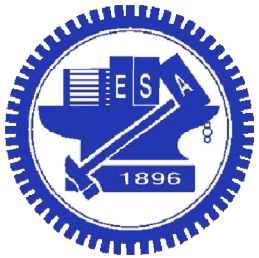
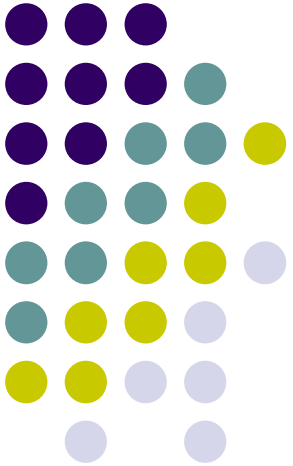
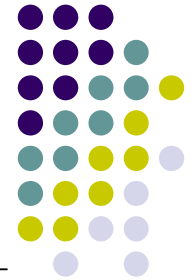
# Lab-2: Profiling m4v_dec on GR-XC3S-1500




National Chiao Tung University
Chun-Jen Tsai
3/28/2011



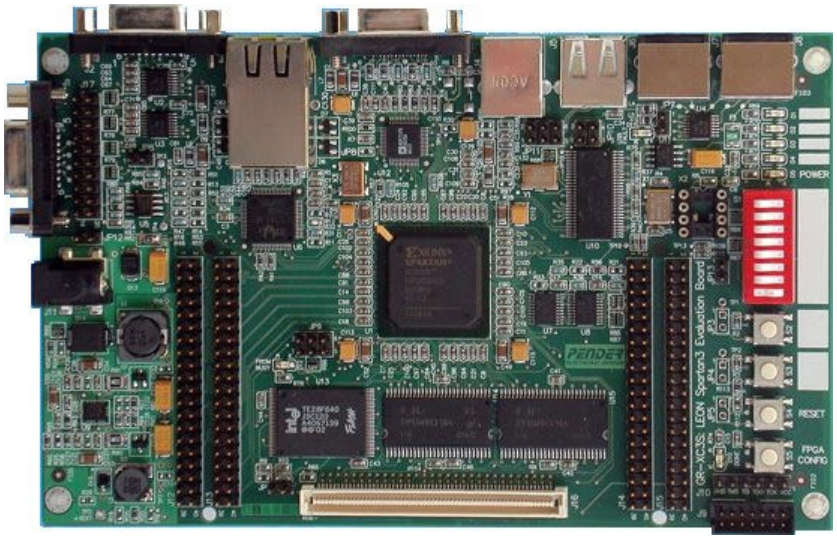
# Profiling with Real-time Timer



- Goal: Profiling m4v_vdec on GR-XC3S-1500 using a real-time timer
- Tasks:
  - Install a real-time timer ISR in the video decoder
  - Use tftp protocol to read/write video data
  - Use the timer to measure the performance of your optimized video decoder from lab1
- Give a demo to TAs and upload a report by the end of 4/8

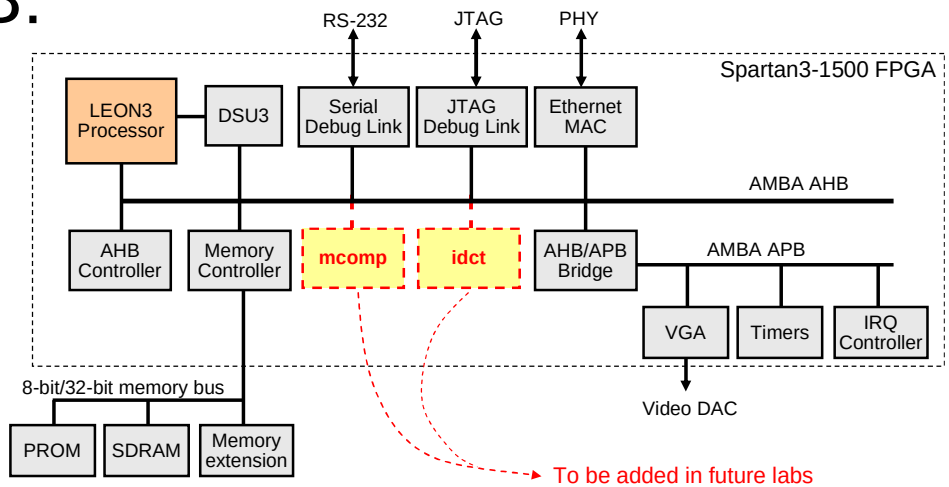
# GR-XC3S-1500 Development Board

- The system core IC is a Xilinx Spartan III FPGA
- Features
  - FPGA: XC3S-1500-FG456-4C FPGA
  - On-board memory
    - ✓ 8 MB Flash
    - ✓ 64 MB SDRAM
  - On-board I/O interfaces
    - ✓ 10/100 Ethernet PHY
    - ✓ 24-bit VGA Video DAC
    - ✓ USB 2.0 PHY
    - ✓ Two UART Transceivers
    - ✓ JTAG port



# GRLIB IP Library

- The board supports a reusable IP library, GRLIB
  - Designed for system-on-chip (SoC) development
  - Based on AMBA bus protocol
- Standard IPs in GRLIB:
  - LEON3 processor core
  - BUS controllers
  - Memory controller
  - Debug support unit
  - Interrupt controller
  - Timer
  - I/O controllers: UART, Ethernet, VGA, USB, …



# Debug Support Unit (DSU)

- GRLIB has a DSU IP, which is an AHB slave
  - Accessible by any AHB master (e.g. debug interfaces)
  - A debug Interface can generate read or write transfers to any address on the AHB bus through DSU
- DSU can also be used to access
  - Processor registers
  - Instruction trace buffer
  - AHB trace buffer

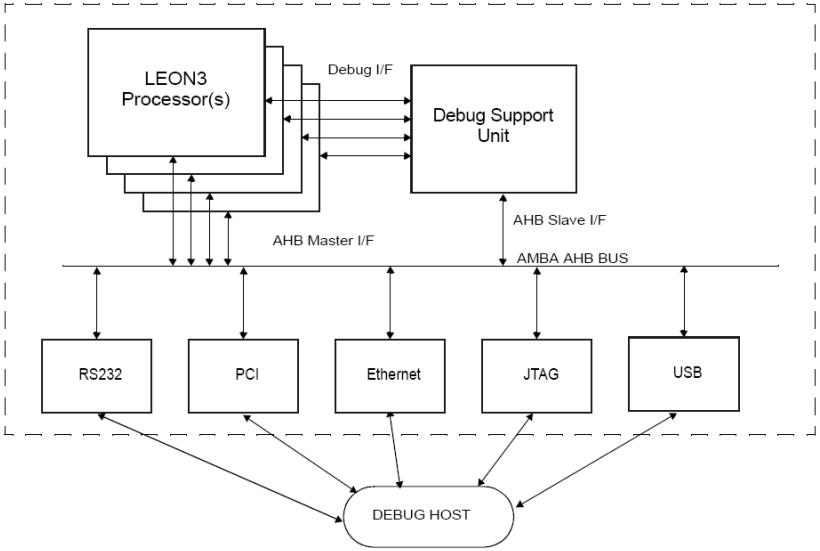
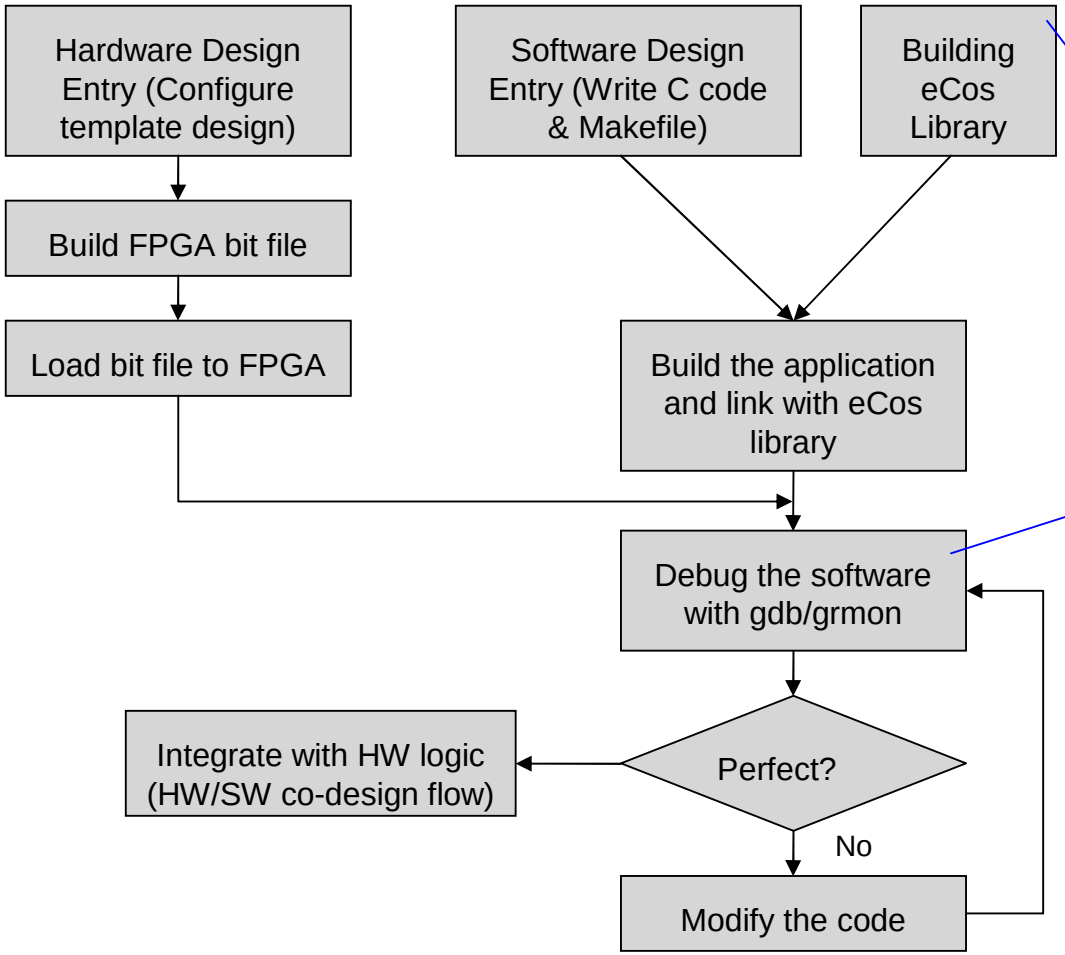



*Figure 53.* LEON3/DSU Connection

# System Development Flow†



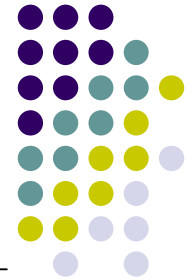

eCos is an open source library that provides basic OS kernel functions to your applications;

In our case, we need eCos to provide us tftp and timer support

At this moment, the HW IDCT logic is substituted with a C model function

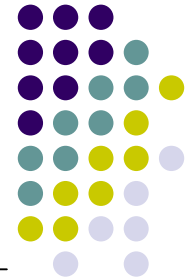
† Illustration by Michael Wu, 2008

# Setup Cygwin Environment



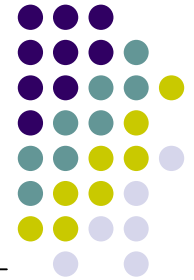
- Installing the latest Cygwin under Win32 by running http://www.cygwin.com/setup.exe
- In addition to the default Cygwin packages, make sure the following packages are installed:
  - automake, gcc, gdb, make, sharutils, tcltk, wget
  - If you want to use GHDL under Cygwin, you may also need the mpfr package

# BCC Cross-Compiler Installation



- Download the BCC package, sparc-elf-3.4.4-1.0.29d-cygwin.tar.bz2, from:
  ftp://graisler.com/bcc/bcc/bin/windows/old
- Under a Cygwin console, type
  $ mkdir /opt
  $ tar xjf sparc-elf-3.4.4-1.0.29d-cygwin.tar.bz2 –C /opt
- Modify the PATH variable by adding the following line to .bashrc in your home directory:
  - export PATH=/opt/sparc-elf-3.4.4/bin:$PATH

# eCos Installation

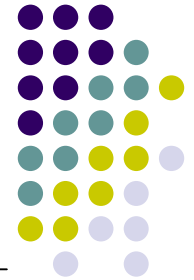


- An eCos port (ecos-rep-1.0.8.tar.gz) to GRLIB can be downloaded from
  ftp://gaisler.com/ecos/ecos/src

- Under a Cygwin console, type
  $ mkdir /home/soc/std_id
  $ tar xzf ecos-rep-1.0.8.tar.gz –C /home/soc/std_id

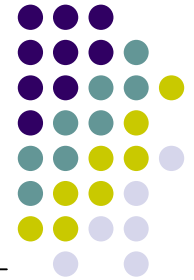
Each student create your own subdirectory (use student ID) under the system account

# Installing eCos Configuration Tool



- Download and install the eCos configuration tool, configtool-2.11-setup.exe, for Windows from: ftp://gaisler.com/ecos/ecos/bin/configtool/windows/
- Run eCos configuration tool
  - Setup paths
    - ✓ Tools → Paths → Build Tools = c:\cygwin\opt\sparc-elf-3.4.4\bin
    - ✓ Tools → Paths → User Tools = c:\cygwin\bin
    - ✓ Build → Repository = c:\cygwin\home\soc\std_id\ecos-rep-1.0.8

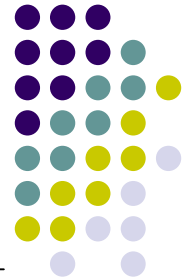
# Building eCos Library



- We have to build an eCos with timer and tftp support
- Inside eCos configuration tool†
  - Select: Build → Templates
    - ✓ Hardware: LEON3 processor with GRETH ethernet
    - ✓ Packages: net
  - Search for “CYGHWR_NET_DRIVER_ETH0_ADDRS”
    - ✓ Edit addresses according to environment setup
  - Select: File → Save
    - ✓ Save the configuration as student_id/ecos_leon/leon.ecc
  - Select: Build → Library

† See *soc10_leon_tutorial.doc* for detail settings

# Building the Application

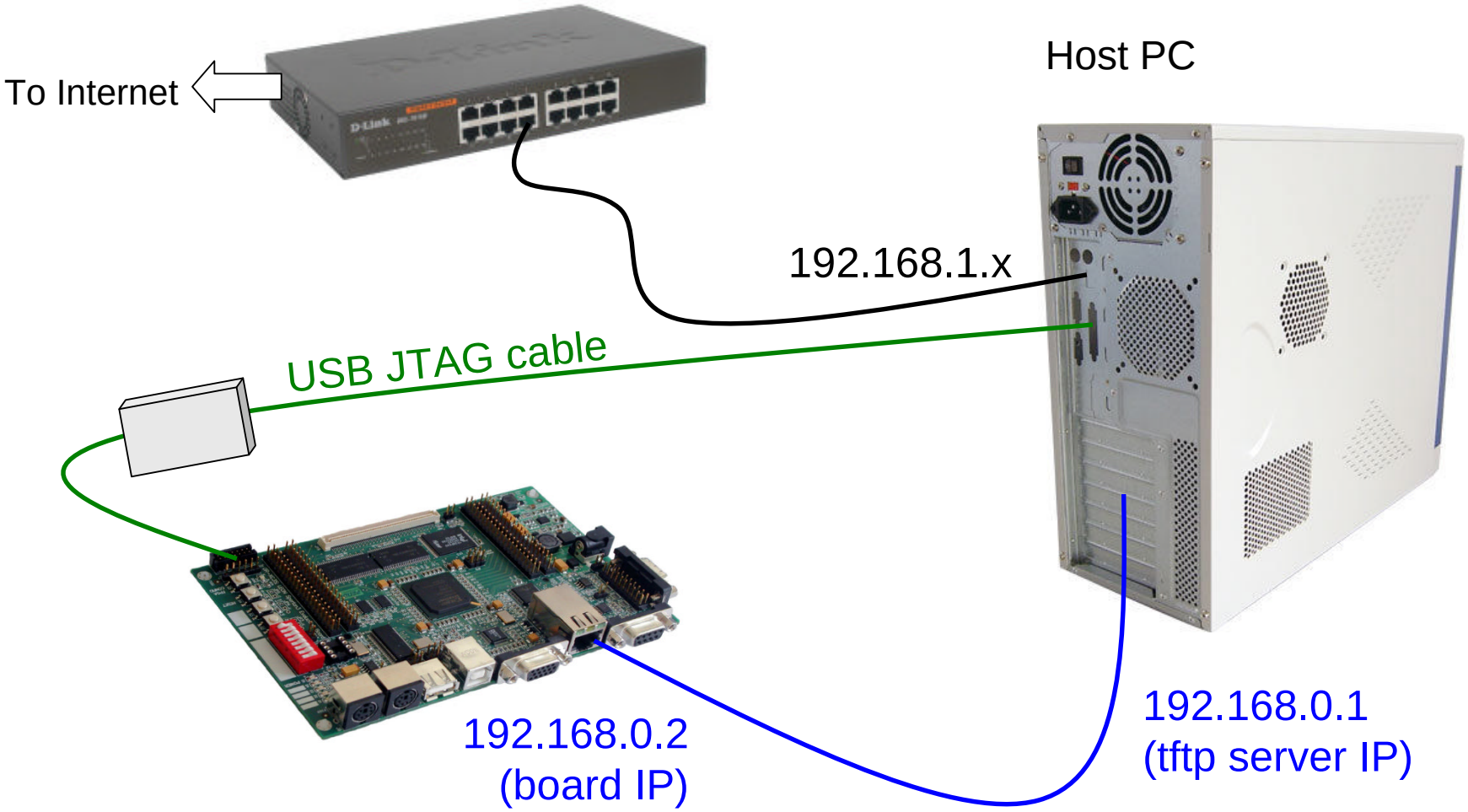


- Unzip the lab package under your work directory

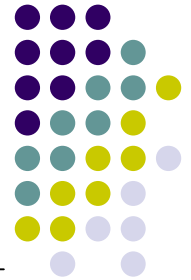
```
+-- ecos_leon/
|
+-- lab2_pkg/ +-- bitstream/
              +-- leon_bit_files/
              +-- m4v_dec_ecos/ +-- src/
                                +-- Makefile
                                +-- readme.txt
                                +-- timer_example.c
```

- Simply type “make” in m4v_dec_ecos/, and you will have an m4v_dec.elf executable

# System Setup

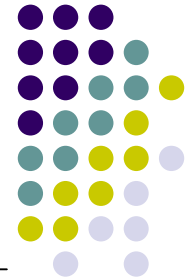


# GRMON Debug Monitor



- GRMON is a general debug monitor for the LEON processor, it supports
  - Downloading and execution of LEON applications
  - Breakpoint and watchpoint management
  - USB, JTAG, RS232, PCI, and Ethernet debug links
  - Remote connection to GNU debugger (gdb)
  - Access to all system registers and memory
  - Built-in disassembler and trace buffer management

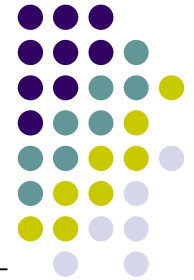
# Installation of GRMON



- Download the GRMON package, grmon-eval-1.1.39.tar.gz, from: ftp://gaisler.com/grmon/grmon
- Under a Cygwin console, type
  $ tar xzf grmon-eval-1.1.39.tar.gz –C /opt

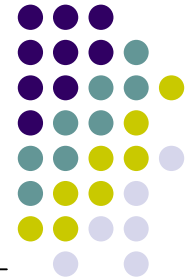
- Modify the PATH variable by adding the following line to .bashrc in your home directory:
  - export PATH=/opt/grmon-eval/cygein/bin:$PATH

# Uploading FPGA Bit File (1/2)



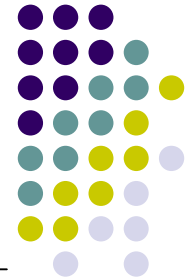
- Although we don't have to synthesize HW logic in this lab, you still need the HW EDA tool in order to configure the FPGA
- Download and install ISE WebPACK 10.1 from http://www.xilinx.com/tools/webpack.htm
  - The latest version is 11.1, but 10.1 is used in this course

# Uploading FPGA Bit File (2/2)



- Run iMPACT in the ISE suite
  - Select: “Create a new project (.ipf)” → “OK”
  - Select: “Configure devices using Boundary-Scan (JTAG)”
  - You will see a scan chain with 3 devices: xcf04s, xcf01s, and xc3s1500
  - Select “Bypass” for the first 2 devices
  - For xc3s1500, open the bit file that comes with the lab package (under leon_bit_files/)
  - Right click on xc3s1500 and select “Program”
  - Select “Verify” and click “OK”
  - Wait for a blue “Program Succeeded” message to appear

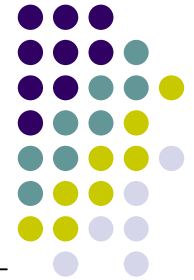
# Control the Board from GRMON



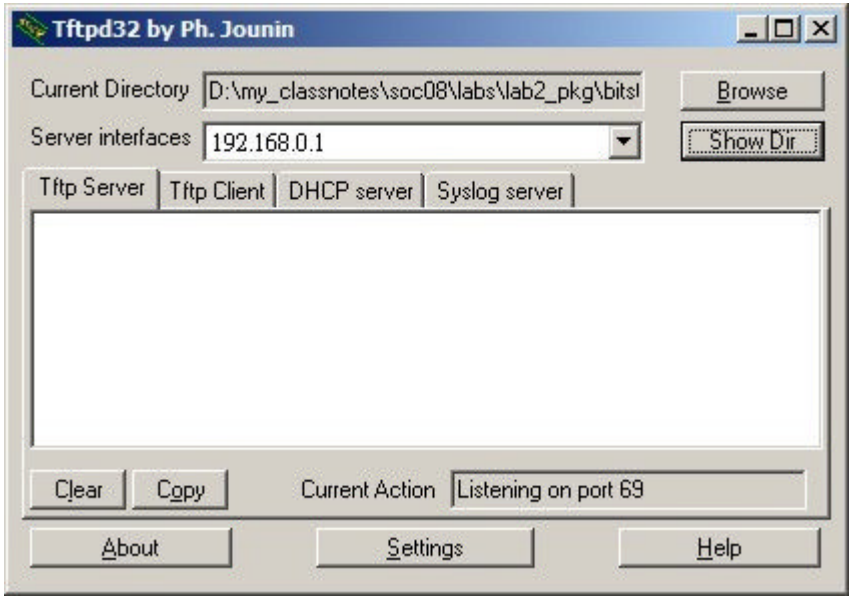
- Under Cygwin prompt, type $ grmon-eval –eth –u
  - This tells GRMON to use the Ethernet debug interface
  - The –u flag let application pipe the output to grmon console

```
GRMON LEON debug monitor v1.1.39 evaluation version
Copyright (C) 2004-2008 Aeroflex Gaisler - all rights reserved.
For latest updates, go to http://www.gaisler.com/
Comments or bug-reports to support@gaisler.com
This evaluation version will expire on 2/11/2010
ethernet startup.
GRLIB build version: 4075
initialising ..............
detected frequency:   9 MHz
Component                            Vendor
LEON3 SPARC V8 Processor           Gaisler Research
        . . .
Modular Timer Unit                  Gaisler Research
General purpose I/O port           Gaisler Research
Use command 'info sys' to print a detailed report of attached cores
grlib>
```

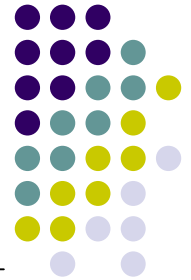
# Run a tftp Server on Host PC



- We must run a tftp server on the host PC in order to send/receive data to/from the board
- It is recommended that you use the tftp32 server (http://tftpd32.jounin.net/):



# Decoder Execution



- Under Cygwin console, type the commands:

  ~/$ cd <your working directory>

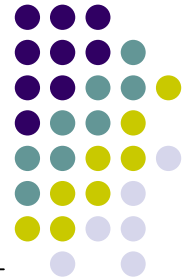
  ~/$ grmon-eval –eth –u

  ……. some GRMON startup messages ……..

  grlib > load m4v_dec.elf

  grlib > run

# Sample Execution Result

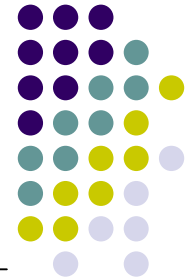


```
Reading bitstream using tftp...
bitstream_size = 83860, err = 0
Initializing decoder ...
Decoding frames: 0...16...32...48...64...80...96...112...128...144...

IDCT Computation:          9118.00 ms (46.61% of total decoding time)
Inverse Quantization:      1009.00 ms ( 5.16% of total decoding time)
Motion Compensation:       4152.00 ms (21.22% of total decoding time)
Boundary Extension:         559.00 ms ( 2.86% of total decoding time)
Boundary Removal:           646.00 ms ( 3.30% of total decoding time)
Block Data Transfer:       1724.00 ms ( 8.81% of total decoding time)
DC/AC Prediction:           160.00 ms ( 0.82% of total decoding time)
VLC Decoding:               349.00 ms ( 1.78% of total decoding time)
Total decoding time:      19564.00 ms, we measured 17717.00 ms (90.56%)

Writing decoded YCbCr frames using tftp...
Finished decoding.
```

# General Purpose Timer in GRLIB



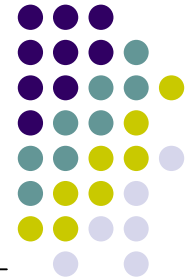
- The Leon platform contains GPTIMER IP† which you can use for profiling purposes
- By default, there will be two hardware timers
  - The first one is used by eCos
  - The second one is free for you to use
- To use the timer, you need its interrupt ID,

```
grlib> info sys

03.01:011   Gaisler Research  Modular Timer Unit (ver 0x0)
            irq 8
            apb: 80000300 - 80000400
            8-bit scaler, 2 * 32-bit timers, divisor 10
```

† See *GRLIB IP Core User's Manual*, Chapter 35 GPTIMER - General Purpose Timer Unit

# Installing of Timer ISR in eCos (1/3)



- Timer register definitions

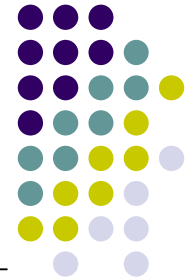
```
#include <cyg/kernel/kapi.h>
#include <cyg/hal/hal_io.h>

#define TICKS_PER_MS 40000 /* 40 MHz per second/1000 */

/* The IRQ value '8' displayed in 'info sys' is for  */
/* the first timer, which is used by eCos. We will   */
/* use the 2nd timer. Therefore, the IRQ value is 9. */
#define CYGNUM_HAL_INTERRUPT_TIMER2 (8+1)

/* The base address of timer registers is again */
/* obtained by 'info sys'.                      */
#define TIMER_BASE           0x80000300
#define SCALER_RELOAD_VALUE  TIMER_BASE + 0x04
#define TIMER2_RELOAD_VALUE  TIMER_BASE + 0x24
#define TIMER2_CTRL_REGISTER TIMER_BASE + 0x28
```

# Installing of Timer ISR in eCos (2/3)

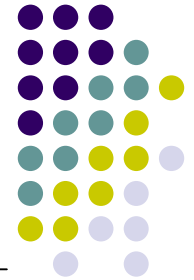


- Timer ISR installation:

```
cyg_interrupt_create(CYGNUM_HAL_INTERRUPT_TIMER2,
    0, data, my_isr, NULL, &handle, &isr_struct);
cyg_interrupt_attach(handle);
cyg_interrupt_unmask(CYGNUM_HAL_INTERRUPT_TIMER2);

/* initialize the timer to 1 ms per tick. For a 40MHz */
/* clock, 40000/prescaler_value equals 1 ms.          */
HAL_READ_UINT32(SCALER_RELOAD_VALUE, prescaler_value);
HAL_WRITE_UINT32(TIMER2_RELOAD_VALUE,
                    TICKS_PER_MS/prescaler_value);
/* set the control register */
HAL_READ_UINT32(TIMER2_CTRL_REGISTER, ctrl_reg);
ctrl_reg |= 0xA; /* enable the interrupt */
ctrl_reg |= 0x1; /* start ticking */
HAL_WRITE_UINT32(TIMER2_CTRL_REGISTER, ctrl_reg);
```

# Installing of Timer ISR in eCos (3/3)



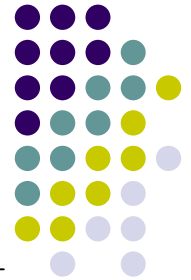
- Timer ISR

```
cyg_uint32
my_isr(cyg_vector_t vector, cyg_addrword_t data)
{
    long *counter = (long *) data;
    (*counter)++;
    cyg_interrupt_acknowledge(vector);
    return CYG_ISR_HANDLED;
}
```

- Uninstallation

```
cyg_interrupt_mask(CYGNUM_HAL_INTERRUPT_TIMER2);
cyg_interrupt_detach(handle);
cyg_interrupt_delete(handle);
```

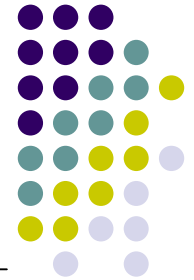
# Using tftp Protocol in Your Code



```
#include <network.h>
#include <tftp_support.h>

int main(int argc, char**argv)
{
    struct sockaddr_in host;
    int err, size, max_size, yuv_size;
    char *ifname, *ofname, *bit_buf, yuv_buf;
    . . .
    /* initialize network interface */
    init_all_network_interfaces();
    memset((char *) &host, 0, sizeof(host));
    host.sin_len = sizeof(host);
    host.sin_family = AF_INET;
    host.sin_addr = eth0_bootp_data.bp_siaddr;
    host.sin_port = 0;
    /* retrieve video bitstream */
    size = tftp_get(ifname, &host, bit_buf,
                    max_size, TFTP_OCTET, &err);
    . . .
    /* output decoded YCbCr frames */
    tftp_put(ofname, &host, yuv_buf, yuv_size, TFTP_OCTET, &err);
}
```

# Final Remark



When it comes to system design,
the completeness of your solution
depends on
the effort you put into it ….

and it shows!