Architecture design of a SoC

How to build a SoC like iPhone

Tell you the secret of iPhone and its pros and cons

Teach you how to build your own SoC solution

Kenny Liu

Preface

I, Kenny, is the chief system architect in a SoC LLC for many years. When I am in that company, I am in charge of design the whole system including chip, hardware and software. I have finished many successful SoC architecture designs. In the working days, I found that almost nobody knew the whole system even in a SoC company. Every member and every department only focused on only a part of the whole system. When we are facing a problem, we always have to think it in a system point, it's the value of system architect.

In order to "make the world more plat", I want to teach more people and more companies how to design a SoC.

We use the following measures to write this document:

- Data collection, such as training document, specification, datasheet, articles, thesis.
- Solution crack, analyze the target solution's architecture, performance, technologies, etc.
- Prototype building, such as implement a test system, do some demo, test the performance, etc.
- Report writing. Including background knowledge, information source, system architecture, key technology, test report, sample code, etc.

I hope you can get the following benefits from this document:

- Improve technical capacity. Thus they can do what they don't know how to do it before.
- Lower technical risk. From our report, they will get a technical prototype, thus there will be low technical risk for them because we have verified it for them.
- Shorten developing duration. From our report, they will get a system design, thus they can just use it as a reference design and give some customization.
- Get more confidence. Once they know how their rivals developed a system, they will become more confident with their own system.
- Low the developing cost. Wrong direction, failed project will cost a lot of money. Using a verified prototype, it will save a lot of money.

We have paid so much effort to write this document, but now, you can get it via Internet. I hope it will be helpful for you.

Contact: MSN: liujianjun88@hotmail.com . You can get latest version from http://www.itconsult2000.com/product/SoC-iPhone.pdf .

Content

Tables and figures

1 What's iPhone

On January 9, 2007, Apple CEO Steve Jobs took the stage at the Macworld Expo in San Francisco to pre-announce a new mobile phone, the Apple iPhone. He was characteristically enthusiastic about its prospects. "We're going to make some history together," he said. "Today Apple is going to reinvent the phone." (Jobs, 2007)

Even though it had not yet been released, the iPhone quickly became one of the most discussed new technology products, outstripping the coverage of other very prominent mobile phone products that have been shipping for years. A search for the term "iPhone" on Google returns about 53 million hits. For comparison, "Razr" (Motorola's iconic slim phone) returns 23 million hits, and "Treo" (Palm's smartphone) returns about 30 million.

1.1 Technology of iPhone

IPhone used the following technologies.

1.1.1 Hardware features

The iPhone is a "candybar" style device (ie, without flip cover or slider). The front of the device is dominated by a touch-sensitive 320x480 screen, and it does not have a physical keypad. It weighs 135 grams: heavy for a simple "feature" phone (which tend to range from 80g to 120g), but light for a "smart" phone (which range from 110g to more than 200g). Hardware features include a quad-band GSM/Edge cellular radio (without 3G), WiFi (802.11 b and g), Bluetooth 2.0, and a two megapixel camera.

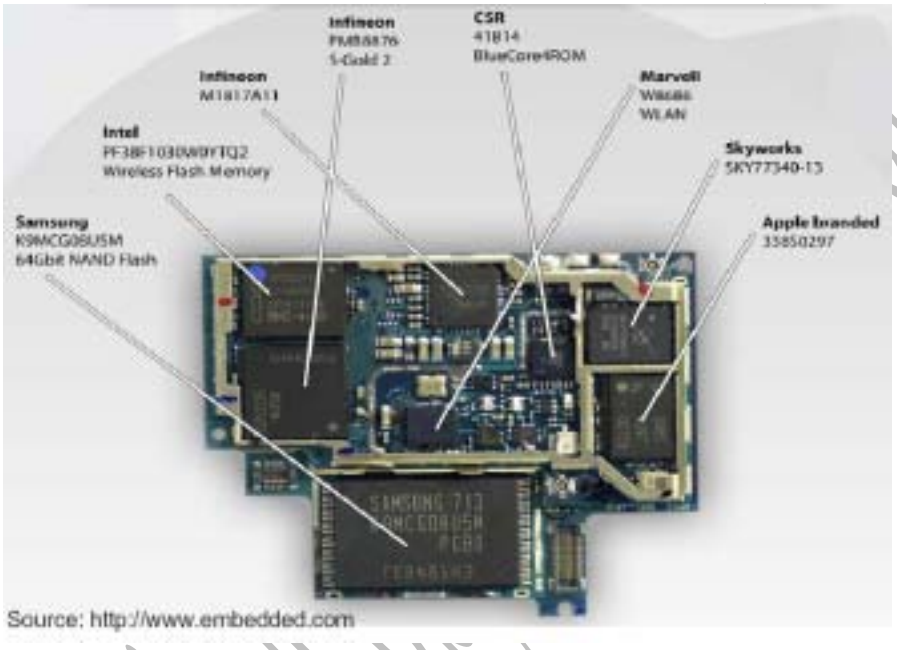
Screen size	3.5 inches
Screen resolution	320 by 480 at 160 ppi
Input method	Multi-touch
Operating system	OS X
Storage	4GB or 8GB
GSM	Quad-band (MHz: 850, 900, 1800, 1900)
Wireless data	Wi-Fi (802.11b/g) + EDGE + Bluetooth 2.0
Camera	2.0 mega pixels
Battery	Talk / Video / Browsing - Up to 5 hours Audio playback - Up to 16 hours
Dimensions	4.5 x 2.4 x 0.48 inches / 115 x 61 x 11.6mm
Weight	4.8 ounces / 135 grams

Specification of iPhone

The iPhone's hardware features and configuration are comparable to those of other "smartphones" already on the market. For example, the HTC 3600 has a form factor and weight similar to the iPhone. The HTC device lacks 802.11g and has a lowerresolution screen, but has a UMTS (3G) radio, a memory card slot, and two built-in cameras.

1.1.2 PCB layout

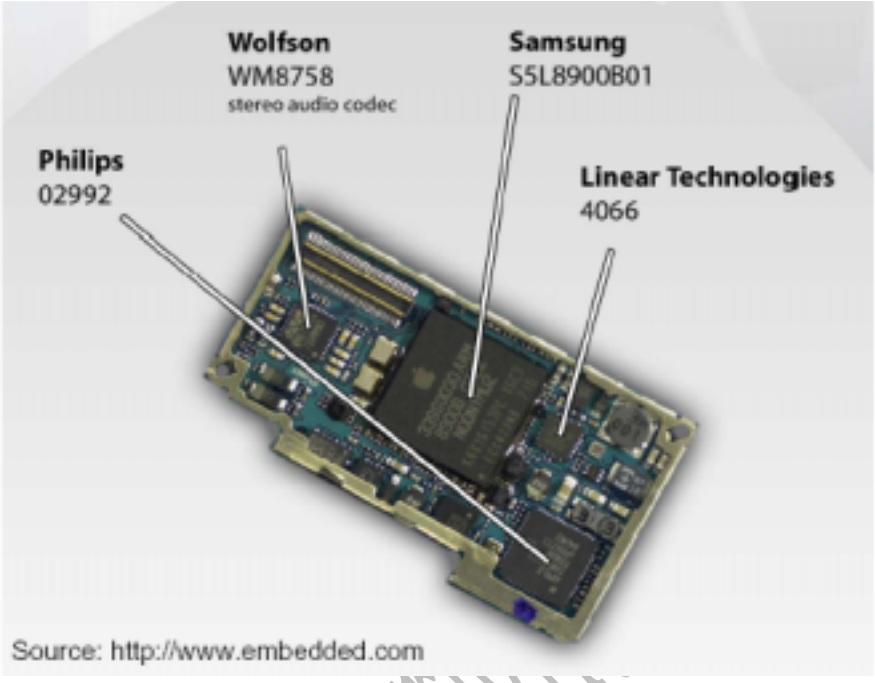
This is PCB layout of iPhone.



Up side of iPhone PCB

Maker (Brand)	Function	Mark	Dimension(mm)	Notes
Infineon	Baseband IC	337S3235 G0704	10x10x1	BGA
Intel	Flash Memory (32MB+16MB)	1030W0YTQ2	8x10x1	BGA
Infineon	Power Management IC	338S0289 G0716	6.5x6.5x0.8	QFJ
Skyworks	Power Amp Module (GSM/EDGE)	SKY77340-13	6x8x1	SOJ
Epcos (Infineon)	RF Transceiver+FEM	338S0297 G0717	6x9x1	BGA
Hosonic	Occilator (26MHz) for B-T	26.000 723	2x2.5x0.55	Can type
N/A	Band Pass Filter	N/A	2x2.5x1	LTCC
N/A	N/A	2150717	2x2x0.8	QFJ
N/A	Power Amp Module (2.4GHz)	B8 BA6N	2x2x0.8	QFJ

Marvell	W-LAN Chip	W8686B13	4x5x0.5	WL-CSP
CSR	Bluetooth Chip	41B14	4x4.5x0.5	WL-CSP
Hosonic	Occilator(40MHz) for W-LAN	40.000 723	2x2.5x0.55	Can type



Down side of iPhone PCB

Maker (Brand)	Function	Mark	Dimension(mm)	Notes
APPLE (Samsung)	Application Processor	339S0030 ARM	14x14x1.0	BGA
Linear Technology	USB Power Manager & Battery Charger	4066 N3016	4x4x0.75	QFJ
APPLE (Broadcom)	I/O Controller	338S0459 P2TFY	8x8x0.8	QFJ
Samsung	Nand Flash Memory(4GB)	K9HBG08U1 M	18.5x12x1.0	SOP
National Semiconductor	Display Interface Serializer	M73RF LM25 12SM	4x4x0.8	BGA
ST Microelectronics	3D Accelerator asensor	713 302D	3x5x0.9	MEMS, SON-14, LGA
Hosonic	Occilator (24MHz)For MP3	24.000 873	2x2.5x0.55	Can Sealed type
Hosonic	Occilator (27MHz)	27.000 719	2x2.5x0.55	Can Sealed

				type
Epson-Toyoc om	Occilator (32KHz) for Timer	A710Y	3.2x1.5x0.55	
Wolfson	Stereo audio CODEC	WM8758BG	10x10x0.8	QFJ

1.1.3 Software features

The iPhone incorporates a version of Apple's MacOS X with its Safari web browser. Depending on how the browser is implemented, this is potentially a differentiator for Apple as most mobile browsers can display only a subset of the web content available on PCs. However, the iPhone is not planned to include Java or Flash support, limiting its ability to display some websites. The iPhone also includes Yahoo email support, Google Maps and associated location-based services, iPod music and video playback, a suite of phone apps (address book, threaded SMS, calendar, dialer, visual voicemail [pick messages from a printed list on screen]), photo management software, and some widgets provided by Apple. The visual voicemail product appears to be unique, and of course iTunes is not offered for other companies' devices (other than a few Motorola mobile phones). Yahoo e-mail and Google Maps are available for other mobile devices.

1.1.4 UI features

Apple made a distinctive departure from the industry norm in this area. Unlike other touchscreen smartphones, the device has no stylus, and is designed to be entirely finger-driven. The interface can sense multiple simultaneous finger presses, enabling the user to manipulate the interface via gestures (something that Apple says it is patenting). For example, pinching the fingers together is used to reduce the magnification on a web page. This sort of "multi-touch" interface has been discussed in the past, but Apple is one of the first companies to implement it in a commercial product. It won't be possible to judge the success of Apple's approach until the iPhone ships.

As is typical of many Apple products, the iPhone's interface also includes some minor features uncommon in most competing products and designed to spotlight the company's attention to detail. One is a proximity sensor that turns off the touchscreen when the device is close to the user's face (so incidental contact with the user's ear won't activate functions of the phone). Another is the use of position sensors to rotate the screen's image appropriately when the device is turned sideways.

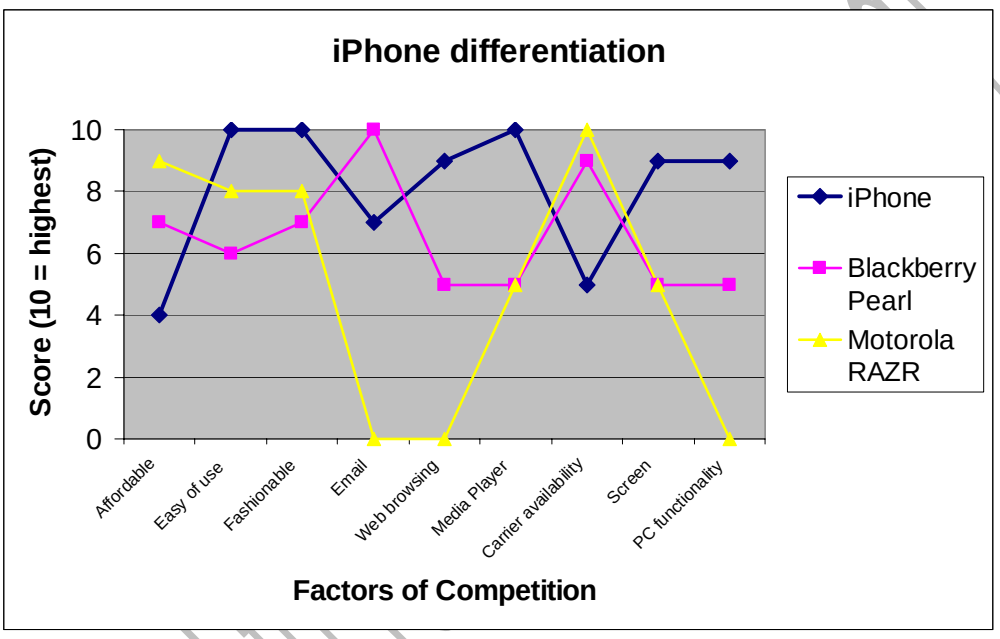
1.2 Strategy

IPhone used many new strategy for its success.

1.2.1 Market strategy

Intended for style-conscious cell phone consumers who would like to browse the Internet and enjoy entertainment to-go, the iPhone is a smart phone that combines a web browser, email, iPod and personal computer into a single, fashionable, easy-to-use device.

The following strategy canvas illustrates how the iPhone is differentiated from the BlackBerry Pearl (a smart phone) and Motorola's RAZR (a popular, stylish consumer phone)



iPhone's Market strategy

There have been over 100 million iPods sold. In 2006, the following amounts of systems were sold:

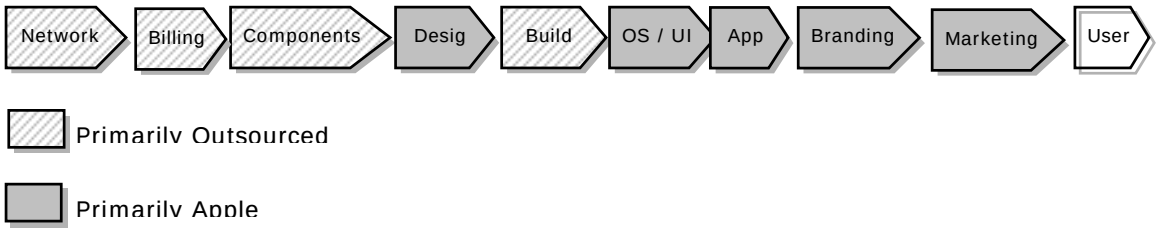
System	Number sold
Game Consoles	26 M
Digital Cameras	94 M
Mp3 Players	135 M with iPod having 80% market share
PCs	209 M
Mobile Phones	957 M

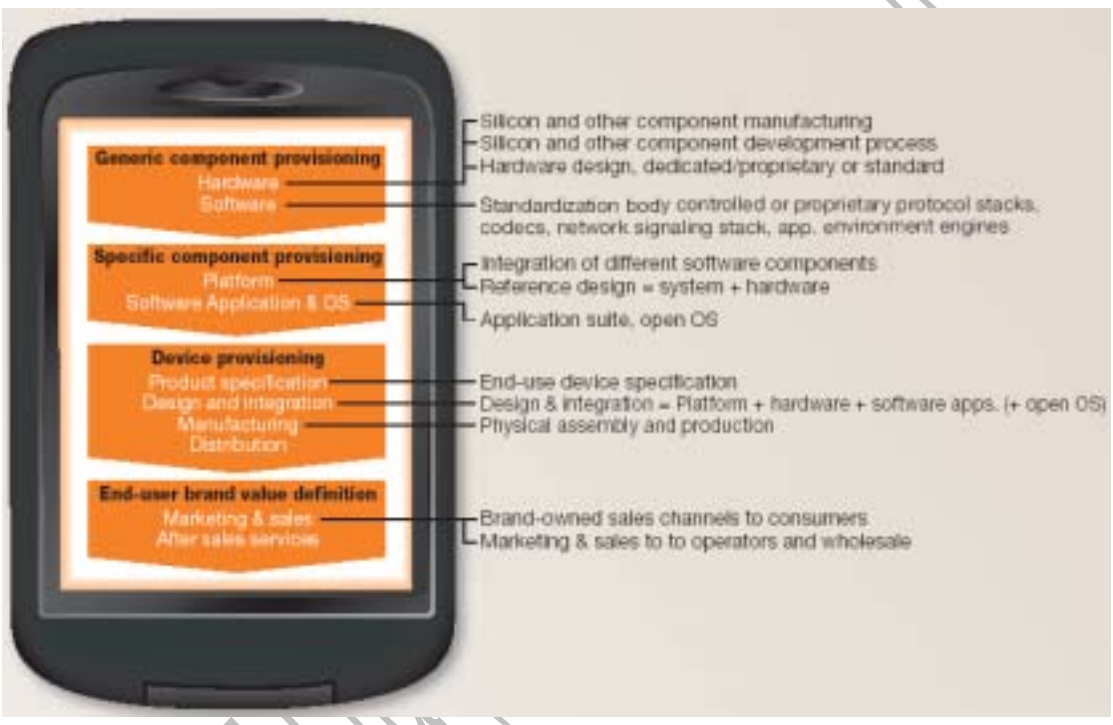
Apple's goal is to achieve 1% market share which is 10 M phones by 2008. They are going after the world market.

1.2.2 Development strategy

The value chain below illustrates the context in which Apple delivers value to the

customer with the iPhone:





Value chain of iPhone

1.2.3 Communities

Make communities as partners.

1.2.3.1 Conceitedsoftware.com

Provice free software application.

1.2.3.2 Modmyifone.com

Provice free software application.

1.2.3.3 Iphone.freecoder.org

Provice free software application.

1.3 Summary

1.3.1 Cost and profit

1.3.1.1 Cost

Quantity: 10 M Sets/Year

Unit: US cent

Name	Producer	Size (mm)	Description	Count	Price	Total
Baseband IC	Infineon	10x10x1		1	1,200	1,200
Flash Memory	Intel	8x10x1	32MB+16MB	1	1,500	1,500
Power Management IC	Infineon	6.5x6.5x0.8		1	300	300
Power Amp Module	Skyworks	6x8x1	GSM/EDGE	1	150	150
RF Transceiver＋FEM	EPCOS(Infineon)	6x9x1	GSM/EDGE	1	500	500
		2x2x0.8	SP4T	1	150	150
Power Amp Module		2x2x0.8	2.4GHz	1	130	130
Application Processor	Apple(Samsung)	14x14x1.0		1	1,600	1,600
USB Power Manager & Battery Charger	Linear Technology	4x4x0.75		1	300	300
I/O	APPLE (Broadcom)	8x8x0.8		1	200	200
Nand Flash Memory	Samsung	18.5x12x1.0	4GB	1	2,800	2,800
Display Interface IC	National Semiconduct	4x4x0.8		1	300	300
Audio Codec	Wolfson	10x10x0.8		1	100	100
Flash Memory	SST	5x6x0.5	8Mb(512Kx16)	1	250	250
Other IC				11	80	880
Others				30	10	300
IC total						10,660

LCD	Sharp, TMD, Epson	3.5inch	TFT(480x320)	1	2,400	2,400
Back light	Balda	3.5inch		1	1,500	1,500
Display						3,900
W-LAN IC	Marvell	4x5x0.5	WL-CSP	1	700	700
Bluetooth IC	CSR	4x4.5x0.5	WL-CSP	1	250	250
RF						1,175
Resistance, etc.						187
Sensor						2,000
Board						820
Total						19,708

Hardware cost of iPhone

Thus the total hardware cost for iPhone (4GB) is US$197.

1.3.1.2 Profit

Pricing:

Model	Price
4gb model	499
8gb model	599

Availability:

Ship Date	Ship Date
United States	June 2007
Europe	December 2007
Asia	2008

1.3.2 Pros and cons

Some notable features available in many other smartphones are missing from the iPhone, including:

• A physical QWERTY keyboard such as that found in the Palm Treo and RIM Blackberry devices. Instead, there is a virtual keyboard on the touchscreen. Apple claims this is an advantage because mobile device keyboards have such small keys that they are hard to use.

• The ability to install third party applications. At release Apple said the iPhone will be a closed device -- only Apple will be able to offer new applications for it.

• The ability to read and edit Microsoft office documents, which is important to business users who often receive them as e-mail attachments.

• A user-replaceable battery. Like the iPod family but unlike most mobile phones, the iPhone's battery cannot be replaced by the user. While this allows Apple to make the device thinner because the battery does not require a separate housing, the user can't carry a backup battery, and when the battery wears out the device

must be returned for service. Palm Computing tried a similar approach with its early smartphones, but gave up due to customer complaints.

2 Architecture design

2.1 User cases analysis

Use UML as modelling tools.

2.2 Requirement definition

What our product definition here. Just some like iPhone.

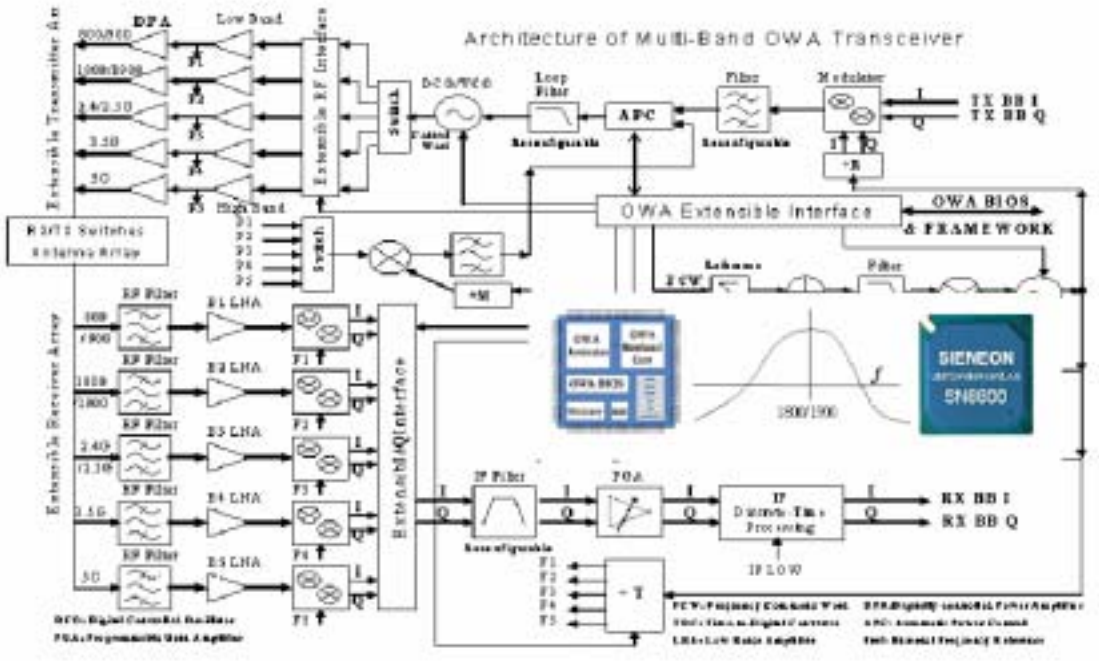
2.3 Comparation of popular architectures

Compare some popular solutions here.

2.3.1 RF transceiver

We focus on new solutions.

2.3.1.1 OWA Multi-band Transceiver



OWA Multi-band Transceiver Multi-band Transceiver

Features of this architecture is:

- Software defined radio: A radio in which the preset operating parameters including inter alia frequency range, modulation type, and/or output power limitations can be re-set or altered by software.
- Reconfigurable radio: Reconfigurable radios are radios whose hardware configuration and software can be changed under software control.
- Cognitive radio: A radio or system that senses and is aware of its operational environment and can dynamically, autonomously, and intelligently adjust its

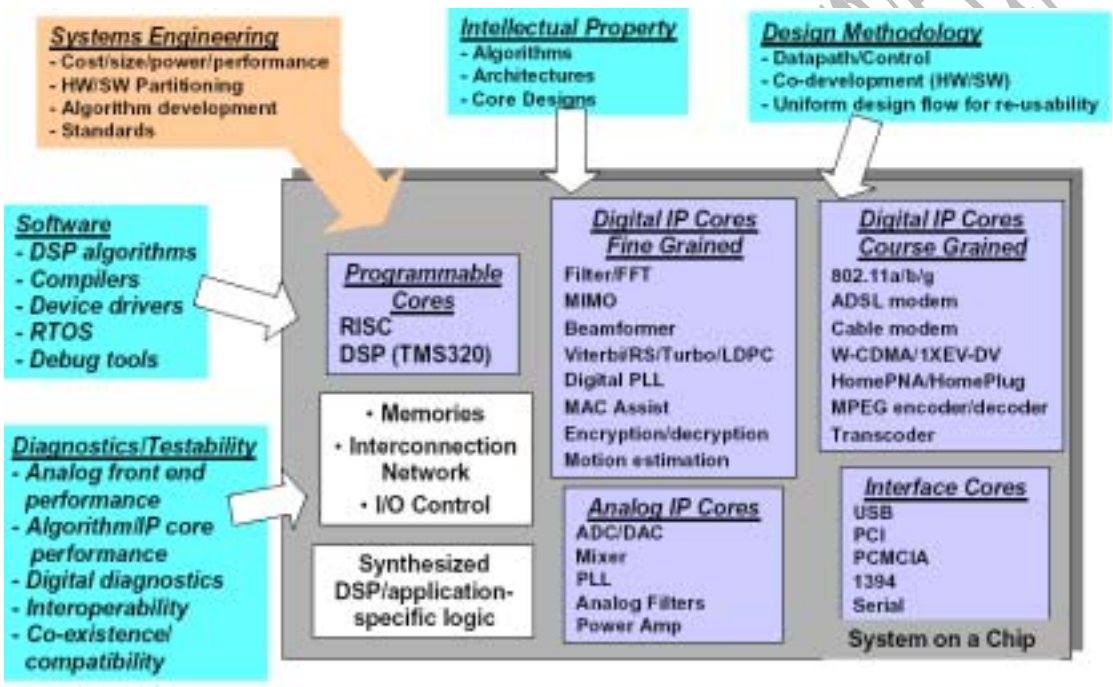
radio operating parameters.

- Open Wireless Architecture (OWA): defining the open interfaces in wireless networks and systems so that users can buy different parts from various vendors. OWA system is upgradeable and extensible. In addition, the system can support various standards through open interface parameters. OWA will converge with open computer architecture and open network architecture.

2.3.2 Baseband chip

We focus on new solutions.

2.3.2.1 Texas Instruments



Chip architecture of Texas Instruments

On-chip functionality is increasing rapidly

- Digital / analog integration
- Software, analog, protocol stacks, system interfaces, applications
- >> 100 Kgates per sq. mm.

Recovering non-recurring engineering/R&D expense

- Requires increased hardware and software re-use

Programmable DSP will drive the broadband communications market coupled with re-usable applicationspecific IP blocks

2.3.3 Desttop frameworks and OS

2.3.3.1 Android

The following diagram shows the major components of the Android operating system. Each section is described in more detail below.

Software architecture of Android

Applications

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

Application Framework

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser

Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data

A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files

A Notification Manager that enables all applications to display custom alerts in

the status bar
An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack

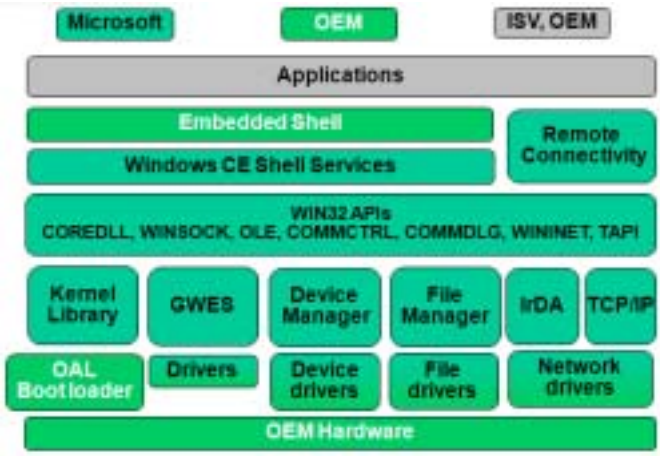
Libraries
Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:
System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
Media Libraries - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view
SGL - the underlying 2D graphics engine
3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
FreeType - bitmap and vector font rendering
SQLite - a powerful and lightweight relational database engine available to all applications

Android Runtime
Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.
Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.
The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Linux Kernel
Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stac
The following is the Software architecture of Windows mobile:

2.3.3.2 Windows mobile

Microsoft provide Windows mobile as their latest version.



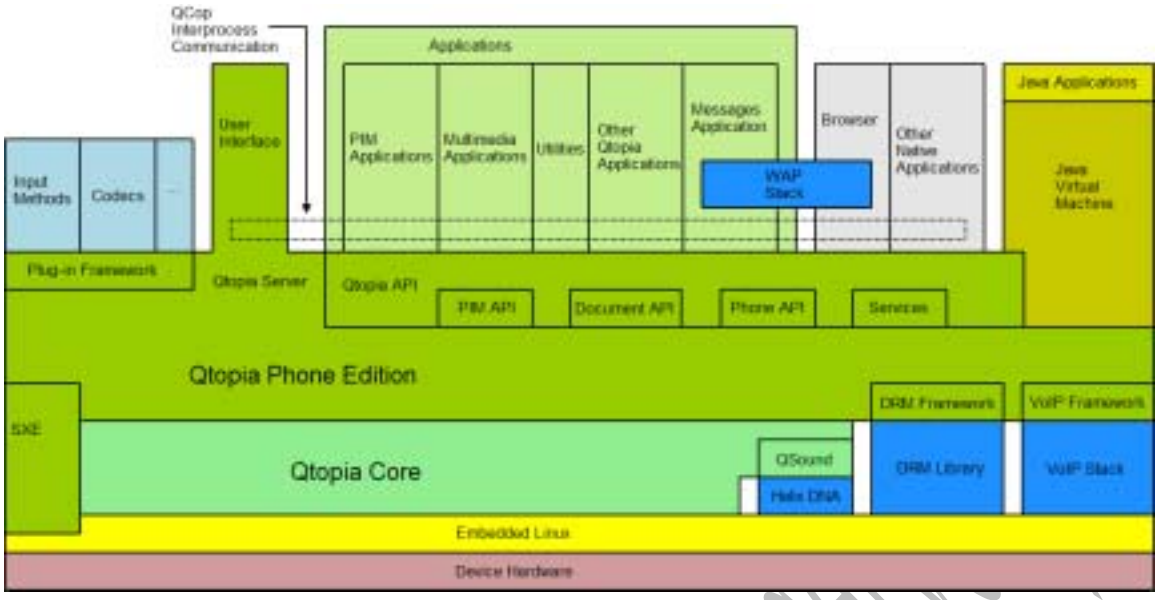
Software architecture of Windows mobile

GWES module:
Graphics, Windows, and Event manager
Graphic output(display and print)
User input: keyboard, stylus, mouse, etc.
Windows management: message routing, etc.
GWES is the most componentized WinCE module
GWES exports only a subset of the Win32 API functions

Kernel, GWES, Filesys, and Communications:
Each module is divided into components
Build an OS image that fits your needs
Windows CE configurations: MINKERN, MININPUT, MINCOMM, MAXALL, IESAMPLE. . .
Components can be added, deleted, or replaced
Execute In Place (XIP) from ROM

2.3.3.3 Qtopia

The following is the Software architecture of Qtopia:



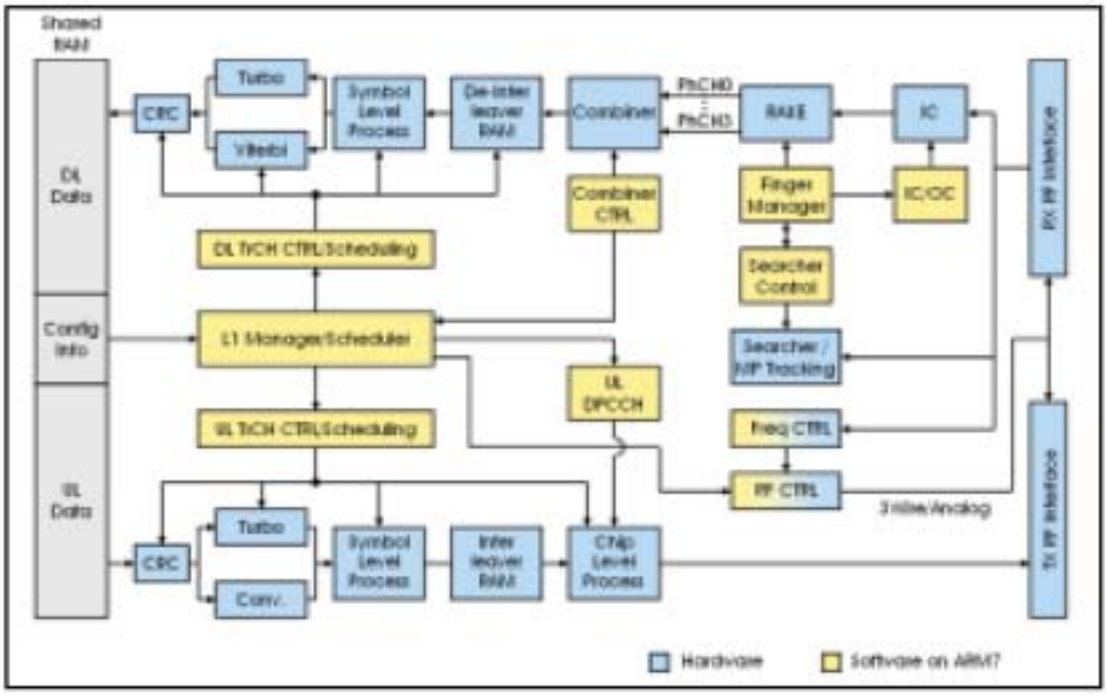
Software architecture of Qtopia

2.4 Our architecture strategy

What's our choice?

2.4.1 Splitting telecommunication features

Mapping it into hardware and software.
Hardware is expensive but low power comsumption.



Hardware / Software Split

The first sets of trade-offs that need to be considered relate to the classic DSP vs

HW choice. When the design for this SPINNERchip was begun, the WCDMA standards were matured, allowing us to consider a HW approach as a viable, lower power consumption approach to the design. One cannot dispute that a dedicated HW approach will always have lower power consumption and out perform a similar solution that is DSP based. In our specific implementation we have achieved a 25% power consumption savings compared to an equivalent implementation done with multiple DSP cores. Of course, one would argue that you give up flexibility by going down the dedicated HW path. This will be addressed further in the article.

The second issue addressed was scalability of the design to support future revisions of the specifications, mainly surrounding higher data rate implementations. The current design is fixed to support up to 384kb/s in both the uplink and downlink. Future versions will require up to 14.4Mb/s in support of HSDPA. As data rates increase, DSP based implementations become increasing complex, as scaling requires adding parallel processors, with inherent timing issues to contend with. HW architectures are inherently parallel, enabling a ramp to higher data rates with minimal impact on the overall HW structure. In fact, to support a 2Mb/s class modem with the SPINNER device would require expanding a few of the memory blocks and some minor SW changes, but no major HW restructuring.

There were two aspects to flexibility that needed to be considered. Firstly, the concern of casting processor blocks into hard coded logic; vs. a more flexible SW based DSP approach. In our implementation we have utilized HW blocks to handle of the constant tasks, such as Convolutional coding, Viterbi decoding and Turbo encoding/decoding, etc and retain the flexibility required for these blocks by writing a SW framework around the blocks for rapid programmable control and reconfiguration. An ARM7TDMI-S® is at the center of this control logic. The second aspect of flexibility that was required was the ability to connect our device to multiple 2G/2.5G host processors, and most importantly be able to do so without requiring HW changes on either our SPINNER device or on the various host devices we would connect to. This second level of flexibility was achieved by utilizing an SRAM memory bus as the interface between the two processors.

Based on the architecture described above, it is clear that our ARM7™ solution retains a lot of the flexibility of a DSP solution. Furthermore, with the timing and implementation of the data path fixed and deterministic, we are able to easily extend the control software without affecting the overall real-time design of the system. Furthermore, small change to optimize the SW implementation does not influence the overall design requiring an extended re-testing cycle. These features of our system design are amongst the most important contributing factors to ensuring that the system can meet all the performance requirements of a 3G system. In general, one can design a complex DSP system to support a 384kbps

connection. The difficult part is ensuring that you can still do this while the network requests a complex configuration of measurements etc. from the UE. In this case, the hardware data path protects the throughput that can be supported by the overall solution as opposed to sacrificing some performance in order to meet network requirements.

With regard to achieving an optimum physical implementation size for the digital baseband co-processor and required analog functions, we compared the size implications of parallel DSP cores, with their associated local memory vs. an approach of dedicated, but configurable HW. A single RISC based ARM solution, with its simplified instructions set and compact code size yielded a more compact and scalable solution than a parallel DSP core implementation with its combined larger instruction set, code size and local memory requirement. We concluded we could achieve a higher performance solution with the HW based approach, in a physical die size that for both scalability reasons as well as power consumption, would be better than a multi-core DSP implementation. In addition, we included the required analog functionality (transmit/ receive DAC/ADC, GP DACs). This further improved the cost and size impact by alleviating the need for external discrete analog components. We were able to further optimize the SPINNER solution by writing a customized RTOS for all scheduling functions. This provided significant savings on both code size (2K vs 16K) and a drop from 55 to 37 MIPs. The SPINNER architecture was conceptualized to share certain tasks with the host processor chip (protocol stack engine, AMR Vocoding, etc) , but at the same time minimize the traffic between the two devices so as to reserve the maximum bandwidth for data traffic. As mentioned prior, we chose to put algorithms which are fixed into HW, and be able to control or parameterize these HW blocks in a rapid fashion, some as often as every slot (667us). This hybrid HW/SW split gives us the maximum benefit of HW processing power and low power consumption, while retaining the needed flexibility for various scenarios that would need to be covered by the UMTS specification.

The SW control of the hardware data pipe is best handled by an MCU due to its relatively short pipeline and low interrupt latency; an ARM7TDMI® in our case. This is imperative for time critical support of items such as power control, which requires a less than 45us response. Furthermore, an MCU based control solution is best suited for code with many branch statements, such as this implementation vs. a code set with high computational requirements, where a DSP would be more appropriate. So the choice of putting the low frequency, low computational requirements onto the ARM, such as TFCI decoding, power control, etc. provides us with a performance advantage and retains the flexibility required to optimize these functions during field testing. The hybrid split of HW/SW also facilitates the use of a serial port for debugging, which facilitates complete visibility of both hardware and software registers.

2.4.2 Splitting multimedia features

Mapping it into hardware and software

3 Chip design

3.1 Telecommunication subsystem

Data flow

3.2 Multimedia subsystem

Data flow

3.3 Application subsystem

ARM core

3.4 Bandwidth analysis

Data flow

3.4.1 Streamed Video

From RF to video

3.4.2 VoIP

From RF to voice

3.4.3 Video recording

From camera to LCD and FLASH

3.5 Power consumption analysis

Power consumption

3.5.1 Idle state

Chip with paging

3.5.2 Multimedia state

Playing video

3.5.3 Talking state

Making a call

4 PCB design

4.1 Goal

4.1.1 Low cost

Less components. Less layers PCB.

4.1.2 Easy to manufacture

Simple tools and steps.

4.1.3 Easy to debug

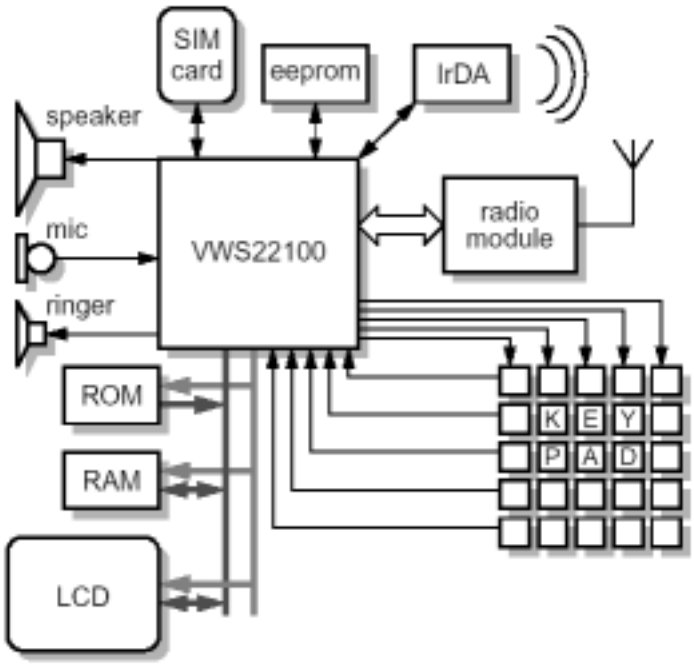
Powerful debug ports.

4.2 Pinout definition

Useful and reasonable

4.3 Layout design

GSM systems usually combine a digital signal processor core (which handles all the real-time data traffic) with a general-purpose controller (which handles the protocols and user interface).



Sample PCB of GSM/GPRS

4.3.1 RF block

Link RF

4.3.2 Baseband block

Link baseband

4.3.3 Audio block

Link audio

4.3.4 Peripherals block

Link peripherals

5 Software design

5.1 Telecommunication subsystem

Protocol stack

5.1.1 Data plane

Data, such as layer 1.

5.1.1.1 GSM/GPRS/EDGE

2G

5.1.1.2 GSM/GPRS/EDGE/WCDMA/HSDPA

3G

5.1.1.3 GSM/GPRS/EDGE/WCDMA/HSDPA/LTE

4G

5.1.2 Signal plane

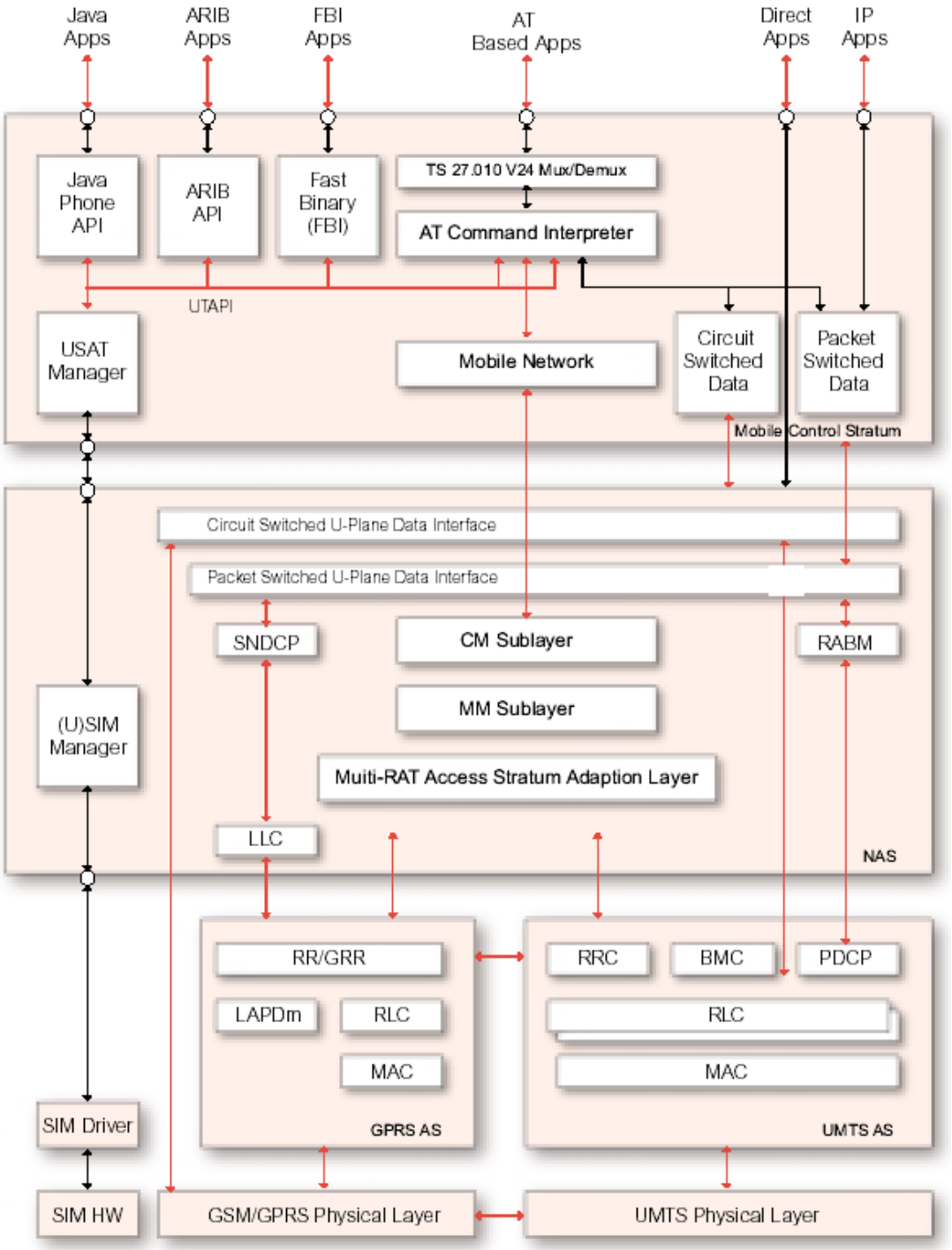
Layer 2 and layer 3

5.1.2.1 GSM/GPRS/EDGE

2G

5.1.2.2 GSM/GPRS/EDGE/WCDMA/HSDPA

UbiNetics' flagship IP offering is the PS300, a complete 3G protocol stack supporting multi-mode operation for GSM, GPRS and W-CDMA networks.



Architecture for 3G protocol stack

EDGE

HSDPA

5.1.2.3 GSM/GPRS/EDGE/WCDMA/HSDPA/LTE

4G

5.1.3 Modem system software

OS, AT, etc

5.1.4 Footprint analysis

Memory size

5.2 Multimedia subsystem

Audio, video

5.2.1 Accelerator drivers

Video Accelerator

5.2.2 Equalizer drivers

Audio effects

5.2.3 Camera drivers

Camera input

5.3 Application subsystem

Application OS

5.3.1 Boot code in ROM

RedBoot boots FLASH

RedBoot is supposed to be a next generation bootloader from Red Hat, replacing CygMon and GDB stubs with a firmware supporting a very wide range of hardware. the RedBoot package is fairly well documented, including a RedBoot User's Guide that provides actual examples of its use on more than a dozen different systems. RedBoot's web site is located at http://sources.redhat.com/redboot/.

RedBoot functions: parse FLASH file system and then loading image.
Size of RedBoot is about 50 KB?

5.3.2 Linux OS in FLASH

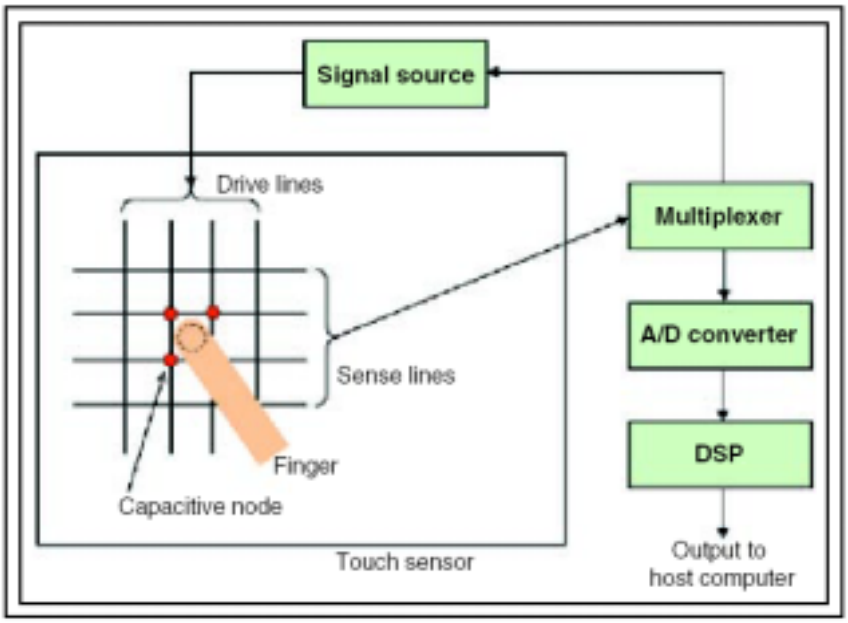
Linux image and basic file system.

5.3.3 Input drivers

LCD and touch panel, sensors

5.3.3.1 Touch panel

Until the product is actually available, the best source of information on the iPhone's touch screen is Apple's patent applications. The most relevant one is #2006-0097991, dated May 11, 2006 and entitled "Multipoint Touch Screen."
The patent describes two different implementations of projected-capacitive touch technology. The first, which Apple calls "self capacitance," is a simple passive array of 24 × 36 sensing electrodes in a single plane. This is commonly known as a "matrix" touch panel, and it is commonly used in applications such as industrial control panels, membrane keyboards, and other situations where a limited number of well-defined areas on a surface need to be made touch sensitive. Since it is basically a low-resolution architecture, it is not regularly applied to displays. The second implementation of projected-capacitive touch technology described in Apple's patent application is a more traditional structure consisting of two sets of parallel rows of transparent conductors, one on each side of a substrate, perpendicular to each other. Apple calls this implementation "mutual capacitance." From a sensor point of view, it is essentially the same concept as Touch International's ExtremeTouch™ product.



A simplified view of Apple iPhone's projected-capacitive touch screen.

There are six fundamental touch-vocabulary elements (gestures) in the iPhone user interface (UI):

- Single tap to select or activate something.
- Double tap to change the display format.

- Drag and drop to move something.
- A stroke (“swipe” or “flick”) up/down/ left/right to scroll.
- “Pinching” two fingers together to shrink something.
- “Spreading” (un-pinching) two fingers apart to enlarge something.

5.3.3.2 Sensors

Sensors

5.3.4 Desktop frameworks

Desttop layout, application management, etc

5.4 Boot sequence

We make the world more plat

6 Service design

We suppose 3rd part will provide these kinds of application.

6.1 Phone services

6.1.1 Calls

Make calls

6.1.2 SMS/EMS/MMS

Send, receive SMS.

6.1.3 Phone books

Contacts

6.1.4 Fax

Send, receive Fax.

6.2 Connection services

Connect to each other.

6.2.1 WLAN

For free internet access.

6.2.2 Bluetooth

For personal access.

6.3 Hot internet services

Embedded hot internet services.

6.3.1 Browser

Opera, etc

6.3.2 e-Mail

Pop3, SMTP, etc.

6.3.3 YouTube

Free video

6.3.4 GoogleMap

GPS may be more useful.

6.3.5 Yahoo finance

Stocks

6.3.6 Instant messager

ICQ, MSN, etc

6.3.7 SIP

Skype is also a choice.

6.4 Internet multimedia

iTune

6.4.1 Music

Download mp3 for iStore.

6.4.2 Video

Download video for iStore.

6.4.3 Radio

Listen radio online

6.4.4 TV

Watch TV online

6.5 Games

Embedded some games

6.5.1 Simulators

NES

6.6 PC services

6.6.1 Programming language

Perl, php, Python, Ruby, Tcl

6.6.1.1 Perl

Script language

6.6.1.2 Php

Script language

6.6.2 Command interface

SSH, VPN

6.6.3 Office

PDF

6.7 SIM service

Cope phone bool

7 Tools design

7.1 Developing tools

7.1.1 Compilation tools

Gcc, gdb

7.1.2 Debug tools

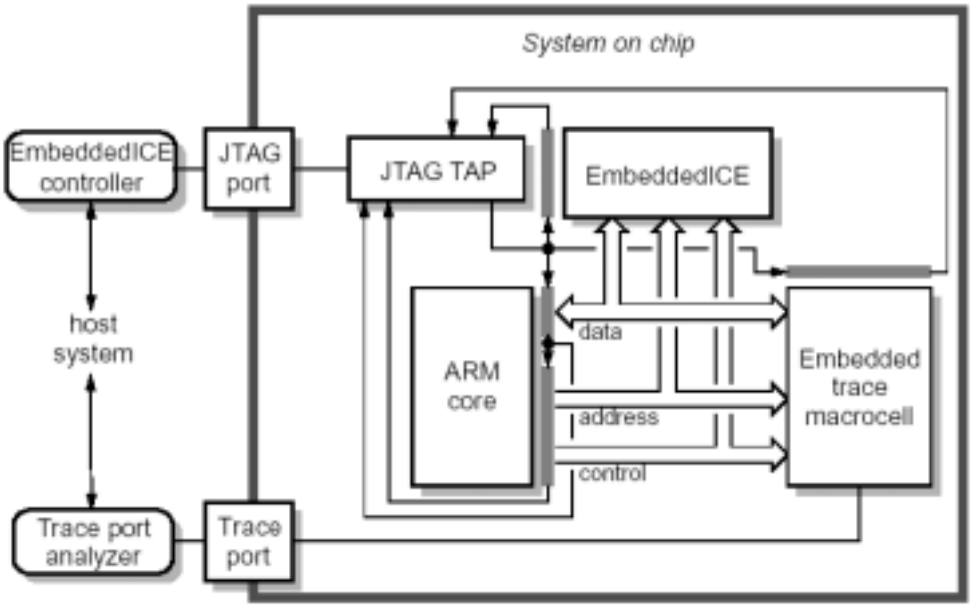
7.1.2.1 What's JTAG?

JTAG: the JTAG port is an IEEE standard interface to on-chip scan register that are used to support the production testing of printed circuit boards. The scan registers allow signals to be sent out of one chip through its pins, across the PCB under test, into a second chip, an scanned out from there. All the PCB solder joints, tracks an vias are tested in this way.
JTAG stands for Joint Test Action Group, the committee set up in the 1980s to set up this standard.
Since the chip has this interface for PCB testing, it is convenient to use it for other serial access purposes such as the ARM debug scan chains - it is efficient to support debug this way as it does not require any more pins.

7.1.2.2 Debug tools

Burying a processor deep inside an SoC makes observing its activity much harder. The facilities required for good access to processor activity or software debugging include:

- breakpoint/watchpoint hardware: this will stop processor execution whenever a particular instruction or data location is accessed
- mechanisms to examine processor and memory state
- a 'trace' facility to see a record of processor address and data bus activity around some triggering event ARM systems provide the first facility through the EmbeddedICE system, the second through various scan chain mechanisms, and the third through the Embedded Trace macrocell.



Debug tool of ARM

7.1.2.3 On-chip debug

Breakpoints - stopping execution for debug pruposes

A typical breakpoint debug sequence might be where the user wants to examine the values in the processor's registers at a particular point in the program execution:

- the debug hardware is initialized through the JTAG serial scan chains to set the breakpoint
- the application runs in real time until the hardware detects the breakpoint trigger
- the processor then stops and enters debug mode
- the debug hardware then presents a 'STM all registers' instruction by scanning it in through the JTAG port ono the processor's data bus
- the processor's clock is then cycled via scan chain control, causing it to load and execute the STM up to the point where it issues the first register value on its data output
- the first register value is captured and shifted out to the debugger via the scan chain
- the processor's clock is cycled again to yield the second register value, which is captured and scanned out, and so on.

Trace information

If an on-chip core is running at high speed (e.g. over 100 MHz) it would require too many pins to get all the address and data information off chip on every cycle. Instead, on-chip compression hardware is used to reduce the data rate through the pins.

Instruction addresses usually run sequentially, so consecutive addresses can be assumed with no data required.

The off-chip debugger has access to the code that is running, so it knows when the processor is executing a branch. It only needs one bit of information - whether or not the branch was taken- to be able to track the program address.

So, the only time a full program address must be sent from on chip to off chip is when the processor executes a subroutine return, table jump, or similar computed branch. These are relatively infrequent in typical code.

7.1.3 Simulation tools

simulator

7.2 Analysis tools

Coverage, bandwidth

7.3 Manufacture tools

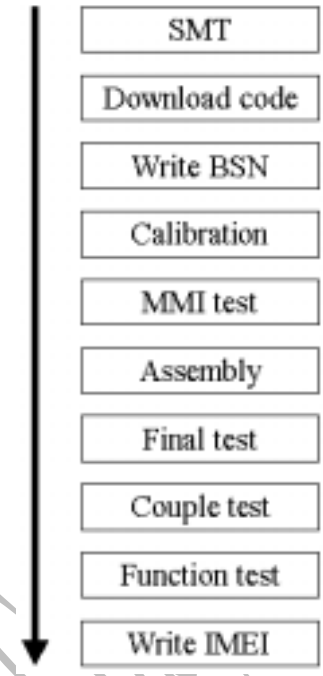
Manufacture

8 Manufacture design

This is the last step.

8.1 Manufacture flow

Sample flow



Manufacture flow of MTK

8.2 Software downloading

Download software to FLASH, such as basic frameworks.

8.3 Calibration

Calibrate RF, battery and audio variances.

8.4 Factory Settings

Set IMEI, defaulting settings, etc.

8.5 Resource downloading

Download multimedia resource into file system of the FLASH.
Download applications into file system of the FLASH.

9 More than iPhone

9.1 Goal

Multimedia + PC experience + Online Game + Internet

9.2 Less power consumption

Our solution and key technology

9.3 GPS

Our solution and key technology

9.4 Online game

Our solution and key technology

10 ROI analysis

Return On Investment

10.1 Cost analysis

10.1.1 Chip cost

Die cost

10.1.2 PCB cost

BOM

10.1.3 Software cost

License fee

10.1.4 Developing cost

Schedule and HR

10.1.5 Summary

Total cost

10.2 Return analysis

Market share

10.3 Conclusion

Have a try

11 More reference

11.1 Related media

Where you can find this document?

Organization:
www.mpsoc-forum.org
www.design-reuse.com
www.edatechforum.com

Journals:
www.edn.com
www.cellphone.eetchina.com

11.2 Related documents

Since I cannot write all details just in one document, please refer to related documents for more details:

11.2.1 Chip design

How to design the telecommunication subsystem?

11.2.2 Software design

How to design a modem?
How to design layer 1 for a protocol stack?

11.2.3 Tools design

How to measure the footprint?

11.3 Related prototype packages

Deliver package:

- Design document
- C source code of prototype
- Test tools
- test report

11.3.1 Multi-touch package

Multi-touch is …

11.3.2 Fax package

Fax is …

11.4 More services

If you meet any problem in implementation phase, please contact with author for help. Author can provide more service for you based on your needs, such as detailed bug shooting, give training, act as project manager, join technical review etc.

11.5 Contact with author

I started to write this document since 2008/02/25 when I got an iPhone with me. But before the writing, I have studied iPhone for several months and have collected many data about iPhone already.