Review of the Apollo Testbed Project

Massoud Pedram

University of Southern California
Dept. of EE Systems
Los Angeles CA 90089

Dec. 4, 2003

Outline

- BitsyX versus AT2
- Fine-Grained Dynamic Voltage and Frequency Scaling
- Power Reduction in the Display System
- Wake on Wireless
- Power Management of the Main Memory System

BitsyX versus AT2

BitsyX Platform

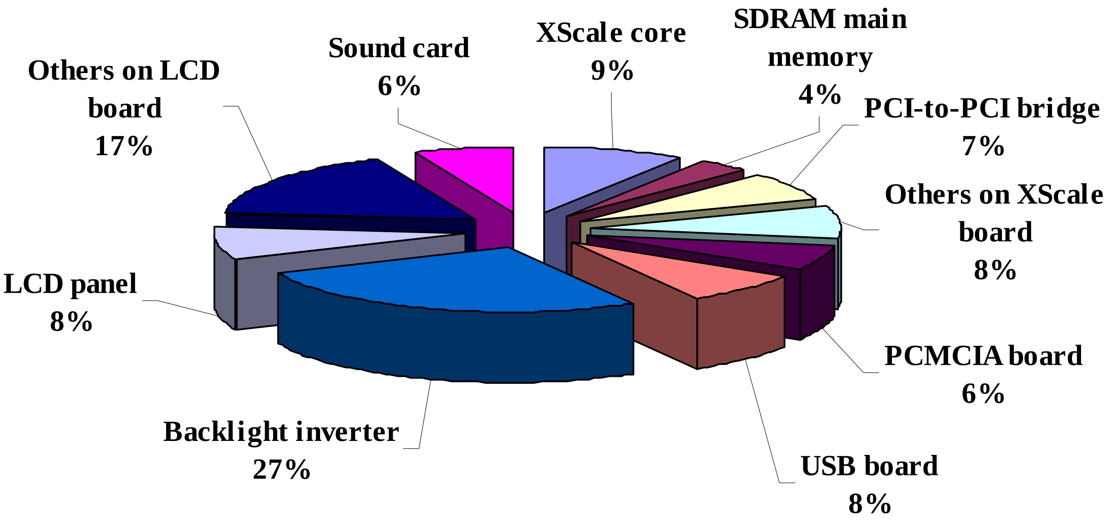
- **Specification**
  - Intel PXA255 32-bit XScale processor up to 400MHz
  - Intel SA-1111 companion chip
  - 64MB SDRAM on-board main memory
  - 32MB on-board Flash memory
  - USB 1.1 host and client functions
  - LCD panel interface for GUI
  - ADSmartIO for GPIOs
  - AC97 codec audio interface

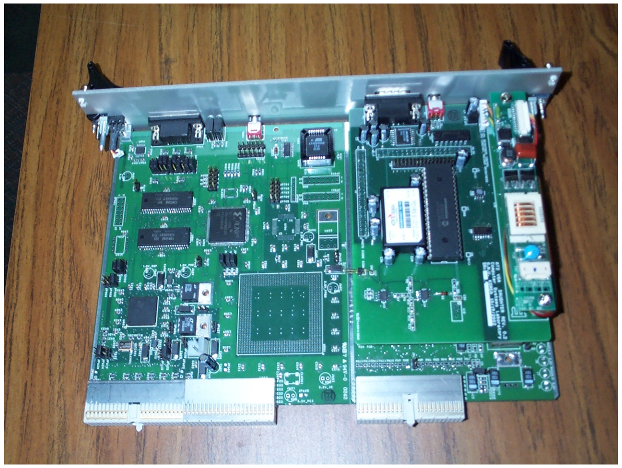
BitsyX Platform (cont’d)

- **Power consumption**
  - Running mode (9.4W)
    - PXA255 processor core in turbo mode
    - SDRAM main memory in operating mode
    - SA1111 companion chip in run mode
    - Backlight on
    - LCD panel on
  - Sleep mode (480mW)
    - PXA255 processor core in sleep mode
    - SDRAM main memory in self-refresh mode
    - SA1111 companion chip in reset
    - Backlight off
    - LCD panel off

AT2 Platform

- **With 6.4 inch transmissive LCD panel (LP064V1)**
  - The display system consumes about 52% of the total system power consumption (10.9W)





BitsyX Platform (cont'd)

- BitsyX versus AT2

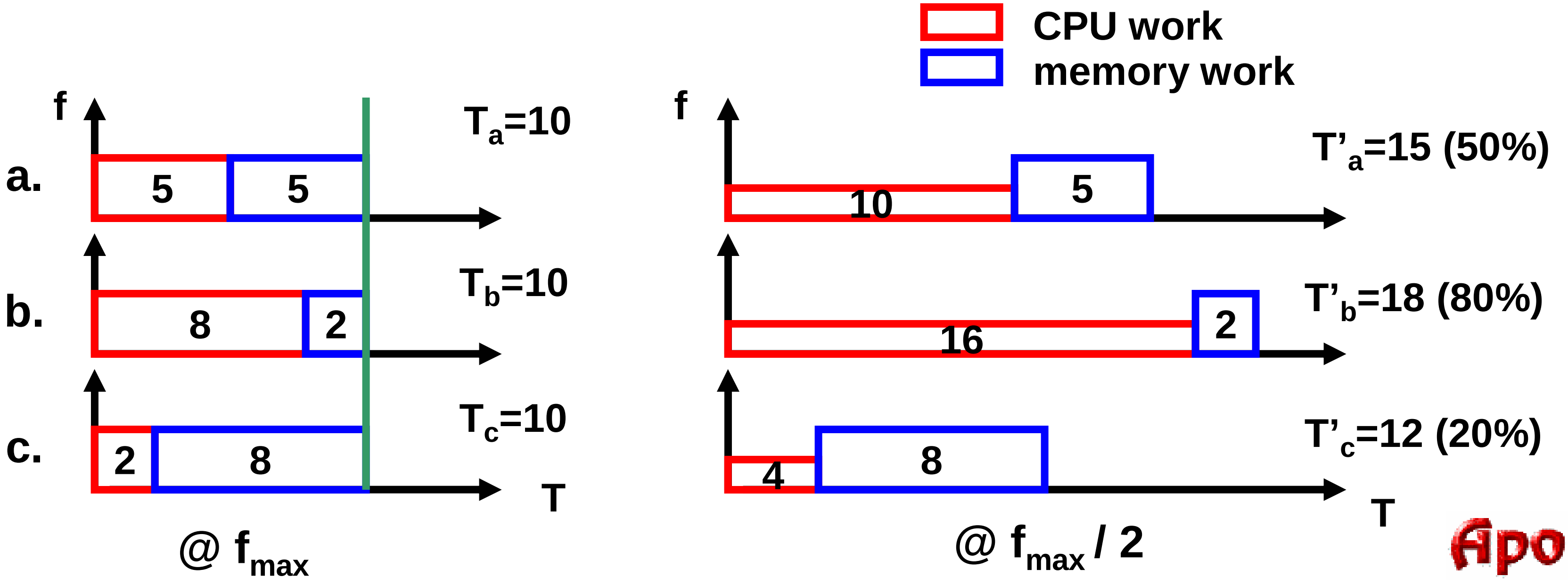
	BitsyX	AT2
Processor	Intel PXA255 Xscale @400MHz	Intel 80200 Xscale @733Mz
Companion chip	Intel StrongARM1111	Xilinx Virtex-E FPGA
Main memory	Onboard 32-bit 64MB SDRAM @100MHz	DIMM 64-bit 128MB SDRAM @100MHz
Flash memory	Onboard 32MB	Onboard 4MB
Frame buffer memory	Shared with main memory	Dedicated 16MB SDRAM
Display	6.4 inch 640x480 18-bit color TFT	10.4/6.4 inch 640x480 18-bit color TFT
Sound	AC97 codec	Crystal CS4630-CM PCI sound card
PCMCIA	1 type I and II	2 type I and II or 1 type III
USB	USB 1.1 host/client	USB 2.0 host
Input device	PS/2 keyboard, touch screen	USB keyboard, USB mouse, touch screen
Note	Not expandable	Expandable using PCI bus
Power consumption	9.4W (Measured, typical)	10.9W (typical with 6.4 inch display)

•Note1: BitsyX power consumption is 9.4W while that of AT2 is 10.9W
•Note 2: The backlight consumes 6.0W on BitsyX, whereas it consumes 2.7W on AT2
•Note 3: Performance of the AT2 is two times higher than that of BitsyX

Fine-Grained Dynamic Voltage and Frequency Scaling

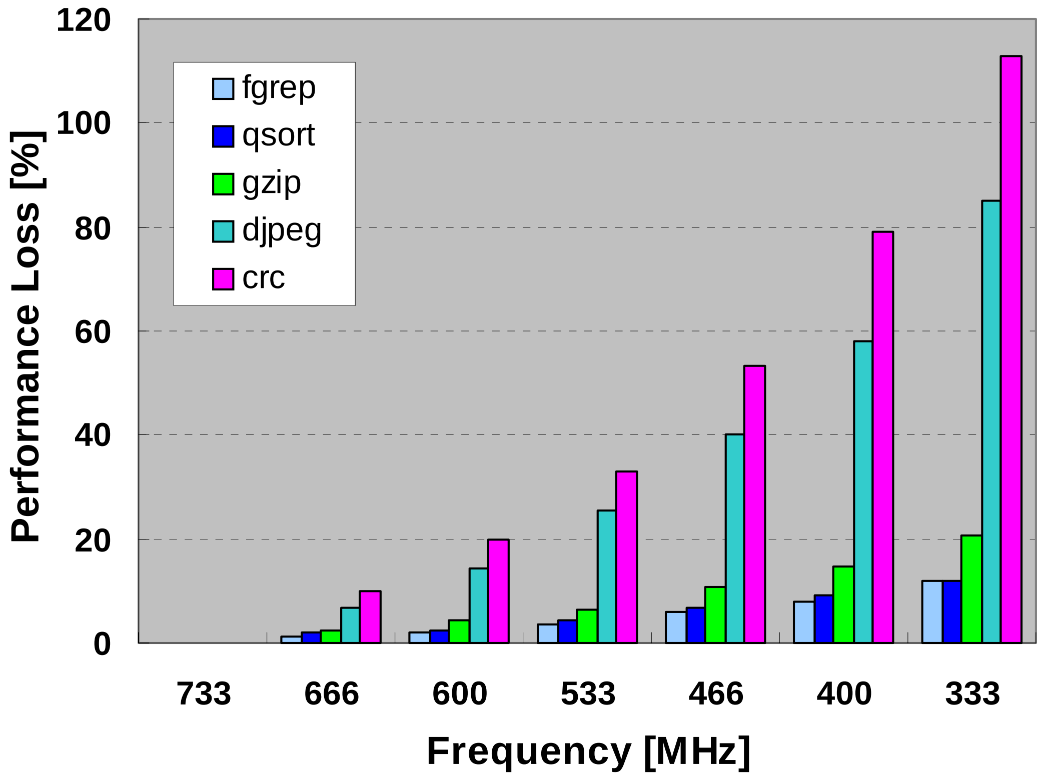
Energy and Performance Trade-off

- **Dynamic Voltage and Frequency Scaling (DVFS)**
  - Energy is saved at the cost of a delay increase
- **A program execution sequence consists of CPU and memory instructions**
- **The CPU has to stall until the external memory access is completed**
  - With DVFS, CPU energy is saved with little performance loss



Motivation

- Performance degradation at different frequencies



- More CPU energy savings is possible with a given performance loss target for memory-intensive applications

The Program Execution Time

- The amount of CPU and memory workload for an application program must be determined
- Execution time of a program is the sum of the On-chip (CPU work) and the Off-chip Latency (memory work)
  - $T = T_{onchip} + T_{offchip}$
- T_{onchip} : Varies with the CPU frequency
  - Cache hit
  - Stall due to data dependency
  - TLB hit, …
- $T_{offchip}$: Is invariant with the CPU frequency
  - Access to external memory such as SDRAM and frame buffer memory through the PCI, which is in turn due to a cache miss

Calculating the Target Frequency

- **T = T$_{onchip}$ + T$_{offchip}$**

$$T_{onchip} = \frac{\sum_{i=1}^{n} CPI_{onchip}^{i}}{f_{cpu}} = \frac{n \cdot CPI_{onchip}^{avg}}{f_{cpu}} \qquad T_{offchip} = \frac{\sum_{j=1}^{m} CPI_{offchip}^{j}}{f_{mem}} = T - T_{onchip}$$

n	: number of onchip instructions
CPI$_{onchip}$	: CPU clocks per instruction
f$_{cpu}$	: CPU clock frequency (variable)

m	: number of offchip events
CPI$_{offchip}$	: memory clocks per offchip event
f$_{mem}$	: memory clock frequency (fixed)

- **The β value of a program is defined as the ratio of T$_{onchip}$ to T$_{offchip}$ for that program. Given a performance loss factor, the target CPU frequency is calculated as:**

$$f_{target} = \frac{f_{\max}}{1 + PF_{loss} \cdot \left[1 + \beta \cdot \left(\frac{f_{\max}}{f_{cpu}} \right) \right]}$$

$$\begin{array}{lcl} PF_{loss} = 0 & \Rightarrow & f_{target} = f_{\max} \\ PF_{loss} \uparrow & \Rightarrow & f_{target} \downarrow \\ PF_{loss} \downarrow & \Rightarrow & f_{target} \uparrow \\ T_{offchip} \uparrow & \Rightarrow & f_{target} \downarrow \end{array}$$

Calculating the β Factor

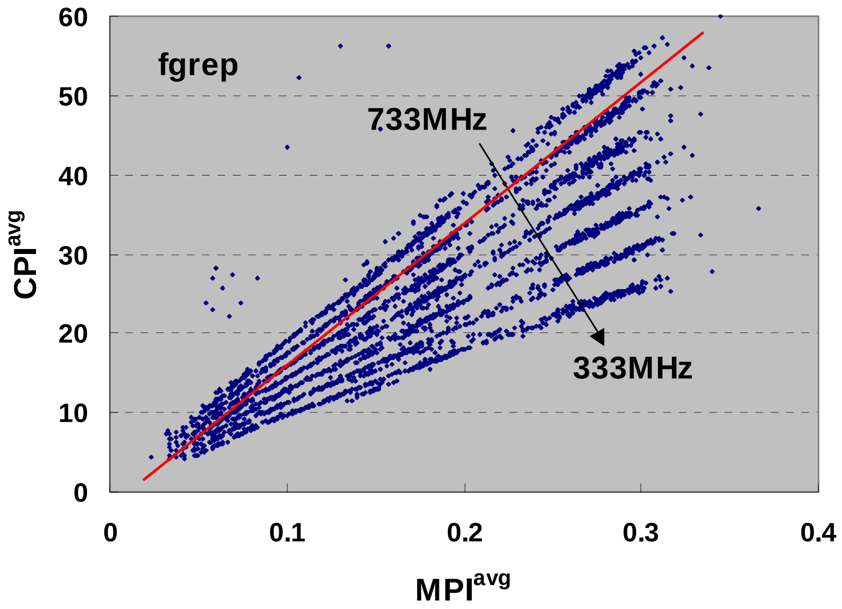
- **Must calculate the β Factor of a program**
  - Need to get T_{onchip} and $T_{offchip}$ values
  - T_{onchip} depends on f_{cpu}, n, and CPI_{onchip}
  - $T_{offchip}$ is calculated from T and T_{onchip}
- **Must calculate CPI_{onchip} and $T_{offchip}$**
- **Use the Performance Monitoring Unit**

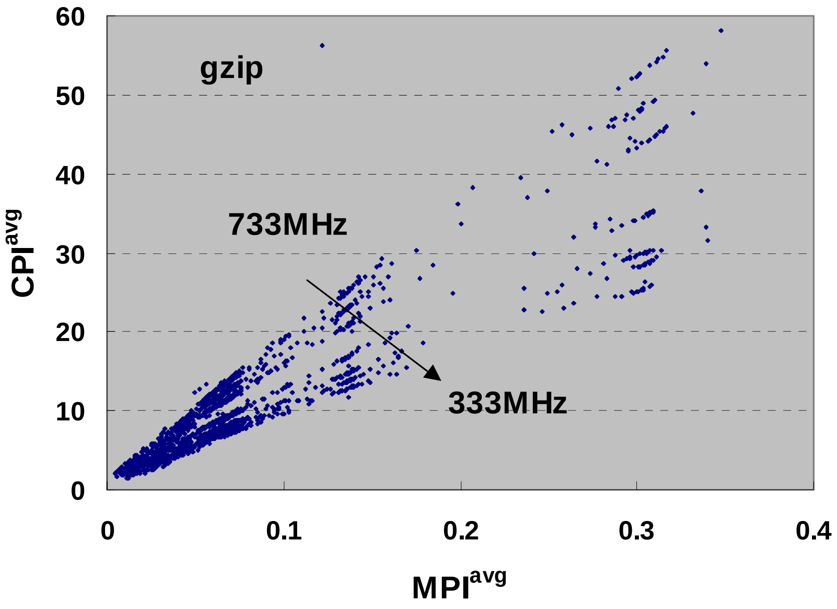
Performance Monitoring Unit (PMU)

- **PMU on the XScale chip can report up to 20 different dynamic events during execution of a program**
  - Cache hit/miss counts
  - TLB hit/miss counts
  - No. of external memory accesses
  - Total no. of instructions being executed
  - Branch misprediction count
  - Data stalls, ...
- **However, only two events can be monitored and reported at any given time**
- **For DVFS, we use PMU to generate statistics for**
  - Total no. of instructions being executed (n=INSTR)
  - No. of external memory accesses (m=MEM)
- **We also record the no. of clock cycles from the beginning of the program execution (CCNT)**

Plot of CPI vs. MPI

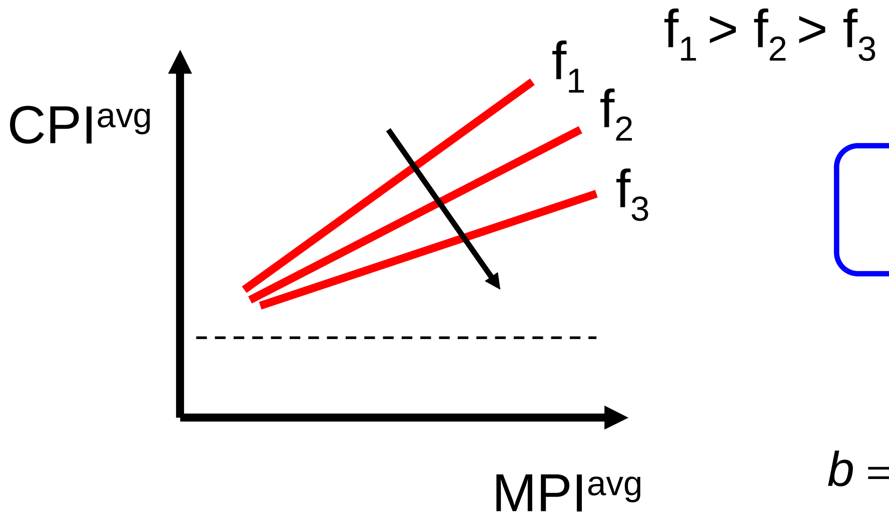
- PMU is read at every OS quantum (~50msec)
- We define MPI as the ratio of memory access count to the total instruction count
  - CPI^{avg} = CCNT / INSTR, during a quantum
  - MPI^{avg} = MEM / INSTR, during a quantum
- A plot of CPI^{avg} vs. MPI^{avg} with changing frequency is provided below





Regression Equation Modeling

- **A linear regression equation can be generated for each CPU clock frequency**



$$\boxed{CPI^{avg} = b(f)*MPI^{avg} + c}$$

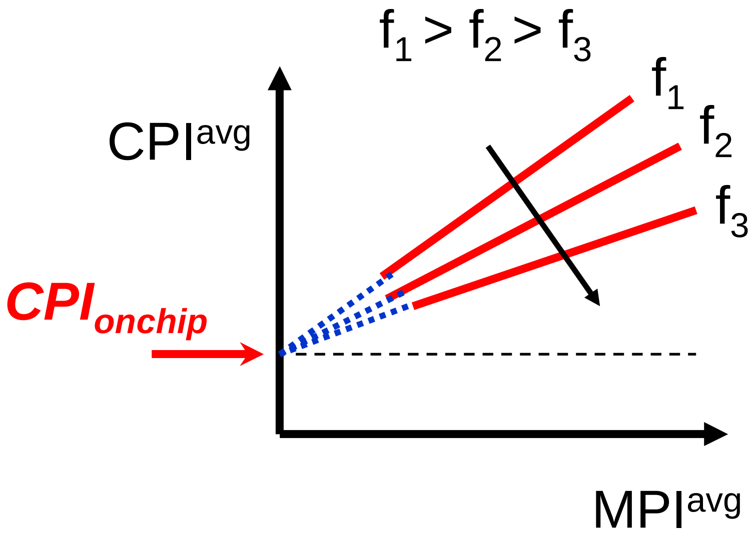
$$b = \frac{N\cdot(\sum_{i=t}^{t-N+1} x_i \cdot y_i) - (\sum_{i=t}^{t-N+1} x_i)\cdot(\sum_{i=t}^{t-N+1} y_i)}{N\cdot(\sum_{i=t}^{t-N+1} x_i{}^2) - (\sum_{i=1}^{t-N+1} x_i)^2},$$

$$c = \frac{\sum_{i=t}^{t-N+1} y_i}{N} - b\cdot\frac{\sum_{i=t}^{t-N+1} x_i}{N}$$

N : No. of regression points, e.g., 25
x_i : MPI^{avg} for the i^th point
y_i : CPI^{avg} for the i^th point

Calculating CPI_{onchip} and $T_{offchip}$

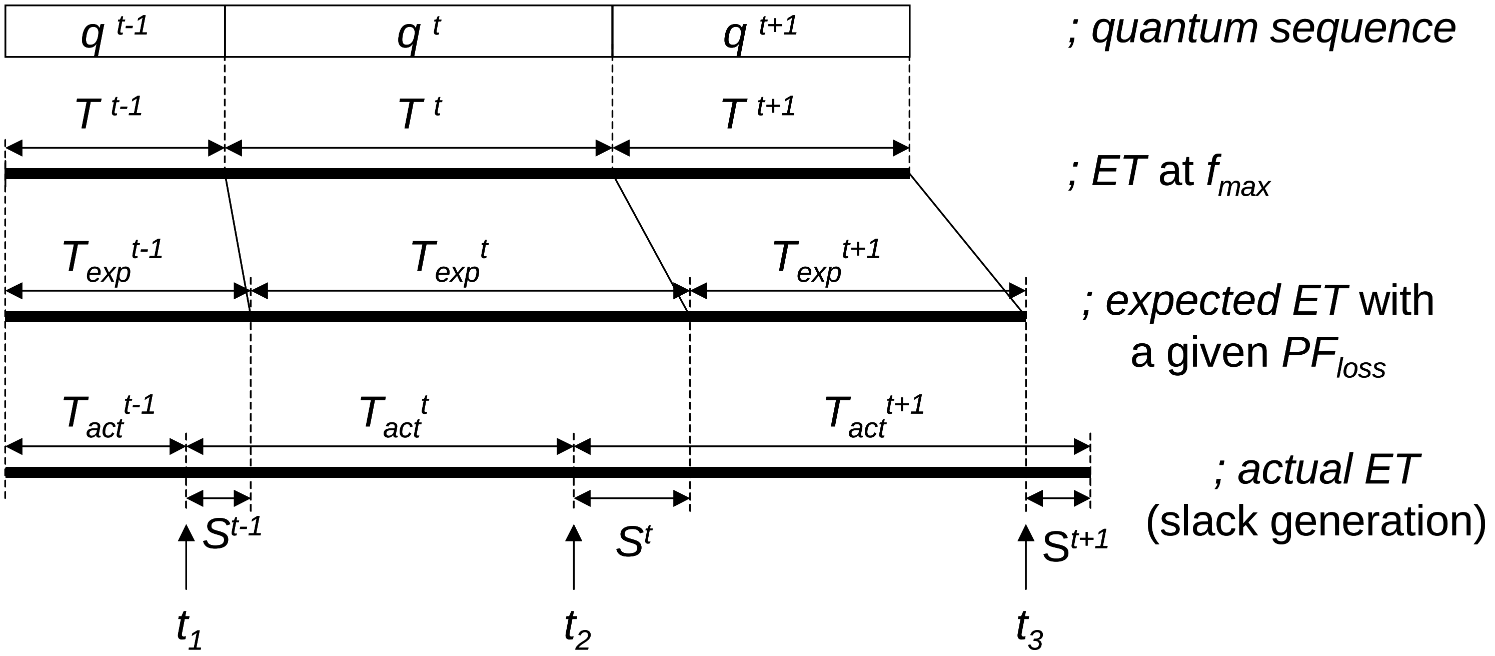
- **Notice that CPI_{onchip} denotes the CPI value without the offchip access; So it is equal to the y intercept of the CPI vs. MPI plot:**



- **We calculate $T_{offchip}$ directly as shown below:**
  - $T = T_{onchip} + T_{offchip} = CCNT/ f_{cpu}$
  - $T_{offchip} = CCNT/f_{cpu} - T_{onchip}$

Prediction Error Adjustment (I)

- **Error adjustment**



$$S^{t-1} = T_{exp}^{t-1} - T_{act}^{t-1}$$

$$\begin{aligned} S^{t} &= T_{ex[}^{t} + T_{exp}^{t-1} - T_{act}^{t} - T_{act}^{t-1} \\ &= T_{exp}^{t} - T_{act}^{t} + S^{t-1} \end{aligned}$$

$$\begin{aligned} S^{t+1} &= T_{exp}^{t+1} + T_{exp}^{t} + T_{exp}^{t-1} - T_{act}^{t+1} - T_{act}^{t} - T_{act}^{t-1} \\ &= T_{exp}^{t+1} - T_{act}^{t+1} + S^{t} \end{aligned}$$

ET : Execution time

$T_{exp}^{k} = T^{k} \bullet (1+PF_{loss})$
$(k = t\text{-}1, t, t\text{+}1)$

Prediction Error Adjustment (II)

- **Target frequency selection**
  - without error adjustment

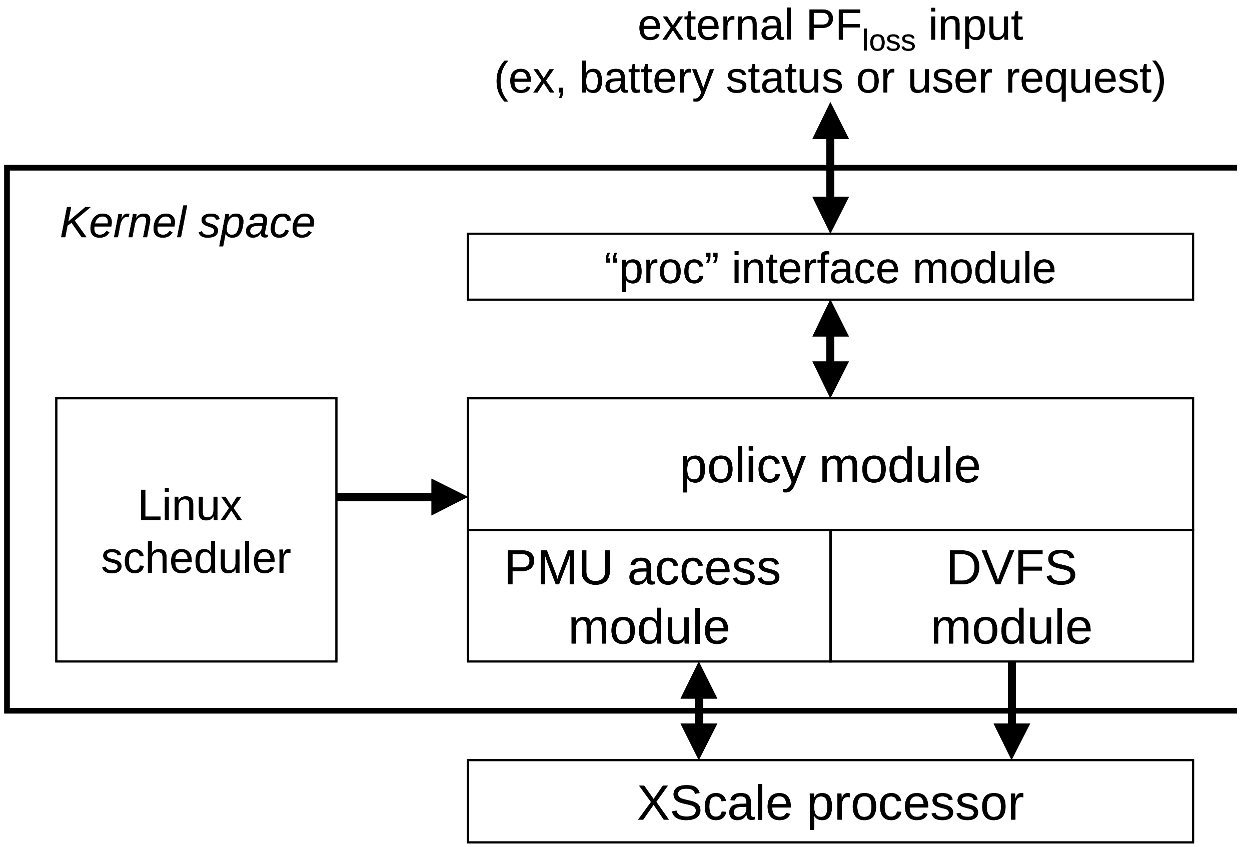
$$f^{t+1} = \frac{f_{\max}}{1 + PF_{loss} \cdot \left[1 + \beta^{t} \cdot \left(\frac{f_{\max}}{f_{cpu}} \right) \right]} \qquad \beta^{t} \equiv \frac{T^{t}{}_{offchip}}{T^{t}{}_{onchip}}$$

  - with error adjustment

$$f^{t+1} = \frac{f_{\max}}{1 + PF_{loss} \cdot \left[1 + \left(\beta^{t} + \frac{S^{t}}{PF_{loss} \cdot T^{t}_{act}} \right) \cdot \left(\frac{f_{\max}}{f^{t}} \right) \right]}$$

Implementation (I)

- **Offchip Latency-driven DVFS (OL-DVFS)**
  - Software architecture



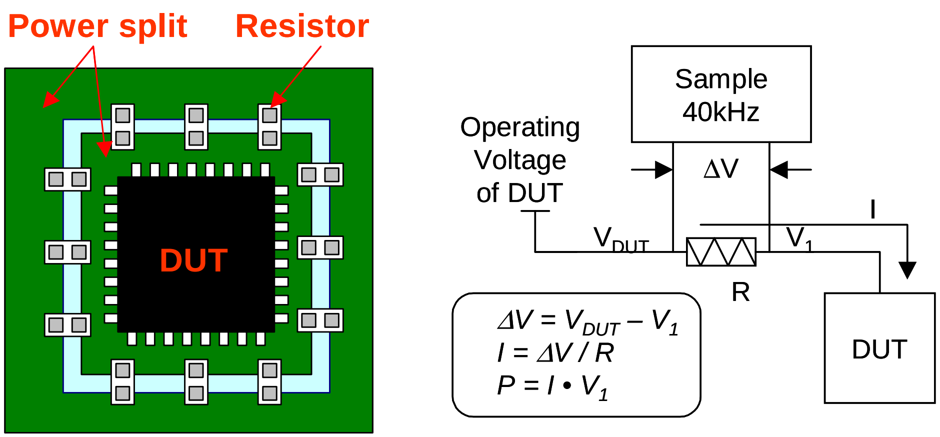
Implementation (II)

- A voltage is mapped to each CPU frequency
- Voltage control circuitry is on-board
- Power measurement with DAQ (Data Acquisition)

CPU Freq. vs. Volt. Relation

Frequency (MHz)	Voltage (V)
333	0.91
400	0.99
466	1.05
533	1.12
600	1.19
666	1.26
733	1.49

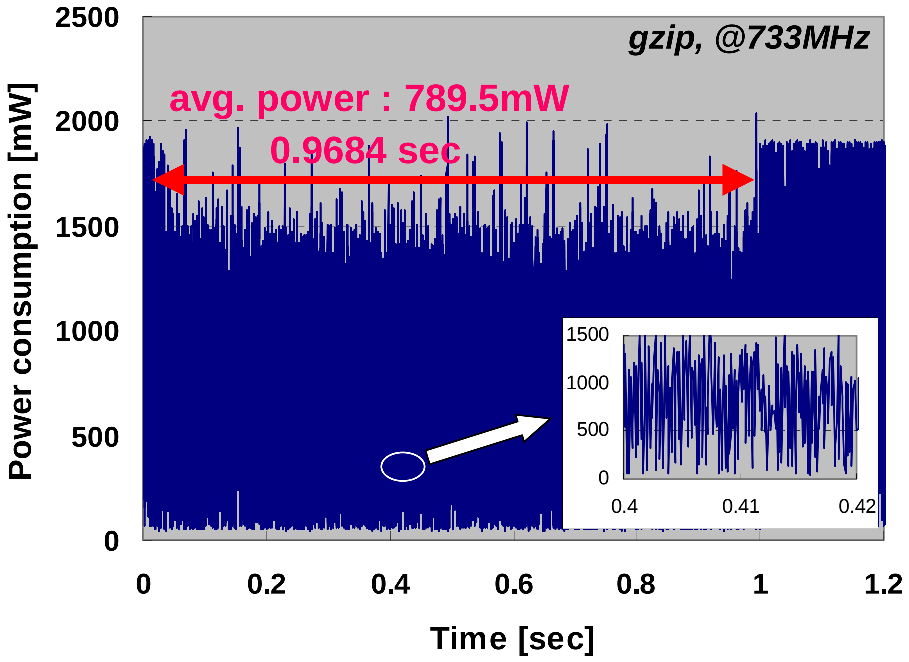
Data Acquisition system



Experimental Results (I)

- **Power consumption vs. performance degradation**

without OL-DVFS

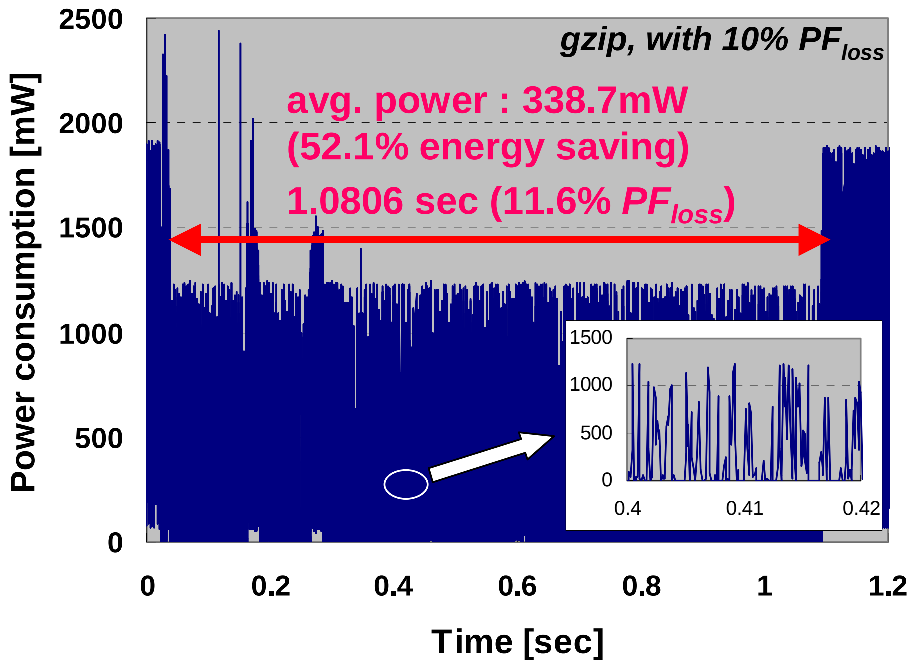


gzip, @733MHz

avg. power : 789.5mW

0.9684 sec

with OL-DVFS



gzip, with 10% PF_{loss}

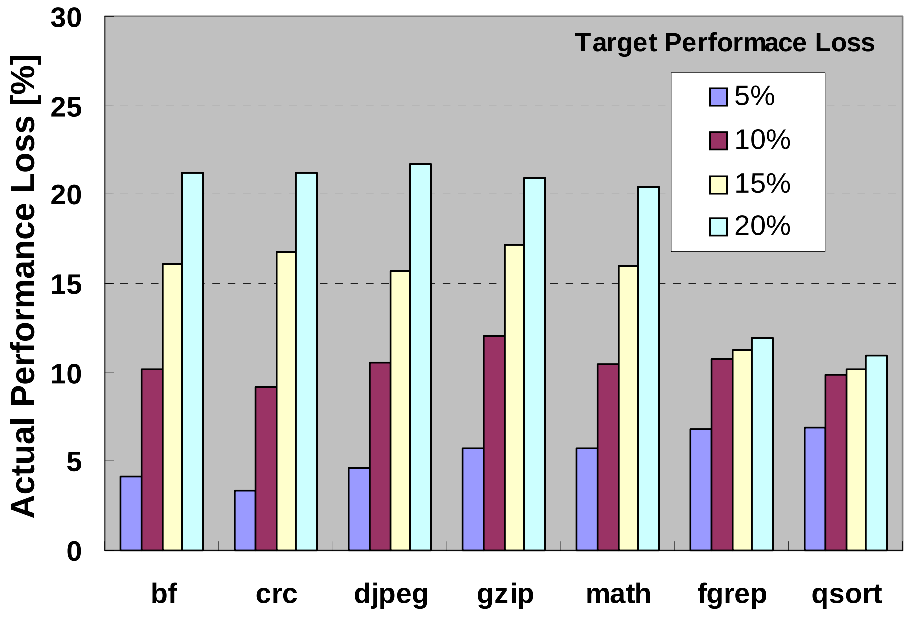
avg. power : 338.7mW

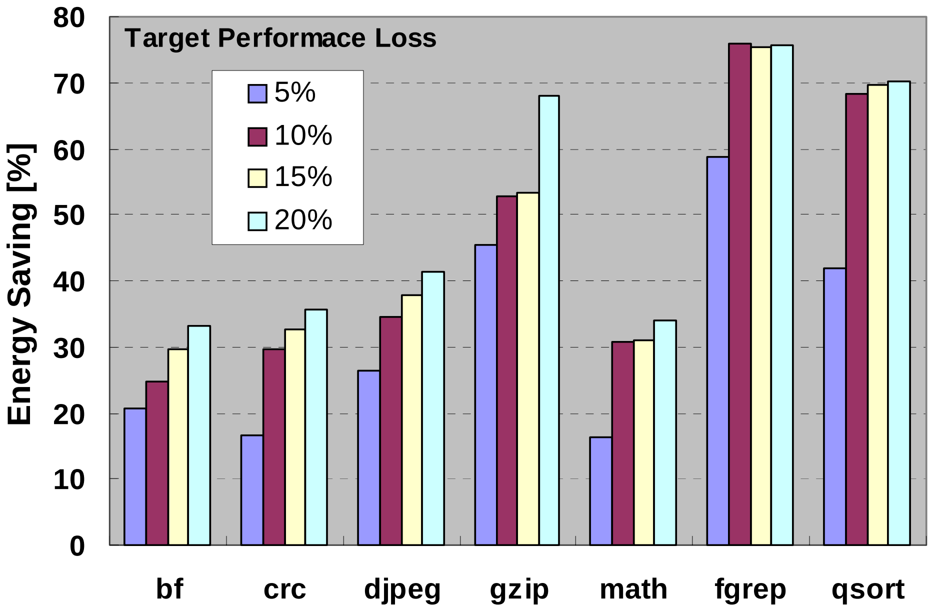
(52.1% energy saving)

1.0806 sec (11.6% PF_{loss})

Results (II)

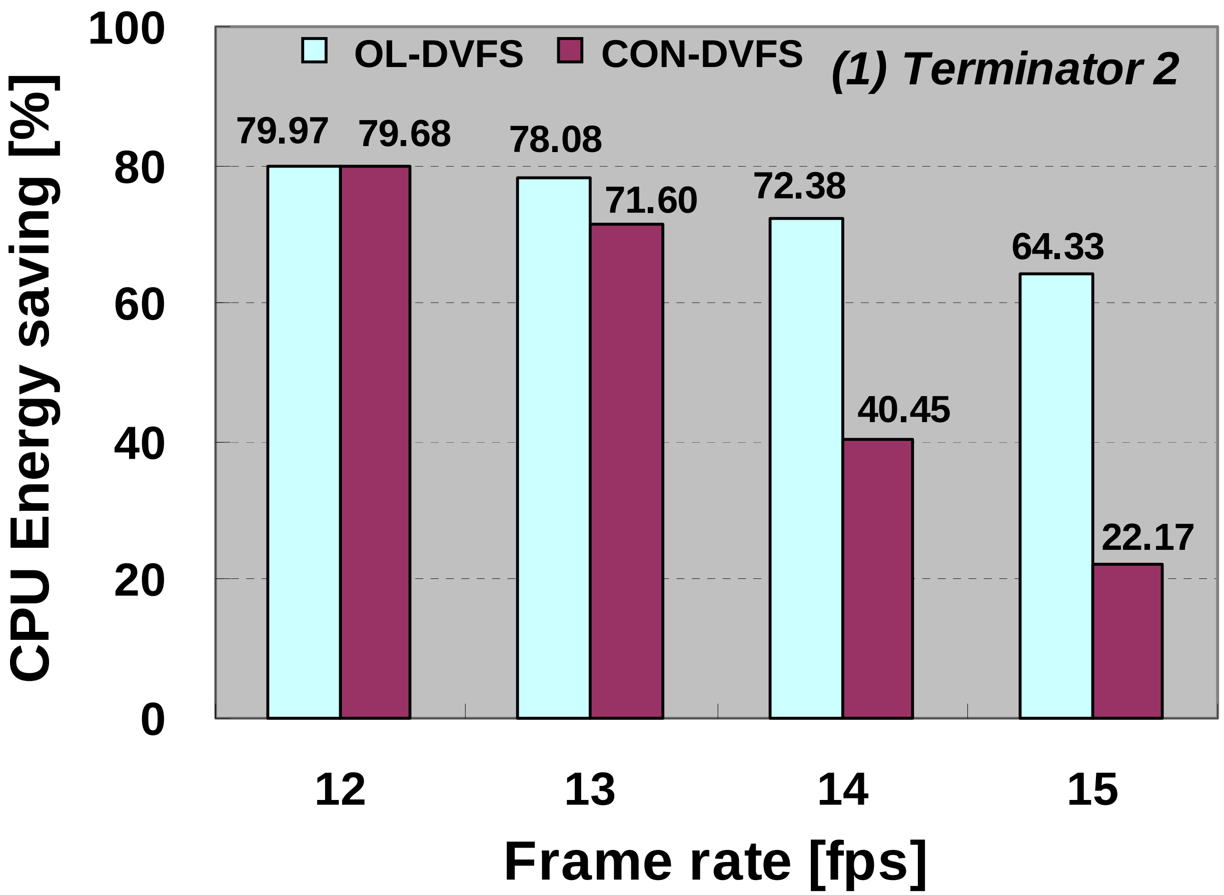
- **Measured PF_{loss} with a variable performance loss target ranging from 5% to 20%**





Results (III)

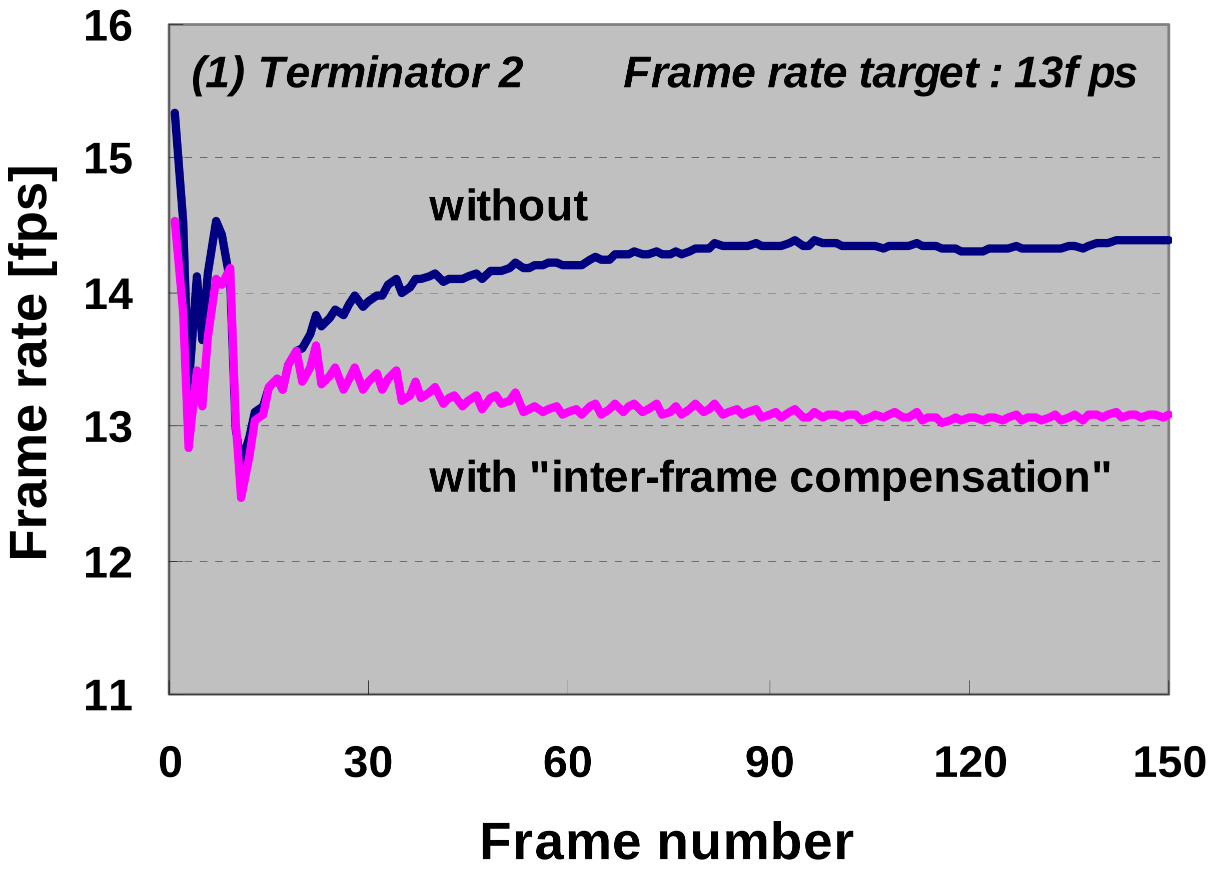
- MPEG2 video playback



OL-DVFS : off-chip latency driven DVFS
CON-DVFS : conventional DVFS

Results (IV)

- Effect of the error compensation method

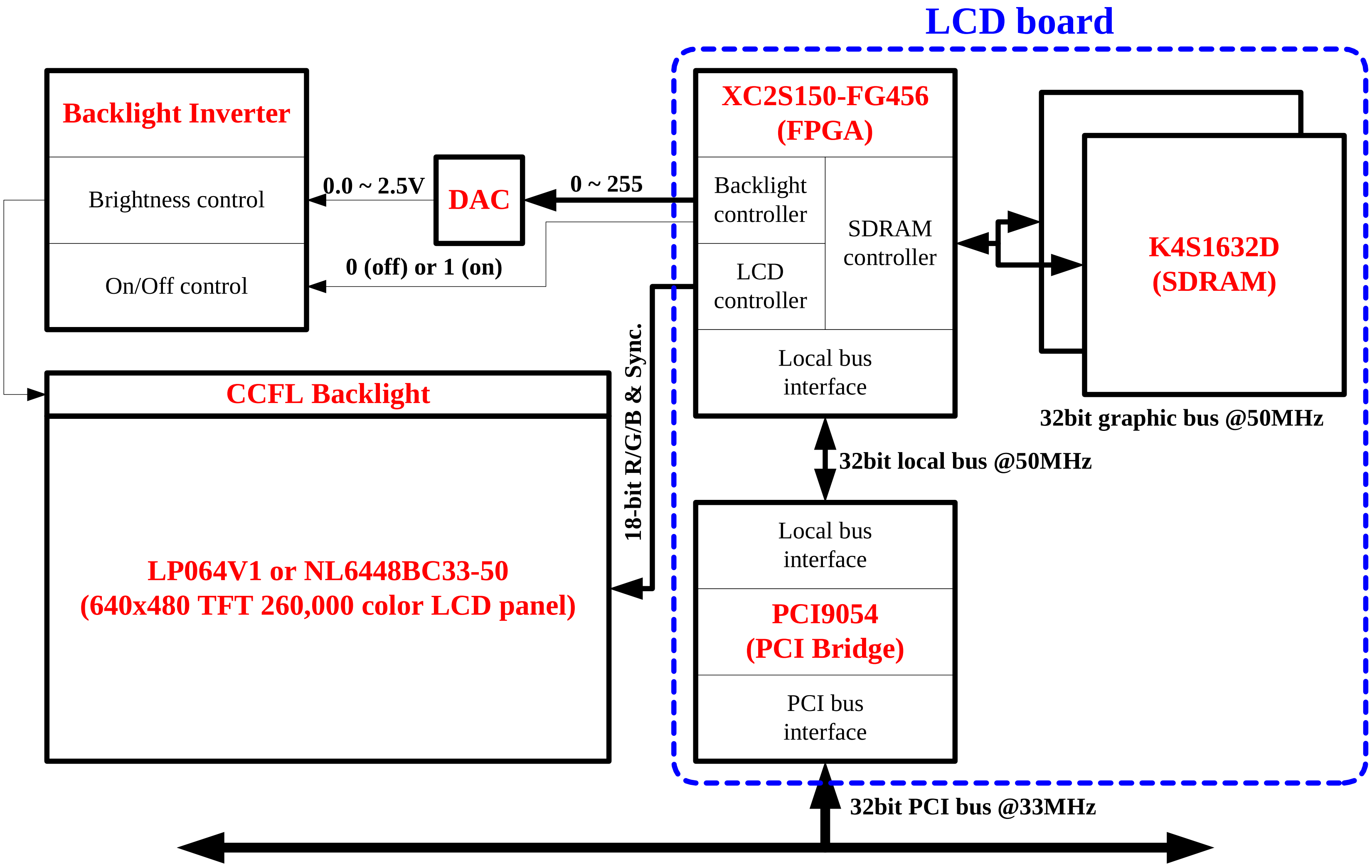


OL-DVFS on BitsyX Platform

- **PXA255 processor specification**
  - Frequency : 100MHz ~ 400MHz
  - Voltage : 0.85V ~ 1.3V
- **Fewer PMU events compared to the XScale processor**
  - MEM is not available
  - Cache miss count can be an alternative
- **Lack of information about hardware for DVFS**
  - On-board DC-DC converter for variable voltage generator
  - Voltage control scheme using I^2C
- **Experimental results**
  - System power consumption is lower by 144mW at 200MHz compared to that at 400MHz. This is however less than the value of 233mW saving reported in the data sheet
  - Unstable; it sometimes hangs after voltage scaling
- **Ongoing work**
  - Fine tuning of voltage level without causing any system crash
  - Check the compatibility of Cache miss count for OL-DVFS

Power Reduction in the Display System

Display Architecture



Display Specification

- **LCD board specification**
  - PCI9054 PCI bridge (PLX)
    - 32bit 33MHz PCI bus interface
    - 32bit 50MHz local bus interface
  - XC2S150 FPGA LCD controller (Xilinx)
    - Up to 150k gates logic capacity
    - SDRAM controller for frame buffer memory
    - LCD timing generator for LCD panel signals
      - Vsync, Hsync, Dtmg and so on
    - Backlight brightness controller
      - With DAC (Digital Analog Converter)
  - K4S1632D SDRAM frame buffer (Samsung)
    - 16MBytes capacity

Display Specification (cont'd)

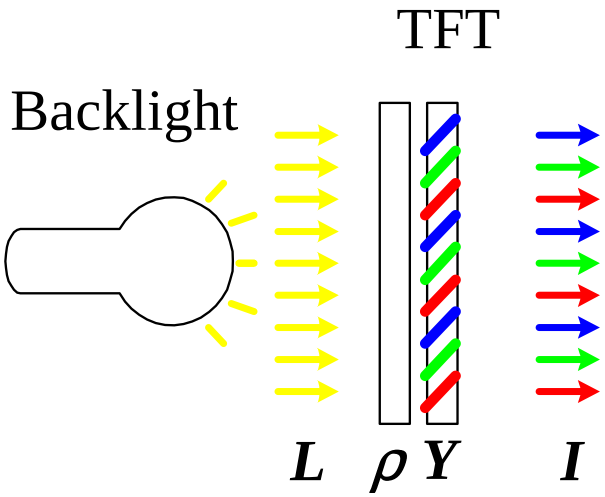
- **LCD panel specification**
  - LP064V1 (LG-Philips)
    - 6.4 inch 640x480 VGA standard resolution
    - 260,000 colors (18-bit RGB)
    - Transmissive type with CCFL backlight
  - NL6448BC33-50 (NEC)
    - 10.4 inch 640x480 VGA standard resolution
    - 260,000 colors (18-bit RGB)
    - Transflective type with CCFL backlight

Dynamic Backlight Luminance Scaling

- **Perceived intensity *I* by human eyes**

$$I = \rho \times L \times Y$$

  - ρ : the transmittance of the LCD panel
  - L : the luminance of the backlight
  - Y : the brightness of the image



DBLS Screen Shots

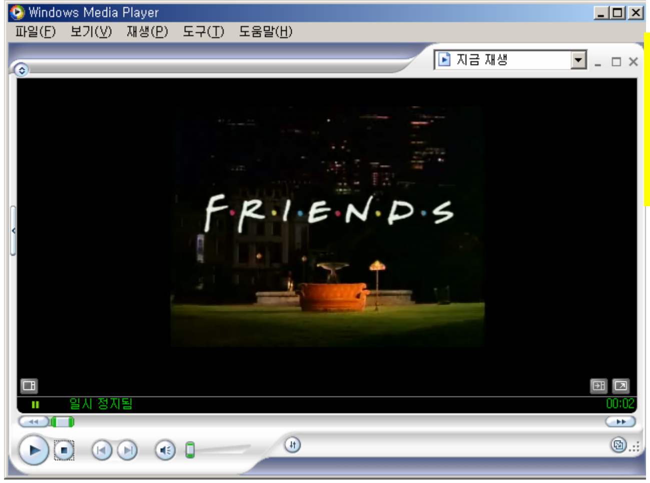
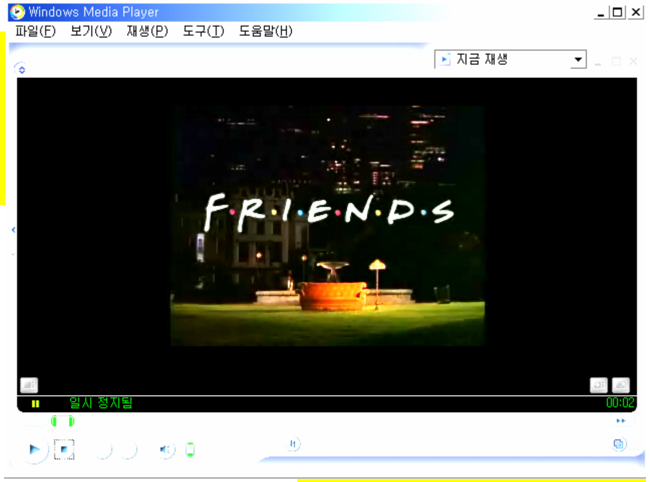
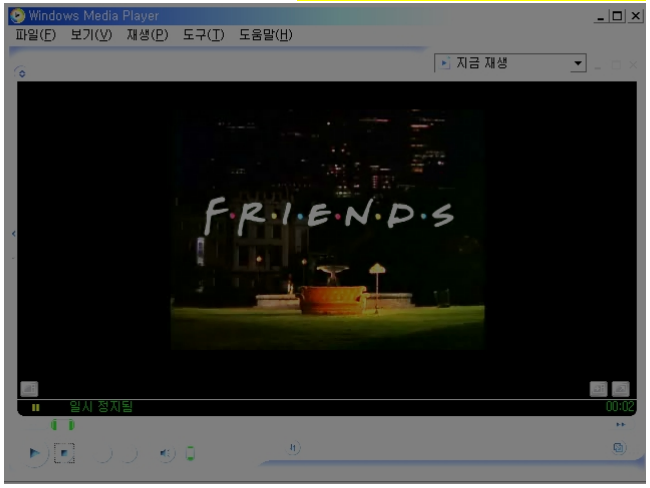


Image brightness Doubling
$Y' = Y/2$



Backlight Dimming by half
$L' = L/2$

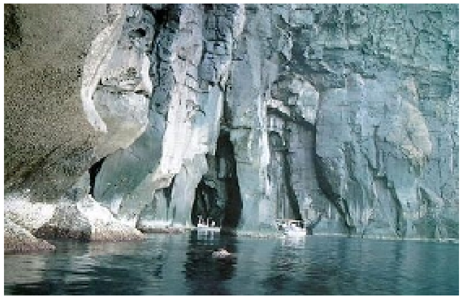


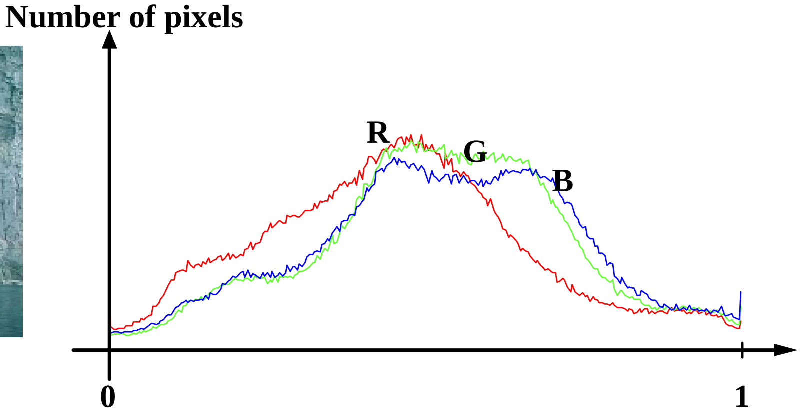
30% of the pixels are saturated after image brightness enhancement

DBLS (cont'd)

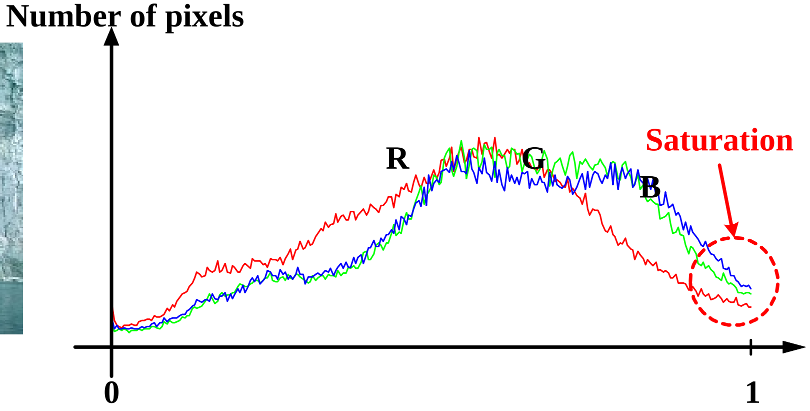
- Most of the overhead is caused by
  - Heavy frame buffer traffic for frequent frame buffer read
    - R/G/B spectrum analysis of the current image on the LCD panel
  - Heavy frame buffer traffic for frequent frame buffer write
    - Frame buffer contents updated with the corresponding brightness enhanced image

Original

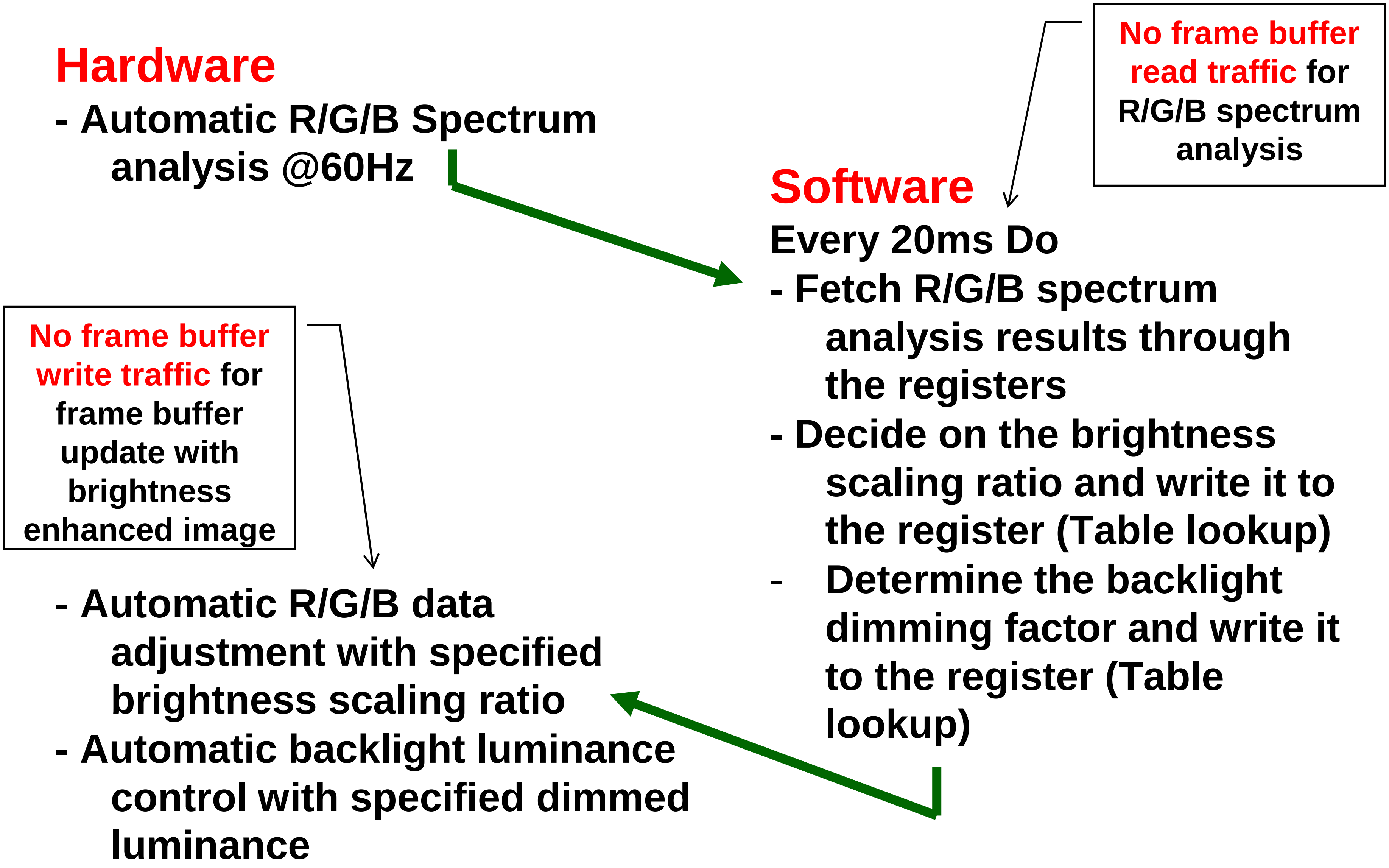




Brightness enhanced



Hardware-aided DBLS



Hardware

- Automatic R/G/B Spectrum analysis @60Hz

No frame buffer read traffic for R/G/B spectrum analysis

Software

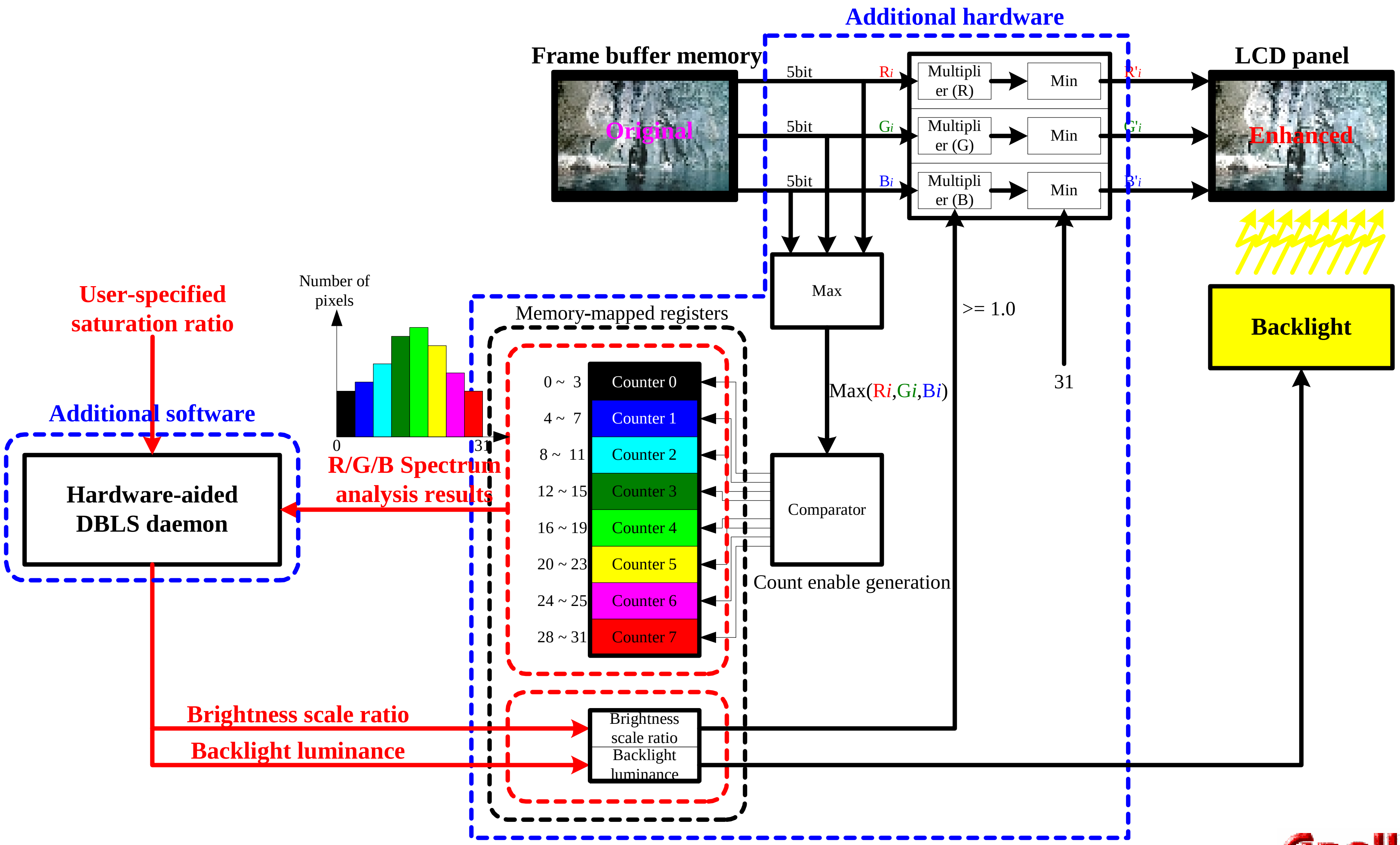
Every 20ms Do

- Fetch R/G/B spectrum analysis results through the registers
- Decide on the brightness scaling ratio and write it to the register (Table lookup)
- Determine the backlight dimming factor and write it to the register (Table lookup)

No frame buffer write traffic for frame buffer update with brightness enhanced image

- Automatic R/G/B data adjustment with specified brightness scaling ratio
- Automatic backlight luminance control with specified dimmed luminance

Hardware-aided DBLS (cont'd)



Additional hardware

Frame buffer memory

Original

5bit

5bit

5bit

R_i

G_i

B_i

Multiplier (R)

Multiplier (G)

Multiplier (B)

Min

Min

Min

R'_i

G'_i

B'_i

LCD panel

Enhanced

Backlight

Max

>= 1.0

31

Max(R_i,G_i,B_i)

Comparator

Count enable generation

Memory-mapped registers

0 ~ 3	Counter 0
4 ~ 7	Counter 1
8 ~ 11	Counter 2
12 ~ 15	Counter 3
16 ~ 19	Counter 4
20 ~ 23	Counter 5
24 ~ 25	Counter 6
28 ~ 31	Counter 7

Brightness scale ratio

Backlight luminance

Number of pixels

0

31

R/G/B Spectrum analysis results

User-specified saturation ratio

Additional software

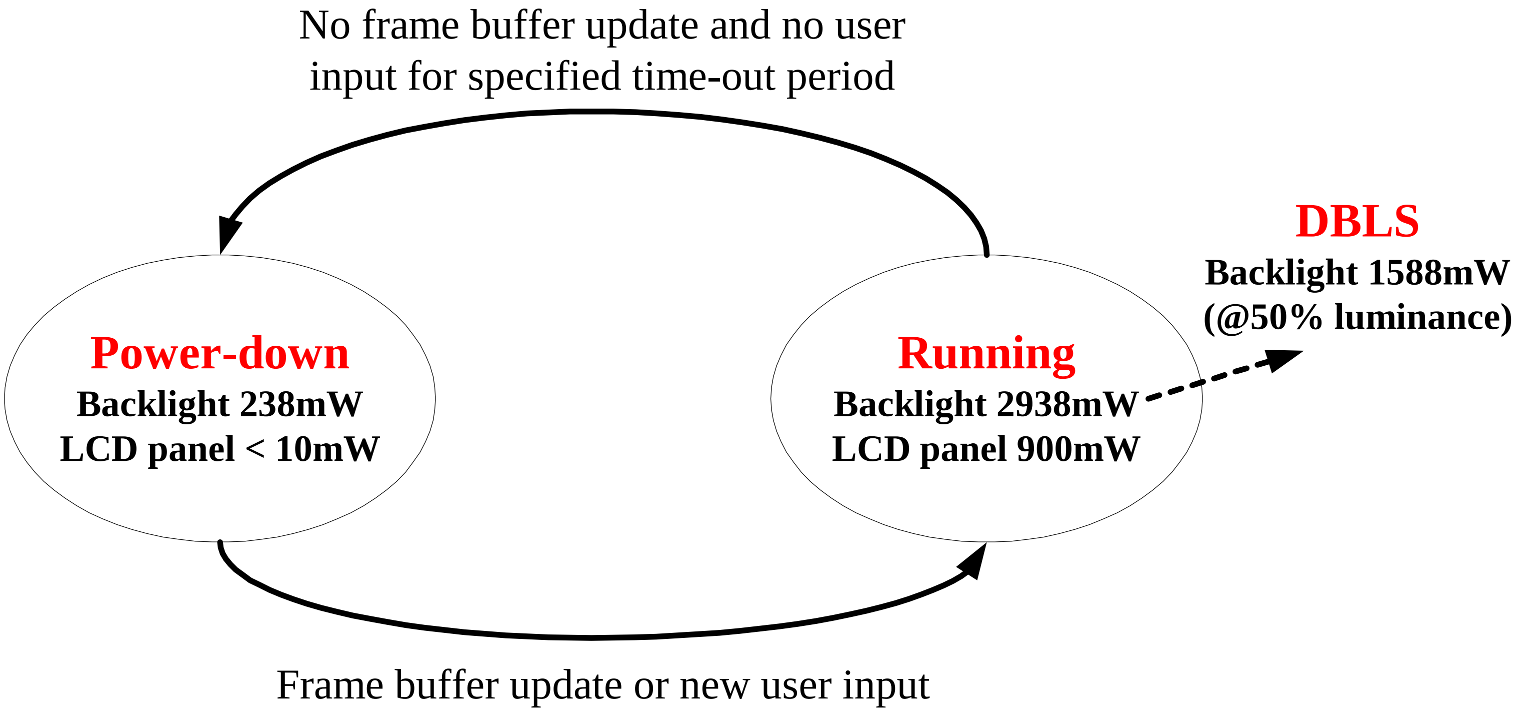
Hardware-aided DBLS daemon

Brightness scale ratio

Backlight luminance

Power Consumption Results

- **Transmissive LCD display system**



No frame buffer update and no user input for specified time-out period

Power-down
Backlight 238mW
LCD panel < 10mW

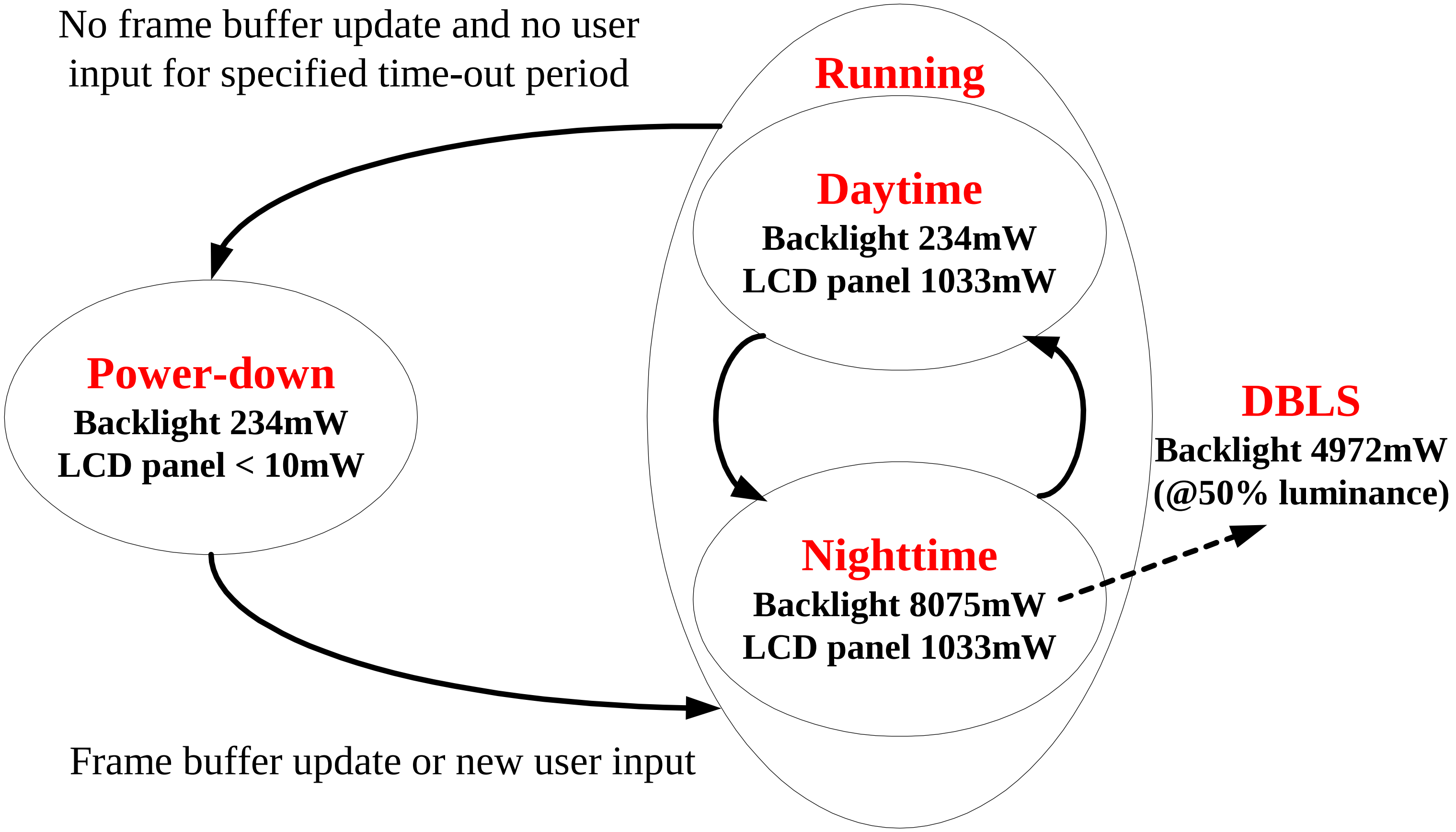
Running
Backlight 2938mW
LCD panel 900mW

DBLS
Backlight 1588mW
(@50% luminance)

Frame buffer update or new user input

Power Consumption Results (cont'd)

- **Transflective LCD display system**



No frame buffer update and no user input for specified time-out period

Running

Daytime
Backlight 234mW
LCD panel 1033mW

Power-down
Backlight 234mW
LCD panel < 10mW

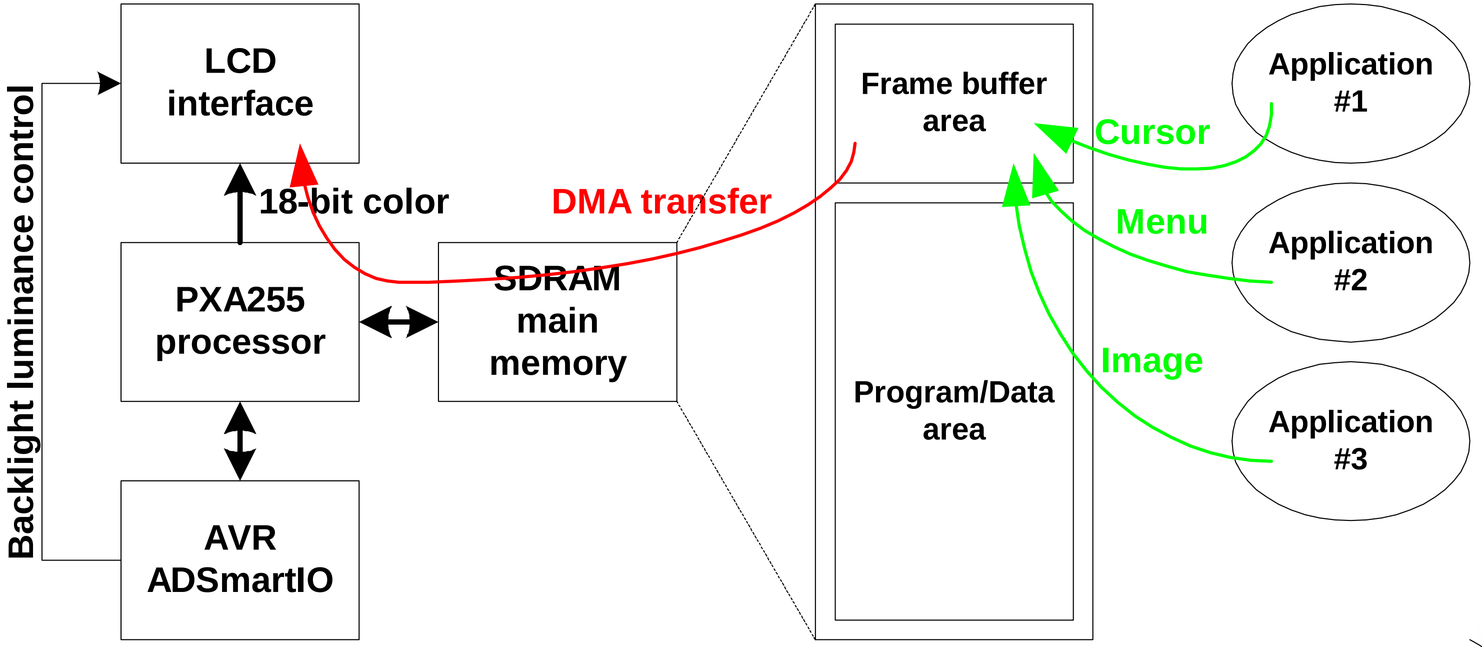
DBLS
Backlight 4972mW
(@50% luminance)

Nighttime
Backlight 8075mW
LCD panel 1033mW

Frame buffer update or new user input

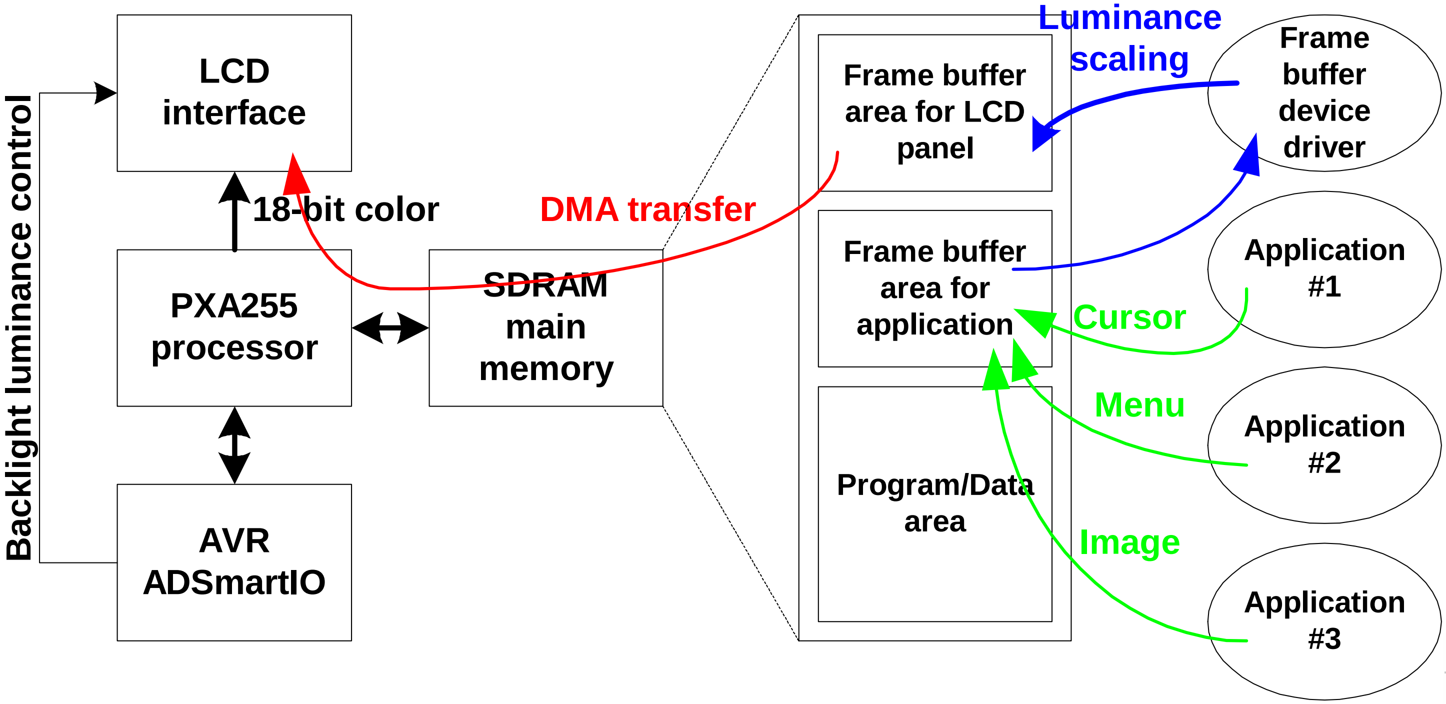
DBLS on BitsyX

- **Display system on BitsyX**
  - Some portion of the SDRAM main memory is reserved as the frame buffer memory
  - DMA controller prefetches the graphics data to periodically refresh the LCD panel
  - ADSmartIO AVR processor is used to control the backlight luminance of the LCD panel



DBLS on BitsyX (cont'd)

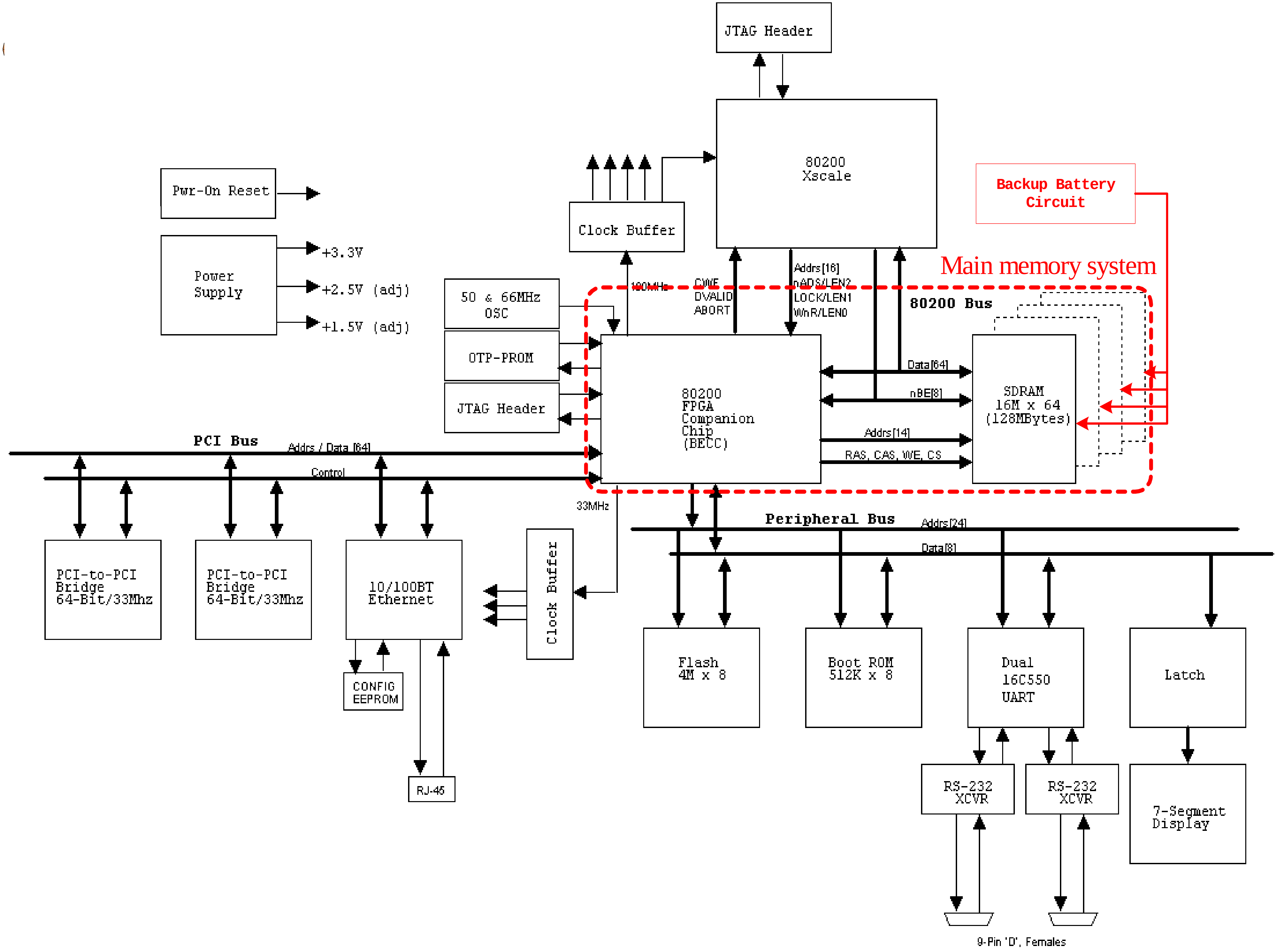
- **Our approach**
  - Allocate one more frame buffer area in the memory
    - Separate frame buffer areas for the LCD panel and the applications
    - Applications update rate of the frame buffer are rather low
  - Modify frame buffer device driver
    - Periodically update the frame buffer area for the LCD panel with luminance scaled image derived from the frame buffer area for application (update frequency in 1-20 Hz range)
    - Simultaneously dim the backlight



Power Management of the Main Memory System

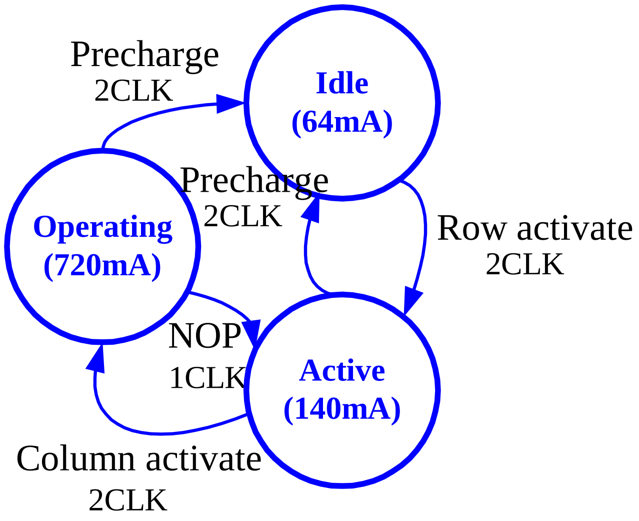
Main Memory System Specification

- **New XScale board's main memory system specification**
  - SDRAM specification
    - Four K4S561632 SDRAM chips (Samsung)
      - 64-bit data bus width
      - Operating @100MHz
  - Memory controller specification
    - XCV200E FPGA (Xilinx)
      - Up to 200k gates
      - Operating @100MHz
      - Active-page control for SDRAM devices
    - Memory controller is implemented with a FPGA and we can modify its functionality by rewriting and recompiling its VHDL codes


JTAG Header
80200
Xscale
Backup Battery
Circuit
Main memory system
Pwr-On Reset
Clock Buffer
Power
Supply
+3.3V
+2.5V (adj)
+1.5V (adj)
100MHz
CWE
DVALID
ABORT
Addrs[16]
nADS/LEN2
LOCK/LEN1
WnR/LEN0
80200 Bus
50 & 66MHz
OSC
OTP-PROM
JTAG Header
80200
FPGA
Companion
Chip
(BECC)
Data[64]
nBE[8]
Addrs[14]
RAS, CAS, WE, CS
SDRAM
16M x 64
(128MBytes)
PCI Bus
Addrs / Data [64]
Control
33MHz
Peripheral Bus
Addrs[24]
Data[8]
PCI-to-PCI
Bridge
64-Bit/33Mhz
PCI-to-PCI
Bridge
64-Bit/33Mhz
10/100BT
Ethernet
Clock Buffer
CONFIG
EEPROM
RJ-45
Flash
4M x 8
Boot ROM
512K x 8
Dual
16C550
UART
Latch
RS-232
XCVR
RS-232
XCVR
7-Segment
Display
9-Pin 'D', Females

Main Memory Controller

- **SDRAM state diagram**
  - When all banks (4 banks) are closed, SDRAM is in the IDLE state
  - When there is any activated bank (open row), SDRAM is in the ACTIVE state
  - When memory controller is reading from or writing to SDRAM, SDRAM is in the OPERATING state



Memory controller command

Row activate: Activate selected bank by BA[1:0]

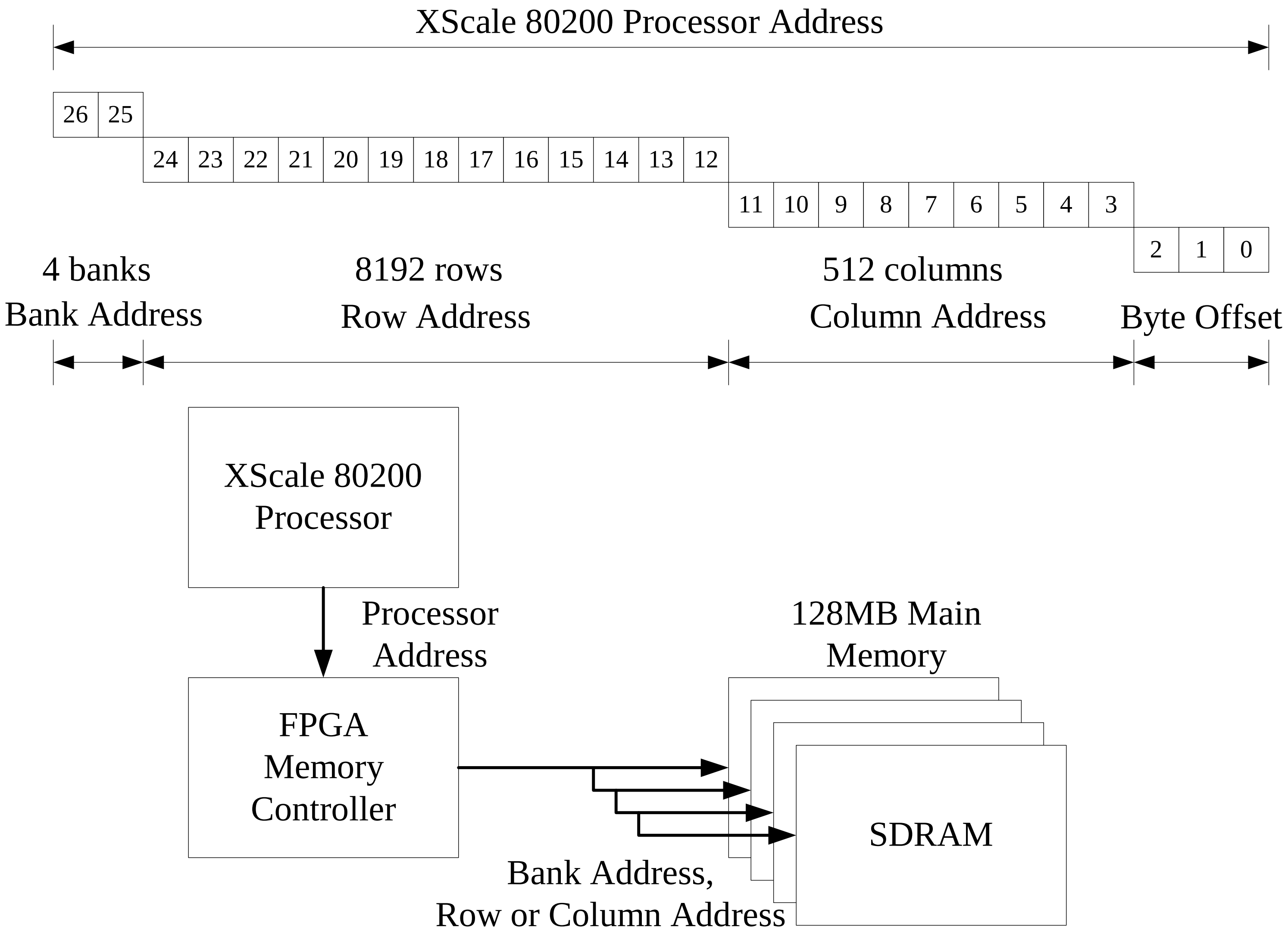
Precharge: Close selected bank by BA[1:0] or all the banks

Column activate: Read from SDRAM or write to SDRAM

NOP: No operation

Main Memory Controller (cont’d)

- **Address mapping**



XScale 80200 Processor Address

26 25	24 23 22 21 20 19 18 17 16 15 14 13 12	11 10 9 8 7 6 5 4 3	2 1 0
4 banks Bank Address	8192 rows Row Address	512 columns Column Address	Byte Offset

Main Memory Controller (cont'd)

- **Auto-precharge control**
  - Operation
    - Typical SDRAM access requires a row-address issue followed by a column address issue
    - After a burst access, the controller closes the row
      - IDLE → ACTIVE → OPERATING → IDLE
  - Advantage
    - Low standby power consumption
  - Disadvantage
    - No performance gain

Main Memory Controller (cont’d)

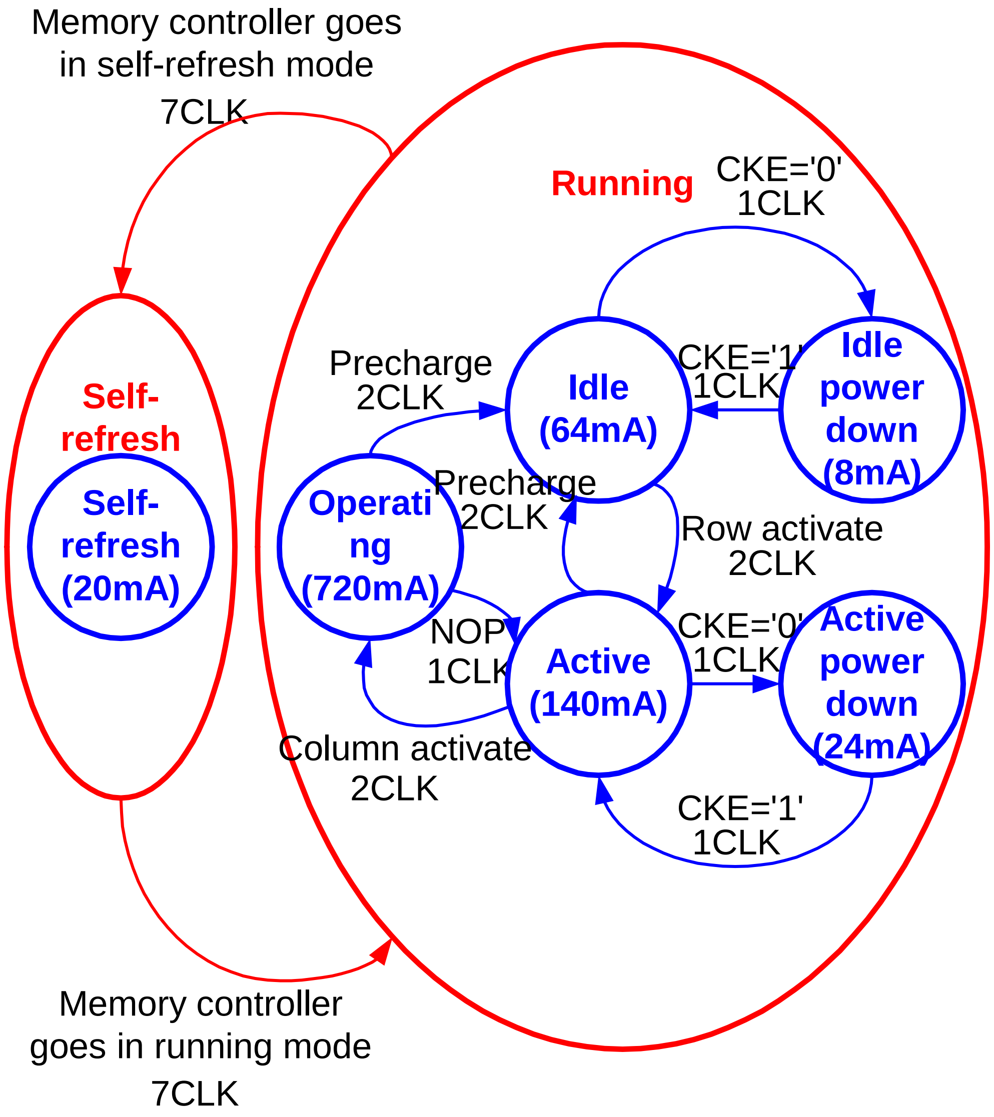
- **Active-page control**
  - Operation
    - After a burst access, SDRAM can remain in the ACTIVE state
      - IDLE → ACTIVE → OPERATING → ACTIVE
    - If the row address of the next access is equal to the current one (case of a row hit)
      - ACTIVE → OPERATING → ACTIVE
    - Else (case of a row miss), else open row must be closed so that a new row can be opened
      - ACTIVE → IDLE → ACTIVE → OPERATING → ACTIVE
  - Advantage
    - Performance gain due to row-hit if row-hit ratio is over 50%
  - Disadvantage
    - High standby power consumption

Power management

- **Running mode**
  - W/o the power-down mode enable (Normal)
  - W/ the power-down mode enable
    - Sends SDRAM's to power-down state for power-reduction when there is no pending memory request
- **Self-refresh mode**
    - For data retention with secondary power source such as a backup battery
    - For fast boot support

Memory controller

Main memory device state/power diagram (@3.3V)



↙ Transition by power management software

↙ Transition by memory controller

○ Memory controller state

○ Memory device state

Memory controller state

Running: System is running normally

Self-refresh: System is disconnected from power source

Memory device state

Idle: All banks are closed.

Active: There is at least one opened bank.

Operating: Memory controller is reading data from the bank or writing data to the bank.

Idle power down: Memory controller is deasserting CKE (clock enable) signal in Idle state.

Active power down: Memory controller is deasserting CKE signal in Active state.

Self-refresh: Memory device is internally performing refresh operation by itself.

Power Consumption Results

- Power consumption of SDRAM devices in the new main memory system

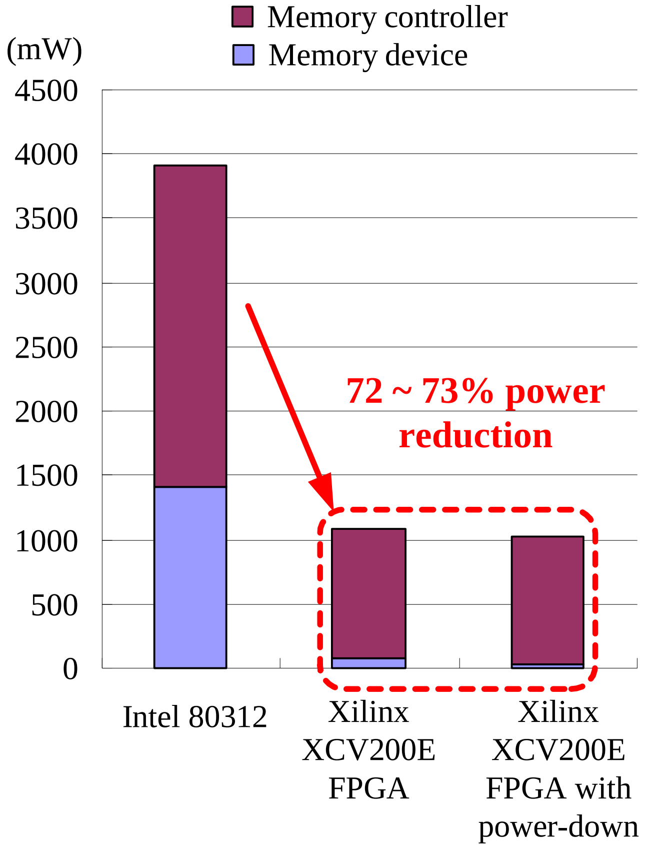
Mode			Power (mW)	Ratio (%)	Note
Self-refresh			13	100.0	
Running	Idle	Normal	83	100.0	
		Power-down enable	26	68.7	
	Busy	Normal	389	100.0	18.0 fps
		Power-down enable	297	23.7	17.8 fps

*** Note that previous main memory power consumption always exceeds 1.4W**

- Power consumption of main memory controller
  - Intel 80312 memory controller consumes 2.5W
  - New FPGA memory controller consumes 1.0W

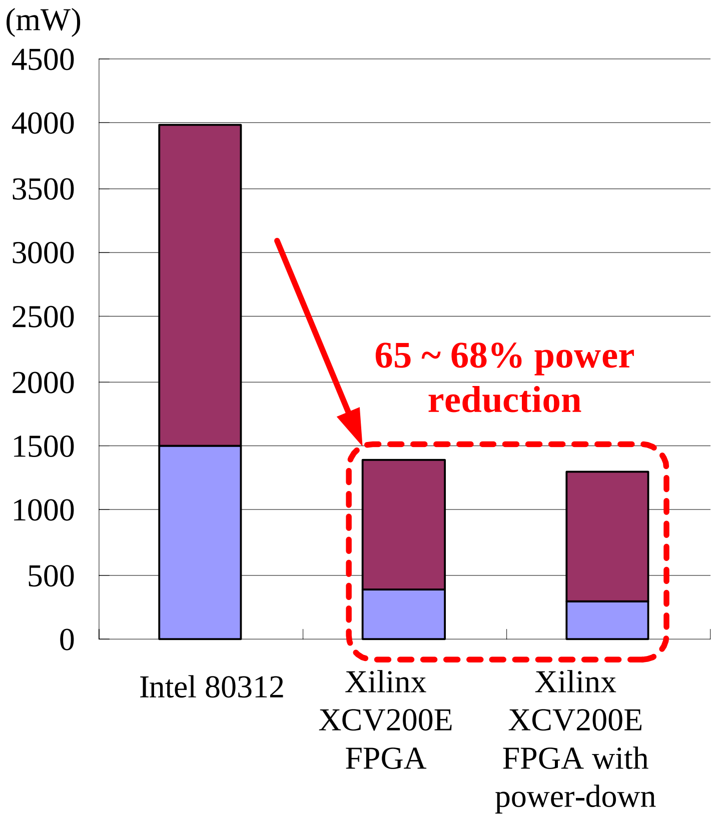
Power Consumption Results (cont’d)

- **Power consumption of the main memory system (Memory controller + SDRAM devices)**



72 ~ 73% power reduction

When system is idle



65 ~ 68% power reduction

When system is busy (MPEG2)

Memory Management in AT2

- **Automatic switching between auto-precharge and active-page mode of the memory controller**
  - Most row-hit occurs within only a few clock cycles of previous access
  - If the row-hit ratio is above 0.5, we switch to the active-page mode
  - Otherwise, we switch to the auto-precharge mode
  - The row-hit ratio is profiled through the row-hit history window in the FPGA memory controller
- **Delayed-precharge mode with automatic threshold timeout value**
  - After a burst access, SDRAM can remain in the ACTIVE state for up to specified timeout period
  - According to the row-hit history, the FPGA memory controller dynamically changes the timeout value for issuing the precharge command

Memory Management in BitsyX

- **APD (Automatic Power-Down) function of BitsyX platform**
  - If SDRAM's are not accessed, the pxa255 will immediately put the SDRAM's into the power-down mode
  - If the SDRAM's are accessed while they are in the power-down mode, there will be a latency penalty of one memory clock cycle to wakeup the SDRAM's
- **Kernel module for memory power management**
  - Disable the APD function when a memory-intensive application is running
  - Enable the APD function when a CPU-intensive application is running

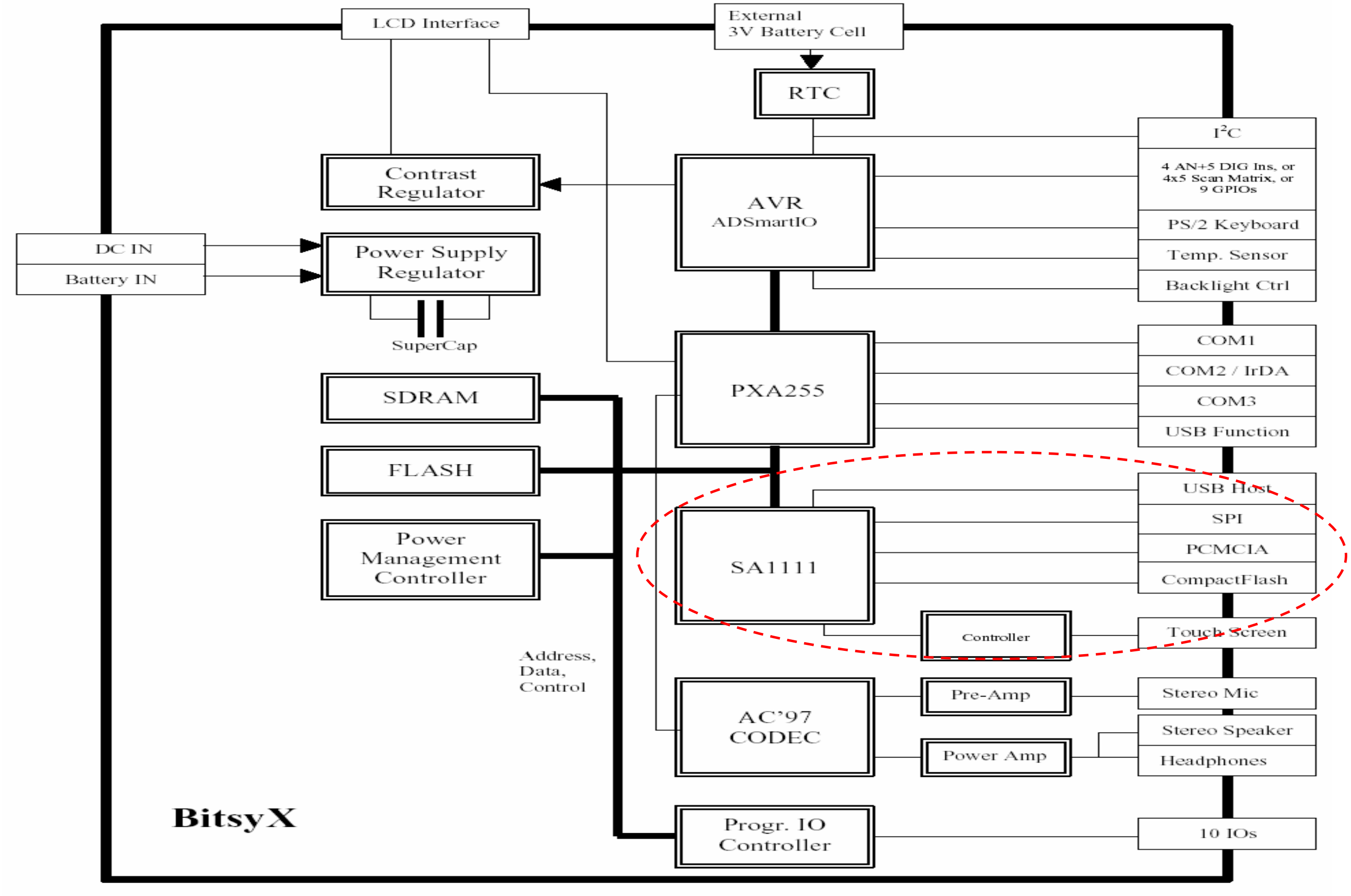
Wake on Wireless

Wake on Wireless

- **Problem statement: The BitsyX system is in its low power, sleep state but the wireless LAN is still kept powered on. When the WLAN receives a data packet addressed to it, it has to wake-up the system while ensuring that no data is lost in this process**
- **BitsyX sleep state definition:**
  - Processor (PXA 255) in sleep state
    - PXA defines 4 power states turbo, run, idle and sleep
  - SA1111 in its reset state
  - Most peripherals are turned off (except RTC, ADSmartIO and devices that wake up the system)
  - The system is brought out of the sleep state by either one of: RTC, shorting reset pins, a specified set of interrupts managed by the ADSmartIO (keypad, touch-screen)

Wake on Wireless (Cnt'd)

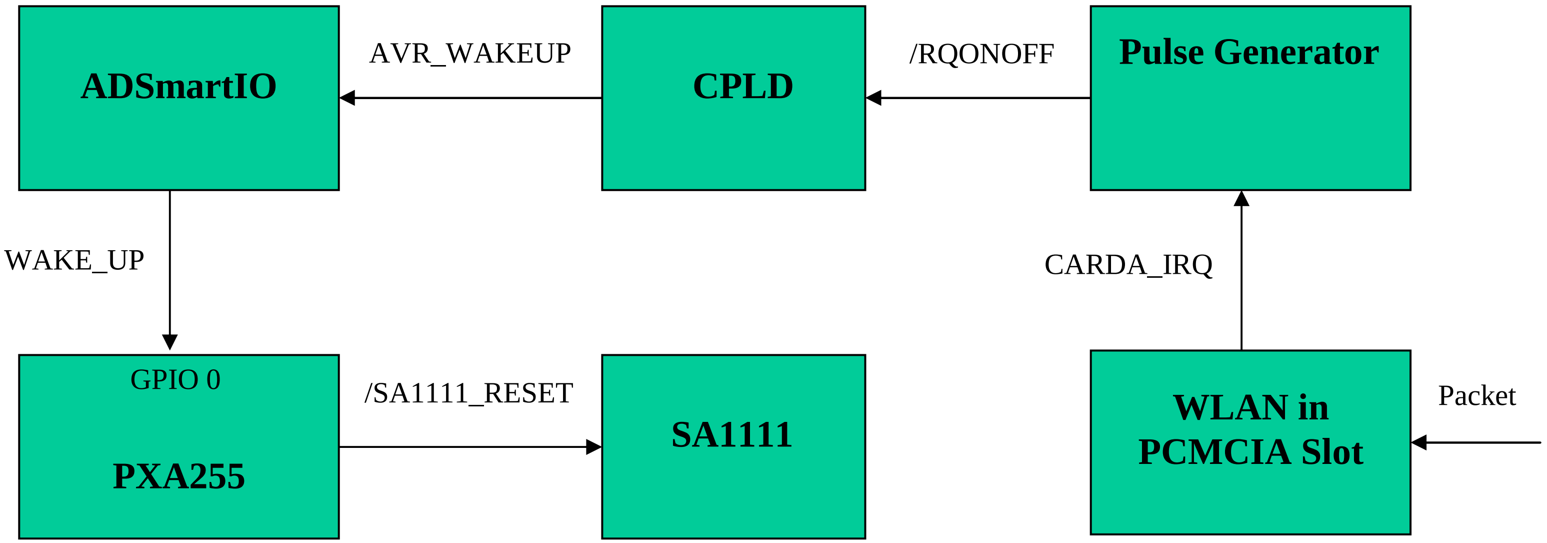
- **BitsyX architecture**
  - The wireless LAN is connected to the Compact Flash (CF) slot
  - The CF slot is controlled by the SA1111 companion chip



Proposed Solution

- **Devise a way to wake up the BitsyX system that has been put in the sleep-mode in response to an incoming packet for the wireless LAN card on the PCMCIA slot. Our mechanism is given below.**
  - Initially, the system is in sleep mode. A packet arrives at the wireless LAN card in a PCMCIA slot.
  - The wireless LAN card generates a CARDA_IRQ signal and this signal goes to a pulse generator.
  - The pulse generator converts this signal as a voltage pulse of fixed duration; this pulse is then sent to the CPLD /RQONOFF input pin.
  - The CPLD generates an AVR_WAKEUP signal to the ADSmartIO chip.
  - The ADSmartIO generates a WAKEUP signal to the PXA255 GPIO0 input pin.
  - The PAX255 generates a SA1111_RESET signal to the SA1111.
  - Eventually, the system gets back to the normal mode.

Block Diagram



ADSmartIO
AVR_WAKEUP
CPLD
/RQONOFF
Pulse Generator
WAKE_UP
CARDA_IRQ
GPIO 0
PXA255
/SA1111_RESET
SA1111
WLAN in PCMCIA Slot
Packet