AUTOMATED TEST SUITE – A VALIDATION PACKAGE FOR MOBILE CHIPSETS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI DESIGN and EMBEDDED SYSTEM

By

SRIKANTH NERELLA

Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela

2007

AUTOMATED TEST SUITE – A VALIDATION PACKAGE FOR MOBILE CHIPSETS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI DESIGN and EMBEDDED SYSTEM

By

SRIKANTH NERELLA

Under the Guidance of

Prof. K. K. MAHAPATRA

Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela
2007

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the Thesis Report entitled ***"AUTOMATED TEST SUITE – A VALIDATION PACKAGE FOR MOBILE CHIPSETS"*** submitted by Mr. **SRIKANTH NERELLA (20507006)** in partial fulfillment of the requirements for the award of Master of Technology degree Electronics and Communication Engineering with specialization in "**VLSI DESIGN and EMBEDDED SYSTEM**" during session 2006-2007 at National Institute Of Technology, Rourkela (Deemed University) and is an authentic work by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university/institute for the award of any Degree/Diploma.

Prof. K. K.MAHAPATRA
Dept. of E.C.E
National institute of Technology
Rourkela-769008
Email: kkm@nitrkl.ac.in

Date:

ACKNOWLEDGEMENTS

With great sense of pleasure and privilege, I express my deep sense of gratitude to Mr. K.K.MAHAPATRA, Professor, department of Electronics and Communication Engineering, NIT Rourkela for his valuable suggestions, sagacious guidance, scholarly advice and comprehensive critical remarks during the course of investigation.

I want to thank all my teachers **Dr. G.S. Rath**, **Dr. G. Panda, Dr. S.K. Patra and Dr. S.K. Meher** for providing a solid background for my studies and research thereafter. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I am most obliged and grateful to my guide Mr. Raheem Baig, Team lead, RFWS – Sysval group, Analog Devices India (P) Ltd, Hyderabad Design Center, for giving training and guidance in completing the project.

My stay at Analog Devices Inc., Hyderabad, has been a rich learning process and memorable experience. I was helped and guided by lot of people at Analog Devices Inc., Hyderabad. I am greatly indebted to all the staff members of Analog Devices India (Pvt.) Ltd, HDC, HYDERABAD for making me feel as a part of the institution and their co-operation during my Dissertation work

I also extend my thanks to entire faculty of Department of Electronics and Communication Engineering, National Institute of Technology Rourkela, Rourkela who have encouraged me throughout the course of Master's Degree.

Finally, I am thankful to one and all that have directly or indirectly rendered their valuable guidance and encouragement during my Dissertation work.

SRIKANTH.N

Table of contents

ABSTRACT

With the diminishing sizes of transistors, it is now possible to incorporate complex systems on a single die. The chipsets used in mobile phone handsets are a good example of such a complex systems. The design, validation, hardware development and software development of such complex chipsets is an intricate task which also consumes lot of time. With the increasing competition, time-to-market factor plays a crucial role in the development of a product. There is a constant need for an automated platform which would help designers at various stages in the development process of a complex product with minimum efforts.

Automated Test Suite (ATS) is an automated Diagnostic Test System that enables the users to automate, validation procedures for any ADI DBB (ANALOG DEVICES digital base band) chipsets and H/W platforms in a user-friendly environment. This software follows the Host-Target model ensuring easy implementation of test cases so that the user can concentrate on the testing module only. It provides good modularity and reusability with simple structure.

ATS has several features, these are:

- Communication between Host and Target via RS232 or USB,
- Usable with ANVIL evaluation boards
- Remote execution of test routines on target from host.
- Enables h/w platform testing – can be extended for performance and characterization testing.
- Backwards and forwards extendable to other chipset families and h/w platforms.
- Provides ‘Help’ feature for all the tests to user.
- Script based testing ensures customizable test routine development.
- Test result log - HTML based details and summary of test results
- GUI based test tool gives user-friendly interface. Debug tools have been developed for some of the Hardware modules (LED, GPIO)

This project was implemented in three phases. First phase includes the formulating of ATS architecture and provide sample implementation for LEMANS (AD6900 MSP 500) DBB. The second phase is adding new platform DIONE (AD6722 MSP 430) DBB to the existing ATS. Third phase concentrates on designing and implementing new ATS architecture for efficient performance and to reduce development time for ATS.

List of figures

Figure No **Figure Title** **Page No**

Glossary

ABB	Analog Base Band Processor (AD6535, AD6555)
ADI	Analog Devices Inc.
ADS	ARM Developer Suite
ANVIL	ADI's evaluation board for wireless chipset development.
ARM	Micro processor from ARM Ltd. Either ARM7TDMI or ARM926EJ-S
ASPORT	Audio Serial Port
BSPORT	Base Band Serial Port
CAST	CAST (Chipset Abstraction Software Technology) drivers
DBB	Digital Baseband processor (AD6525/6/7/8/9, AD6532 and AD6758)
DSP	Digital Signal Processor. Either ADSP218x or ADSP21535
DMA	Direct Memory Access
EVB	Evaluation Board
GPIO	General Purpose input/output Pin
GPT	General Purpose Timer
GSM	Global System for Mobile
GUI	Graphical User Interface
ISR	Interrupt Service Routine
JTAG	Joint test action group
KBD	Keyboard
MCU	Micro Controller Unit (ARM)
MSP	Mixed signal processor
NFC	NAND Flash Controller
N/A	Not applicable
RF	Radio Frequency
TAC	Target Application Code
TCS	Target Communication System
TS	Target Services
UART	Universal asynchronous receive transmit
USB	Universal serial bus
WDT	Watch Dog Timer

Chapter 1

INTRODUCTION

This chapter gives overview and organization of thesis.

INTRODUCTION

1.1 *Problem Statement:*

The embedded product designs are getting larger day by day and are being integrated onto single chips. So the design of these highly integrated modules must be equipped with extensive validation for each module & combinations of different modules. Now a day the product success depends on delivering it with in time to market, but the validation of all the modules & interdependencies between them increases the development time of embedded product.

For example, the Global System for Mobile communication (GSM) chipsets needs to meet the growing standards and support different peripheral integration. This demands the chipsets to have high capability to handle different applications, like Audio and Video Processing, Camera and consumer applications with internet connectivity (GPRS).All these modules consume a lot of power & has to be provided with additional circuitry to minimize power loss. Also these chipsets are made to work with more than one core. This thesis concentrates on developing an automated validation system to reduce development time of mobile chipsets that are being developed by commercial organizations. In this dissertation the mobile SOCs (system on chip) developed by ANALOG DEVICES Inc (ADI) are taken as examples.

1.2 Motivation:

The ADI msp chipsets contains DSP and the famous ARM controllers that are used in every product now a day. In addition to two processor cores these MSP chips have several IP vendor ICs that are needed for mobile applications processing, different types of memory interfacings, keypads, LCD displays, wireless equipment and sensors etc. As the wireless chipset is having various modules embedded into it, after fabrication validating each of the modules for every chip becomes difficult task. Validation is immediate process after fabrication of any chip. The biggest challenge with multiple processors is debugging. The code on each individual device may be debugged—the tools and techniques are well understood. The challenge arises with interactions between the two processors. There is a clear need for debugging technology that addresses the issue of debugging the system—that is, multicore debugging. So this work explores maximum possible technologies to be applied for implementation.

1.3 Background and literature survey:

In the past decade, there has been an extensive growth in mobile communications technology. In order to standardize this mobile communication across various service providers, the European Telecommunication Standards Institute (ETSI) has proposed a mobile telecommunication standard called Global System for Mobile Communications (GSM). GSM is a digital cellular radio network technology and it is now being operated in over 200 countries world wide.

A GSM network is composed of several functional entities, whose functions and interfaces are specified in the GSM standards. The GSM network can be divided into three broad parts. The Mobile Station is carried by the subscriber. The Base Station Subsystem controls the radio link with the Mobile Station. The Network Subsystem, the main part of which is the Mobile services Switching Center (MSC), performs the switching of calls between the mobile users, and between mobile and fixed network users.

A mobile station may be referred to as a “handset”, a “mobile”, a “portable terminal” or “mobile equipment”. Besides providing a transceiver for transmission and reception of voice and data, the mobile station also performs a number of very demanding tasks such as authentication, handover and channel encoding as per the GSM standards.

The number of mobile equipment subscribers has been growing extensively because of its ease of use. Many vendors having being developing the hardware of the mobile phone equipment and Analog Devices is one of the key players in this industry. Analog devices have developed a wide range of mobile equipment hardware to meet the varied requirements of the mobile equipment subscribers.

The embedded tool development articles propose several types of mechanisms for validation of chipsets. [14] The ADI has set of platforms that provide some possibility for validation like *SLIVER* and *CAST* which are not much efficient and interactive. The ARM and BLACKFIN architectures have been surveyed and programming tools for multicore programming (for both MCU and DSP) are used.

1.4 Current Status:

The evaluation of new chips and systems typically requires the development of custom hardware and software, which usually has to be modified and re-complied for even small changes in test conditions, when validating them. Thus there is a great need for an automated platform for running the validation of each module after the fabrication of chip. In the absence of automated platform, this can be a laborious and time-consuming process. Developers for long have been looking for a solution that will enable them to re-use applications (test, evaluation or demo) with minimal porting to new chipsets. Automated test suite (ATS) is developed to meet the requirements. This package tests all the modules simultaneously and reports performance results of every module. To implement this proposed ATS design, the embedded product must have several modules integrated into it for full-fledged implementation. The mobile chipsets are the most suitable products which have lot of IP vendor ICs integrated and interlinked. The ADI's MSP (mixed signal processor) mobile chipsets are selected for implementation.

1.5 Thesis contribution:

Automated Test Suite (ATS) is a validation package developed for efficient validation of wireless chipsets. It covers the several layers of embedded product development cycle like:

1. Hardware abstraction:
    - Memory management operations like cache, buffers and register controlling.
    - Working with programmable devices like CPLD.
2. Device driver usage (USB, UART, CAST drivers).
3. Firm ware development:
    - Maintaining OS related tasks for multitasking
    - Handling interrupts & exceptions
4. Software development for GUI & interacting with processor kit through Perl etc...

This software would help engineers to validate Digital Base band chipsets. I will be involved in the complete development process of this project right for formulating requirements, design, creating test plan, implementation, testing, archiving and documentation. It will help me in understanding all the phases in a software development life cycle. Also I will be acquainted with ADI's digital and analog base band chipsets used in mobile phone handsets.

1.6 Organization of report:

This report is organized as:

Chapter 2: Provides explanation of ATS architecture and implementation protocol.

Chapter 3: provides brief explanation of ADI's mobile chipsets and their evaluation boards.

Chapter 4: Explains the building procedure for each module validation and implementation details of non-interrupt based modules.

Chapter 5: Implementation of interrupt based modules and handling interrupt service routines.

Chapter 6: Designing of new ATS architecture and implementation for efficient validation.

Chapter 7: Conclusions and future work.

Chapter 2

ATS ARCHITECURE

This chapter briefly explains ATS development architecture.

2.1 Introduction:

ATS development is mainly of two parts: The HOST & target packages to be implemented. It is developed on FORGE environment which is used for communication between HOST PC & target for module ids & log back results. FORGE is developed for ADI MSP chipsets for debugging & development of applications. It provides some target services that are used in ATS. The FORGE acts as an operating system for ATS tasks.

The used FORGE services are like:

- interrupt handling
- Watchdog timer configurations for dead locks maintenance etc...
- UART communication for massage passing, acknowledgements and error handling

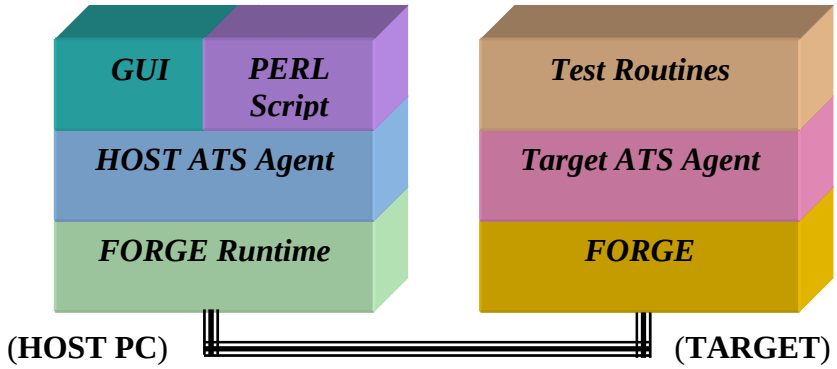

Fig 2.1: Layered model of ATS architecture

The physical communication exists between FRT (FORGE run time) and FORGE target in the board. This figure explains the different layers in ATS validation package.

ATS GUI:

This is the main application windows environment i.e. visible to user which gives friendly environment for validation. Graphical user interface (GUI) is designed to scan existing folders & files and to list the available board types and test modules. This allows the user to easily add any new board or new module for future enhancement. While target modules can be designed & built independently, making it easy to port any standalone test case. On the target side, one dedicated folder is provided for each test module that gives easy understanding & porting of new tests. Separate folders/files for utility functions & message parsing; provide separation of tests from other generalized service functions. [13]

Programming details are hidden from the user by only providing ARM executables. XML files are used for listing & identifying of test modules, provide easy upgradeability and simplify addition of a new module. Same id lists for module and test are maintained in both HOST side (xml tables) and target side (pointer tables).

2.2 Target design:

The chipset evaluation board is to be loaded with ARM assembly code for validating each module. The downloading onto flash memory is done through a tool called micro loader which communicates to the chip through boot code stored on ROM. Initially it will download some configuration code on byte basis and then it will send complete binary file as burst packets after configuring the target memory controllers.

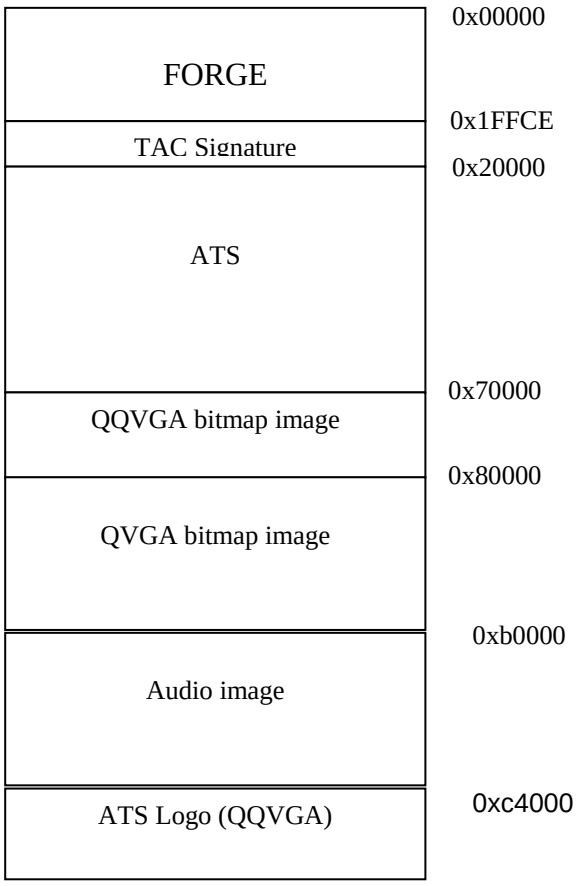

Fig 2.2: Target memory map

In any processor memory map, the memory is scattered into several regions based on attributes and type of data. For example the standard C compiled memory can be divided as read only (RO), read write (RW), zero initialize (ZO) regions. The scatter file provided by ARM compilers is used for scattering memory into these regions by providing base & limit addresses for each region. This scattering of data allows the processor to efficiently locate and use the data as per specified attributes (shared/common, read only/ read write).

In this ATS design the complete code is built by referring to flash memory which will be loaded by code (RO), initialized RW data and read only data. The advantage of storing it in flash is though the power is lost or the board gets reset, the downloaded data will remain same.

The flash image is loaded into memory as shown in figure 1.2, FORGE TAC is the starting code which provides some target services used in ATS, and it acts as an OS for handling tasks. Third segment is the actual executing task code for the validation suite. The other segments are the data files used in testing of display and audio modules.

Additional data files for any new modules can be added in the next address locations of flash memory.

2.3 Framework design:

In any embedded design for interactive communication between the HOST PC & target board there must be a protocol mechanism that handles synchronization, error handling and uninterrupted communication through physical media. In ATS, one protocol named as diagAgent act as a layer between target modules and the HOST GUI.

This protocol uses FORGE target communication system embedded with UART communication protocol. The task insertion facility and watchdog monitoring for DBB while waiting for parameters from HOST are used from FORGE services.

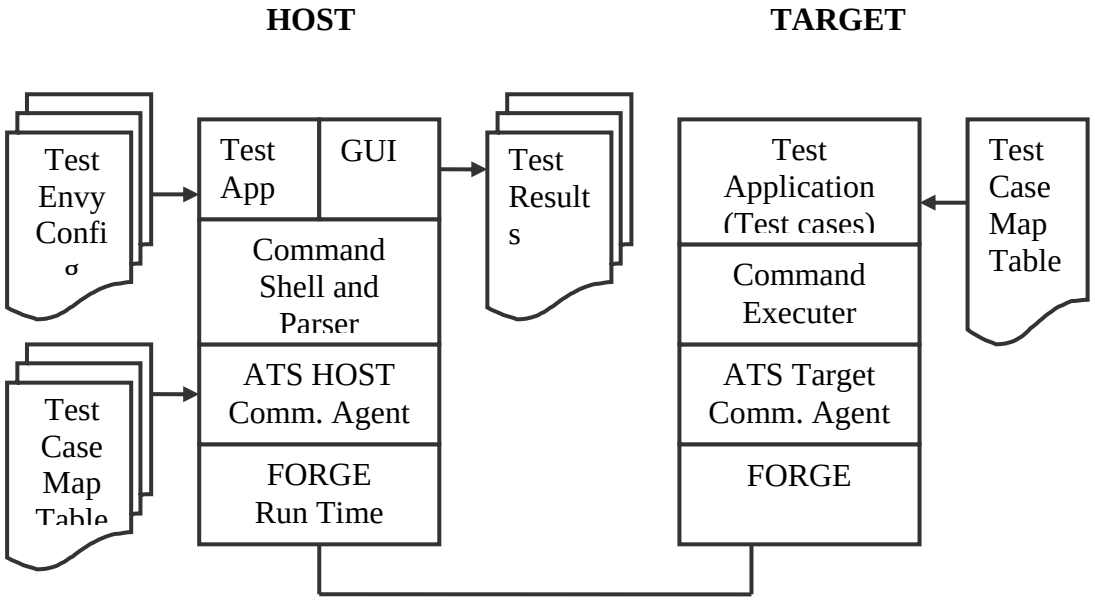

Fig 2.3: Test Suite Frameworks

The HOST script forms a request message and sends it to the DaigAgent. On reception of the request message DiagAgent decode the message and calls a diag functions with user supplied parameters. On completion of this task DiagAgent sends back the response to the HOST. HOST software logs the response message in proper format in an HTML file.[13]

The Target Software is a single TAC and it has two parts in it.

i. Diag Agent runs on FORGE and work as a communication agent between HOST and Target.

ii. Diag Applications are diagnostic routines for the functional blocks. These functions can be registered in the target and HOST database and they can be invoked with PERL script form the HOST. With proper ids on the target, the tests can be run with the help of PERL script. This way the automation of the diagnostic environment can be achieved.

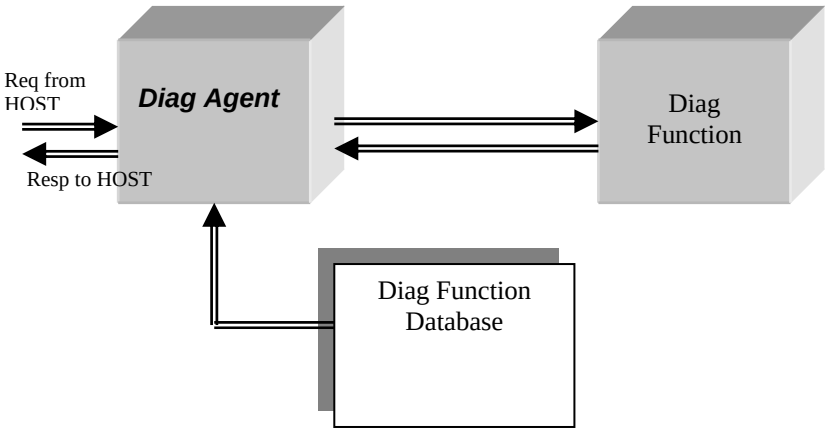

Fig 2.4: Target Diagnostic Software Architecture on Anvil

The diag function database provides the selection of ids for specific module and function.

This Diag agent consists of two subsections. Functionally, these are

- Message Parser
- Message Formation

Message Parser: Once the FORGE hands over the message intended for the ATS communication TAC, the Message Parser is invoked, which parses the message and form a parameters buffer and attaches the Function ID to it.

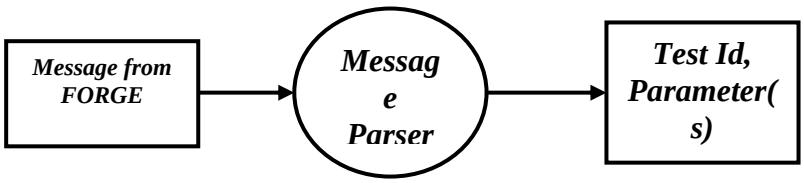

Figure 2.5: Target Message Parser

Message Formation: This sub-module takes the input parameters from user and forms a message and then that message is encapsulated into a FORGE packet and sent it to the Host. It will be used to send any message back to the Host such as error message, test results.

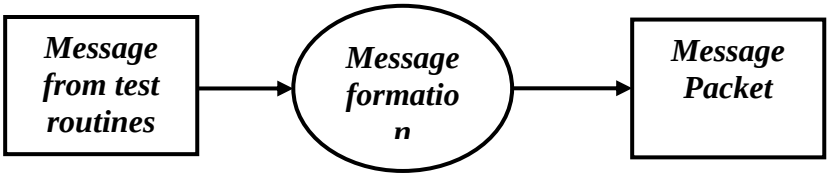

Figure 2.6: Target Message formation

The following diagram shows the software architecture of DiagAgent. The TCS Message Handler gets the message and passes on to the TCS Message Parser. It parses the message and gets Module Id, Function Id and the Parameters if any. Diag function Execution module searches the Diagnostic Function Database and calls the function if it exists. Upon completion of execution of the function, DiagAgent sends the response back to the HOST.

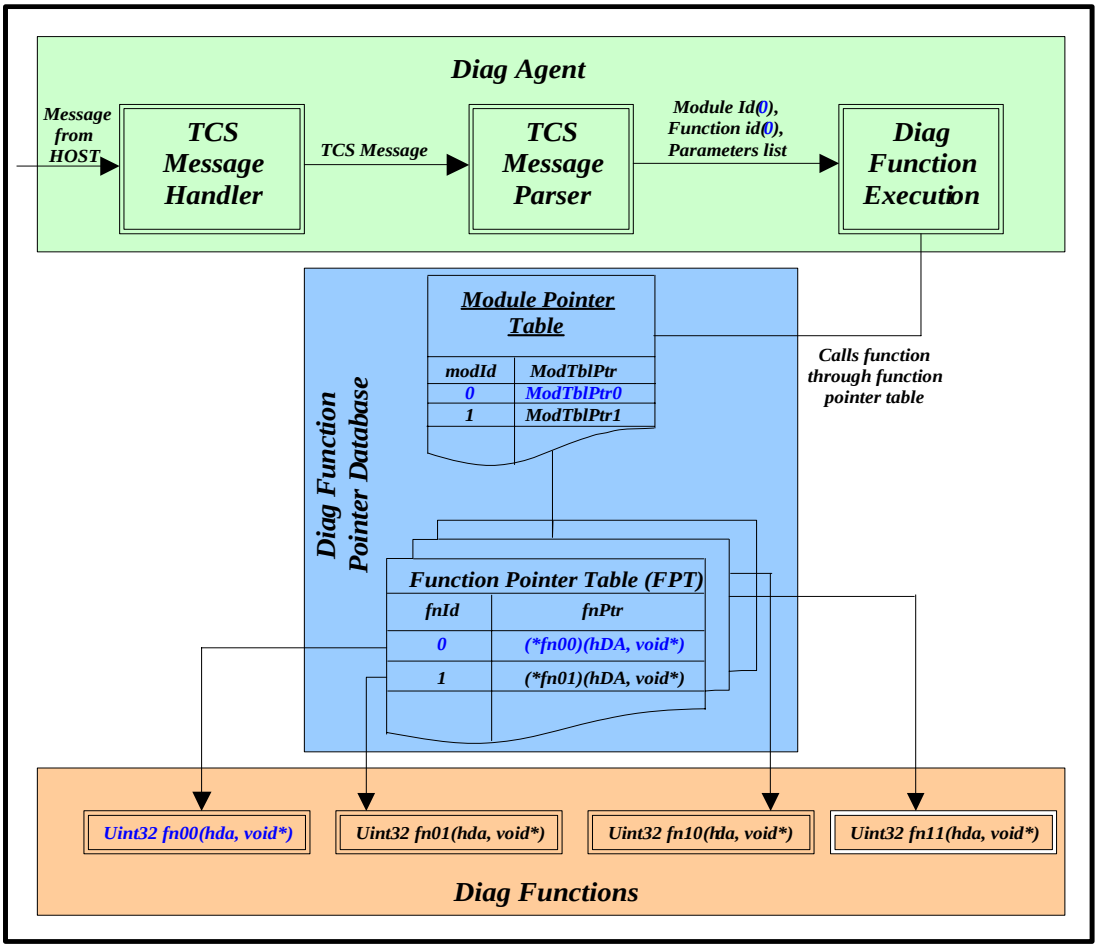

Fig 2.7: Target Diagnostic Software Data flow

- ***Diagnostic Functions:*** Diagnostic Functions are part of the same TAC as DiagAgent. They are functions used to test the functional blocks. These functions need to be registered in the Diagnostic Function Database in the target to make them available to the HOST. HOST also maintains same mapping of the functions, a script on the HOST can invoke these diagnostic functions and get the result back.
- ***Diagnostic function database:*** The Diagnostic Software on the Target maintains a database of diagnostic functions available to the HOST. The DiagAgent calls appropriate functions using this database. It maintains two types of tables. The Module Pointer Table contains the pointer to the Function Pointer Table of the functional block. Each Functional block maintains a table of diagnostic function pointers.

The proposed ATS has several features that are necessary for any project development: [13]

1. Remote execution of test routines on target from host with parameters passing facilities hence same test may be run with different parameters values, which can be changed dynamically.

   This is useful in

   a. Testing different memory locations by passing different addresses for test.
   b. Setting & Testing clock frequencies easily & dynamically, so that the performance at different frequencies of each module & its combinations can be evaluated.
2. GUI based debugging tool enables to debug hardware through user friendly interfaces, the VB language is used to develop GUI facility.
3. Test result logging for keen verification & displaying status messages in every step of the test, so that user can be informed of what is happening in the chip.
4. *upgradeability:*

   The function/module tables are used to differentiate between executing test functions & different modules in the chip by maintaining unique ids for each function and for module. This facilitates easy adding of new tests and new modules in the upcoming chip.
5. Backwards and forwards extendable to other chipset families and h/w platforms.
6. *modularity:*

   By maintaining separate files and folder structures for each module, the code will be understandable and easy for porting into new chips.

Chapter 3

MOBILE CHIPSETS & EVALUATION BOARDS

This chapter contains explanation of ADIs mobile chipsets and their evaluation boards.

Analog devices has developed the AD20msp500/ GSM base band chipset which implements a combination of a GSM system and an advanced analog and digital signal processing technology to provide a new mobile equipment design. Figure 1 illustrates the design of the mobile equipment. It is composed of two main chipsets, AD6532 - the digital base band chipset and AD6536 – the analog base band chipset. The AD20msp500 chipset is integrated with a radio chip named Othello that performs the transmission and reception of radio signals according to GSM standards. These three units are described in brief below.

3.1 Mixed signal processor system:

Digital Base band: The DBB digital engine includes two high-performance, low power programmable processors, the DSP engine and the ARM7 RISC micro-controller engine. The DSP engine uses the ADSP Blackfin DSP core, operating at 65 MIPS, and handling the vocoder and channel codec functions. The ARM7 RISC micro-controller engine, operating at 39 MIPS, handles the protocol stack and man-machine interface (MMI) functions. The digital processor also includes ciphering and Viterbi functions and a range of debugging tools and interfaces for the FLASH memory, keypad, display, SIM card, IrDA, and USC ports. [8]

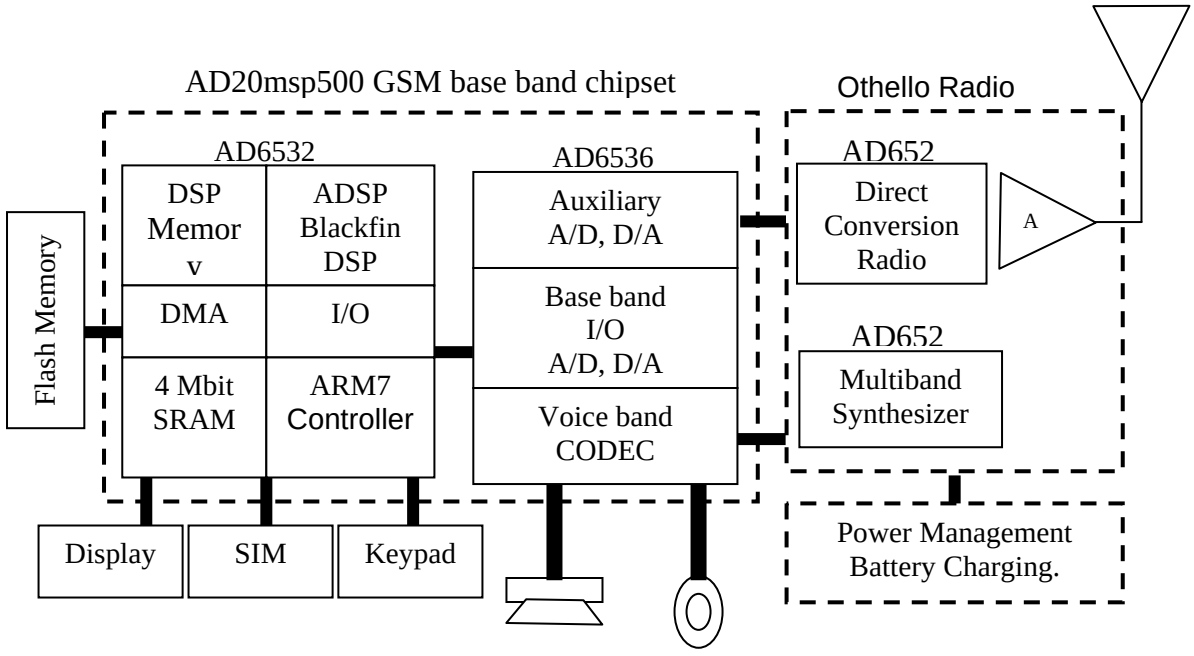

Figure 3.1: AD20msp500 GSM base band chipset integrated with a radio chip.

Analog Base band: This is a mixed signal chip that performs the Analog to Digital conversions, Digital to Analog conversions and filtering for the speech channel such as microphone and speaker, as well as the modulation and demodulation for the up and down links of the communication channel. It provides the power distribution to different modules with different levels as needed. This chip also supports additional ADC for functions such as battery and temperature monitoring.

Radio chipset: The Radio Frequency (RF) chip performs the all RF applications. The radio consists of two integrated circuits, the AD6523 Zero-IF Transceiver and the AD6524 Multi-band Synthesizer. The AD6523 contains the main functions necessary for both a direct conversion receiver and a direct VCO transmitter, known as the Virtual-IF transmitter. The AD6524 is a fractional-N synthesizer that features extremely fast lock times to enable advanced data services over cellular telephones—such as high-speed circuit-switched data (HSCSD) and general packet radio services (GPRS).

3.2 Digital Base band Chipset:

This is complete single chip programmable Digital Base Band processor. Figure 1.2 illustrates its internal architecture. It consists of MCU subsystem, DSP subsystem, System Memory, External Devices and Peripheral subsystem. MCU subsystem includes ARM MCU, System DMA, Boot Code ROM, and the System Peripherals. DSP subsystem includes the Blackfin DSP, L1 and L2 cache memories, DSP peripherals, DSP DMA controller, DSP Cipher coprocessor, and External coprocessor interface. System Memory contains on chip fast static RAM that stores code and data for either processor. External Devices includes Flash, Display and other devices. The ARM MCU and the Blackfin DSP processors each have its own control functions and they both have shared access to all peripherals and memory sub systems. All system resources are memory mapped. The two processor systems have their own dedicated DMA controllers, interrupt controllers and local control / status registers. [7]

The ARM controller and DSP system architectures and operations are explained briefly in appendix, the other peripherals and bus system provided in chip are explained here.

The internal DBB system can be partitioned into five main subsystems.

- MCU Subsystem: ARM926EJ-S core, MCU Bus Interface Unit, MCU Data Cache, MCU Instruction Cache, Boot ROM, and Embedded Trace Macro cell.

- DSP Subsystem: Blackfin DSP core, DSP Bus Interface Unit, L1 and L2 Memories, DSP Peripherals and DSPDMA controller.
- Peripheral Subsystem: Communications peripherals, house keeping peripherals, man Machine interface peripherals, and SYSDMA controller.
- Applications Subsystem: Parallel Port Interface, LCD controller, and APPDMA controller.
- Memory Subsystem: System SRAM and External Bus Interface, including SDRAM and FLASH interfaces.

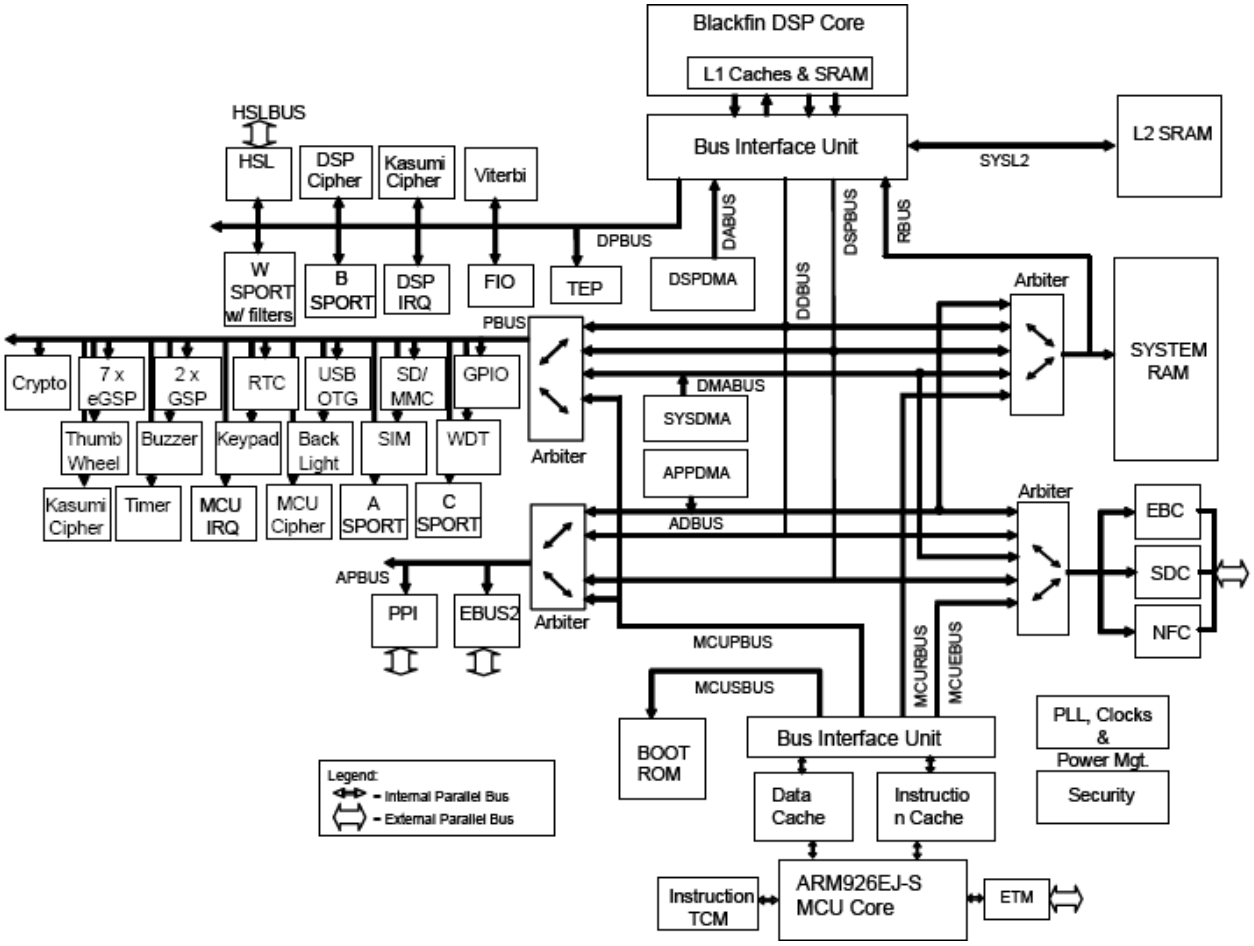

Figure 3.2: DBB internal architecture.

The DSP system has peripherals like BSPORT (base band serial port), WSPORT (wide band CDMA serial port), timing processor and high speed logger. The MCU system has peripherals like ASPORT (audio serial port), CSPORT (control serial port), and embedded macro cell (ETM) for debugging purposes. Thus the MCU is used for controlling operations for ABB through ASPORT (for audio) and CSPORT for interactive controlling. The DSP is used for transferring voice channel data for GSM system through BSPORT and WSPORT.

The system is served by 4 main bus systems:

1. **SBUS:** The local MCU system bus.
2. **PBUS:** The peripheral bus, shared between the MCU, the system DMA, the DSP DMA, and the DSP.
3. **EBUS:** The parallel interface to off-chip FLASH, SRAM memory.
4. **RBUS:** The interface to on-chip system RAM, shared between the MCU, the System DMA, the DSP DMA and DSP.

System buses PBUS, RBUS and EBUS are shared busses these are driven via an arbitration system. Each bus has its own programmable arbiters that can be programmed to define the arbitration priorities and specific function for each arbiter. Figure 1.2 illustrates the brief architecture of DBB. The DBB chip includes house keeping units such as the timers and interrupts controllers.

The Man Machine Interface (MMI) modules provided on PBUS are used for interaction with peripheral devices. The communication between DBB and peripheral devices use serial communication protocols such a SPI, I2C and UART.

3.3 Evaluation boards:

For any chip while developing application different internal signals and outputs from pins must be monitored. But as the chip comes with very small size and with ball pins it is very difficult to debug any application. So the evaluation boards are developed exactly with similar interfaces as end product but added with different types of debug devices. The evaluation boards can be of different sizes based on number and types of debug devices needs to be implemented. The mobile DBB is provided with EVBs named as ANVIL/ANVIL2. The ANVIL board has debug devices like LEDs, 7 segment displays, DIP switches. It provides interfaces to two UART ports, two USB ports, JTAG connector (for debugging). The board has slots for mobile accessory devices like SD/MMC, SIM, audio devices with hook switch, key pad, LCD displays, backlights etc. [9]

To control and multiplexing these devices with proper pins of the DBB chip, there must be a controller. The CPLD device is provided on board is programmed to control all these debug and interface devices. It has some registers that can be configured to multiplex the pins and to change the modes of operations.

The CPLD has interrupt service which can be fired through two sources. The USB interrupt and DIP switch interrupt. The switch state change interrupt is implemented by comparing the current state of the switches with a latched switch state. Whenever the current switch state differs from the latched switch state and interrupt to the digital baseband processor will be generated. [9]

Once the evaluation board is designed, the different modules in it should be validated for proper operation. So this automated package includes the validation of DBB modules and EVB devices. The ATS package consists of 25 modules implemented in it. In next chapters each module's implementation is described.

Chapter 4

TARGET MODULE IMPLEMENTATION

This chapter briefly explains implementation of ATS target modules and their development procedure.

For any application/software to run on embedded chip, the chip should be loaded with some binary code that must be able to communicate with HOST PC through specified communication protocol and do the specified task on every module. This chapter explains how the ARM binary code is developed and implementation details of each module.

4.1 Building procedure and linking:

The ATS target code is a single TAC built for flash memory. It contains the binary form of ARM assembly code for each module, all linked together. The batch file system provided on windows is used for building with different options and to link all the existing modules. The ADS (ARM developer suite) compiler version1.2 is used to compile and link the developed C code. The PERL program is used to keep list of the files to be compiled according to the options provided by developer.

The sample code written for developing the binary code is:

```
rem -- libraries path - CAST and FORGE
set DIR_FORGEHOME=C:\Program Files\Analog Devices Inc\Forge
set DIR_FRL=%DIR_FORGEHOME%\Target_Software\MCU\library\FRL
set DIR_CAST=%DIR_FORGEHOME%\Target_Software\MCU\library\CAST

set DIR_FRLINC=%DIR_FRL%\inc
set DIR_CASTINC=%DIR_CAST%\inc
set DIR_FRLGENLIB=%DIR_FRL%\libs
set DIR_FRLMSPLIB=%DIR_FRL%\libs
set DIR_CASTLIB=%DIR_CAST%\libs

rem -- ADS tools
set ADS_TCC="%ARMCONF%\tcc"
set ADS_LINK="%ARMCONF%\armlink"
set ADS_TOBIN="%ARMCONF%\fromelf"

rem -- ATS build folder
set DIR_ROOT=.\..
set DIR_TCASE=%DIR_ROOT%\tcase
set DIR_SHAREDINC=%DIR_ROOT%\shared\inc
set DIR_OBJ=%DIR_ROOT%\obj
set DIR_IMAGE=%DIR_ROOT%\image\%DBB%
```

*echo ********** ats agent complilation **********
rem ATS bnuild options
set ADS_COMPILE_OPTION=-cpu %CPU% -g+ -O0 -zc -apcs /interwork -W -E+i -D__APCS_INTERWORK -D%PLATFORM% -D%DBB% -D%SDBB% -I "%DIR_FRLINC%" -I "%DIR_SHAREDINC%" -I "%DIR_CASTINC%" -fpu none*

set ADS_LINK_OPTION=-map -list %DIR_IMAGE%\ADSv1.2\memorymap.txt -symbols "%DIR_CASTLIB%"\%PLATFORM%-%SDBB%-ADS.lib "%DIR_FRLGENLIB%"\frlgen-ADS.alf "%DIR_FRLMSPLIB%"\frlmsp-ADS.alf

The compile and link options are set as required, the option switches used in command changes the behavior of binary code being developed. Once the command options are prepared the C files can be compiled as:

*rem -- compile diagagent
%ADS_TCC% -c %DIR_AGENT%\diagagent.c -o %DIR_OBJ%\diagagent.o ADS_COMPILE_OPTION%
%ADS_TCC% -c %DIR_UTIL%\diagutil.c -o %DIR_OBJ%\diagutil.o ADS_COMPILE_OPTION%*

The RO (read only) and RW (read write) base addresses must be defined according to the chipset used. The compile switches (#define) will separate the code based on options provided in batch file.

In ATS total 25 modules are implemented and up to 8 modules are interrupt based modules that are complex and huge. The interrupt based modules implementation is explained in next chapter.

4.2 Clock setting:

In an embedded chip there exist different types of peripheral modules that are capable of operating at different clock speeds. In DBB chip there is a clock distribution mechanism that has one PLL and other clocks for modules like bus, DSP processor, SDRAM clock etc. are derived from basic frequency. The power consumed by chip can be reduced through proper handling and disabling clocks for unused modules. The DBB chips are controlled by clock gating logic and clock divider selectors.

1. **PLLs and their configuration:**

Phase locked loop is a well known circuit that is used to generate and stabilize the clock frequency as required. The following figure shows how the clocks in chip are being divided and distributed to different modules.

The chip has main clock input (CLKIN) generally as 26 MHz. The input block provides filtering and amplification and generates SYSCLK which will be propagated to all the modules in chip based on ratios.

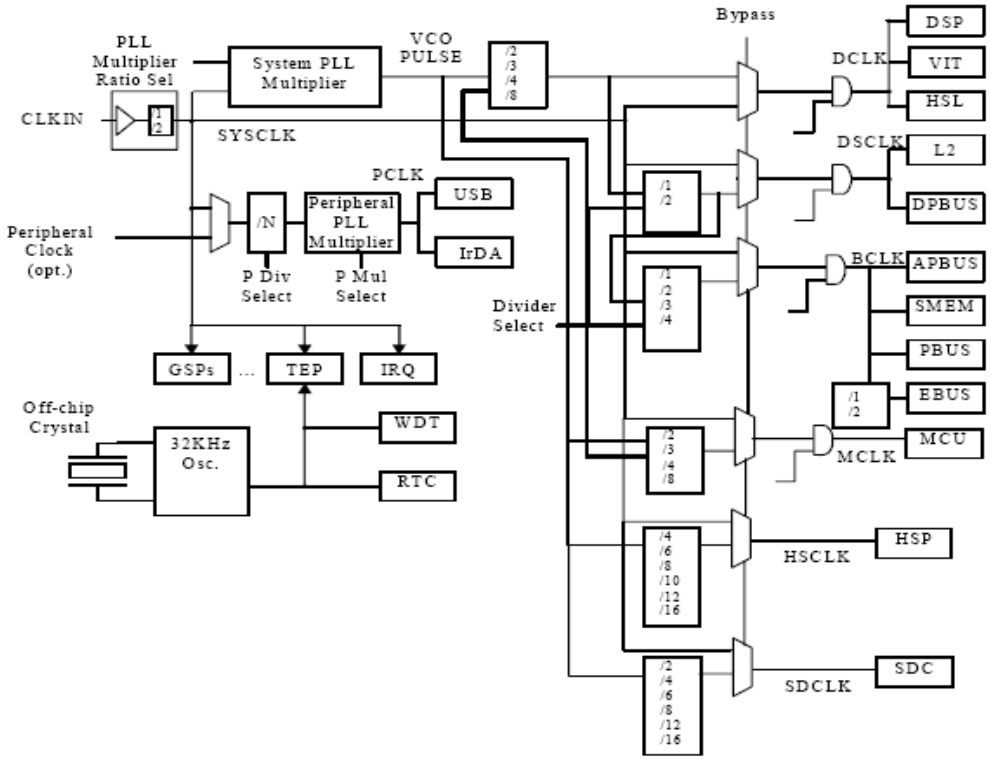

Fig 4.1 Clock distribution mechanism for different modules

2. **Implementation:**

The required clock setting will be passed from HOST GUI as a parameter and it should be with proper configuration (like multiples of 13) for stable processing. There is a dependency on the clock setting of each module, like the BCLK (bus clock) is derived from DSCLK which is again derived from DCLK (DSP clock). The clock ratios for each clock (MCLK, SDCLK, BCLK, DCLK, and DSCLK) are calculated and set by setting clock configuration registers.

4.3 LCD displays (QVGA & QQVGA):

1. **Types of LCD displays:**

   Liquid crystal displays are the commonly used devices for display purpose in mobile devices. There are different vendors that are providing LCD devices, in which the applications must be able to interact with all the types. The LCDs used in ANVIL2 platform are Panasonic, TFS and ZEUS. Each model has its own drivers to interact, so the code must consider the configuration and interfacing to each type of LCD.

2. **Configuring and interfacing to LCD display:**

   The LCD segment is connected on APBUS provided by DBB. The data and commands are sent through APBUS. The display is divided into pixels which will glow according to value received; each pixel has 3 color levels (red, blue, and green) and based on the proportion of each color the color of pixel will be identified. In this module 16bit RGB format is used, thus for each pixel there exist 48bit data. The picture quality depends on the resolution of device; it defines the spacing between each pixel. In ANVIL2 EVB two resolution displays are used.

   QVGA (Quarter Video Graphics Array) pixel size: 320 X 240.

   The name is derived from the fact that it offers 1/4 of the 640 × 480, maximum resolution of the original IBM VGA display technology

   QQVGA (Quarter QVGA) pixel size: 160 X 120.

   The CAST drivers provide interfacing to display devices through, initialize & configure controller functions. The initialize function will acquire the GPIO pins to send command & data. The configure function will provide different options to set contrast range, color mode, display mode (full screen or partial) and pixel positions etc...

3. **Implementation:**

   In ATS both QVGA, QQVGA are implemented separately to test them for text and image displays. In test there will be some delay between display of text and image. The image data is stored in flash memory and the starting address is passed to display function. The stored images are configured to pixel settings of both QVGA & QQVGA separately.

4.4 General purpose I/O (GPIO):

The number of pins to interface to the external world is one of the bottlenecks for chip size reduction. So this general purpose pins provides the concept of multiplexing the pins for different functions.

1. **GPIO modes of operation:**

   The GPIOs are located on PBUS and each of the pins can be configured as inputs/outputs. Each GPIO consists of separate input, input enable, output and output enable functions. If a GPIO input enable is set, the state of the GPIO pin is sampled on each read cycle. If the input function is not used, the input enable should be cleared to save power. If the output enable is set, the pad drives the value of the output register on each cycle. If the output enable is cleared, the output register can be used as a temporary storage area.

   The following figure shows the block diagram of a GPIO.

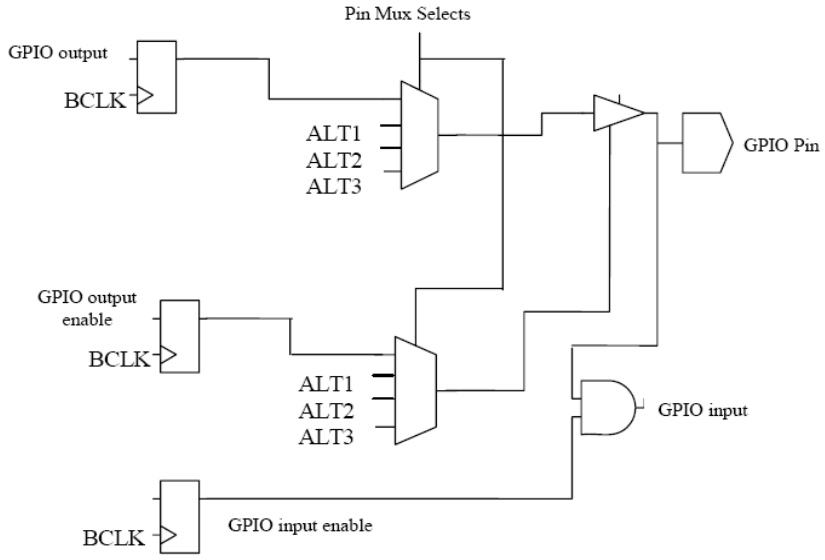

Fig 4.2: GPIO Block Diagram

The ALTx is the mode of operation for pin; each pin has four modes of operations. By default ALT0 will be selected. The mode can be changed by changing bit fields of mode configuration registers, provided for each bit. Thus the mux selects shown in fig will be derived from bit fields.

For programming purposes, the GPIOs are grouped into banks of 16, but each input or output bit can be individually set or cleared. The external pins of DBB are assigned multiple, selectable functions. This increases the applicability of DBB to many different system applications. After booting, the pins must be configured for the intended application. Every pin has a primary function, and most pins have up to three

alternate functions. The grouping of pins provides easy changing of modes and controlling for input/output by setting all the bit fields of grouped pins.

2. **Implementation:**

   There are two types of verifications for GPIOs.

   The GPIO is tested by sending some square signal onto it continuously, in a normal testing (tests only one pin that is fixed in code).

   The other type of test is a very flexible test, in which one debug window is provided to give any pin number, its mode, state (input/output) and value (for input only). Thus the user can test any of the pin in different modes. There is a provision to send one toggle signal onto the pin (with output state) for continuously or only once.

4.5 NOR flash:

In embedded products, there must be a non volatile memory to store the application code, which should not be erased. When the mobile is switched on the code and data are loaded from flash memory. Some applications like messages, contacts etc. write data onto flash memory dynamically, and must be able to access them even after rebooting.

There are two types of flash memories:

NAND flash, designed with NAND type of latches.

NOR flash, designed with NOR type of latches.

NOR flash is characterized by fast random access to bytes or pages of information. NAND Flash is characterized by slow initial access to pages of information with a serial readout from a buffer. NOR Flash is best used in applications where code storage is required and data requirements are moderate (files are small but numerous, and accessed frequently). NAND Flash is geared toward applications where files are large and infrequently accessed.

1. **Introduction:**

   This type of flash memory is generally used in mobiles; it provides a feature called “Execute-In-Place”. The code can be programmed and can be able to execute from same location, thus there is no need for downloading a redundant copy into power hungry RAM for execution, and thus reduces parts count, reduces power consumption, and system cost. As the NOR flash used to execute the code directly, it should be validated for speed of execution in different modes.

2. Modes of operation:

NOR parts have the ‘Read-While-Write’ (RWW) feature, thus single NOR part can be internally partitioned so the processor can be writing to one array while simultaneously reading from the other array. This improves system performance and also removes the need for additional nonvolatile memory devices. When a FLASH partition is taken out of read array mode for programming or erasing, attempted reads to that partition return incorrect data. To prevent this, the MCU is permitted to lock the EBUS arbitration during programming and erasing. This prevents another EBUS master from reading FLASH during this time. MCU arbitration locking is configured in the EBUS arbiter registers.

It has mainly two modes of operation.

Asynchronous page mode:

It will be done by accessing page by page asynchronously, the wait states between page reads are handled in two distinctive parts: the initial access and the remaining paged accesses. The flash interface controller provided on DBB allows configuring page sizes and number of wait states between the each access.

Synchronous burst mode:

In this mode, once the burst FLASH device has latched the address, the data will be read as four half words. By default the flash will be asynchronous mode.

3. Implementation:

This module tests the functionality of NOR flashes in different modes. The objective of this test is to change the mode of flash, run a sample counter program from flash memory, and measure the time taken for execution.

But in ATS all the code is built & downloaded by referring to NOR flash memory, so it is not possible to change the mode of operation while the code is executing from flash. So two small mode changing programs (binaries) are copied to internal SRAM, by the test routine itself and the code is run from internal SRAM. In this test first one counter program will be executed from flash in asynchronous mode and the time will be measured, then the mode will be changed to burst and same counter is executed again. The time will be measured with the help of real time clock (RTC) provided in DBB chip.

4.6 NAND flash:

1. **Advantages over NOR flash:**

   NAND is widely known for size reduction, thus the NAND flash provides high density and available with less cost. NAND flash devices do not support execution of code in place. Hence, the code in NAND flash has to be first moved into a random access memory and then executed, similar to disk like storage devices.

   The programming times are also different. This process takes about 200 μsec for NAND. For NOR this is faster (about 10μsec) but it takes more current. Even though it takes longer for the charge to get into NAND cells, because NAND is programmed a page (528 bytes) at a time from buffer, the total programming time is much faster with NAND than with NOR memory which only programs 1 byte at a time.

2. **NAND flash controller:**

   To improve yields and keep costs down, NAND devices contain randomly located bad blocks in the array. If a memory location is bad, then the entire block is "marked" as a "bad block." Typically, a NAND device starts out with a few bad blocks that are marked by the factory. Usually this is done by writing non- "FFh" data at byte 517 in the first two pages of the block. A programming approach for NAND devices therefore must have a scheme to identify and avoid the bad memory cells when programming the device.

   To handle these bad blocks in the embedded system, an extra "layer" of software is required. In addition, some applications require Error Checking and Correction (ECC) to be calculated for each page of data. The ECC data is used to detect when a memory location "goes bad" and (depending on the ECC algorithm) fix the bad bit. Typically, three bytes of ECC are generated for each page and placed in the "spare area" of the page.

   DBB is integrated with a NFC (NAND flash controller) to handle all these bad blocks and error checking. It acts as interface between the EBS arbiter and the NAND FLASH. Supports page program, Page Read, Block Erase and all allowed operations on NAND FLASH. It features ECC logic to detect 1-bit Error. ECC uses Hamming code to generate 22 bits of ECC data per 256 bytes of data.

Software must handle:

- *Initial bad block detection.* – Software is responsible for reading the bad block information and mapping out the bad blocks. Bad blocks are written with all 00h during shipping, good blocks have FFh.
- *acquired bad block address mapping* – These are the blocks that may be corrupted over time, usually only one or 2 bits can go bad. Hardware can calculate ECC to identify 1 bit data error or ECC error. Software is responsible for identifying the bad bit and correcting the bad data and updating the bad block table if necessary and map these blocks out from being used again. These are relatively simple operations for software to perform, the software-intensive ECC calculation itself is done by hardware with no cycle overhead.
- *wear-leveling functions* – used for increasing life span of the device by ensuring even number of program/erase operations over the completely FLASH memory.

3. **Implementation:**

It validates the block 0 of NAND flash. The CAST drivers are used to initialize, write and read operations. The initialize function will configure NFC with data size; it acquires GPIOs needed and enables chip select.

Write operation will be done page by page, by writing some pre defined random patterns, next the page read is called and the received data is compared with known pattern. Internal SRAM will be used for buffering the data pages in read/write operations.

4.7 Infrared communications:

1. **Introduction:**

IrDA is an open standard defined by the IrDA consortium (Infrared Data Association) for short-range wireless data transfer using infrared radiation. It is connected to DBB through two enhanced generic serial ports (EGSP). Receive data can be edge-detected to convert the modulated data into correct RX data. The IrDA standards have been defined to operate at standard baud rates ranging from 2400 bps to 4 Mbps. The Infrared Data Association has classified these baud rates into three categories, slow, medium, and fast, and defined the physical layer characteristics for each.

Slow IrDA (SIR):

The slow IrDA specification is defined for standard baud rates ranging between 2400 bps and 115200 bps. The data is encoded using RZI (return-to-zero-inverted) modulation with start and stop bits delimiting each byte of data and the pulse duration is 3/16th of the bit duration.

The RZI modulation works as follows:

- If the bit to be transmitted is zero, a 3/16th bit period wide pulse is transmitted in the middle of the bit period.
- If the bit to be transmitted is one, no transmission of signal occurs and the line remains low for the entire bit period.

Hence, in effect the data is reversed during transmission. The reason for logic one being represented as zero is to save the power consumed by the IrDA transmitter. Pulse error rate tolerance should be less than 0.8% for all possible baud rates. There are no error correcting bits used and all error correction must be done by the higher layer software.

Medium IrDA (MIR):

The data is encoded using RZI modulation with a Start flag and Stop flag delimiting each frame of data. The pulse duration is 4/16th of the bit duration. The data modulation is similar to that of Slow IrDA but the bit duration is 4/16th.

Fast IrDA (FIR):

The Fast IrDA specification is defined for a 4 Mbps baud rate. This is not supported by the IRDA module provided on DBB.

2. Developing standalone binary for slave board:

To validate the IRDA communication there should be additional IRDA module to interface with validating board. So one slave ANVIL2 board, containing code for IRDA communication needs to be setup. This slave board code is no need to interact with any HOST communication, so it doesn't need any ATS framework interfaces. This standalone binary is developed directly by using startup code provided by CAST drivers. It doesn't contain any FORGE TAC.

Once the board is loaded with binary it will wait for selection of transmit/receive from keypad, then it will communicate with any other IRDA module.

3. **Implementation:**

   The procedure for implementing IRDA is:

   - Configure CPLD on ANVIL2 to route GPIO pins to IRDA device.
   - Acquire the two EGSP ports and GPIO pins to operate for IRDA communication.
   - Configure IRDA registers with desired mode (Tx/Rx), speed and baud rate.
   - Send data if it is transmit mode, if it is receive mode enable watch dog timer and wait for receiving the data.

4.8 Monitor port:

1. **Introduction:**

   In developing applications for mobile systems, there is a need to monitor some signals for verifications which exist internal to the chip. Thus DBB provides a Monitor Port that allows internal hardware signals to be observed on the MONITOR bus for debug purposes. The MONITOR [12:0] bus can be selected as an alternative function of GPIO pins. There are several Monitor Port signal groups, each of which provides access to a specific set of internal signals. The module is selected via the Monitor Port Module Register. For each module, the Monitor Select Register is used to select a particular class of internal signals.

2. **Testing for clock signals:**

   This module is tested by configuring it to display one group of clock signals. So by changing operating clock for DBB the different frequency signals can be observed on monitor port i.e. connected by GPIO [12:0].

   The ANVIL2 CPLD has to be configured for selecting monitor port and GPIO modes are changed to get monitor port signal. The proper group and port signal is selected for clock signals to get connected to monitor port.

4.9 ANVIL2 CPLD interrupt:

As discussed in third chapter the ANVIL2 platforms provides different type of debug devices, interfacing mechanisms to communication peripherals etc… To control all these devices and to interface with the chip properly it needs a programmable controller. Thus the EVB is provided with ALTERA CPLD for this purpose. It has one interrupt service provided for debug devices; this module validates the interrupt facility provided by ANVIL2 CPLD.

1. **CPLD introduction:**

   Complex programmable logic device (CPLD) is a programmable logic device with complexity between that of PALs and FPGAs, and architectural features of both. The building block of a CPLD is the macro cell, which contains logic implementing disjunctive normal form expressions and more specialized logic operations.

   Features:

   - Non-volatile configuration memory. Unlike many FPGAs, an external configuration ROM isn't required, and the CPLD can function immediately on system start-up.
   - Large number of gates available. CPLDs typically have the equivalent of thousands to tens of thousands of logic gates, allowing implementation of moderately complicated data processing devices.

   This CPLD handles the configurations provided for different connectors like monitor port, audio in/out, RF interface etc. It interfaces with the digital baseband processor via EBUS1 and presents to the digital baseband processor a set of registers used to configure and monitor certain board resources. These resources include driving the 16 diagnostic LEDs, driving the two 7-segment displays, reading the state of the 16 diagnostic switches, providing an interface to the Debug USB port and configuring several digital multiplexers that control signal switching between the digital baseband and various plug-in modules.

   The device is programmed from a PC serial or parallel port with a simple programmer cable that plugs into a JTAG header on the motherboard. After programming, the device programming file resides in the configuration flash memory of the CPLD. The programming information is transferred from the on chip configuration flash memory (CFM) to the globally distributed SRAM programming bits of the CPLD every time the system is powered up. This transfer takes a maximum of 300 microseconds to complete.

2. **Implementation:**

   This module first initializes the keypad and then unmasks CPLD DIP switch interrupt. Then the interrupt status is keep on monitored and if any flag rises, the value of DIP switch will be displayed on LCD display. This test continuously runs until user presses any key on keypad.

4.10 Debug devices:

1. **Introduction:**

   The ANVIL platform is designed to provide maximum possible debug devices while programming the chip. The basic debug devices on ANVIL2 platform are:

   LED:

   It has 16 LEDs arranged in form of 4X4 matrix, thus any half word data can be sent to this address for debugging display.

   Seven segment display:

   Two display segments are provided on board, thus any half word data can be sent to this address for debugging display.

   DIP switch:

   Dual-in-line package (DIP) switch is used to send inputs to the chip. It contains total 16 switches, thus one half words can be sending as input.

2. **Implementation:**

   These debug devices also needs to be validated after the fabrication, the each module is implemented as:

   Seven segment display:

   Each 7 segment display has 8 segments (including decimal point, DP) and two display devices are provided on ANVIL platform. In this module all the 16 segments are switched on and off with some delay between them. Next each display device, will cycle through all hexadecimal digits, from 0 to F.

   DIP switch:

   The state of the 16 DIP switches is stored in register, thus it will be read for a fixed time frame and display the status of switches on LCD display, LEDs and seven segment displays. Any change in switches status will be monitored until, test exit by pressing any key on keypad.

   LED: It has two types of tests.

   Walk through test:

   Each LED is switched ON and OFF with some delay, it will be done through all the LEDs, finally all the LEDs will be switched on simultaneously, by sending 0xFFh to this address.

Debug test:

In this there is a debug window provided on HOST, and that is designed same as LED arrangement on board. By selecting the desired LEDs they can be switched on dynamically.

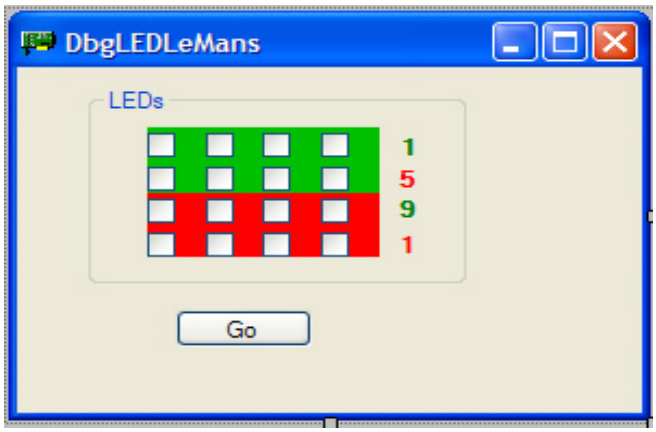

Fig 4.3: LED output window

Thus the LED can be validated interactively, through this mechanism.

4.11 Accessory devices:

1. **Introduction:**

The mobile system provides mobile accessory device support also like buzzer tone generator, hook switch (input switch from headset), back lights (key pad, LCD display lights), service lights etc.

Buzzer tone generator:

DBB contains a Tone Generator module (located on PBUS) that can be used to output custom tones defined by duration, frequency, and power level. This can be made available via GPIO interface. It has three registers dedicated to configure duration, frequency and volume level.

Hook switch:

This module is provided by ABB, which can be controlled through CSPORT (control serial port). The switch is provided on EVB and connected to audio connecter and ABB.

Back lights:

The backlights controller consists of 3 independent backlight circuits, each of which can be programmed for output frequency and duty cycle. They can be individually enabled, defined via the backlight CTRL register. Each of the backlights and the service light generates a Pulse Width Modulated (PWM) signal. All these light controllers take input from SYSCLK (derived from PLL). The number of service

lights and back lights varies based on DBB chipset. The backlights for displays must be on to work with any of LCD displays, otherwise the data sent to displays is not visible.

Service lights:

This is another optional peripheral support provided on chip, this is also same as back lights but not dedicated.

Each light controller (back light or service light) has one configuration register, which allows to configure frequency, duty cycle and pre scale parameter.

Duty Cycle – The Duty Cycle defines the reset point for the corresponding Backlight Output (i.e. every time the counter reaches the value defined in Duty Cycle, the output of the corresponding backlight is reset to 0 in the next cycle). The output is set again when the counter is reset. The Duty Cycle should be programmed to any value between 0 and (FreqSel – 1).

Frequency Select – Defines the rollover point for the corresponding Backlight counter. Every time the counter reaches the value defined in FreqSel, it resets in the next cycle.

Pre Scaling Factor – The corresponding Backlight is clocked with a clock that is divided based on the Pre Scale Factor:

2. Implementation:

Buzzer tone generator:

GPIO pins which are connected to hook switch are acquired and the configuration registers are programmed with some set of frequency, duration and volume level. The output signal from buzzer is verified through oscilloscope. The signal will be generated for some duration and then is disabled.

Hook switch:

By using CSPORT the command to enable the hook switch is sent and the press or release status of switch is displayed on LCD display and 7 segment displays.

Back lights and service lights:

For service light the GPIO pin is acquired and corresponding configuration register is programmed with desired frequency and duty cycle. The light is enabled for some time, and then it is disabled. The LED provided on EVB for this service light is toggled for some time.

The backlights for QVGA, QQVGA displays and keypad are controlled through ABB. Thus the CSPORT is configured and the control commands with desired

frequency and duty cycle are passed through CSPORT. All the backlights are toggled for some time sequentially.

4.12 Debug UART communication:

For any chip the interfacing with outside world is depends on the communication protocols implemented in it. The mobile chip provides interfaces like USC (universal System Connector), USB (Universal Serial Bus), GPIOs and UART (universal asynchronous receive and transmit) communication ports.

1. **Introduction:**

   The EVB platform provides two UART ports, download and debug ports.

   The download port is acquired by FORGE once the target code is downloaded and ATS is running in the system. The debug port validation is provided in ATS with help of hyper terminal service provided by windows. The chip has same interface for both UART ports, only the EVB provides two ports to provide additional debugging capacity.

2. **Implementation:**

   The download port and debug port are connected to EGSP modules. So the EGSP port is acquired with desired baud rate and port selection. The CAST drivers are used to transmit and receive data through UART port. The test will send back the data received from port and checks for string “end”, if the string found the test will exit.

   The hyper terminal on PC is configured with same baud rate of target and is connected with the UART port. Once the test starts, the characters sending from PC are echoed back to hyper terminal.

Chapter 5

IMPLEMENTATION OF INTERRUPT BASED MODULE

This chapter briefly explains development of interrupt mechanisms in ATS and their implementation.

IMPLEMENTATION OF INTERRUPT BASED MODULES

In any microprocessor when interacting with peripheral devices or in asynchronous communication instead of just waiting in a loop for the response (polling mode), it is necessary to use interrupts. As the other devices are having different speeds of processing, response times and availability checking for task handling, the setting of interrupt for each device based on priority will give good improvement in performance of embedded product. The difference with interrupt driven modules are, there exist an interrupt handler written for each interrupt and will be called once the interrupt is fired. Some modules may have multiple handlers for handling complex tasks.

The DBB chips have some list of dedicated hardware interrupts that are categorized based on priorities, and some set of software interrupts. This chapter discusses the implementation of interrupt driven modules.

The widely used interrupt in ATS is DMA interrupt, for handling DMA transfers between different memory segments and peripherals. It is used in ESRAM, SDRAM, and CAMERA etc...

Interrupt driven modules in ATS are:

1. ESRAM
2. SDRAM
3. key pad
4. debug USB
5. Camera
6. secure and multimedia memory card (SD/MMC)
7. SIM card
8. audio serial port (ASPORT)
9. baseband serial port (BSPORT)

As discussed in above chapter, in embedded products different types of memories are needed, in addition to flash memories some volatile memory also needed for temporary storage of data and code, buffering of any data received from peripherals etc. The EVB has two types of volatile memories provided for different applications.

The basic types of volatile memory are static RAM and dynamic RAM, both have advantages and disadvantages. For large amount of memory requirements DRAM is used due to high density where as for faster response SRAM is used. The DBBs have on chip SRAM and the

EBUS provides interfaces for both SRAM/DRAM to locate externally. So the ANVIL2 EVBs are provided with some blocks of SDRAM and PSRAM. The ATS provides validation of these modules in different modes.

5.1 PSRAM:

The DBB chip has internal SRAM for scratchpad purposes, the objective of providing interface for additional off chip memories is to have high amount of volatile memory. Thus the off chip memories must have high densities, so the DRAM like products are used as off chip memories. PSRAM is a DRAM based structure with interface is like SRAM. It contains built-in refresh and address-control circuitry to make it behave similarly to static RAM (SRAM). PSRAMs can be accessed in standard asynchronous modes, asynchronous page modes, or synchronous burst modes.

This module validates the normal and burst DMA transfer modes of PSRAM.

1. **Introduction to DMA:**

   In mobile applications, data incoming or outgoing through embedded I/O devices must often be managed at high speeds or in large quantities. DMA, a dedicated data transfer device reads incoming data from a device and stores that data in a system memory buffer for later retrieval by the processor. This DMA process occurs transparently from the processor's point of view. DMA also off-loads the processor, which means the processor does not have to execute any instructions to transfer data. Therefore, the processor is not used for handling the data transfer activity and is available for other processing activity.

   A DMA controller can directly access memory and is used to transfer data from one memory location to another, or from an I/O device to memory and vice versa. DMA controller manages several DMA channels, each of which can be programmed to perform a sequence of these DMA transfers. A DMA controller typically shares the system memory and I/O bus with the CPU and has both bus master and slave capability. Each DMA controller contains seven controller configuration registers and many individual channel programming registers. Thus by programming the registers, corresponding DMA channel configuration (source address, destination address and notify address etc.) can be defined.

The following diagram shows the DMA subsystem:

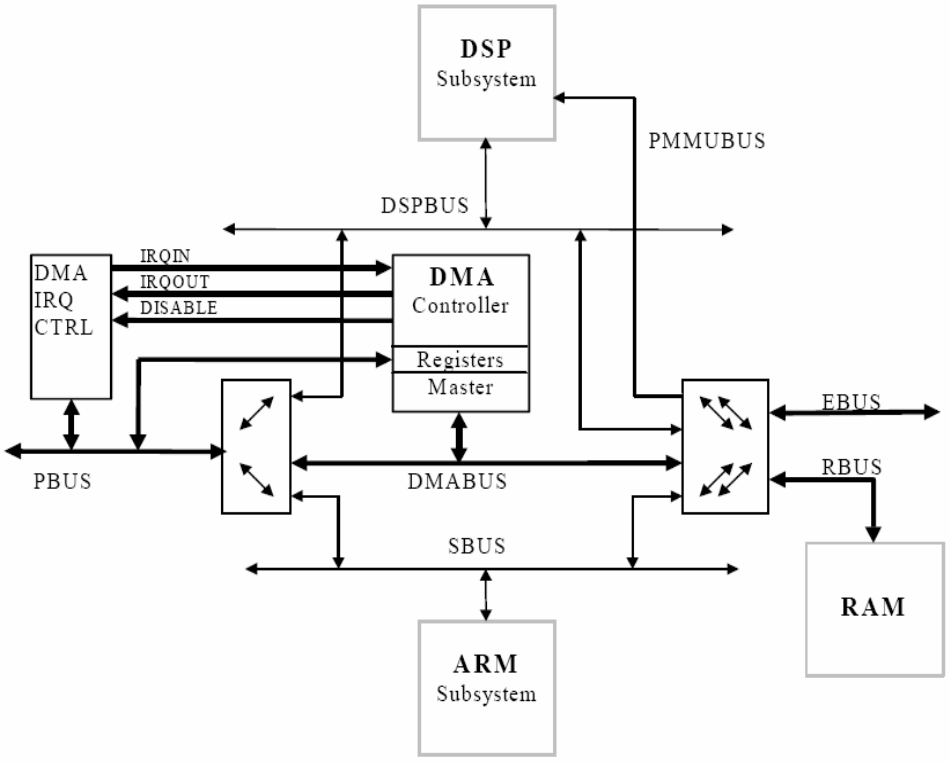

Fig 5.1: DMA subsystem in DBB

DBB provides three DMA controllers to relieve the MCU and DSP from repetitive operations, such as servicing IO devices and performing transfers from one memory block to another. The DMA subsystem includes the DSP DMA Controller (DSPDMA), the System DMA Controller (SYSDMA), and the Applications DMA Controller (APPDMA). Each DMA controller is the sole master of a DMA bus. Using these buses, the DMA controllers have the same access to arbiters for shared memory spaces and input/output devices as the MCU and DSP cores. The DMA controllers permit higher priority channels to interrupt lower priority channels. There are early warning and transfer completion indicators to notify the processor cores that a DMA transfer is ending.

2. **Implementation:**

This module has three set of tests:

Walking through:

In this test of walking ones, one bit setting as '1' will be loaded in each memory location and is shifted bit-by-bit through all 32 bits. For each shifting operation the data is read back from the location and verified.

Memory fill:

In this test each location in the selected block is loaded with the value of inverted address, and is verified for all the locations simultaneously.

DMA burst transfer:

This part configures MCU & ESRAM in DMA mode and sets one interrupt handler. The internal SRAM is filled with patterns of "0xAA55" for the selected DMA transfer size. Then the DMA transfer channel is configured with source address, destination address and notify address (source address + transfer size). After the transfer is completed, the each location is verified for the pattern "0xAA55".

5.2 SDRAM:

1. **Introduction:**

SDRAM is the mainly used memory device for buffering operations. The interfacing to Synchronous dynamic RAM is provided in DBB chip & the memory is located on evaluation board. One SDRAM controller (SDC) is provided on DBB to interface with off chip SDRAM memories. The interface includes timing options to support additional buffers between the processor and SDRAM, to handle the capacitive loads of large memory arrays. The SDC provides controlling for setting normal transfer or DMA burst transfer.

2. **Modes of operation:**

The SDRAM has two modes of operation for recharging the memory cells from capacitor discharge.

I. Auto refresh mode
II. Self refresh mode

Auto refresh mode:

The SDC provides setting of refresh counter for refreshing the data. Auto refresh mode will have a timer setting and refreshes the cells based on timing. Before

executing the Auto-Refresh command, the SDC executes a Precharge All command to the bank.

Self refresh mode:

The Self-Refresh command causes refresh operations to be performed internally by the SDRAM, without any external control. It is like sleep mode in which the SDRAM just contains the data but interfacing is off, if the SDRAM is accessed in self mode it will automatically changes to auto refresh mode.

3. **Implementation:**

   In this module one memory segment is selected and written by sequence of digits. Then the mode is changed to self refresh mode, after which the data is verified by reading. As the data is accessed the mode of SDRAM will automatically changes back to auto refresh mode.

   Other part of test will create one interrupt handler and then configure SDRAM to DMA burst mode, writes block of data from internal SRAM to SDRAM and then verifies it by checking each location.

5.3 Keypad Interface:

1. **Introduction:**

   The DBB offers keypad support for mobile system. The keypad interface supports a row/column decoding of a keyboard matrix of up to 8 rows and 8 columns. The interface generates an interrupt when any key is pressed, and the interrupt is routed through the system interrupt controller to the MCU. Software must scan the columns to determine which key was pressed. This keypad controller gives some set of registers to monitor state of the interrupt, row/column output registers and control register.

2. **Implementation:**

   The keypad is initialized by clearing all the output registers and flushing the key buffer. One interrupt handler is written to check for key press or release. The pressed key is read and compared with the sequence of keys. Thus all the keys provided on EVB are verified.

5.4 Debug USB cable:

The EVB platform provides two USB ports, download and debug ports. The GPIO pins corresponding to debug USB are acquired and the interrupt handler is set for transmit/receive.

The CAST library functions are used to interface with USB. The USB 2.0 standard protocol is used. The test will check the id returned by USB cable connected to PC and enumerates it.

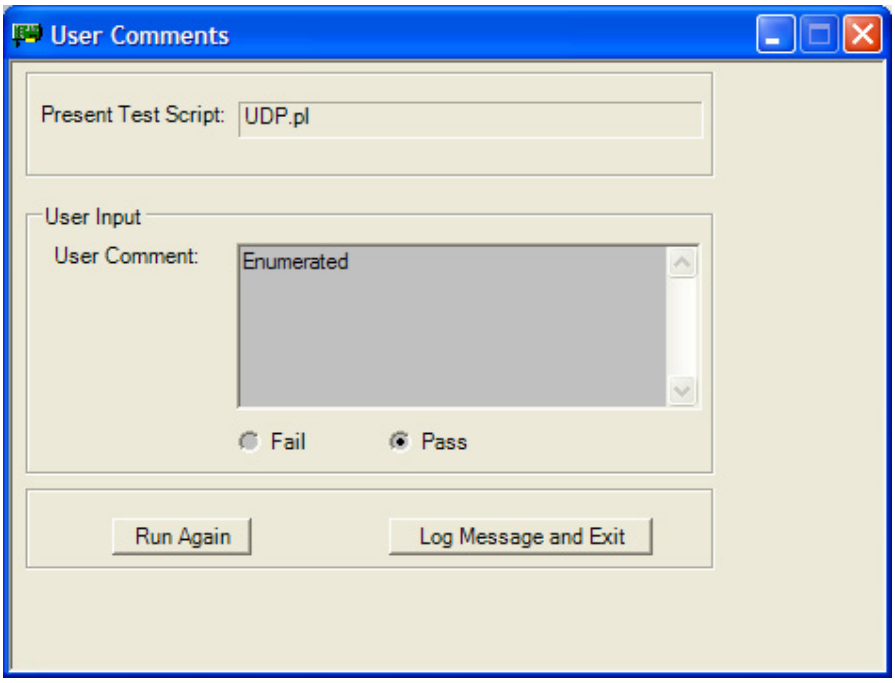

Fig 5.2: USB test output (common for all tests)

The figure shows the output result of ATS for debug USB test, the result window provides facility to enter comments for result logging.

5.5 Secure data and multimedia memory card (SD/MMC):

1. **Introduction:**

   Now days the applications in mobiles are growing and the data files in mobile are increasing. Thus the usage memory flash cards in mobile became necessary. The DBB provides interfaces to external memory cards that can be replaced. It provides an interface for Multi-Media Cards (MMC), Secure Digital Memory Cards (SD Card), and Secure Digital Input/Output Cards (SDIO) which are standard in mobile applications. The interface has card detection and interrupts handling facilities.

   This provides set of registers to power/clock control, response and status registers, interrupt mask registers etc. it has same interrupt provided for both transmit and receive, thus it can do half duplex communication only.

2. **Implementation:**

   The procedure of implementing SD/MMC module is:

   The MCU is set at 52 MHz and one interrupt handler is set for card detect and GPIO pin corresponding to SD/MMC interface. If the card is inserted in slot provided on EVB, the interrupt is raised and card is initialized with required configuration.

The transmit operation is started by configuring data length, data timer timeout. The transfer interrupt is setup with transmit handler and the FIFO empty interrupts are unmasked. Thus once the block of data is transmitted the empty interrupt is raised and next block transfer is started.

After the transmit operation is over, the receive operation is started in same way, the interrupt handler for FIFO empty interrupt is changed to receive handler. Thus the receive operation is done with specified configuration. Finally card deselect command is passed and the test exits.

5.6 SIM (Subscriber Identity Module) card:

1. **Introduction:**

   SIM is a removable smart card for mobile phones. SIM cards securely store the service-subscriber key used to identify a mobile phone. DBB contains a dedicated smart card interface, conforming to the ISO/IEC7816-3 (Identification Card) standard. The SIM interface module functions as an asynchronous serial communications port. It is half-duplex and shares hardware between receive and transmit modes. The interface is symmetric, with the same timing for receive and transmit modes. The interface provides set of registers for status, data, control registers for interrupts and bit rates.

2. **Implementation:**

   The GPIO pin is acquired (configured with necessary mode and state) and the interrupt handler is set with SIM interrupt. The handler uses the CAST library resources for transmit and receive. The SIM detect and un_detect checking is also provided in this module.

5.7 Audio serial port (ASPORT):

1. **Introduction:**

   The mobile systems have audio processing applications and this mobile solution provides different types of general audio transmission and audio interfaces. The ASPORT is dedicated serial port for audio communication between DBB & ABB. In a phone call the audio decoder, encoder and ASPORT operate in a full duplex mode simultaneously sending and receiving voice samples. Monophonic audio samples or stereo sample pairs can be sent from the DBB to the ABB via ASPORT.

The ASPORT has the following capabilities:

- Communication applications with digital sample rates of 8 or 16 kHz.
- Supports simultaneous stereo decoding and monophonic encoding for applications with digital sample rates of 8 kHz or 16 kHz.
- Supports monophonic and stereo audio data requirements.
- Provides DMA transmit (AsportTxI) and receive (AsportRxI) interrupts, mapped to both the DSP DMA and System DMA controllers.
- Allows programmers the flexibility of using SYSDMA, DSPDMA, DSP or MCU for data transfers. The transmit and receive buffer registers have sufficient capacity for the latency that the DSP or MCU needs to service a data interrupt.
- Provides clock gating for reduced dynamic power.

The ASPORT is a three-pin serial interface to the Analog Baseband (ABB) chip. The frequency of data transfer is controlled by the ABB. The ABB initiates data transfer requests via the frame sync pin. Data then flows between DBB and the ABB in 16-bit packets at SYSCLK frequency on the data input and data output pins. The data transfers may be half-duplex or full-duplex.

The Audio I/O section provides interfaces for up to two microphones and four speakers. The two audio input channels can also be configured as line inputs. An analog I/O port is provided mainly to accommodate FM radio ICs. The Audio section provides interfaces that enable the user to realize phone call voice band audio uplink and downlink, polyphonic ring tone play, MP3 playback, gaming sound play, voice recording, and FM radio play. The Audio I/O has a set of control register bit fields accessible via the CSPORT interface. Uncompressed 16 bit Audio samples are transmitted and received by the DBB chip over the ASPORT serial interface. The samples are transmitted at the user selected sampling rate. The Audio Output signal path consists of a decoder, analog adder, programmable gain amplifier, analog switch and driver amplifiers connected to each speaker in the system. Two separate Audio decoder channels are used for stereo output, while only one is used when doing monophonic output. AOUT1 and AOUT2 support mono only.

2. **Implementation:**

The control register provided for ASPORT can configure the half/full duplex transfer mono or stereo communications and size of transfer etc. The ANVIL2 EVB provides switching between different audio interfaces of AOUT1, AOUT2 or AIN1, AIN2 etc.

This module has different tests:

Audio Sweep:

The ASPORT channel is configured as stereo mode with double frame synchronization. Thus the frame synch occurs for a 32-bit L/R separately for each channel. The configurations for audio output like gain, frequency are passed through CSPORT. The audio file formatted as 8 kHz stereo data is stored on flash memory and the starting address passed to transmit function.

Second time the audio channel is configured as stereo mode with single frame synchronization, thus the frame synch occur once for a 32-bit L/R pair for each channel. Same audio file is played thus the both synch modes are validated.

AIN1 to AOUT1 loopback:

In this part the handset interface provided on EVB is verified. The interrupt handlers are set for transmit and receive on ASPORT, the ABB is configured to route the in/out audio signals to handset connector. Thus the same handset is used to transmit and receive the voice. The channel is configured as mono with full duplex, as the handset is used.

AIN3 to AOUT3 loopback:

This is also same loopback test, but the ABB configured for audio in 3 and out 3 ports. The interrupt handler is keep on monitoring the tx/rx status and transfers the data. The loopback test runs continuously till the key press.

5.8 Base band serial port (BSPORT):

1. **Introduction:**

DBB has a baseband serial port (BSPORT) in the DSP subsystem to support GSM, GPRS, and EGPRS data transfers. The BSPORT is a full duplex serial port interface between the DBB and an analog baseband (ABB) device. The BSPORT may also be configured to operate in half duplex mode to reduce the ABB pin count requirements. In both cases, the BSPORT is used to transmit uplink burst data and receive I/Q samples.

2. **Implementation:**

   The code will detect type of ABB based on EVB configuration register and configures BSPORT control registers to communicate with specific ABB. Then the transfer operations are initiated. The transmit and receive operations are done sequentially.

5.9 CAMERA:

1. **Introduction:**

   Now day's camera devices are widely used in mobile systems for single capture & video capture that needs support for different resolutions and handling of memory and power consumption.

   In order to run camera application continuously, the image data i.e. captured has to be stored in a buffer and send it to LCD display frame by frame. SDRAM provided on EVB is used as a buffer for each frame. This will test different combinations of data transfers between PPI & EBUS. It needs different DMA transfers and buffers for captured frames. The DBB provides Parallel Peripheral Interface (PPI) in application subsystem which will be mainly used for camera application to take input from camera sensor. The EBUS is used to send the buffered data to LCD display.

   In ANVIL2 evaluation boards we have two LCD displays with different resolutions as QQVGA (360 X 240) & QVGA (160 X 120) and interface support for different vendor's sensors like micron/ Omni vision (OV2630) etc. So the module has different switches & configurations for each device based on their drivers.

2. **Operation:**

   The PPI is configure to take raw RYB data for each pixel based on resolution through DMA channel and store it in a buffer. Then another DMA channel is configured to take the data from memory and send it onto EBUS which is connected to LCD device.

3. **Implementation:**

   The figure shows the flow diagram for camera module implementation. The SDRAM provided externally on ANVIL2 is used as buffer to store captured frames. The I2C communication is used to communicate with the interface circuitry provided on camera sensor.

   All the initializations for SDRAM, I2C, LCD, key board and PPI are done. As this application is developed based on FORGE, the code starts from *taskstart.* The camera type is detected by scanning the configuration register and then the specific drivers are enabled. Then the configuration needed to display image with required size are

done for both input (PPI configuration) and output (image configuration). The interrupt routines are set for two DMA channels and are enabled.

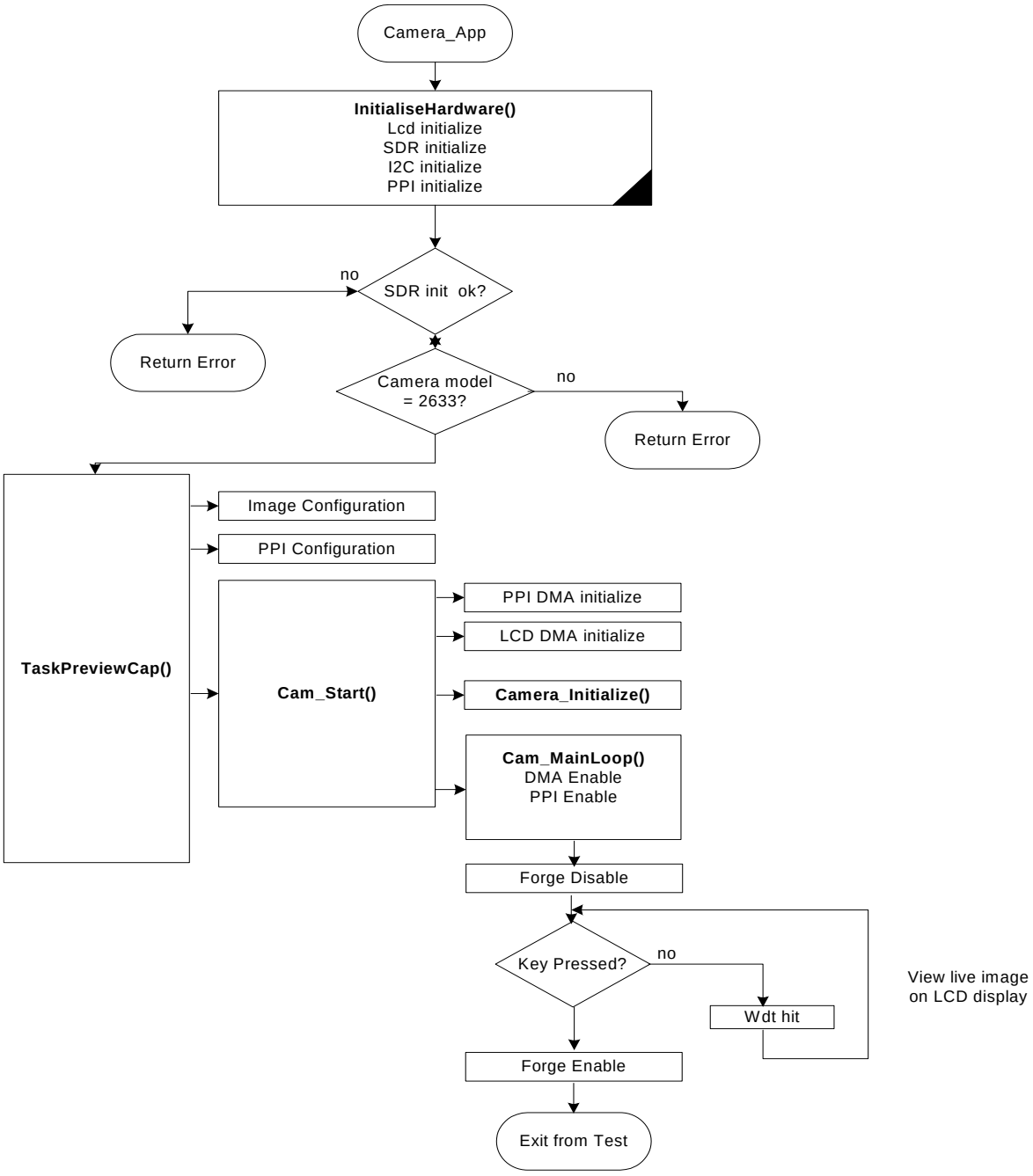

Fig 5.3 flow chart for camera module

Thus the data transfers will start and the code runs till the key press. Each frame is captured from sensor and stored in SDRAM. Then the data is sending to display through EBUS. During this operation the chip is monitored by watch dog timer for proper operation.

Chapter 6

ATS REARCHITECURE

This chapter explains development of ATS rearchitecture for efficient implementation and execution.

6.1 Need for changing architecture:

The ATS is developed based on FORGE platform which provides different types of target services to interact with DBB chipsets. In ATS main use with FORGE is only the communication between HOST & target. The other services provided by FORGE are not used in ATS. Thus adding of big software like FORGE only for communication purpose is redundant.

Also for any new DBB chip bring up of FORGE needs some duration & ATS development has to wait until the FORGE is complete to use its services. This delays the development of ATS and validation with in the time is not possible. So these issues forces to remove FORGE in ATS and re implement its architecture with own communication mechanism. In old ATS as it is id based, there should be some memory allotted to store the ids of each module. This redundant memory is removed in new ATS by removing id based structure.

6.2 Framework design:

Framework is the layer between ATS target validation code and HOST GUI. It has both HOST & target sub layers to interface with target modules, GUI and it contains serial communication protocol implemented for transfer of data between HOST & target.

The main aim of ATS is to reduce bring up time of ATS and minimize hardware dependency. Thus the framework is developed by using minimum resources, just it has basic power up initialization and UART interrupt handlers for transfer of data. Same ATS GUI is used in this architecture, but instead of Perl files it runs the executable provided by framework.

In new ATS the test code is downloaded dynamically before testing. Thus there is no dependency between each module and with framework. The framework code is built and downloaded on to flash memory. It runs from flash once the board gets boot up. Then it will wait in infinite loop for command from HOST and provides memory slot to download the test code from PC.

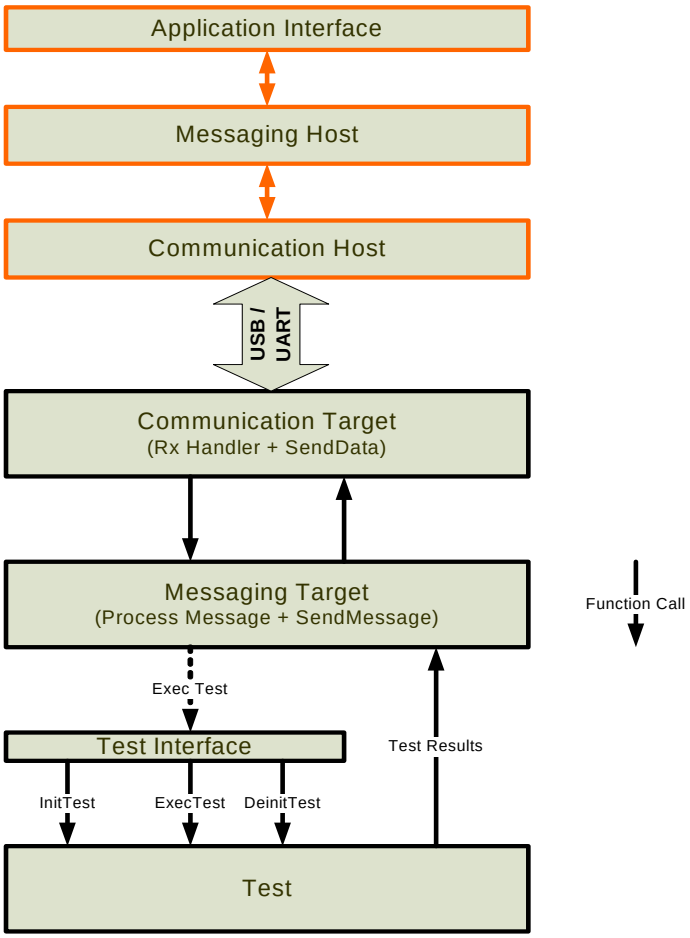

Fig 6.1: New ATS framework model

The figure shows the connections between the layers in ATS. The framework has only UART receive interrupt enabled, to receive the commands from HOST, to download the tests to ISRAM and for messaging services. The communication between HOST & target is through USB/UART protocols. The application interface provided in framework will interact with GUI for taking the commands, parameters and sending the log messages.

Host and Target framework software features:

- Download to ISRAM
- Execute code from ISRAM
- Send data to Test – configuration and input parameters
- Receive data from Test – results
- Download and execute multiple tests in sequence

6.3 Building and linking:

In new ATS the framework provides the some target services and startup functions to include the code. The test code is called from the functions provided by framework. The code is stored in internal SRAM, thus the RW & RO base addresses are set with ISRAM memory

and the ADS compiler is used to build the ARM code. The batch file system is used to list the files to be compiled and the libraries needed for linking. The ARM executable generated will be given as input for GUI for downloading on to board.

6.4 Implementation:

The interface between the target communication layer and test modules is through the 3 functions: *init test, exec test, Deinit test*. The target provides messaging service for sending any feedback messages to the HOST.

In this structure each module is independent of others, thus the module will called directly from the functions provided by framework. One utility file is provided, which has implementation of some target services provided by FORGE. This provides utility services like: display on LCD, delay, setting & clearing the interrupts, switch on the back lights etc.

The new framework is designed such that the target development is not effected by its architecture. Thus there is no need to change target modules implementation model and only the framework layer has to be applied for all the modules. NOR flash memory validation is effected by new model and has to be designed with new architecture.

NOR flash implementation:

Original implementation:

In old ATS the complete code is built and loaded in NOR flash memory and is executed directly from flash. For each module the code segment is selected by framework and is executed. In NOR flash test there is a need to change the modes of operations for effective validation of NOR flash device, but as the code is running from flash itself it is not possible to change the modes. So some mode changing program is written in assembly and is copied onto internal SRAM, then the control is passed to it. Thus the modes of NOR flash can be changed and is validated.

New implementation:

In new architecture all the code is copied and executed from internal SRAM, thus the NOR flash modes can be directly changed through control register configurations. Thus it eliminates the redundant copying of code segments from other locations. This avoids the inclusion of assembly coding in the development of ATS modules.

All other modules can be directly ported to new implementation by applying the framework interface layer.

6.5 Advantages:

The new ATS has several advantages:

- As the framework doesn't contain much resources, the bring up time for any new chip is small. Thus the framework can be developed faster.
- Each test module is independent so if any module is not properly working, it won't effect the development of other modules.
- As the test is downloaded dynamically, at the time of testing. Any new modules can be added easily by just copying the binary files.
- This is not id based testing, so no need to keep the database of ids for each module. It reduces the memory used by test.
- Common structure for every module helps easy understanding of module implementation and new modules can be easily ported.
- PERL usage for invoking FORGE run time (FRT) and passing of parameters is removed completely, thus the ATS developer load of using PERL is eliminated.
- FORGE is completely removed, thus there is no dependency and the development can be started directly once the framework is done.

6.6 Disadvantages:

- In this new ATS as the test code is downloaded dynamically, so it takes long time for each module for validation.
- The common utility functions are building and downloaded with each module every time; this increases the size of each binary and the time of validation for each module.

Conclusions and Future scope

Conclusions:

ATS plays major role in validating chipsets of ADI. This helps the developers to concentrate on the application development instead of processor related issues. It also helps faster development validation applications for new chips. Since ATS concentrates on both HOST and target side helps the users to have flexibility in developing TACs that uses interactive controlling between HOST and target.

The new ATS architecture gives more flexibility and efficient implementation of validation software. As it uses very less hardware resources in framework layer, its bringup is faster and the modules can be developed independently to each other which reduces development time of ATS.

Thus the implemented new architecture gives very good implementation and performance compared to existing ATS architecture.

Future Scope:

The common functions can be built with target framework to load on to flash. Thus there is no need to download same code again and again, so reduces the size of binaries.

The common services included in framework will increase the size and dependencies in framework. This may delay bring up of framework in turn development of ATS package. This can be eliminated by developing target services based on any switch or parameter. Thus once the services are ready then only they can be included in framework and flash memory.

Currently the ATS communication with target is through UART cable which will be slower, so it is better to implement USB protocol for faster downloading and interactive execution.

APPENDIX A

Introduction to ARM architecture:

ARM controllers are widely used cores in embedded system development; it offers different type of cores and variants to select as per need. Based on the product requirement the user can select the variants like floating point unit, DS Processing unit, Java acceleration (Jazelle™), security (Trust Zone™), Intelligent Energy Manager (IEM) and thumb unit.

For example the architecture named **ARM7TDMI** will have:

ARM 7 instruction set, thumb instruction set, debug, ice, multiplier variants embedded into it.

ARM926EJ-S will have:

ARM9 instruction set with enhanced DSP unit (E-DSP), jazella, and synthesizable variants.

The ARM is a reduced instruction set computer (RISC) and incorporates the typical RISC features like:

- A large uniform register file
- A load/store architecture, where data processing operations only operate on register contents, not directly on memory contents
- Simple addressing modes, with all load/store addresses being determined from register contents and instruction fields only
- Uniform and fixed-length instruction fields, to simplify instruction decode.

In addition, the ARM architecture gives:

- Control over both the ALU and shifter in every data-processing instruction to maximize the use of ALU and shifter
- Auto-increment and Auto-decrement addressing modes to optimize program loops

The MultiTrace solution is a new addition to the ARM development tools that is designed to collect information from ARM core-based SoCs containing an Embedded Trace Macro cell™ (ETM) non-intrusively and without stopping the CPU, by monitoring instructions and data busses at full core speeds.

It provides seven modes of operation:

- User Mode.
- Privileged Modes: Used to service interrupts or exceptions.
- Fast interrupt mode (FIQ Mode)-used for data transfer.

- Interrupt Mode (IRQ Mode).
- Supervisor Mode-protected mode for operating system.
- Abort Mode.
- System Mode.
- Undefined Mode.

All these modes have different sets of registers. The registers are divided in groups to reduce copying time of current status by using some registers directly. The following figure shows the register model for each operating mode.

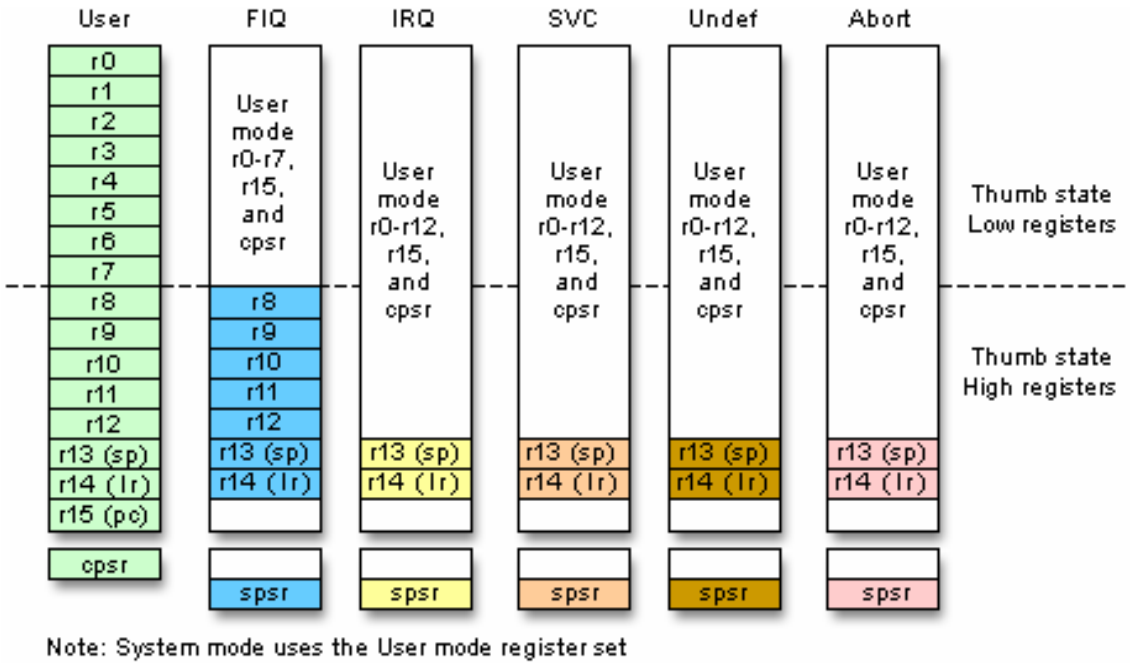

Fig A.1: Register mapping model for different operating modes of ARM processor

The CPSR (Current Program Status Register) contains Interrupt control bits, status of ALU flags, current mode is provided for each mode. The CPSR is copied to SPSR whenever the mode is changed.

Only user mode doesn't contain SPSR (Saved Program Status Register) by default user mode will be enabled and no need to store other modes status.

Thumb code is the concept used for reducing the code density for non complex applications. It provides some 16 bit instruction set and the complex instructions like interrupt handling and branching are not provided. The ARM-thumb interworking module provides instruction to change the states between ARM/thumb for any interrupt handling.

ARM developer suite (ADS):

The ARM controller can only be run with its own instruction set. To convert the high level applications into ARM assembly the ARM developers provided different tools, ADS is the tool used for developing ATS applications.

ADS enable to write and debug applications for the ARM family of RISC processors. It also provides facilities to develop, build, and debug C, C++, and ARM assembly language programs. The linker can link ARM code and Thumb code, and automatically generates interworking veneers to switch processor state when required. The linker also automatically generates Long Branch veneers, where required, to extend the range of branch instructions.

The linker supports command-line options that enable user to specify separate locations for code and data within the system memory map. Alternatively, user can use scatter-load description files to specify the memory locations, at both load and execution time, of individual code and data sections in your output image. This enables user to create complex images spanning multiple memories. As this mobile chip has different types of memories and deals with applications that are switching dynamically between the memories, the ADS best suites for ATS.

When armlink is used to construct an executable image, it:

- Resolves symbolic references between the input object files
- Extracts object modules from libraries to satisfy symbolic references
- Sorts input sections according to their attributes and names, and merges similarly attributed and named sections into contiguous chunks
- Eliminates duplicate copies of debug sections
- Organizes object fragments into memory regions according to the grouping and placement information provided
- Relocates relocatable values and generates an executable image.

REFERENCES

Web resources:

1. http://www.arm.com/pdfs/DDI0210C_7tdmi_r4p1_trm.pdf (ARM7TDMI technical reference manual)
2. http://www.arm.com/pdfs/DDI0198D_926_TRM.pdf (ARM9 reference)
3. http://www.analog.com/UploadedFiles/Data_Sheets/ADSP-BF531_BF532.pdf
4. http://www.analog.com/UploadedFiles/Associated_Docs/68928504ADSP_BF54x_Blackfin_Processor_Hardware_Reference.pdf
5. http://www.analog.com/UploadedFiles/Application_Notes/4280679728866EE068v09.pdf
6. http://www.analog.com/UploadedFiles/Associated_Docs/23439574Blackfin_PRM_1.1.pdf (blackfin processor programming reference)
7. DUI0064D_ADS1_2_GettingStarted.pdf Reference for Arm Developer Suite http://www.arm.com/documentation/Software_Development_Tools

Internal documents:

8. *"AD6522-Hercules-UserManual version 2.0 July 04 2006"* reference document for MSP430 chipset.
9. *"Lemans Specification revision 1.0 September 18 2006"* reference document for MSP500 chipset.
10. *"Anvil 2 CPLD specification version 3.0 December 15 2006"* reference document for mobile chip evaluation boards.
11. *"StratosC User's Guide release January* 24 2005*"* technical reference document for ABB processor.
12. *"ATS user guide"* reference document for ATS architecture and its implementation.
13. Forge User Manual August 12, 2004

Others:

14. *"embedded systems design" official* publication of the embedded systems conference
15. Steve heath. Embedded Systems Design. New Delhi: Published by Elsevier