving the ICA data (i.e. unmixing matrix, independent component samples) directly to the UART for outputting (as in the case of the original design), the `ip_isr()` unpacks and pushes them into a software FIFO for de-artifact processing. The software FIFO facilitates data passing from `ip_isr()` to `main()`, such that intensive de-artifact computations can be implemented in the `main()` function instead. As a result, the `ip_isr()` can be kept as light and short as possible.

The `main()` function loops infinitely to retrieve and process any data that becomes available in the FIFO. The `main()` function performs the de-artifact function as illustrated in Figure 3-3. Matrix inversion is performed on the first 16 samples (see Figure 4-8) according to a fast inverse calculation method [77][78] to generate the mixing matrix. The independent component (i.e. channel) considered by the user as the artifact is removed by zeroing out each

incoming sample in that channel. The remaining independent component channels are then remixed, packed and finally output as artifact-free EEG provided to UCSD's brain dialer system.

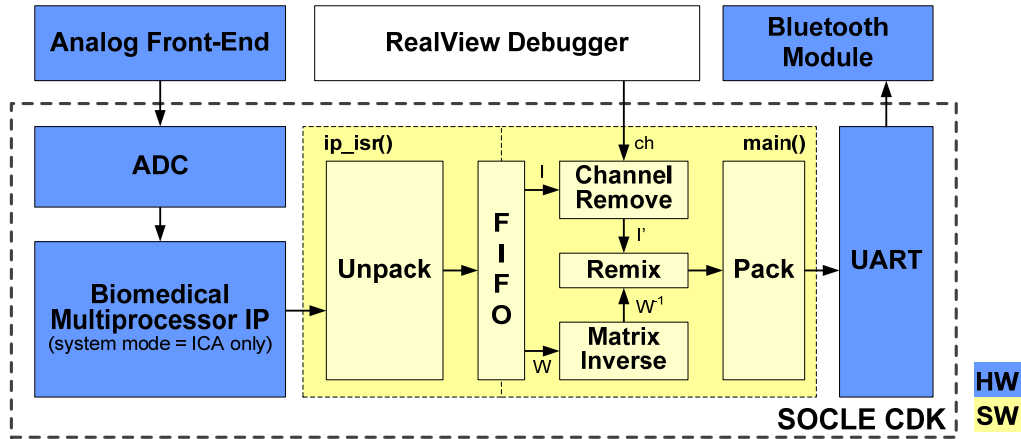


Figure 4-20 Software extension to support de-artifact function

#### 4.4.2.3 Results of Integration

A photograph of the actual system integration and evaluation setup is shown in Figure 4-21. The laptop on the left is set up to run UCSD's VisualStim software and the other configured for real-time EEG display and CCA analysis. In between are 1) the test subject wearing the EEG acquisition module connected to 2) the SOCLE CDK on which the proposed biomedical multiprocessor system offering the ICA/de-artifact solution is implemented. EEG electrode placement is chosen for optimal observation of the visual cortex, so as to maximize the SNR of the SSVEP.

Prior to operating the system according to its target application, a series of progressive sanity checks were performed first. These checks included 1) stabilizing the Bluetooth connection; 2) checking the integrity of golden test patterns (ex. sine waves) transmitted from the SOCLE CDK to the analysis laptop; and 3) ensuring the quality of EEG signal acquisition by visually checking for alpha waves (by closing the eyes) and eye blink artifacts on UCSD's real-time EEG display software. Finally, the matrix inverse and remixing operations in Figure 4-20 were also verified by checking that raw input EEG can be reconstructed when all four independent component signals are remixed.



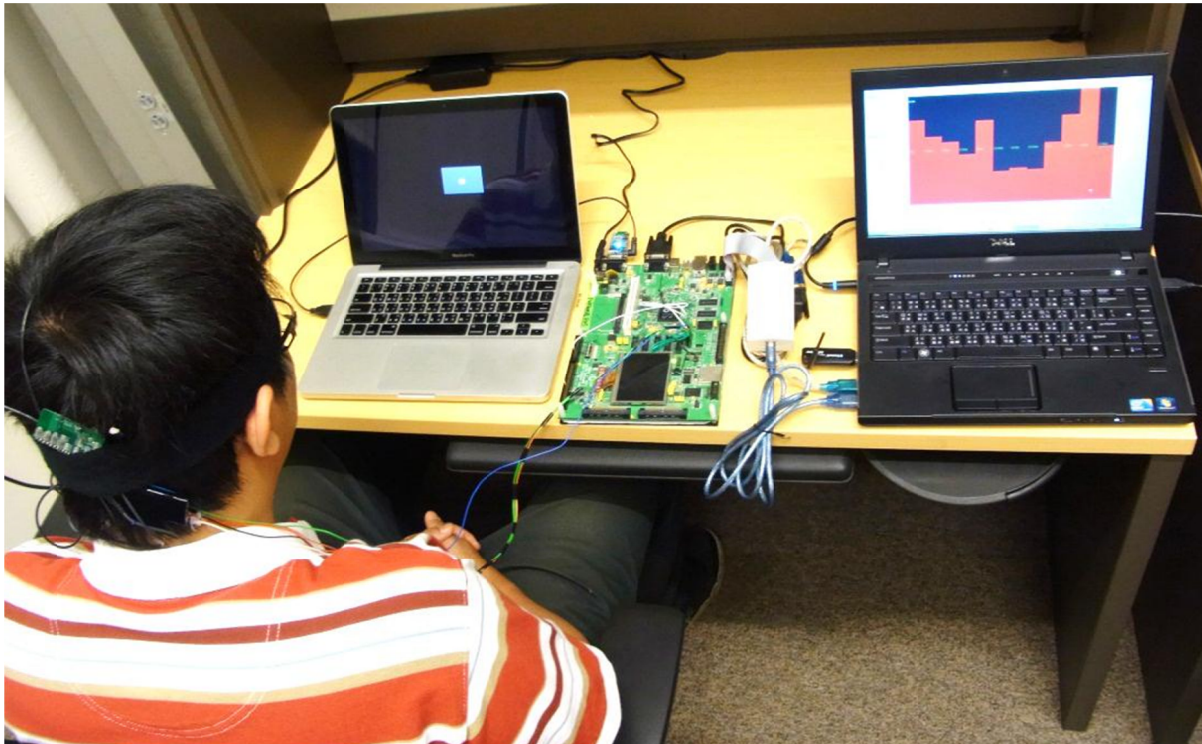


Figure 4-21 Actual system integration and evaluation setup, showing real-time CCA plot of 9Hz SSVEP without ICA.

With the system stabilized, the setup is operated according to the target application. Figure 4-21 shows a snapshot of the system running for the real-time detection of 9Hz SSVEP. The CCA analysis software calculates a power spectrum estimate of the received EEG and displays the result as a bar graph plot of 12 vertical bars corresponding to the 9Hz up to the 11.75Hz frequencies. Taller bars suggest higher correlation with a particular frequency.

The CCA plot is initially generated from 2 seconds worth of EEG and afterwards recalculated and refreshed every second taking into account new EEG data until the SSVEP is detected. The particular frequency that stays dominant for two consecutive frames is taken as the detected SSVEP. In Figure 4-21, wherein ICA was turned off, CCA failed to identify the 9Hz SSVEP, since the tallest bar corresponds to 11.5Hz. In contrast, in Figure 4-22, the first bar (9Hz) was most prominent for two consecutive frames, indicating the successful detection of 9Hz SSVEP when ICA is enabled.



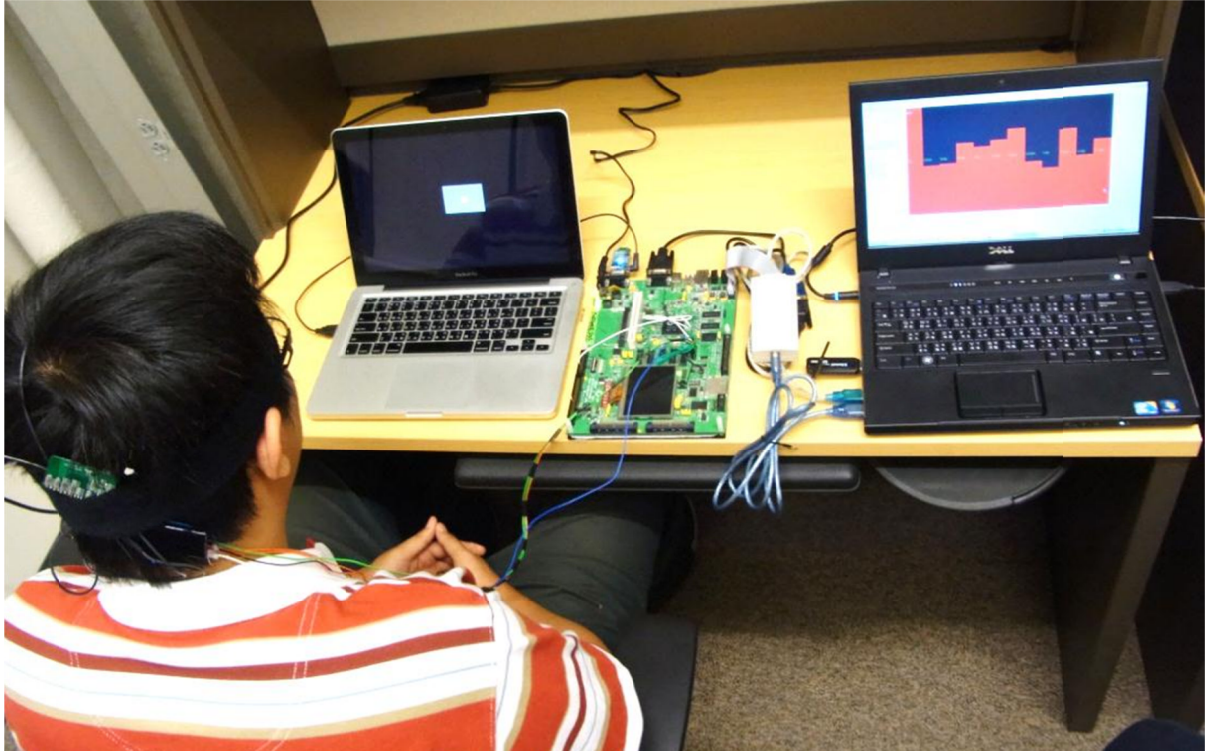


Figure 4-22 Real-time CCA plot of 9Hz SSVEP with ICA enabled

Since it is difficult to compare SSVEP detection performance using the real-time setup, a collection of offline MATLAB tools is employed to help visualize the sequential nature of the SSVEP detection process. Four progressive CCA plots corresponding to 5 seconds of real-time operation are superimposed onto a single plot using MATLAB. Detection latencies greater than 5 seconds are considered unacceptable (from a system point of view) and are therefore excluded from the analysis.

The results in Figure 4-23 to Figure 4-25 show the MATLAB CCA plots of various 5-second EEG segments taken from the 9Hz, 10Hz and 11Hz stimulus test cases, both with and without ICA, for a total of six cases. Table 4-7 shows the summary of the detected SSVEPs in these particular cases. Using raw EEG, false SSVEP detection occurs and in one case fails altogether. In contrast, all SSVEPs are detected correctly when ICA is enabled. These particular results suggest the possible effectiveness of ICA as a means to improve detection of SSVEPs from EEG signals.

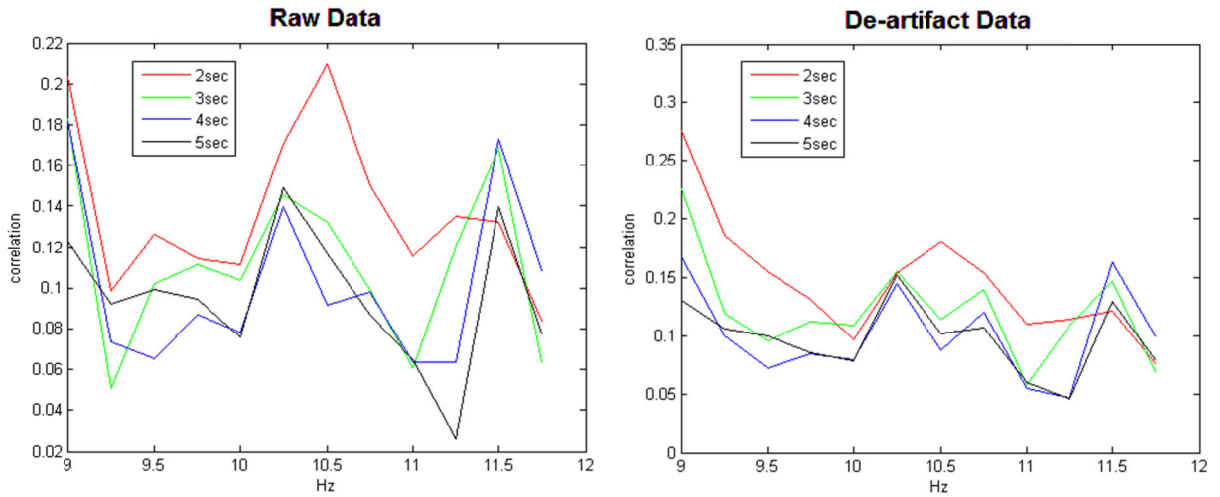


Figure 4-23 MATLAB CCA plot for 9Hz stimulus

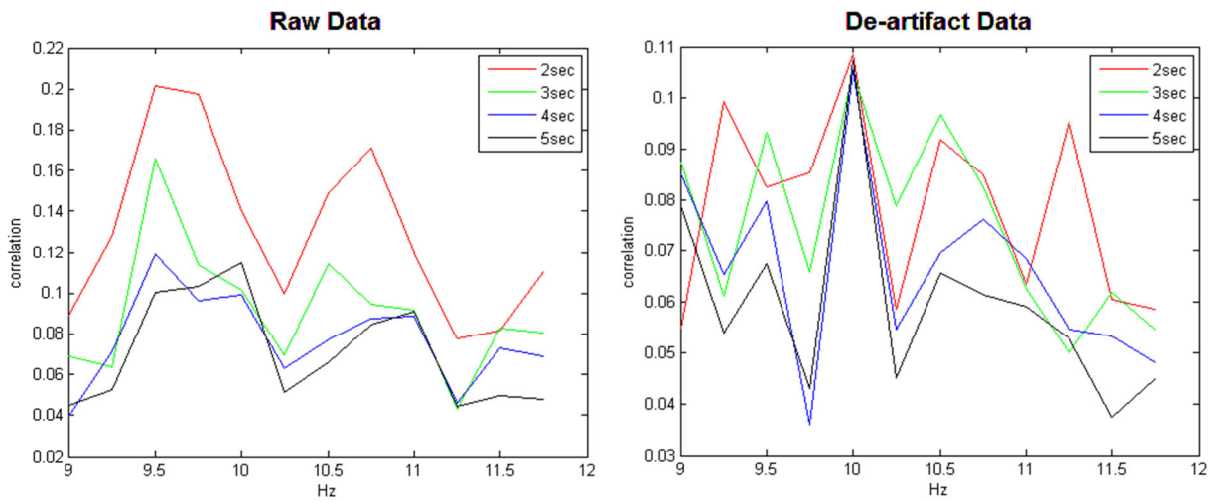


Figure 4-24 MATLAB CCA plot for 10Hz stimulus

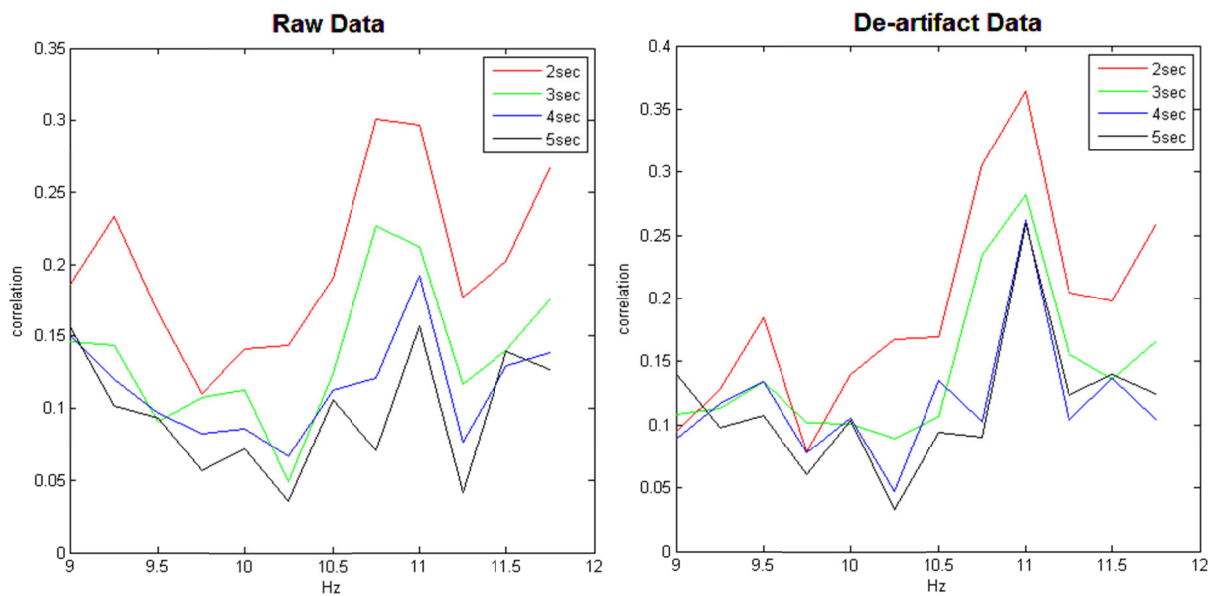


Figure 4-25 MATLAB CCA plot for 11Hz stimulus

Table 4-7 Detected SSVEP frequencies in Figure 4-23 to Figure 4-25

Stimulus Frequency	Detected SSVEP	
	w/o ICA	with ICA
9Hz (Figure 4-23)	FAIL	9Hz
10Hz (Figure 4-24)	9.5Hz	10Hz
11Hz (Figure 4-25)	10.75Hz	11Hz

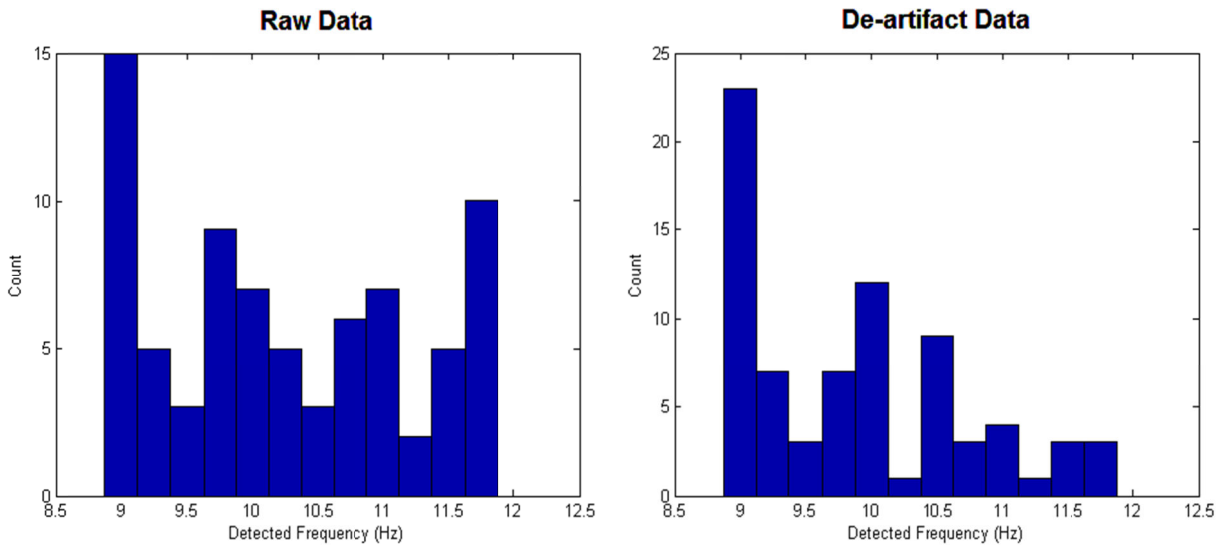
In order to achieve a more thorough and quantitative comparison of SSVEP detection performance with and without ICA, all 6 minutes of recorded EEG data comprising the 6 cases are analyzed (1 minute per case). In each case, histograms of the detected frequencies are generated using both 1-second and 2-second CCA analysis. For the case of 2-second CCA analysis, a sliding window of 1 second is utilized. The results using 1-second and 2-second windows are shown in Figure 4-26 and Figure 4-27 respectively.

When a 1-second CCA window is used, the number of instances the SSVEP was detected correctly increased significantly by 53%, 48% and 44% for the three stimulus frequencies. On the other hand, ICA is only slightly helpful, at times even detrimental to SSVEP detection, when a 2-second CCA window is used. The details of the comparisons are summarized in Table 4-8.

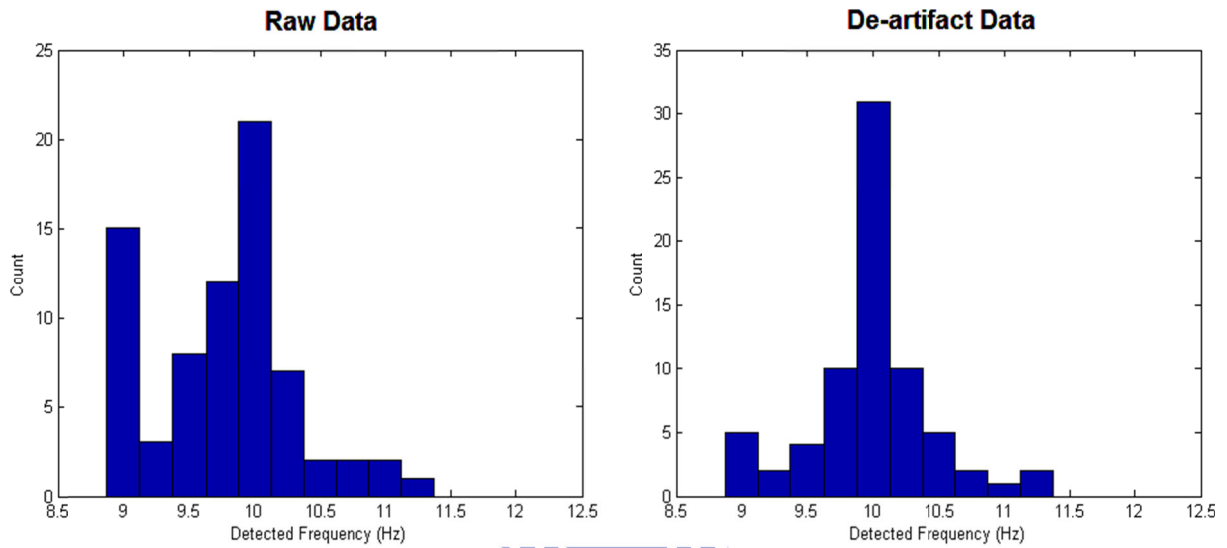
Based on these results, it can be concluded that at least for smaller CCA windows, ICA is an effective means for improving the detection of SSVEPs in EEG signals. Within the context of the brain dialer system, this implies that ICA can assist in improving the accuracy of key detection when faster dial times are desired.

Table 4-8 Improvement in SSVEP detection using ICA, 1s CCA vs 2s CCA

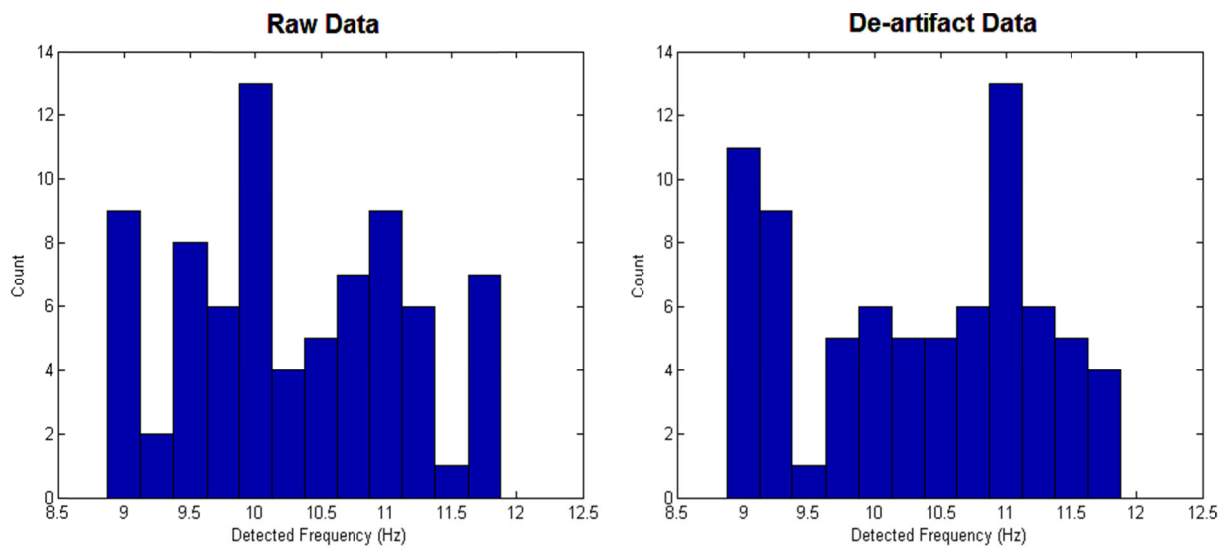
Stimulus Frequency	SSVEP Detection Improvement w/ ICA	
	1-second CCA	2-second CCA
9Hz	53%	-18%
10Hz	48%	20%
11Hz	44%	11%



(a) 9Hz stimulus

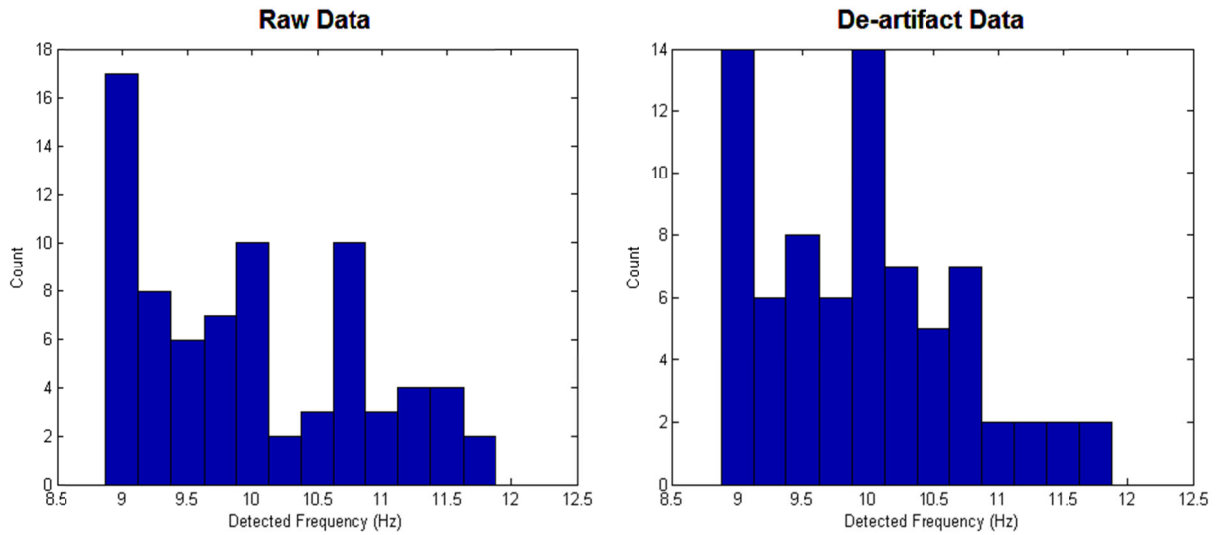


(b) 10Hz stimulus

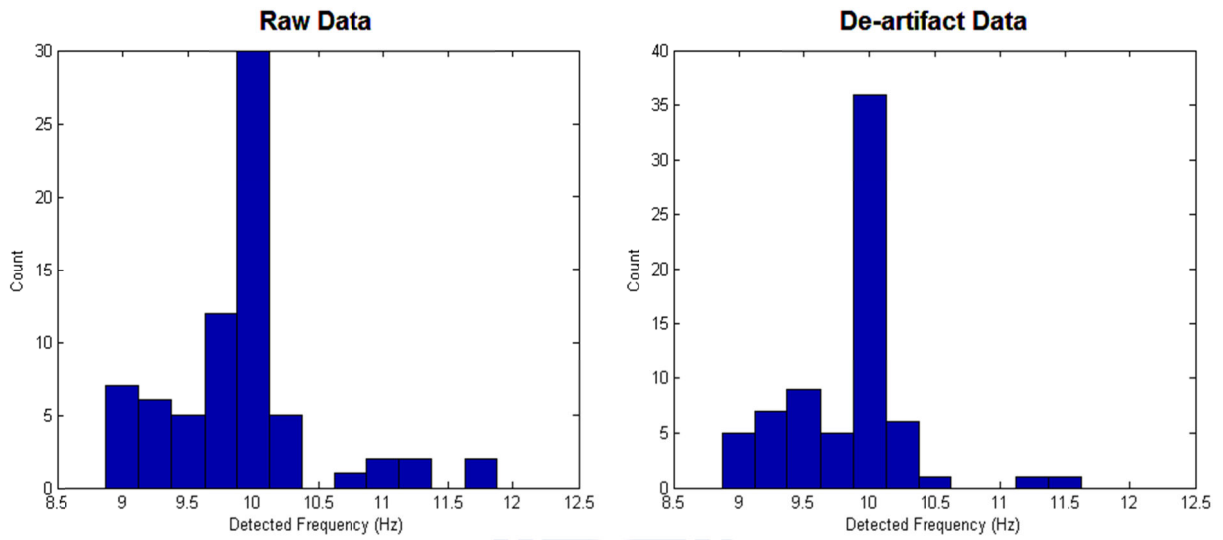


(c) 11Hz stimulus

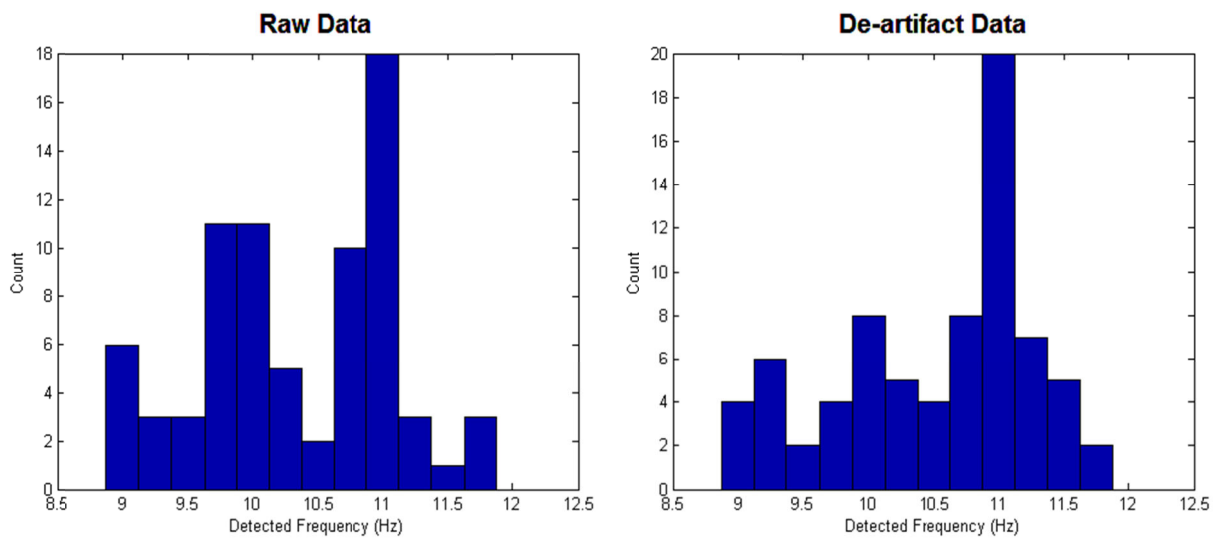
Figure 4-26 Histogram of detected SSVEPs (1-second CCA analysis)



(a) 9Hz stimulus



(b) 10Hz stimulus



(c) 11Hz stimulus

Figure 4-27 Histogram of detected SSVEPs (2-second CCA analysis)

## Chapter 5 Conclusion and Future Work

### 5.1 Conclusion

In this thesis, a highly-integrated biomedical multiprocessor design for portable wireless brain-heart monitoring systems has been developed for the technological advancement of medical emergency care, long-term observation, personal home care and cognitive science.

Various advanced biomedical signal processing algorithms – diffuse optical tomography (DOT) for brain imaging, independent component analysis (ICA) for removing artifacts in brain electroencephalogram (EEG) signals, and heart rate variability (HRV) analysis for monitoring heart electrocardiogram (ECG) signals have been researched, developed, designed and integrated into a complete hardware multiprocessor system.

The design has been functionally verified and implemented both as an IC using UMC 65nm CMOS technology and as an AHB-compatible IP for ARM-based SoCs on a Xilinx FPGA validated using an SOCLE CDK. The biomedical multiprocessor IC features a small chip area of  $1.73\text{mm}^2$  and a low estimated power consumption of 3.6mW and can hence be employed in small, portable and wireless biomedical devices. Alternatively, while the pre-verified biomedical SoC IP can be offered to third-party developers as a proven biomedical solution ready for practical integration into ARM-based SoCs, it can also be reused and extended in more sophisticated applications in the future. In conjunction with an analog front-end sensing circuit and a Bluetooth communications module, the developed biomedical multiprocessor hardware can be packaged as a solution to assist in the detection, diagnosis and monitoring of brain- and heart-related illnesses.

To demonstrate the functionality and real-time operation of the proposed design in the context of complete biomedical monitoring systems, the SoC-based multiprocessor implemented on the SOCLE CDK is featured in two real-time biomedical application settings. In the first application, the proposed design models a patient-worn biomedical sensing device

that services a remote science base station with mixed multichannel biomedical data upon demand. In the second application, the proposed design is integrated with an SSVEP-based BCI brain dialer system to serve as a preprocessor for EEG artifact removal. Based on the experimental results, the proposed biomedical multiprocessor can help improve significantly the brain dialer system's key detection accuracy when reduced dialing times are required.

In conclusion, given the specifications, research, development, validation and demonstration of the work presented, the proposed integrated DOT/ECG/EEG multiprocessor is offered as a proof-of-concept, reference design for the research and development of next generation portable brain-heart monitoring systems.

## **5.2 Future Work**

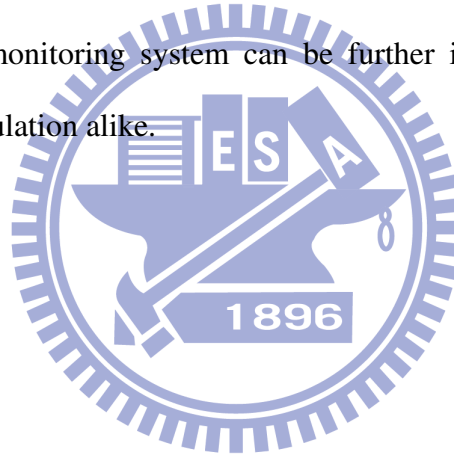
On the implementation level, further size, power and cost reductions can be achieved by integrating the biomedical multiprocessor together with the AFE and wireless communications module using either system-in-package (SIP) technology or mixed-signal IC design. To further reduce power consumption, the next generation design can also adopt more advanced low-power techniques such as power shut-off (PSO) and dynamic voltage and frequency scaling (DVFS). By employing these strategies, the portable biomedical sensing device can operate much longer and thereby improve further the patient's healthcare experience.

From a system-level perspective, change in the way biomedical data is processed and used can also result in significant operational efficiencies. A possible future research direction would be the development of smart, expert system devices that require minimal user management, which prompt action from the user only when a disease condition requiring medical attention is automatically detected. For example, biomedical signal modalities suitable for such type of processing include body temperature, blood pressure and ECG, wherein objective boundary conditions indicating critical illness have already been researched



and clearly defined previously. Thus, acquired biomedical signals can be processed locally and diagnoses be concluded on chip. For ECG, such a system has been developed in [79]. In such systems, there is no need to continuously transmit biomedical data wirelessly over to a science station, and so significant amounts of power can be saved. This way, the operating time of the portable biomedical sensing device can be prolonged even further.

Finally, next generation portable biomedical monitoring systems need not be limited to just the brain and heart. There are many other physiological signals/information that are important for monitoring the human health status. Examples are body temperature, blood oxygen levels (pulse oximetry), blood pressure, blood sugar levels, and many more. By integrating more kinds of biomedical data into one system, the usefulness and efficiency of the integrated biomedical monitoring system can be further increased, benefiting doctors, patients and the general population alike.



## References

- [1] E. Dishman, "Inventing wellness system for aging in place," *Computer*, vol. 37, no. 5, pp. 34–41, May 2004.
- [2] D. A. Boas, D. H. Brooks, E. L. Miller, C. A. DiMarzio, M. Kilmer, R. J. Gaudette, and Q. Zhang, "Imaging the body with diffuse optical tomography," *IEEE Signal Processing Mag.*, vol. 18, pp. 57–75, Nov. 2001.
- [3] P. Schwartz and S. Wolf, "QT interval prolongation as predictor of sudden death in patients with myocardial infarction," *Circulation*, vol.57, p.1074, 1978.
- [4] J. Malmivuo and R. Plonsey, *Bioelectromagnetism: principles and applications of bioelectric and biomagnetic fields*: Oxford University Press, USA, 1995.
- [5] M. Nei, R.T. Ho, et al., "EEG and ECG in Sudden Unexplained Death in Epilepsy." *Epilepsia*, vol.45, no.4, pp.338-345, 2004.
- [6] M.B. Malarvili, M. Mesbah, "Combining newborn EEG and HRV information for automatic seizure detection," *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pp.4756-4759, 20-25 Aug. 2008.
- [7] H. Abdullah, G. Holland, I. Cosic, D. Cvetkovic, "Correlation of sleep EEG frequency bands and heart rate variability," *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp.5014-5017, 3-6 Sept. 2009.
- [8] Cali Fidopiastis, Charles Hughes, "Workshop 1: Use of psychophysiological measures in virtual rehabilitation," *Virtual Rehabilitation, 2008*, pp.xi-xi, 25-27 Aug. 2008.
- [9] E. Watanabe, A. Maki, F. Kawaguchi, Y. Yamashita, H. Koizumi, and Y. Mayanagi, "Noninvasive cerebral blood volume measurement during seizures using multichannel near infrared spectroscopic topography," *J. Biomed. Opt.*, vol. 5, pp. 287-290, 2000.
- [10] Kenneth Revett, Sergio Tenreiro de Magalhães, "Cognitive Biometrics: Challenges for the Future," *Communications in Computer and Information Science*, vol.92, pp.79-86, 2010.
- [11] A. Villringer, J. Planck, C. Hock, L. Schleinkofer, and U. Dirnagl, "Near infrared spectroscopy (NIRS): A new tool to study hemodynamic changes during activation of brain function in human adults," *Neurosci. Lett.*, vol. 154, pp. 101-104, 1993.
- [12] Joseph P. Culver, Bradley L. Schlaggar, Hamid Dehghani, Benjamin W. Zeff, "Diffuse Optical Tomography for Mapping Human Brain Function," *Life Science Systems and Applications Workshop, 2006. IEEE/NLM* , pp.1-2, July 2006.

- [13] G. Strangman, D.A. Boas, and J.P. Sutton, "Non-invasive neuroimaging using near-infrared light," *Biol. Psych.*, vol. 52, no. 7, pp. 679-693, 2002.
- [14] U. Beese, H. Langer, W. Lang, and M. Dinkel, "Comparison of near- infrared spectroscopy and somatosensory evoked potentials for the detection of cerebral ischemia during carotid endarterectomy," *Stroke*, vol. 29, pp. 2032-2037, 1998.
- [15] K. Sakatani, S. Chen, W. Lichty, H. Zuo, and Y.P. Wang, "Cerebral blood oxygenation changes induced by auditory stimulation in newborn infants measured by near infrared spectroscopy," *Early Hum. Dev.*, vol. 55, pp. 229-236, 1999.
- [16] H. Sato, T. Takeuchi, and K.L. Sakai, "Temporal cortex activation during speech recognition: An optical topography study," *Cognition*, vol. 73, pp. B55-B66, 1999.
- [17] Y. Hoshi and M. Tamura, "Dynamic multichannel near-infrared optical imaging of human brain activity," *J. Appl. Physiol.*, vol.75, pp.1842-1846, 1993.
- [18] A. Siegel, J. J. Marota, and David Boas, "Design and evaluation of a continuous-wave diffuse optical tomography system," *Opt. Express*, vol.4, pp.287-298, 1999.
- [19] Y. Lin, G. Lech, S. Nioka, X. Intes, and B.Chance, "Noninvasive, low-noise, fast imaging of blood volume and deoxygenation changes in muscles using light-emitting diode continuous-wave imager," *Review of Scientific Instruments*, vol. 73(8), pp. 3065, 2002.
- [20] "Electrocardiography." Internet: <http://en.wikipedia.org/wiki/Electrocardiography>, Sep. 19, 2011 [Sep. 20, 2011].
- [21] U. Rajendra Acharya, K. Paul Joseph, N. Kannathal, C. Lim, and J. Suri, "Heart rate variability: a review," *Medical and Biological Engineering and Computing*, vol. 44, pp. 1031-1051, 2006.
- [22] F. Roche, V. Pichot, E. Sforza, I. Court-Fortune, D. Duverney, F. Costes, M. Garet, and J.-C. Barthélemy, "Predicting sleep apnoea syndrome from heart period: A time-frequency wavelet analysis," *Eur. Respir. J.*, vol. 22, pp. 937-942, 2003.
- [23] L. Rankine, N. Stevenson, M. Mesbah, and B. Boashash, "A Nonstationary Model of Newborn EEG," *Biomedical Engineering, IEEE Transactions on* , vol.54, no.1, pp.19-28, Jan. 2007.
- [24] M. Ashtiyani, S. Asadi, and P.M. Birgani, "ICA-Based EEG Classification Using Fuzzy C-mean Algorithm," *3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 2008. ICTTA 2008. , pp.1-5, 7-11 April 2008.
- [25] R. Vigarío, "Extraction of ocular artifacts from EEG using independent component analysis," *Electroencephalogr. Clin. Neurophysiol.*, vol. 103, pp. 395-404, 1997.

- [26] R. Vigario, J. Sarela, V. Jousmaki, M. Hamalainen, and E. Oja, "Independent component approach to the analysis of EEG and MEG recordings," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 5, pp. 589-593, May 2000.
- [27] Hosna Ghandeharion and Abbas Erfanian, "A fully automatic ocular artifact suppression from EEG data using higher order statistics: Improved performance by wavelet analysis," *Medical Engineering & Physics*, vol.32, pp.720-729, 2010.
- [28] Muhammad Tahir Akhtar and Christopher J. James, "Focal Artifact Removal from Ongoing EEG - A Hybrid Approach Based on Spatially-Constrained ICA and Wavelet De-noising," *31st Annual International Conference of the IEEE EMBS*, Minneapolis, Minnesota, USA, September 2-6, 2009.
- [29] Pierre Comon, "Independent component analysis, A new concept?," *Signal Processing, Higher Order Statistics*, vol.36, no.3, pp.287-314, 1994.
- [30] A.J. Bell and T.J. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1004–1034, 1995.
- [31] S. Makeig, A. Bell, T.-P. Jung, and T. J. Sejnowski, "Independent component analysis in electro-encephalographic data," in *Advances in Neural Information Processing Systems 8*, M. Mozer et al., Eds. Cambridge, MA: MIT Press, 1996, pp. 145–151.
- [32] Tzyy-Ping Jung and Scott Makeig, "Removing Artifacts from Single Trials," Internet: <http://sccn.ucsd.edu/~scott/tutorial/icatutorial8.html>, [Sep. 20, 2011].
- [33] M. Malik, "Heart rate variability: Standards of measurement, physiological interpretation, and clinical use," *Circulation*, vol. 93, pp.1043–1065, 1996.
- [34] M. N. Levy, "Sympathetic-Parasympathetic Interactions in the Heart," *Circ Res*, vol.29, pp.437-445, Nov. 1971.
- [35] M. Tiainen, H. J. Parikka, M. A. Mäkijärvi, O. S. Takkunen, S. J. Sarna, and R. O. Roine, "Arrhythmias and Heart Rate Variability During and After Therapeutic Hypothermia for Cardiac Arrest," *Critical Care Medicine*, vol. 37, pp. 403-409, 2009.
- [36] S. Kasaoka, T. Nakahara, Y. Kawamura, R. Tsuruta, and T. Maekawa, "Real-time monitoring of heart rate variability in critically ill patients," *Journal of Critical Care*, vol. 25, pp. 313-316, Jun 2010.
- [37] J. Pan and W. J. Tompkins, "A Real-Time QRS Detection Algorithm," *Biomedical Engineering, IEEE Transactions on*, vol. BME-32, pp. 230-236, 1985.
- [38] N. R. Lomb, "Least-squares frequency analysis of unequally spaced data," *Astrophysics and Space Science*, vol. 39, pp. 447-462, 1976.

- [39] G. D. Clifford and L. Tarassenko, "Quantifying errors in spectral estimates of HRV due to beat replacement and resampling," *Biomedical Engineering, IEEE Transactions on*, vol. 52, pp. 630-638, 2005.
- [40] D.C. Yates, and E. Rodriguez-Villegas, "A Key Power Trade-off in Wireless EEG Headset Design," *Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on*, pp.453-456.
- [41] Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, vol., no., pp.46-51, 5-8 Nov. 2007.
- [42] Y. Wongsawat, S. Orintara, T. Tanaka, and K. R. Rao, "Lossless multichannel EEG compression," in *Proc. ISCAS 2006*, pp. 1611–1614.
- [43] N. Sriraam and C. Eswaran, "An Adaptive Error Modeling Scheme for the Lossless Compression of EEG Signals," *Information Technology in Biomedicine, IEEE Transactions on*, vol.12, no.5, pp.587-594, Sept. 2008.
- [44] N. Memon, Kong Xuan, and J. Cinkler, "Context-based lossless and near-lossless compression of EEG signals," *Information Technology in Biomedicine, IEEE Transactions on*, vol.3, no.3, pp.231-238, Sept. 1999.
- [45] Z. Arnavut, "ECG Signal Compression Based on Burrows-Wheeler Transformation and Inversion Ranks of Linear Prediction," *Biomedical Engineering, IEEE Transactions on*, vol.54, no.3, pp.410-418, March 2007.
- [46] C.D. Giurcăneanu, I. Tăbuș and S. Mereuță, "Using contexts and R-R interval estimation in lossless ECG compression," *Computer Methods and Programs in Biomedicine*, vol. 67, no.3, pp.177-186, March 2002.
- [47] Antti Koski, "Lossless ECG encoding," *Computer Methods and Programs in Biomedicine*, vol.52, no.1, pp.23-33, Jan. 1997.
- [48] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *Communications, IEEE Transactions on*, vol.45, no.4, pp.437-444, Apr 1997.
- [49] M.J. Weinberger and G. Seroussi, G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *Image Processing, IEEE Transactions on*, vol.9, no.8, pp.1309-1324, Aug 2000.
- [50] Yuan-Huang Hsu, Chih-Chung Fu, Wai-Chi Fang, and Tzu-Hsien Sang, "A VLSI-inspired image reconstruction algorithm for continuous-wave diffuse optical tomography systems," *Life Science Systems and Applications Workshop, 2009. LiSSA 2009. IEEE/NIH*, vol., no., pp.88-91, 9-10 April 2009.
- [51] C.G.J. Jacobi, "Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen

- vorkommenden Gleichungen numerisch aufzulösen”, *Crelle's Journal*, 30, pp. 51-95, 1846.
- [52] J.G.F. Francis, “The QR Transformation, I”, *The Computer Journal*, vol. 4, no. 3, pages 265-271, 1961.
- [53] J.G.F. Francis, “The QR Transformation, II” *The Computer Journal*, vol. 4, no. 4, pages 332-345, 1962.
- [54] R. A. Rosanoff, J. F. Gloudeman, and S. Levy, “Numerical conditions of stiffness matrix formulations for frame structures,” *Proc. of the Second Conference on Matrix Methods in Structural Mechanics*, WPAFB Dayton, Ohio, 1968.
- [55] J. Demmel and K. Veselić, “Jacobi's method is more accurate than QR,” *SIAM J. Matrix Anal. Appl.*, 13, pp. 1204-1245, 1992.
- [56] Gene H. Golub and Henk A. van der Vorst, “Eigenvalue computation in the 20th century,” *Journal of Computational and Applied Mathematics*, vol.123, no.1-2, pp.35-65, 2000.
- [57] Z. Drmač and K. Veselić, “New fast and accurate Jacobi SVD algorithm: I”, LAPACK Working Note, 169, 2005.
- [58] N. D. Hemkumar, “Efficient VLSI Architectures for Matrix Factorizations,” Ph.D. Dissertation, Rice University, Houston, TX, USA, UMI Order No. GAX95-14181, 1995.
- [59] Y. Liu, C.-S. Bouganis, and P.Y.K. Cheung, “Hardware architectures for eigenvalue computation of real symmetric matrices,” *Computers & Digital Techniques, IET*, vol.3, no.1, pp.72-84, January 2009.
- [60] S. Aslan, E. Oruklu, and J. Saniie, “Realization of area efficient QR factorization using unified division, square root, and inverse square root hardware,” *Electro/Information Technology, 2009. eit '09. IEEE International Conference on*, vol., no., pp.245-250, 7-9 June 2009.
- [61] R. P. Brent and F. T. Luk, “The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays,” *SIAM J. Sci. Statist. Comput.*, vol. 6. pp. 69-84. 1985.
- [62] Jack E. Volder, “The CORDIC Trigonometric Computing Technique,” *IRE Transactions on Electronic Computers*, vol. EC-8, pp.330-334, 1959.
- [63] J. R. Cavallaro and F. T. Luk, “CORDIC Arithmetic for an SVD Processor,” *Journal of Parallel and Distributed Computing*, vol. 5, no. 3, pp. 271-290, 1988.
- [64] A. Ahmedsaid, A. Amira, and A. Bouridane, “Efficient systolic array for singular value



- and eigenvalue decomposition,” *Micro-NanoMechatronics and Human Science, 2003 IEEE International Symposium on* , vol.2, no., pp. 835- 838 Vol. 2, 27-30 Dec. 2003.
- [65] R.T. Gregory, “Computing eigenvalues and eigenvectors of a symmetric matrix on the ILLIAC,” *Math. Tables Aids Comput.*, vol. 7, pp. 215-220, 1953.
- [66] G. E. Forsythe and P. Henrici, “The cyclic Jacobi method for computing the principal values of a complex matrix,” *Trans. Amer. Math. Soc.*, vol. 94, pp. 1-23, 1960.
- [67] H. Tanaka, K. D. Monahan, and D. R. Seals, “Age-predicted maximal heart rate revisited,” *J Am Coll Cardiol*, vol. 37, pp. 153-156, January 1, 2001.
- [68] Shao-Yen Tseng and Wai-Chi Fang, “An EKG system-on-chip for portable time-frequency HRV analysis,” *Consumer Electronics (ICCE), 2011 IEEE International Conference on* , vol., no., pp.559-560, 9-12 Jan. 2011.
- [69] E. Chua and Wai-Chi Fang, “Mixed bio-signal lossless data compressor for portable brain-heart monitoring systems,” *Consumer Electronics, IEEE Transactions on* , vol.57, no.1, pp.267-273, February 2011.
- [70] “Semiconductor device fabrication.” Internet: [http://en.wikipedia.org/wiki/Semiconductor\\_device\\_fabrication](http://en.wikipedia.org/wiki/Semiconductor_device_fabrication), Aug. 5, 2011 [Aug. 23, 2011].
- [71] James Hogan, “The evolution of design methodology.” Internet: <http://www.eetimes.com/electronics-news/4211010/The-evolution-of-design-methodology?pageNumber=1>, Nov. 24, 2010 [Aug.29, 2011].
- [72] Y. T. Wang, Y. Wang, and T. P. Jung, “A cell-phone based brain computer interface for communication in daily life”, *Journal of Neural Engineering*, vol.8, no.2, pp1-5, March 2011.
- [73] “Steady state visually evoked potential.” Internet: [http://en.wikipedia.org/wiki/Steady\\_state\\_visually\\_evoked\\_potential](http://en.wikipedia.org/wiki/Steady_state_visually_evoked_potential), Oct. 5, 2009 [Sep. 14, 2011].
- [74] D. Regan. *Human brain electrophysiology: Evoked potentials and evoked magnetic fields in science and medicine*. New York: Elsevier, 1989, 672 pp.
- [75] C.-T. Lin, L.-W. Ko, M.-H. Chang, J.-R. Duann, J.-Y. Chen, T.-P. Su, and T.-P. Jung, “Review of wireless and wearable electroen-cephalogram systems and brain-computer interfaces a mini-review,” *Gerontology*, vol. 56, pp. 112–119, 2010.
- [76] G. Y. Bin, X. R. Gao, Z. Yan, B. Hong, and S. K. Gao, “An online multi-channel SSVEP-based brain-computer interface using a canonical correlation analysis method,” *J. Neural Eng.*, vol. 6, no. 4, p. 046002, 2009.



- [77] “Inverse of Matrix 4x4 using partitioning.” Internet:  
[http://freevec.org/function/inverse\\_matrix\\_4x4\\_using\\_partitioning](http://freevec.org/function/inverse_matrix_4x4_using_partitioning), Apr. 18, 2008 [Sep. 18, 2011].
- [78] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.
- [79] Hyejung Kim, R.F. Yazicioglu, P. Merken, C. Van Hoof, and Hoi-Jun Yoo, “ECG Signal Compression and Classification Algorithm With Quad Level Vector for ECG Holter System,” *Information Technology in Biomedicine, IEEE Transactions on* , vol.14, no.1, pp.93-100, Jan. 2010.

