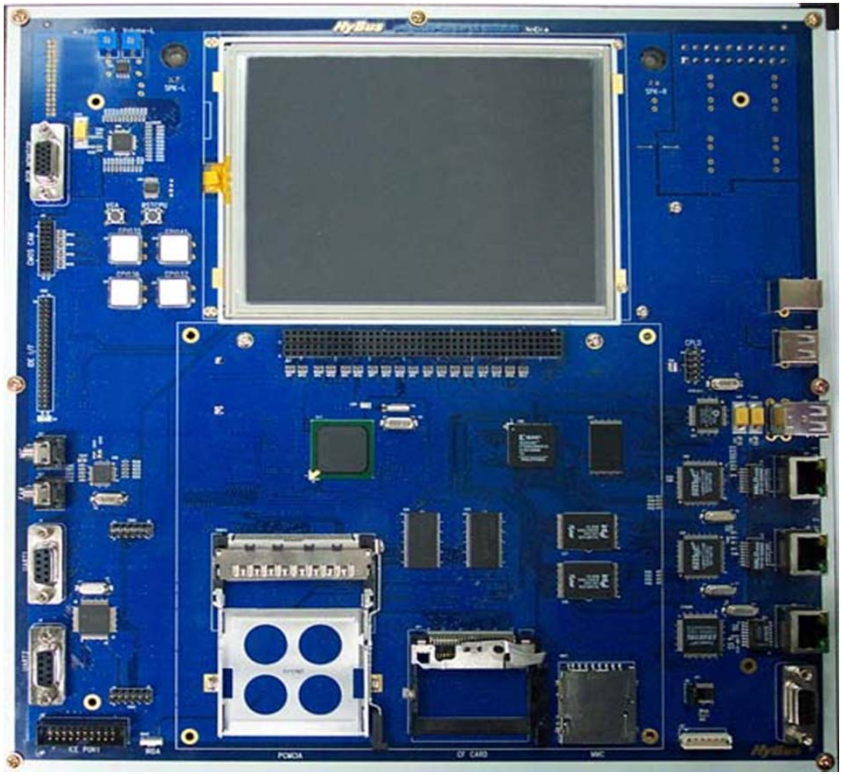
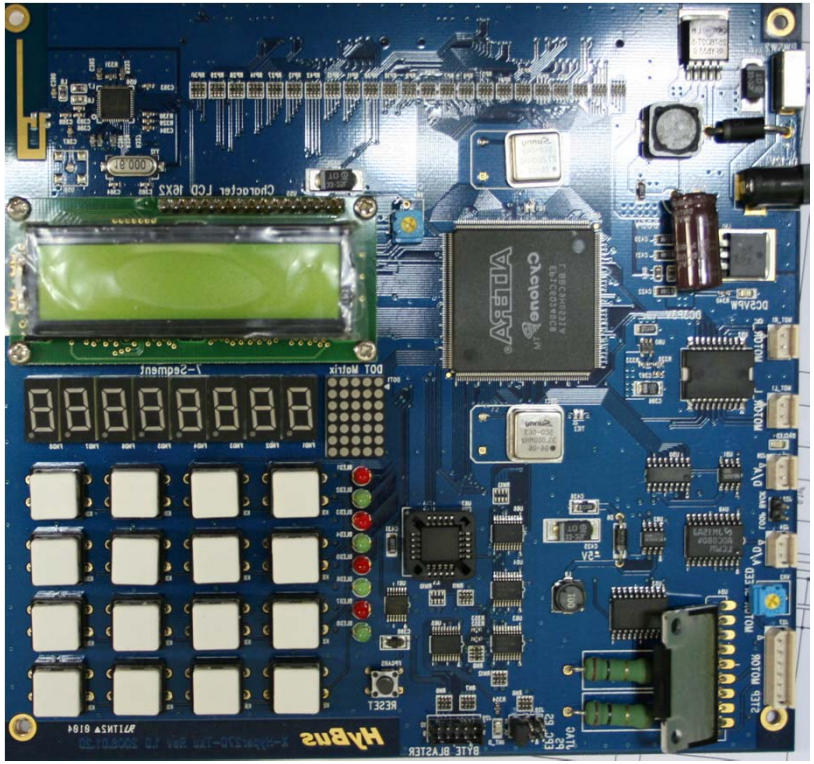
Embedded Systems

Prof. Myung-Eui Lee (A-405)
melee@kut.ac.kr
한국기술교육대학교
Korea University of Technology and Education

Target System

- **Hybus PXA270 Board**





Target Specifications

CPU	PXA270(520MHz)	BULVERDE V5TE(BGA)
Memory	SRAM 256KByte	Internal
	SDRAM 128MByte	K4S561632-TC75 x 4 External
	NOR Flash 32MByte	TE28F128J3C x 2 External
	NAND Flash 64MByte	K9F1208U0M x 1 External
Internal Peripheral (next slide in detail)	Serial x 3Ports (921,600 bps) : 1=Debug	Standard(=IR)/Full Function/Bluetooth
	Display Controller	6.4" TFT LCD
	CMOS Camera I/F	
	USB Host 1.1 1Port – OTG	
	USB Slave 1.1 1Port	
	PCMCIA Slot	
	CF Slot	
	MMC/SD Slot	
External Peripheral	Serial x 4Ports : 2=RS232, 2=TTL	TL16C554(4 port Full Function IC)
	USB Host 2.0 2Port	TD242LP
	IDE I/F	mini IDE
	Analog RGB(VGA Monitor)	THS8135
	Ethernet(10)	CS8900A 1 Port
	Ethernet(10/100)	SMSC91C111 2Port.
	Audio(CS4299) Codec	Internal Speaker
	Touch Panel	
	RTC4513	Internal Battery
CPLD	256 pin CPLD	XC2C256-7FT256C(BGA)

PXA27x Peripherals

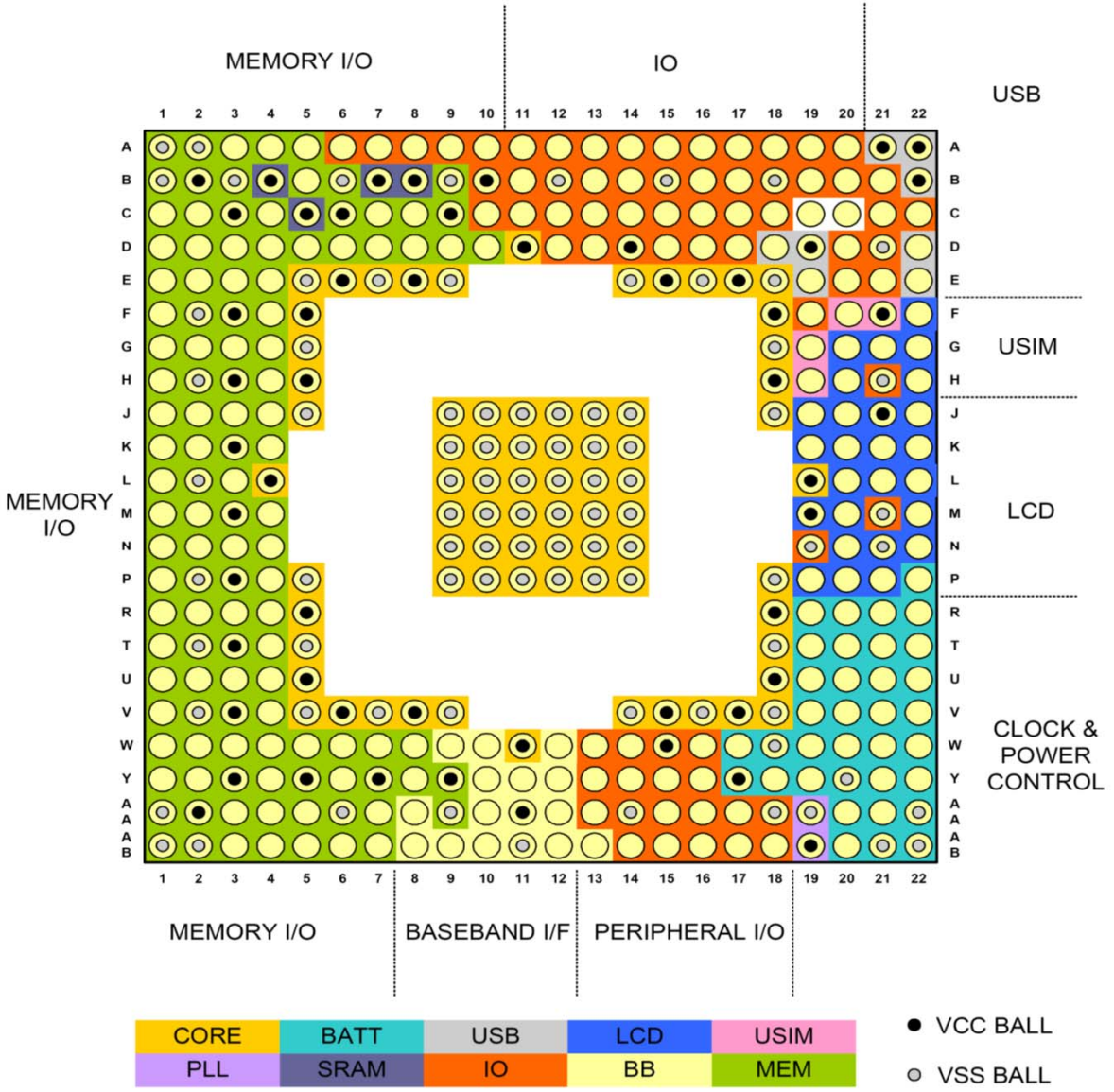
[Internal Peripherals]

- DMA Controller
- Interrupt Controller
- Memory Controller
- SSP (Synchronous Serial Protocol) Port
- I2C Bus Interface
- UART (standard/full function/bluetooth)
- Fast IR
- USB Host (OTG), USB Client
- AC '97 Controller (AC Link Interface)
- Inter-IC Sound (I2S) Controller
- MMC/SD/SDIO Controller
- Memory Stick Host Controller
- Mobile Scalable Link (MSL) Interface
- Universal Subscriber ID Interface
- Quick Capture Interface
- Keypad Interface
- LCD Controller
- Pulse Width Modulator Controller
- General-Purpose I/O Controller
- Real-Time Clock (RTC)
- Operating System Timers
- JTAG Debug Interface

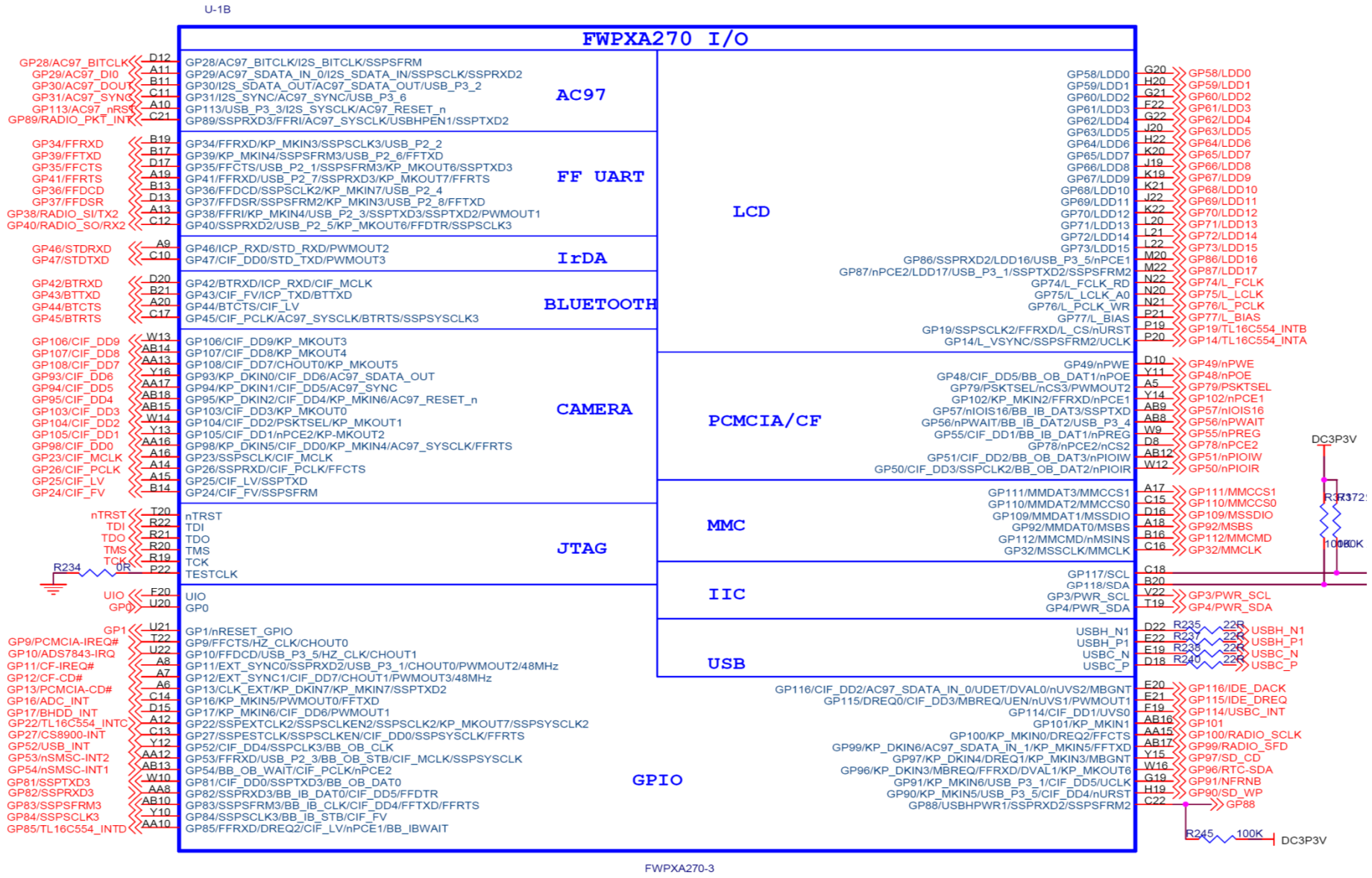
[Peripheral Pins]

- Memory Interface pins
- PCMCIA/Compact Flash card control pins
- LCD controller pins
- Standard UART pins
- Full function UART pins
- Bluetooth UART pins
- MMC controller pins
- SSP pins
- USB client pins
- USB host pins
- AC'97 controller pins
- I2C controller pins
- PWM pins
- Memory stick host controller pins
- Baseband interface pins- Cellular/Wireless Modem
- Keypad interface pins
- Universal subscriber ID interface pins
- Integrated JTAG support
- General-purpose I/O pins
- Clock and Power pins

PXA270 (23mm/360ball)



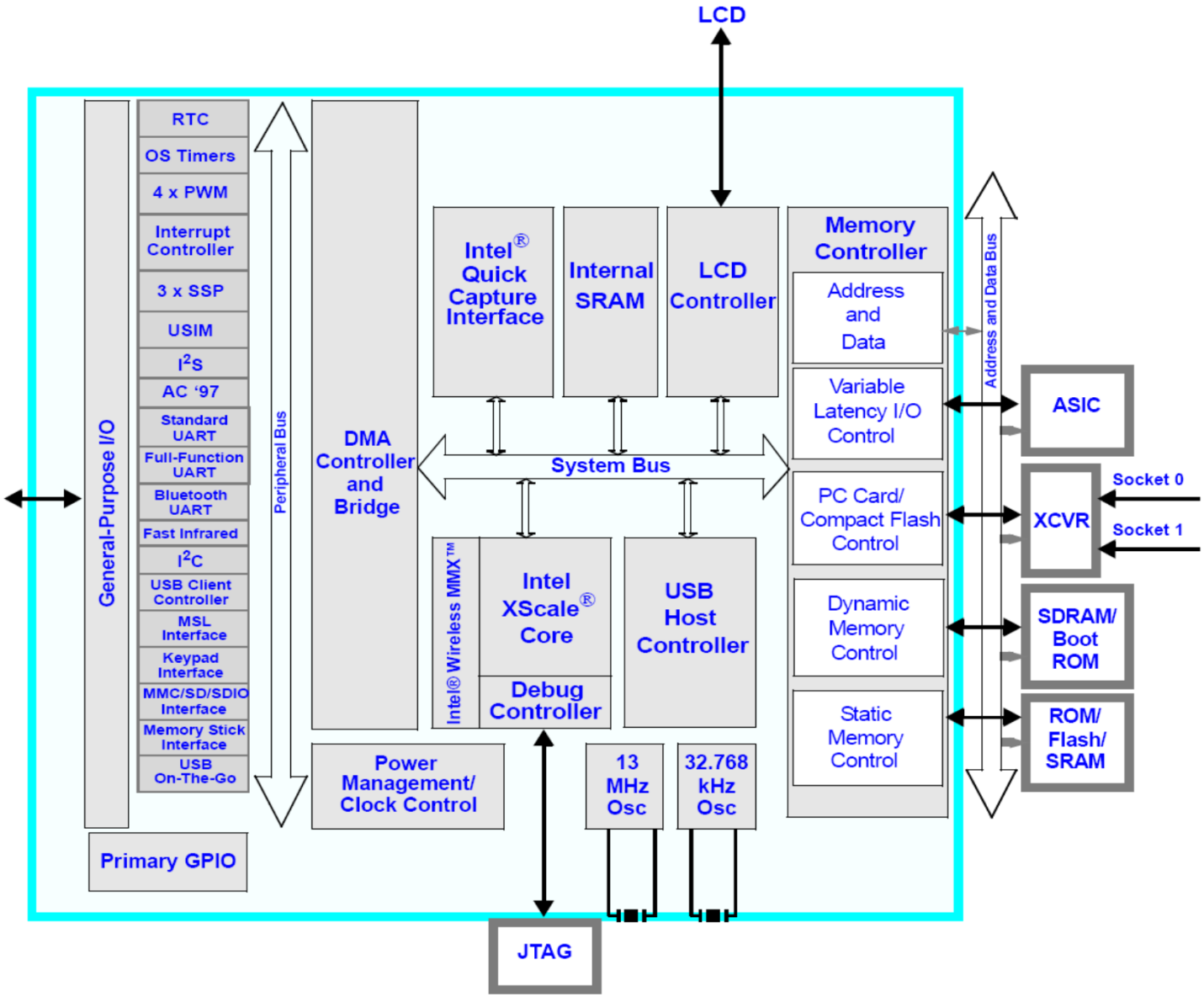
PXA270 I/O Pins



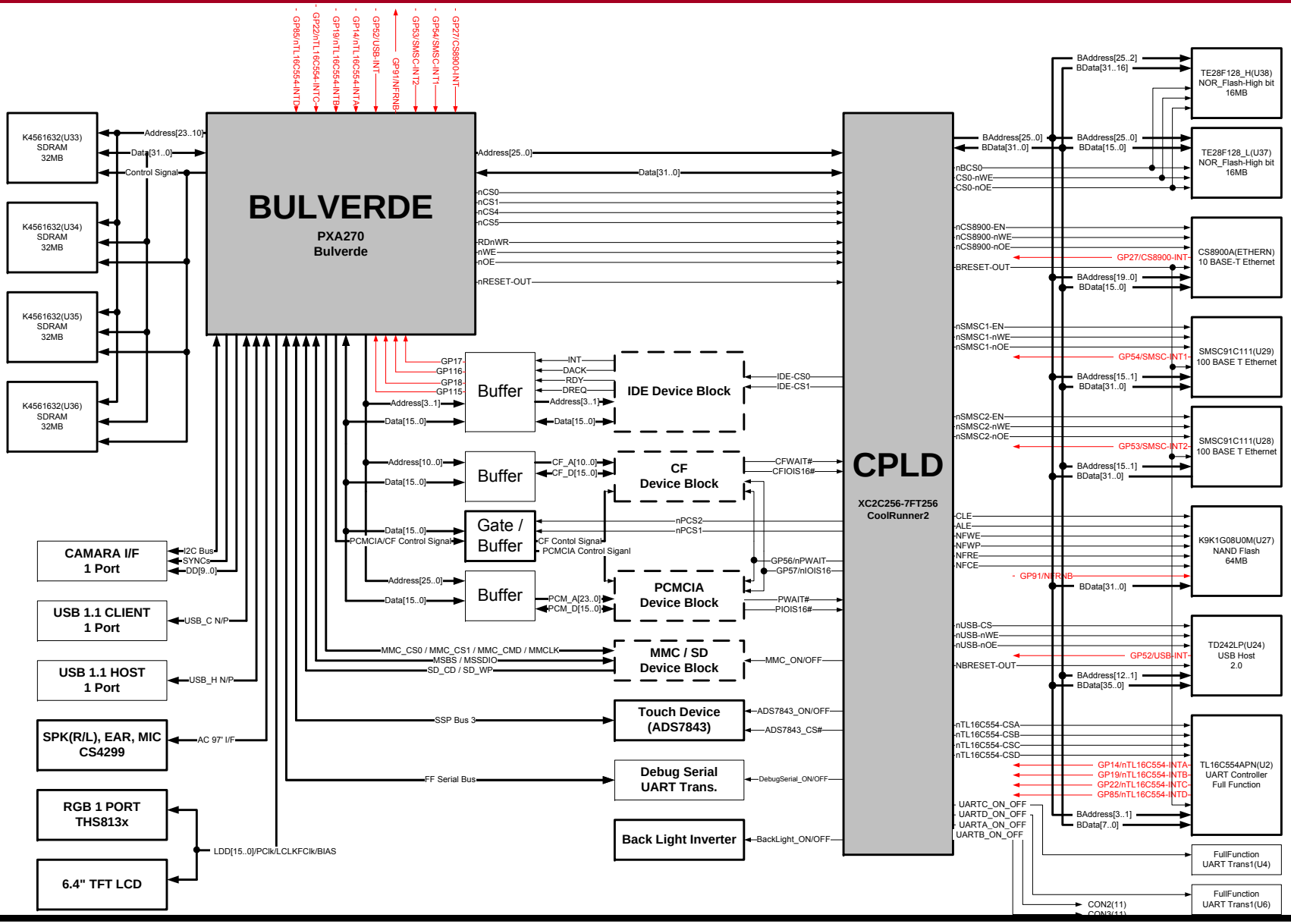
Target Specifications (IEB)

FPGA	EP1C6240PQFP	Cyclone (PQFP-240)
RF Device	CC2420	2.4GHz Zigbee Tranceiver(QLP48)
DC Motor Control	L298	DIP
Step Motor Control	L297	DIP
ADC	ADC0804	
DAC	DAC0800	
7-Segment	7-Segment * 8	
DOT Matrix	DOT Matrix	
Character LCD	Character LCD	16x2

PXA27x Block Diagram



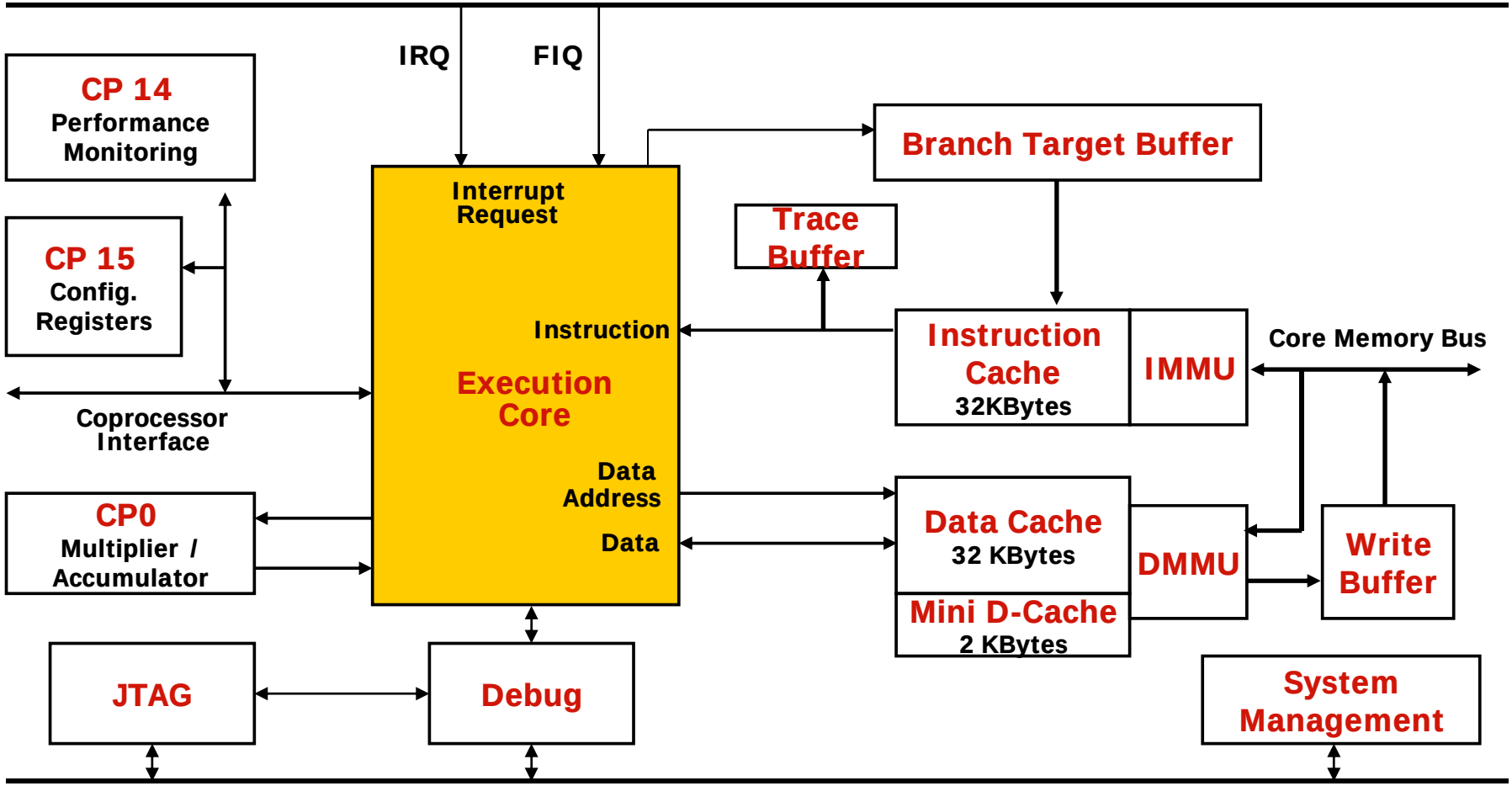
Target Block Diagram



PXA27x SOC

- **PXA27x features**
  - » An integrated **system-on-a-chip** microprocessor for high-performance, low-power, portable, handheld and handset devices.
  - » PXA27x processor family
    - ◆ PXA270 processor in a 13x13mm/23x23mm BGA package
    - ◆ PXA271 processor with 32 Mbytes of Intel StrataFlash® memory and 32 Mbytes of SDRAM
    - ◆ PXA272 processor with 64 Mbytes of Intel StrataFlash® memory
  - » Maximum core frequencies of 624 MHz
    - ◆ Variable core voltage from 0.85 V to 1.55 V
  - » Intel® XScale™ Microarchitecture : refer to Xscale core manual
    - ◆ ARMV5TE Architecture, 32-bit ARM / 16-bit Thumb instruction set
    - ◆ Super-pipelined (integer, multiply-accumulate, and memory)
    - ◆ 32kByte I/D-Cache, 2KByte Mini D-Cache
    - ◆ Memory Management Unit, Branch Target Buffer
    - ◆ Coprocessor, Debug/JTAG, Performance Monitor, Interrupt

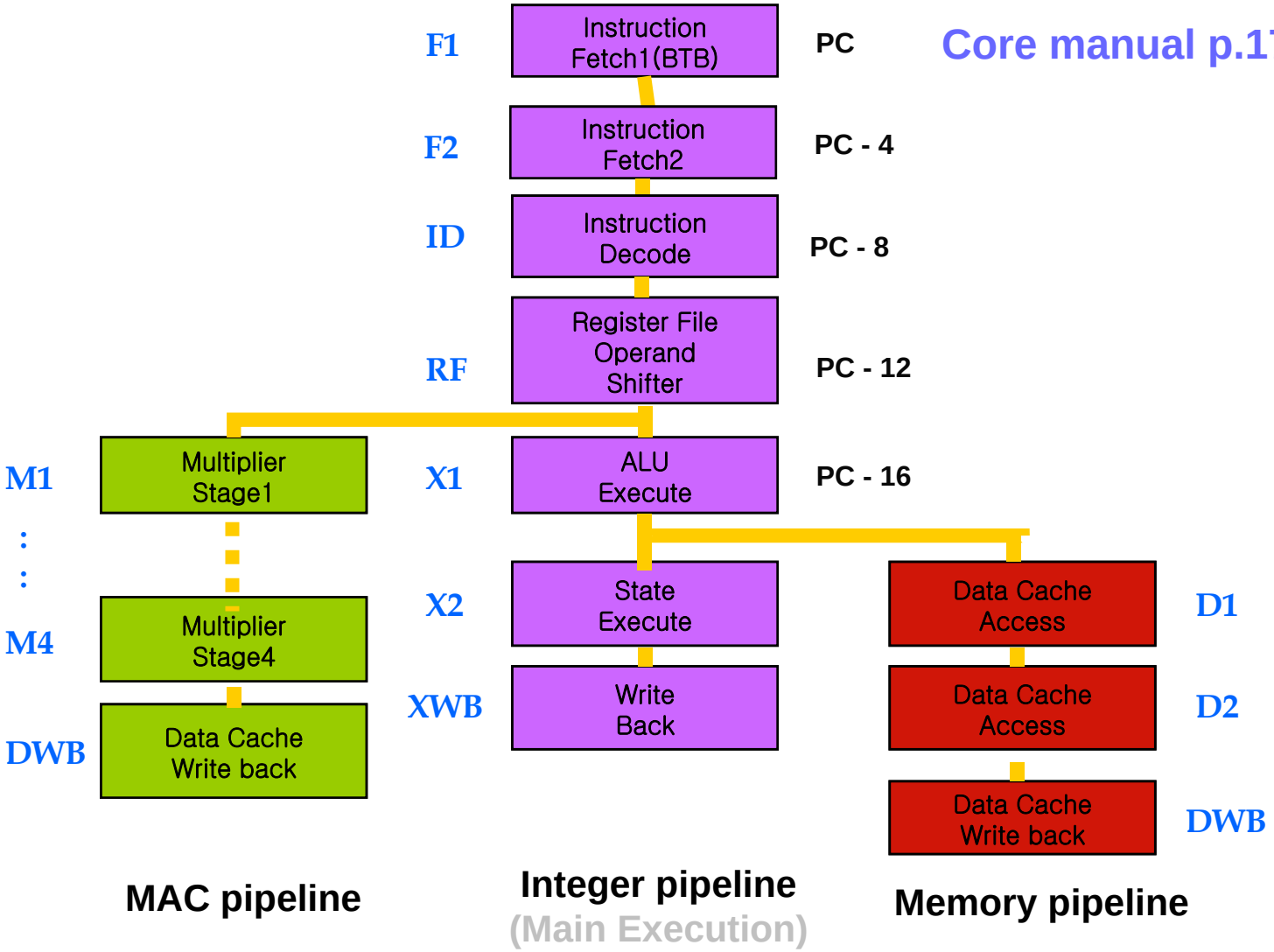
Xscale Microarchitecture



Xscale Microarchitecture

- **Superpipeline** : Core Manual p.177 – 181, MAC : Core Manual p.16, p.182
  - The superpipeline is composed of integer, multiply-accumulate (MAC), and memory pipes.
    - → The integer pipe has **seven** stages: branch target buffer/fetch 1, fetch 2, decode, register file/shift, ALU execute, state execute, and integer writeback.
    - → The memory pipe has **eight** stages that use the first five stages of the integer pipe (BTB/fetch 1 through ALU execute) and then finish with memory stages data cache 1, data cache 2, and data cache writeback.
    - → The MAC pipe has **nine** stages that use the first four stages of the integer pipe (BTB/fetch 1 through register file/shift) and then finish with MAC stages MAC1, MAC2, MAC 3, MAC 4, and data cache writeback.
- **Internal Memory Bus**
  - Simultaneously transfer one 32-bit word of **load** data and one 32-bit word of **store** data every clock cycle (maximum data rate = 2.4 GBytes/sec.)
  - In each direction at 600 MHz, Maximum data rate = 4.8 GBytes/sec.
    - → 64-bits (in each direction 32 bits x 2) = 8-Bytes
    - → 4.8-GBytes/sec. (2.4GBytes/sec. x 2) divided by 8-Bytes = 0.6-G/sec.
    - → 0.6-G/sec. = 600 MHz (maximum core clock frequency)

Superpipeline



Core manual p.177,180

Xscale Microarchitecture

- **Branch Target Buffer (BTB) :** Core Manual p.57-58
  - The 128-entry branch target buffer keeps the pipeline filled with statistically correct branch choices.
  - The BTB contains the address of a branch instruction, the target address associated with the branch instruction, and a previous history of the branch being taken or not taken.
  - The history is recorded as one of four states: strongly taken, weakly taken, weakly not-taken, or strongly not-taken.
  - If the address of the branch instruction hits in the BTB : Core Manual p.58/fig.5-2
    - → And it's history is strongly or weakly taken,
      - the instruction at the branch target address is fetched
    - → And it's history is strongly or weakly not-taken,
      - the next sequential instruction is fetched.
    - → In either case the history is updated.
  - The BTB can be enabled or disabled via coprocessor 15, register 1.
- **Write Buffer (WB)**
  - The WB holds data for storage to memory until the bus controller can act on it.

Xscale Microarchitecture

- **Instruction Memory Management Unit (IMMU)**
  - For instruction prefetches, the IMMU controls logical-to-physical address translation, memory access permissions, memory domain identifications, and attributes (governing operation of the Instruction Cache).
- **Data Memory Management Unit (DMMU)**
  - For data fetches, the DMMU controls logical-to-physical address translation, memory access permissions, memory domain identifications, and attributes (governing operation of the data cache or mini-data cache and write buffer).
- **Interrupts**
  - The microarchitecture responds to normal (IRQ) and (FIQ) fast interrupts.
- **Clock and Power Management**
  - The core is designed with power saving techniques that power-up a functional block only when it is needed.
  - Power management provides power savings with **idle**, **deep-idle**, **standby**, **sleep**, and **deep sleep** modes.
  - Low power modes are selectable by programming CP 14, register 6.

Xscale Microarchitecture

- **Instruction Cache (I-Cache)**
  - The 32Kbyte I-cache can contain high-use multiple code segments or entire programs, allowing the core access to instructions at core frequencies.
- **Data Cache (D-Cache)**
  - The 32Kbyte D-cache can contain high-use data such as lookup tables and filter coefficients, allowing the core access to data at core frequencies.
- **Mini-Data Cache**
  - The Mini-data cache can contain frequently changing data streams such as MPEG video, allowing the core access to data streams at core frequencies.
  - The 2Kbyte Mini-data cache relieves the D-cache of data “thrashing” caused by frequently changing data streams
    - → This prevents core stalls caused by multi-cycle accesses to external memory.
- **Trace Buffer**
  - 256 entry trace buffer captures control flow used in the debug unit.
  - Trace buffer register is controlled by Coprocessor 14(CP) register.

Xscale Microarchitecture

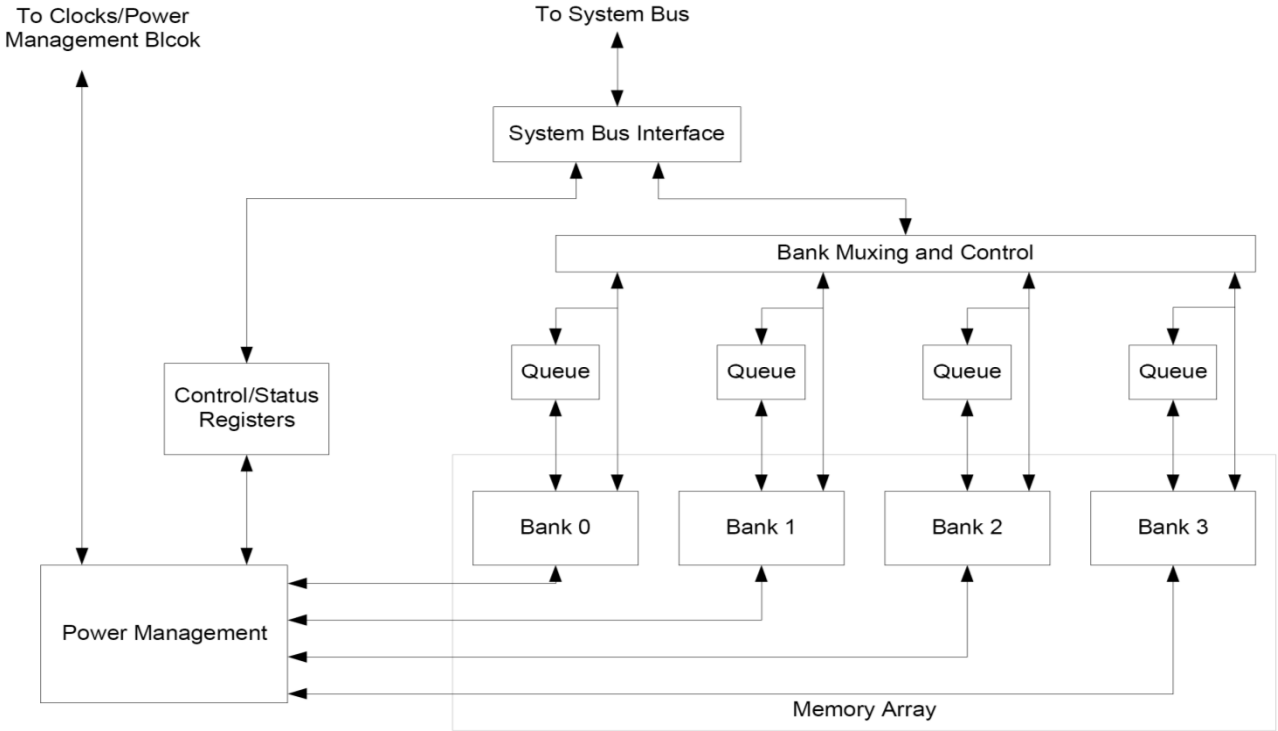
- **Coprocessor Interface**
  - The coprocessor interface uses a 32-bit bus for data transfers (at core frequency) between a coprocessor and core registers.
- **Multiply-Accumulate Coprocessor (CP0= MAC)**
  - PXA27x does not contain floating point hardware (integer only in CP0)
  - For efficient processing of audio media algorithms, CP0 provides 40-bit accumulation of 16x16, dual 16x16 (SIMD), and 16x32 signed multiplies.
- **Coprocessor 14 (CP14) Register** : Core Manual p.96
  - CP14 provides registers that identify or control operation of functions within the microarchitecture.
    - → Performance Monitoring, Core Clock Configuration, Power Mode, Software Debug
- **Coprocessor 15 (CP15) Register** : Core Manual p.80
  - CP15 provides registers that identify or control operation of functions within the microarchitecture.
    - → Cache Type, Cache Operations, Translation Table Base, Fault Status, TLB Operations, Breakpoint

Xscale Microarchitecture

- **Performance Monitoring Unit**
  - The performance monitoring unit contains two 32-bit event counters and one 32-bit clock counter.
  - The event counters can be programmed to monitor i-cache hit rate, data caches hit rate.
- **JTAG**
  - The industry-standard IEEE1149.1 JTAG port consists of a test access port (TAP) controller, boundary-scan register, instruction and data registers, and dedicated signals TDI, TDO, TCK, TMS, and nTRST.
  - The JTAG port can also be used to access the debug unit.
- **Debug Unit**
  - The debug unit is accessed through the JTAG port.
  - The debug unit allows the debugger application code or a debug exception to stop program execution and re-direct execution to a debug handling routine.

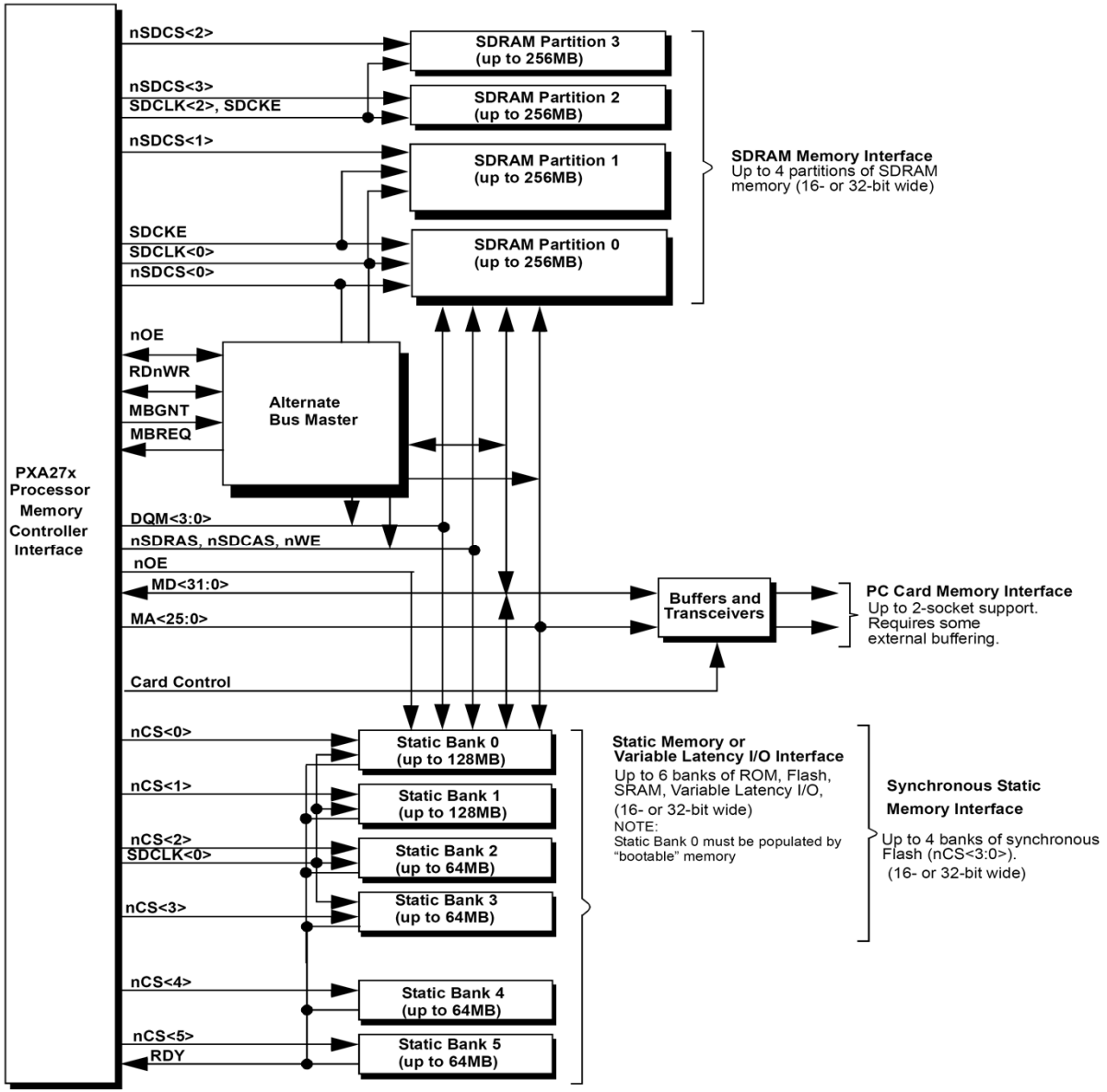
PXA27x Memory Interface

- **System memory interface**
  - Up to 100-MHz SDRAM @ 1.8 V, 2.5 V, 3.0 V or 3.3 V
  - Int. Four Banks of on-chip 64Kbytes **SRAM** : 256 Kbytes
  - Ext. 16, 64, 128, and 256-Mbit **SDRAM** support : 4 partitions
  - Ext. Static memory devices (SRAM, ROM, **NOR-Flash**, or VLIO(**NAND-Flash**)) support : 6 banks



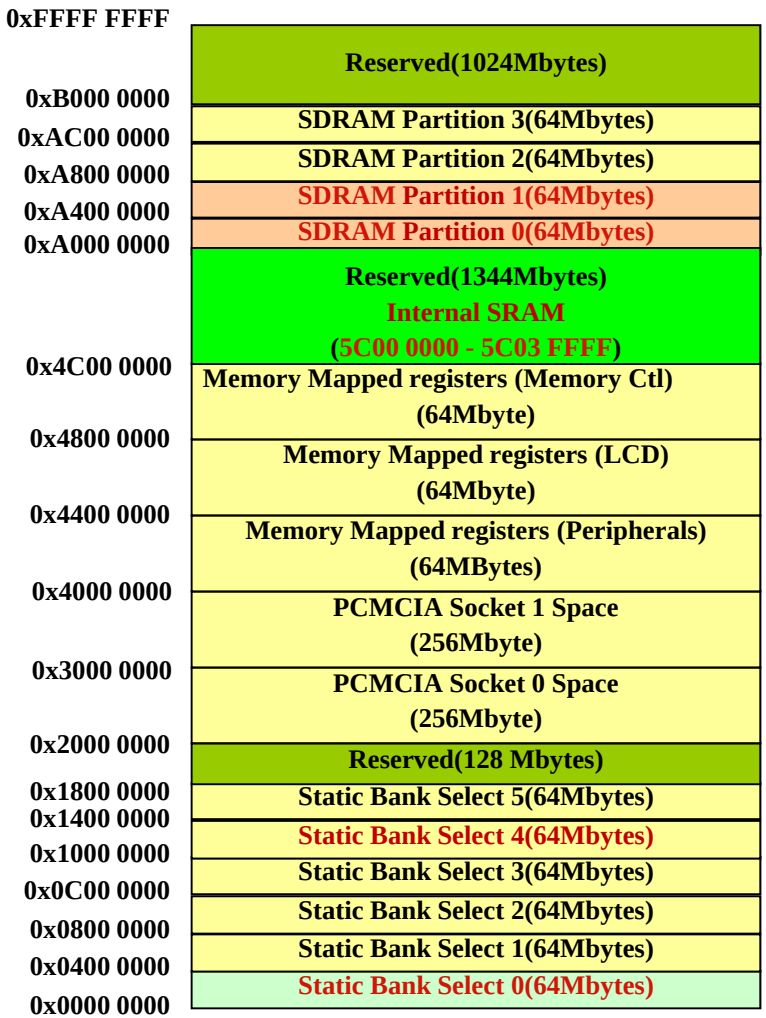
Internal SRAM

External Memory Interface



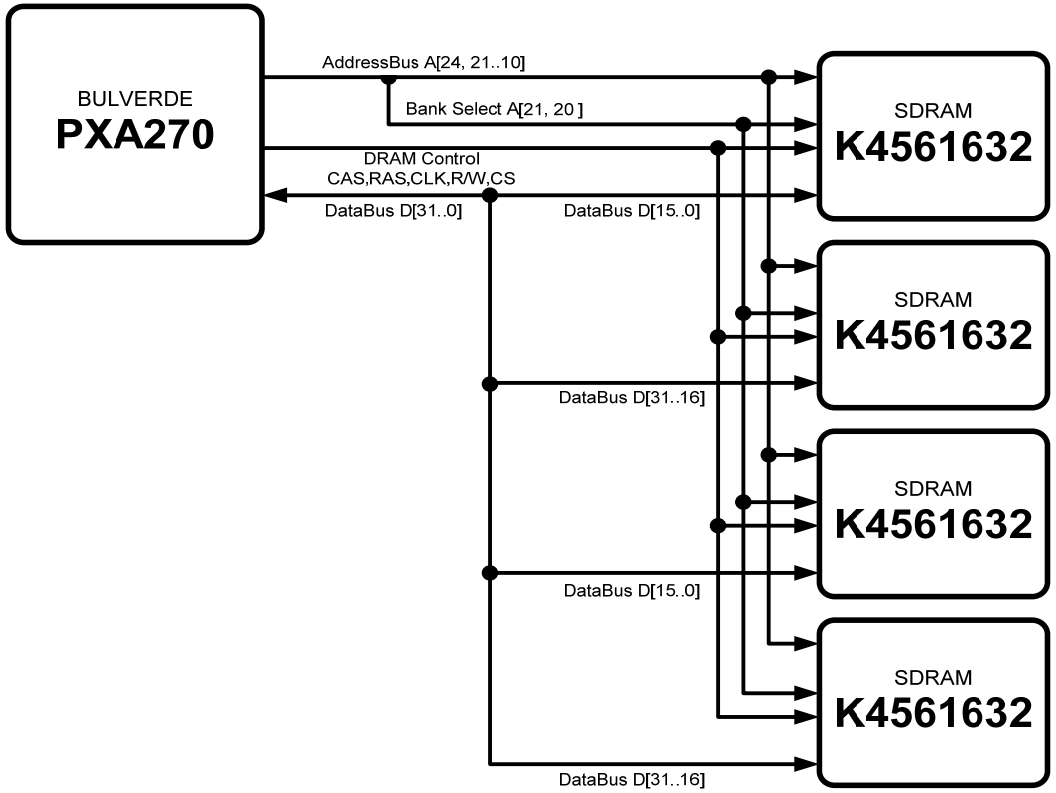
Memory Map- 4G

- **External SDRAM** : 128Mbyte
  - 0xA000 0000 – 0xA7FF FFFF **Part. 0 / 1**
- **Internal SRAM** : 256Kbyte
  - Memory Bank 0 64-Kbyte SRAM
    - → 0x5C00 0000– 0x5C00 FFFF
  - Memory Bank 1 64-Kbyte SRAM
    - → 0x5C01 0000– 0x5C01 FFFF
  - Memory Bank 2 64-Kbyte SRAM
    - → 0x5C02 0000– 0x5C02 FFFF
  - Memory Bank 3 64-Kbyte SRAM
    - → 0x5C03 0000– 0x5C03 FFFF
- **External nand-Flash** : 64Mbyte
  - 0x10F0 0000 **Bank 4** (I/O device)
  - Variable Latency I/O – slide # 20
- **External nor-Flash** : 32Mbyte
  - 0x0000 0000 – 0x001F FFFF **Bank 0**



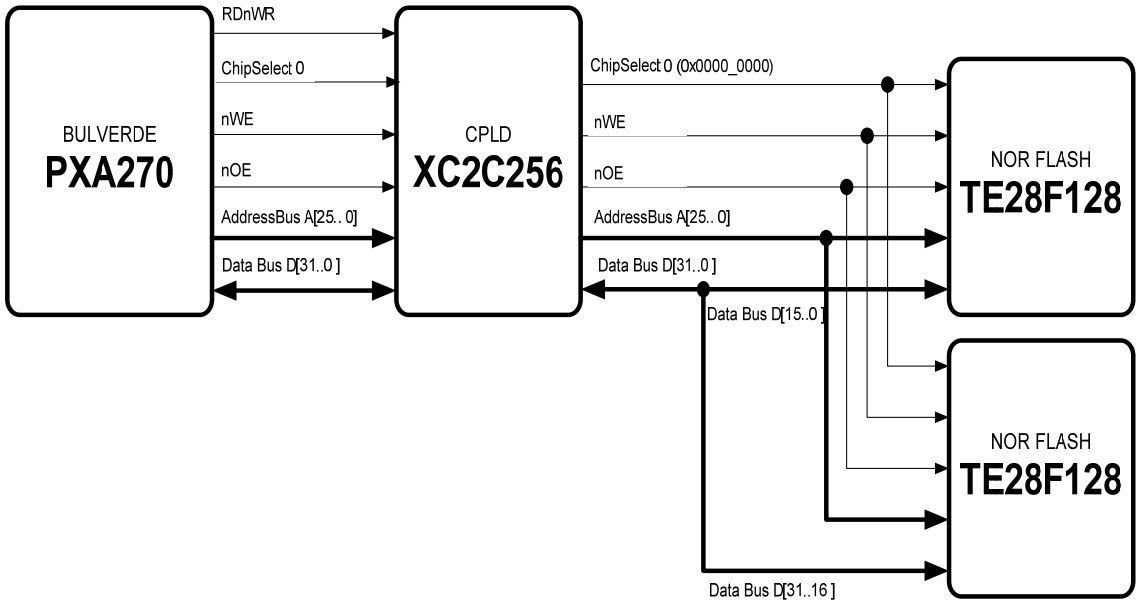
Target Memory

- **SDRAM : K4S561632 (256Mbit) x 4**
  - 16M x 16bit (D15..D0) = 8M x 32bit = 256Mbit
  - 8M x 32bit x 4 = **32M x 32bit** (**128M x 8bit**) **not 128M x 32bit**



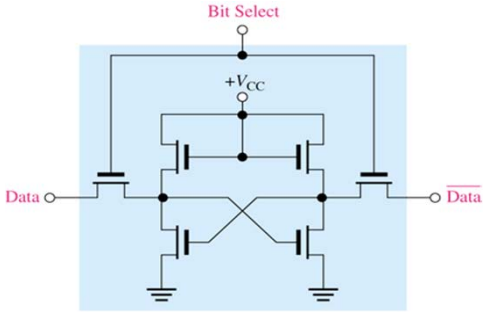
Target Memory

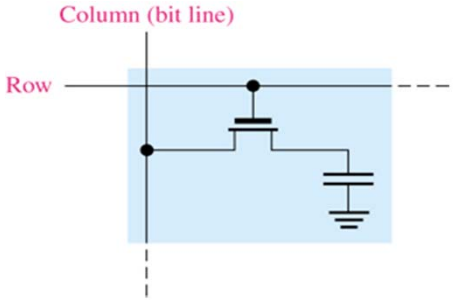
- **NOR Flash : TE28F128 (128Mbit) x 2**
  - 128Mbit = 8M x 16bit = 8Mword(16bit)
  - 8Mword(16bit) x 2 = **8M x 32bit** (**32M x 8bit**) **not 32M x 32bit**



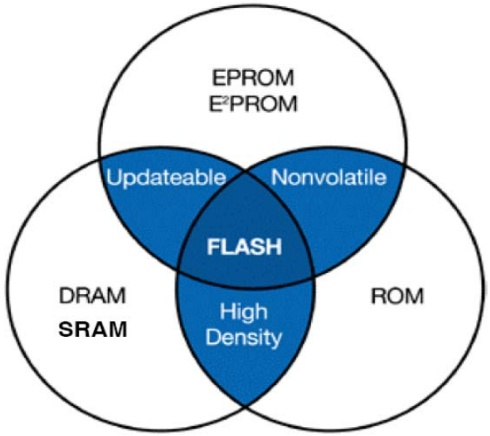
Solid State Memory

- **RAM**(volatile) : **need power to maintain the information stored in the chip.**
  - » **SRAM**
    - Highest speed, low-density memory (limited density drives up cost)
    - Register file, on-chip cache
  - » **DRAM**
    - High-density, low-cost, Refresh





- **ROM** (non volatile)
  - » **UV-EPROM** (Chip Level Erase)
  - » **EE-PROM** (Byte Level Erase)
  - » **Flash Memory** (Block Level Erase)
  - » **Mask ROM**, **OTPROM**



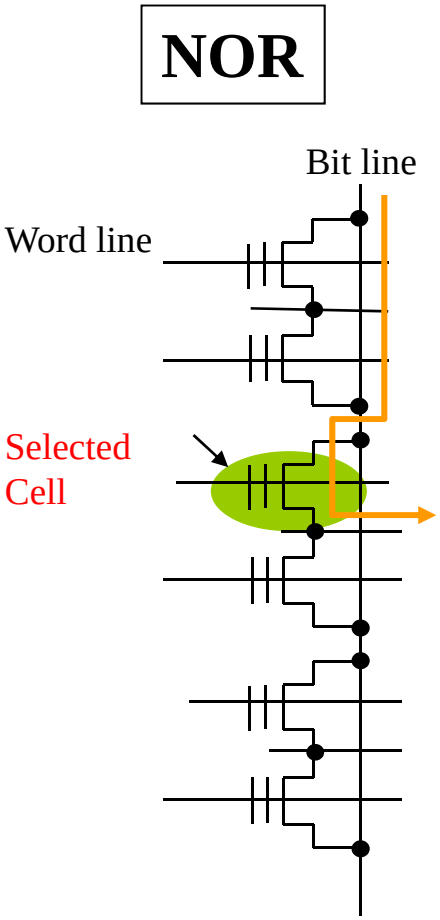
Non-volatile Memory

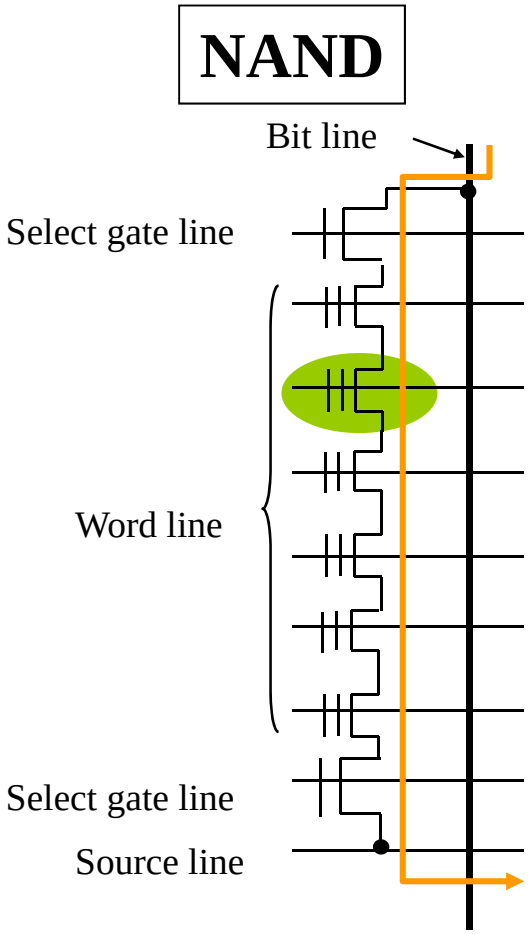
- **ROM** (non volatile)
  - » **Mask ROM**
    - Permanently programmed during the manufacturing process
    - High-density, reliable, low cost
    - Time-consuming mask required, suitable for high production with stable code
  - » **OTPROM**
    - PROM chip that can be programmed only once
  - » **UV-EPROM**
    - High-density memory; must be exposed to ultraviolet light for erasure
  - » **EE-PROM**
    - Electrically byte-erasable; lower reliability, higher cost, lowest density
  - » **Flash Memory**
    - Specific type of EE-PROM that is erased and programmed in large blocks; in early flash the entire chip had to be erased at once.
    - High reliability, Low-cost, high-density, high-speed architecture, low power

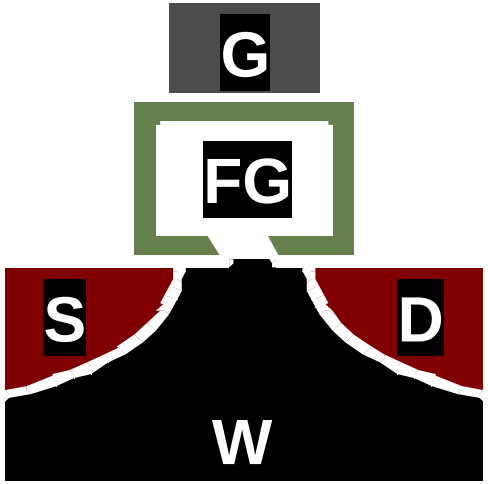
Flash Memory

- **Flash Memory**
  - Unlike EEPROM, it is erased and programmed in blocks.
  - Used as program memory (non-volatile) and in some cases as a data memory
  - Writing using special writing algorithms
    - Writing algorithms are manufacturer specific
  - Manufacturing processes are manufacturer dependent
    - Specifications and properties are different
    - Multiple techniques: NOR, NAND,...
    - NOR allows random-access for reading : XIP (Program Memory)
    - NAND allows only block access : Disk
  - Long erasure time, slower write access times
  - Small, Light-weight, Low-power, Robust, Fast read access times (compared to disks)
  - Limited lifetime (Typically, 100,000 – 1,000,000 program/erase cycles)
  - Bad block management (BBM)
    - Remapping to spare sectors in case of write failure

Flash Memory







Floating Gate (FG) TR

Comparison of NOR and NAND Flash

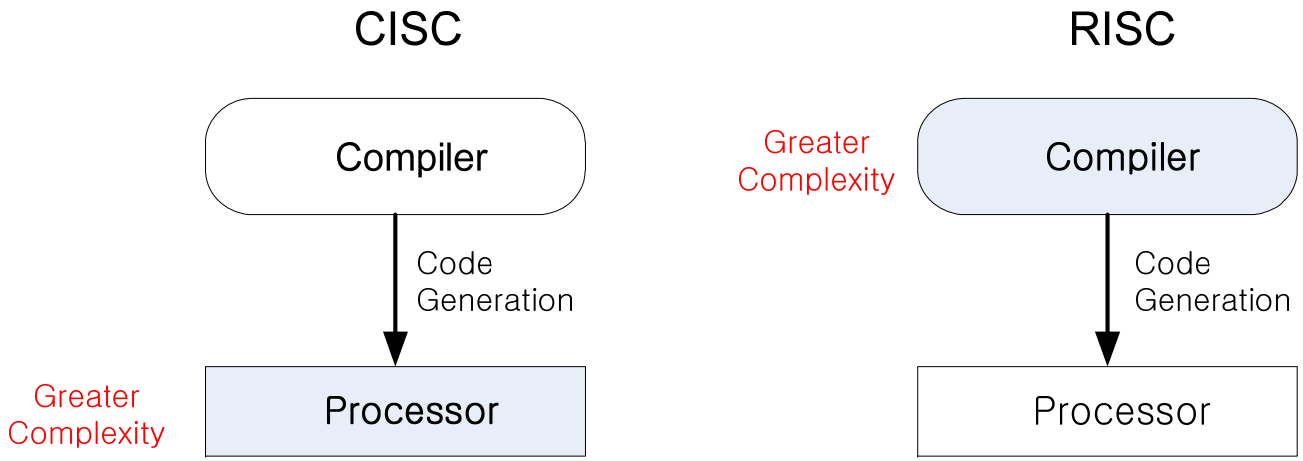
Property \ Type	NOR MT29F2G08A	NAND MT28F128J3
Initiated by	Intel	Toshiba
Random access	Yes	No
Execute in place(XIP)	Yes	No
Read	Fast (0.12 us)	Fast (25 us)
Write	Slow (180 us/32 bytes)	Fast (300 us/2112 bytes)
Erase block	Very slow (typ. 750 ms)	Fast (typ. 2 ms)
Size of cell	Larger	Smaller
Interface	Full memory interface	I/O only
Reliability	Larger	Smaller
Erase cycle	10,000-100,000	100,000-1,000,000
Easy of use	Easy	Complicated
Price	High	Low
Ideal usage	Code storage	Data storage only
Applications	Boot code, Firmware(PC BIOS), Embedded design (Cell phone, PDA, set top box)	Data storage, USB sticks, memory cards(high capacity &low price), digital camera(image storage), MP3 player(music storage)

ARM Processor

- **ARM Features**
  - » Reduced Instruction Set Compute (RISC) architecture
  - » General purpose 32-bit microprocessors
    - ◆ Byte : 8 bits, Halfword : 16 bits (2 bytes), Word : 32 bits (4 bytes)
  - » Three instruction sets (ARM, Thumb, Jazelle)
    - ◆ 32-bit **ARM** Instruction Set
    - ◆ 16-bit **Thumb** Instruction Set
    - ◆ **Jazelle** cores can execute Java byte code
  - » Pipeline
    - ◆ Speeds up execution by fetching the next instruction while other instructions are decoded and executed
  - » Memory Management Unit
    - ◆ Uses a set of translation tables
    - ◆ Provide a virtual to physical address map as well as access permissions
  - » High performance and Very low power consumption

RISC Design

- **Simple but powerful instructions that execute within a single cycle at high clock speed**
  - » Reduce complexity of instructions performed by hardware.
  - » Provide greater flexibility and intelligence in software rather than hardware



RISC Design Philosophy

- **Reduced number of instruction classes**
  - » Provide simple operations that can execute in a single cycle
  - » Compiler or programmer synthesizes complicated operations by combining several simple instructions
- **Pipeline Architecture**
  - » Easy to construct pipeline because each instruction is a fixed length.
  - » Processing instructions are broken down into smaller units that can be executed in parallel by pipeline
- **General purpose registers**
  - » Any register can contain either data or address
  - » Act as the fast local memory
  - » Large general-purpose register set
- **Load-store architecture**
  - » Separate memory accesses from data processing

ARM Processor

- **Architecture Revisions**
  - » Every ARM processor implementation executes a specific ISA
  - » Nomenclature

    ARM {x} {y} {z} {T} {D} {M} {I} {E} {J} {F} {-S}

    x : Family
    y : MMU/MPU
    z : Cache
    T : Thumb 16 bit decoder
    D : JTAG debug
    M : Fast multiplier
    I : EmbeddedICE macrocell
    E : DSP enhanced instructions core manual p.22
    J : Jazelle
    F : VFP (vector floating point)
    S : Synthesizable version

    - All ARM cores after ARM7TDMI include TDMI features
    - Processor family is a group of processor that share the same hardware characteristics
    - JTAG : a serial protocol used by ARM to send and receive debug information
    - EmbeddedICE Macrocell : the debug hardware built into the processor that allows break points and watch points to be set
    - VFP : used to increase performance of graphic applications (ARM does not contain floating-point hardware)
    - Synthesizable : the processor core is supplied as source code that can be compiled into a form easily used by EDA tools

ARM Architecture Version

1

2

3

Early ARM architectures

Halfword and signed halfword / byte support

4

System mode

SA-110

SA-1110

Thumb instruction set

4T

ARM7TDMI

ARM9TDMI

ARM720T

ARM940T

Improved ARM/Thumb Interworking

5TE

Count Leading Zero (CLZ) instruction

Saturated maths

DSP multiply-accumulate instructions

ARM1020E

ARM9E-S

XScale

ARM966E-S

Jazelle

Java bytecode execution

5TEJ

ARM9EJ-S

ARM7EJ-S

ARM926EJ-S

Cross-compiler

- **Compiler**
  - vi hello-x86.c

```
#include <stdio.h>
main()
{
printf("Hello, Welcome to Embedded world.\n");
}
```

  - gcc –o hello-x86 hello.c
  - ./hello-x86
- **Cross-compiler : Toolchain**
  - mkdir /pxa270, cd /mnt/cdrom, cp –a * /pxa270
  - cd /pxa270/toolchain, cp iwmmxt-1.0.0.tgz /opt, cd /opt
  - tar xzvf iwmmxt-1.0.0.tgz
  - vi /root/.bash_profile

```
PATH=$PATH:/opt/iwmmxt-1.0.0/bin
```

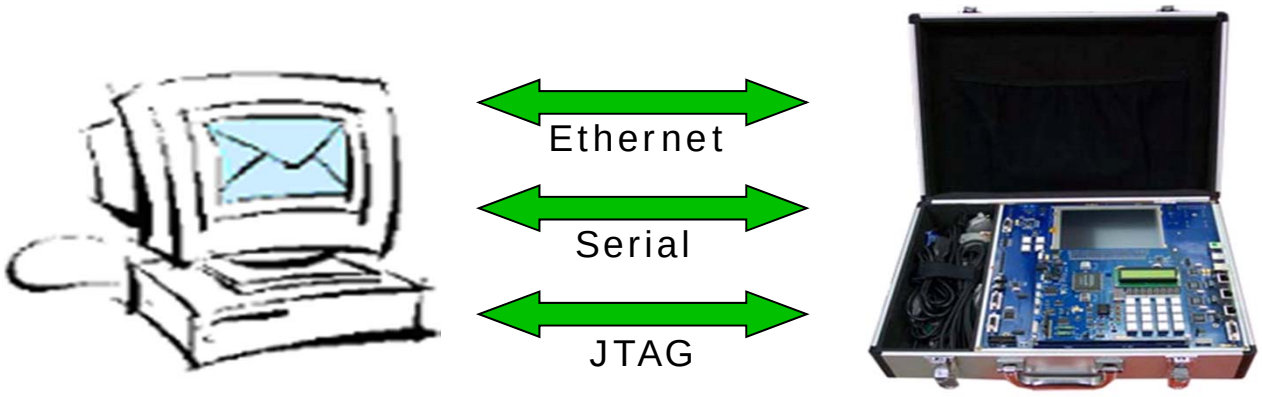
  - source /root/.bash_profile, ctl+alt+BS, log-in

minicom

- **Cross-compile**
  - » arm-linux-gcc –o hello-arm hello.c
  - » ./hello-arm, file hello-arm
- **minicom setup**
  - » minicom –s → Serial port setup
    - A - Serial Device : /dev/ttyS0
    - E - Bps/Par/Bits : 11500 8N1
    - F - Hardware flow control : No
  - » Save setup as dfl
- **Target program run**
  - » Cont-A-Z, Send files, zmodem, Space for selection, Enter
  - » ./hello-arm

Host and Target Communication

- **1. Serial : Minicom**
- **2. Parallel : JTAG** (Joint Test Access Group)
  - » Device test and flash fusing (jflashmm)
- **3. Ethernet : tftp, NFS, tcp/ip**



Host System **Target System**

bootp

- **bootp install**
  - » rpm –qa | grep bootp
  - » rpm –ivh bootp-2.4.3-7.i386.rpm : /pxa270/rpms
  - » bootptab file : /etc/bootptab

    pxa270:ht=ether:ha=0x000b7a88f890:ip=210.68.65.120:sm=255.255.255.0
    - ◆ tag symbols : man bootptab
  - » bootp file : /etc/xinetd.d/bootp

```
service bootps
{
disable             = no
socket_type         = dgram
}
```

  - » service xinetd restart
  - » boot monitor : set myipaddr, destipaddr
  - » bootp command : at boot monitor program

tftp

- **tftp server install**
  - rpm –qa | grep tftp
  - rpm –ivh tftp-server-0.17-9.i386.rpm
  - tftp file : /etc/xinetd.d/tftp

```
service bootps
{
socket_type        = dgram
                :
server_args        = -s /tftpboot
disable            = no
}
```

  - service xinetd restart
  - cp /pxa270/images/zImage /tftpboot
  - tftp zImage kernel (at target)
  - flash kernel
    - Address definition : /pxa270/bootloader/bboot-1.0.1/include/board.h

NFS

- **NFS Install**
  - » ntsysv command : nfsd & portmap service start
  - » Host server system
    - ◆ exports file : /etc/exports
      /nfs 220.68.65.120 (rw, no_root_squash)
      → * : for all IP access
    - ◆ mkdir /nfs, cat > test.txt
    - ◆ service nfs restart
    - ◆ service nfs status
  - » Target system
    - ◆ mount –t nfs 220.68.65.20:/nfs /nfs

samba server

- **samba server settings : 2 ways**
  - 1) Linux System Setting Menu
    - ntsysv → smb service enable
    - system settings →server settings →samba server
    - add→ Basic →Directory (samba) and Permission (Read/Write)
    - add →Access→ Allow access to everyone
    - Preferences→ Server Settings→ Basic →Workgroup (same with clients)
    - Security→ Authentication Mode (Share), & Guest Account (root)
    - service smb restart
  - 2) vi /etc/samba/smb.conf

```
[samba]
        path = /samba/
        writeable = yes
        guest ok = yes
```

- **Windows Client**
  - \\220.68.65.20\samba

Network Configuration

- **Target Network Config.** : boot flash fusing -monitor
  - » ***set*** myipaddr, destipaddr, myhaddr command or config file
  - » setup.c source : myhaddr={0x00,0x08,0xc9,0x62,0x5a,0xeb} random
    destipaddr (host)= 192.168.100.100 → 220.68.65.20 config file
    myipaddr (target)= 192.168.100.50 → 220.68.65.120 config file
- **Target Network Config.** : rootfs.img flash fusing
  - » /etc/rc.d/init.d/network : script file
    - ◆ ifconfig eth0 220.68.65.120 netmask 255.255.255.0 up
    - ◆ route add default gw 220.68.65.254
- **Host / Target Network Config.** : rootfs.img flash fusing
  - » /etc/sysconfig/network : hostname, gateway
  - » /etc/resolv.conf : name server
  - » /etc/sysconfig/network-scripts/ifcfg-eth0 :
    - ◆ IPADDR, NETMASK, HWADDR, GATEWAY, BROADCAST, …
  - » mac address command
    - ◆ ifconfig eth0 down hw ether 00:00:00:00:00:01
    - ◆ ifconfig eth0 up

Network Configuration

- **Target MAC Address Change** : zImage flash fusing
  - /kernel/linux-2.6.11-h270-tku_v1.1/drivers/net/cs89x0.c
  - unsigned char mac[] = {0x00,0x0b,0xf4,0x2f,0x9b,0xe5};