Lecture 7 - Part 1
Getting up to speed with DE1-SoC board: HPS+FPGA Projects

Cristinel Ababei
Dept. of Electrical and Computer Engineering, Marquette University

1. Objective

The objective of this hands-on lecture-tutorial is to learn about how to use the DE1-SoC board to create projects that use both the FPGA fabric and the hardware processor system (HPS). We refer to such systems as HPS+FPGA projects or systems. This is Part 1, of a several part series of such hands-on tutorials.

2. Prerequisites

NOTE: If you have done already any of these installations/settings before (in other contexts), skip them here; no need to re-do them.

2.1: Update your PATH environment variable

Add to your **PATH** environment variable also (make sure you change 21.1 to correct version of your Quartus installation):
C:\intelFPGA_lite\23.1std\quartus\sopc_builder\bin
This will be needed when running sh scripts to generate header files later on. This should have been taken care during the Quartus Prime installation; but, it did not happen in my case.

2.2: Install "Intel SoC FPGA Embedded Development Suite (SoC EDS)"

NOTE: April.2025: When trying to install Cygwin (below) from rocketboards link, a waring message says this SoC EDS is not supported by Intel/Altera anymore? **TODO: investigate this issue.**

SoC EDS is a comprehensive tool suite for embedded software development on Intel FPGA SoC devices. It contains development tools, utility programs, run-time software, and application examples that enable firmware and application software development on Intel SoC hardware platforms.
The SoC EDS is offered in two editions: "Standard" and "Professional". We will use the "Standard" edition.

Follow the guide (Section 2.3.1) from here:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_soc_eds.pdf
to download and install the Intel SoC FPGA Embedded Development Suite (SoC EDS).

Download from here (you must create a free user account with Intel first):
https://www.intel.com/content/www/us/en/software-kit/661082/intel-soc-fpga-embedded-development-suite-soc-eds-standard-edition-software-version-20-1-for-windows.html
Download Standard edition, Version 20.1: **SoCEDSSetup-20.1.0.711-windows.exe**

Because we have installed Quartus Prime Lite, we will modify the default install location to be intelFPGA_lite.
Run the **SoCEDSSetup-20.1.0.711-windows.exe** file to install into:
**C:\intelFPGA_lite\ 23.1std **
It will create a new folder called **embedded/** and install some stuff in existing folders too.
During installation you might be prompted to install device drivers; just install all of them.

2.3: Install "Arm Development Studio - Intel SoC FPGA Edition"

Arm Development Studio is an embedded C/C++ development toolchain designed specifically for Arm-based SoCs, from tiny microcontrollers to custom multicore processors. Designed alongside Arm processor IP, it accelerates system design and software development for Cortex-M, Cortex-R and Cortex-A processors.
Developed in partnership with Arm, the Arm Development Studio (DS) Intel SoC FPGA Edition toolkit is an end-to-end suite of tools for embedded C/C++ software development on any Intel SoC FPGAs. It combines almost all the features of the Arm DS Professional Edition, with powerful FPGA-adaptive debugging capability, providing unmatched visibility and control of your SoC FPGA.
The Arm DS Intel SoC FPGA Edition toolkit is installed as part of the Intel® SoC Embedded Development Suite (SoC EDS). Learn more about the DS tool and get licensing information from Arm.

Follow the guide (Section 2.3.2) from here:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_soc_eds.pdf
to install Arm DS.

Download Arm DS installer archive from the Arm Development Studio for Intel SoC FPGA Edition Download Center for FPGAs website:
https://fpgasoftware.intel.com/armds/
more specifically:
Arm* Development Studio Version 2024.1 for Intel® SoC FPGA

Download the file (pay attention, select Windows Software tab on the download page, by default the Linux option is presented): **DS000-BN-00000-r24p1-00rel0.zip**
Extract the archive, then run the **armds-2024.1.msi** installer from the unzipped archive.
After installation, you should have **Arm DS IDE 2024.1** available as a program to launch.
Launch it to make sure it installed ok.
Do nothing about license for now.

2.4: Install Git

Git - is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Download Git from:
https://git-scm.com/download/win
When you install git for windows, make sure you choose "checkout as-is, commit unix-style line endings"!

2.5: Install Cygwin

Cygwin - offers a Linux-like environment on Windows, and is required by the Embedded Command Shell.
Cygwin is: 1) a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. 2) A DLL (cygwin1.dll) which provides substantial POSIX API functionality.

Follow the guide "Intel® SoC FPGA Embedded Development Suite (SoC EDS) User Guide" (Section 2.3.3) from here:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_soc_eds.pdf

Download **Cygwin**:
https://cygwin.com/setup-x86_64.exe
and place into a folder say: M:\Temp1\setup-x86_64.exe
Open a **cmd** terminal (as Administrator) and goto: C:\intelFPGA_lite\23.1std\embedded\cygwin_setup

>cd C:\intelFPGA_lite\23.1std\embedded\cygwin_setup
>soceds-cygwin-setup.bat M:\Temp1\setup-x86_64.exe
The installer application starts and installs it in: C:\intelFPGA\20.1\embedded\host_tools\cygwin

Finally, also edit environment variables to add to PATH:
C:\intelFPGA_lite\23.1std\embedded\host_tools\cygwin\bin

2.6: Install MinGW

MinGW - used as a platform to build the Newlib library used by the Linaro Bare Metal toolchain

MinGW ("Minimalist GNU for Windows"), formerly mingw32, is a free and open-source software development environment to create Microsoft Windows applications. MinGW includes a port of the GNU Compiler Collection (GCC), GNU Binutils for Windows (assembler, linker, archive manager), a set of freely distributable Windows specific header files and static import libraries which enable the use of the Windows API, a Windows native build of the GNU Project's GNU Debugger, and miscellaneous utilities.

Follow the guide (Section 2.3.4) from here:
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_soc_eds.pdf
to download and install MinGW.

Option 1 (does not work anymore, so, go for Option 2 below):
You can download the MinGW installer to your computer and run as an administrator.
https://rocketboards.org/foswiki/Documentation/SoCEDS#Install_MingGW
The installation should create C:\MinGW on your machine.

NOTE: April.2025: When trying to download and install Cygwin from the link above, I found this message: ***WARNING****: SoC EDS is a discontinued product, which is no longer supported by Altera. The information in this page is provided for archival purposes only! For up-to-date information about HWLIBs, please see this link: Hardware Library (HWLibs). Thank you.* **TODO: Investigate this issue.**

Option 2: Alternatively, I installed MinGW from here:
https://sourceforge.net/projects/mingw/
I selected all packages to install, including msys, etc.

Finally, also edit environment variables to add to PATH:
C:\MinGW\msys\1.0\bin

TODO: Investigate this: Look into more recent MinGW versions like MinGW-w64, and verify if the whole Intel Altera suite of tools would work with that.

2.7: Installing Linaro Bare Metal Toolchain

Linaro Bare Metal Toolchain - is a set of pre-built, validated tools, including an assembler, compiler, linker, and libraries, designed for developing software that runs directly on the hardware (bare metal) of ARM embedded processors. It enables developers to write and compile software that interacts directly with the hardware, without relying on an operating system.

Launch (with right click and then Run as Administrator):
C:\MinGW\msys\1.0\msys.bat
Then:

```
>cd c:/intelFPGA_lite/23.1std/embedded/host_tools/linaro
>./install_linaro.sh
```
The Linaro toolchain and the newlib library are downloaded and compiled.
Upon successful completion, the following are installed in the linaro folder:
• gcc–GNU Compiler
• newlib–Newlib library

If the above installation does not complete successfully, and it ends with errors, then, do the following section; otherwise, skip the next section.

2.8: Install ARM Cross Compiler

Do this only if: The installation of SoC EDS and ARM DS did not install (as it used to…) the cross-compiler **arm-linux-gnueabihf-gcc**. Not sure why. So, let's try to install it on our own. This will allow us to compile my_first_hps example on our laptop (running Windows 10 on an x86 processor). A compiler that runs on one processor architecture (x86) but produces machine code for a different architecture is called a cross-compiler. Usually, we can easily download a compiler that does just that for us (easier done on Linux machines compared to Windows machines). The Intel SoC is based off an ARM core, so we need a compiler that runs on an x86 platform, but converts C code into ARM machine level instructions, which will be executed on the HPS Arm processor inside the FPGA.
Otherwise, we would need to either use a different cross-compiler or install an older version of DS, i.e., DS-5, which was installing in `c:/intelFPGA_lite/20.1/embedded/DS-5` and which was including the cross-compiler **arm-linux-gnueabihf-gcc**.

Create a new folder:
C:\intelFPGA_lite\23.1std\embedded**my_cross_compilers**

NOTE: If while doing the above you get a write error, it may be because you did not or could not run the whole this as Administrator; it is a Windows issue. You can fix it by going with a File Explorer and select C:\intelFPGA_lite, click right on Properties and Edit the permissions to give full access to your User to everything in this folder.

Open an ESC shell by right-click and Run as Administrator:
C:\intelFPGA_lite\23.1std\embedded\ Embedded_Command_Shell.bat

```
>cd C:/intelFPGA_lite/23.1std/embedded/my_cross_compilers
>wget https://releases.linaro.org/components/toolchain/binaries/7.5-2019.12/arm-linux-gnueabihf/gcc-linaro-7.5.0-2019.12-i686-mingw32_arm-linux-gnueabihf.tar.xz
>tar -xvf gcc-linaro-7.5.0-2019.12-i686-mingw32_arm-linux-gnueabihf.tar.xz
>chmod -R 777 *
```

Finish by adding to the environment variable PATH on your computer the following:
C:\intelFPGA_lite\23.1std\embedded\my_cross_compilers\gcc-linaro-7.5.0-2019.12-i686-mingw32_arm-linux-gnueabihf\bin
so that the cross-compiler will be found during compilations with Makefile's later on.

2.9: Windows Subsystem for Linux (WSL 1)

Windows Subsystem for Linux - is a feature of Microsoft Windows that allows the use of a GNU/Linux environment from within Windows, foregoing the overhead of a virtual machine and being an alternative to dual booting.

Nios II EDS was included in Quartus Prime software installation. But, to use it (in some only of our projects/examples), there are additional requirements for using it. That is, we need to also install Windows Subsystem for Linux (WSL) (required from version Quartus Prime Lite Edition 19.1 and Quartus Prime Pro Edition 19.2).

For downloading and installation of Windows Subsystem for Linux on Windows 10/11 simply follow these steps:

Step 1: To enable the Windows Subsystem for Linux feature do the following:
Open PowerShell as Administrator (1.Press the Win+S keyboard shortcut to open Windows Search. 2.Type "PowerShell" into the search bar. 3.From the right pane, select Run as administrator.)
Enable the Windows Subsystem for Linux:

```
>dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

Enable virtual machine feature:

```
>dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Restart your PC to complete the installation.

After that, do again in Power Shell as admin:
> wsl –update
> wsl --set-default-version 1 ← not 2 because it is not supported by Intel ALtera

Alternatively, you could enable the Windows Subsystem for Linux feature by following these steps, which may need also administrative privileges.

1. Right-click on the Windows Start menu icon, choose Search and type "Windows Features". Select the top entry (category Control panel) to enable or turn off Windows-Features. The Windows-Features dialog will be opened.
2. Select in the upcoming dialog the check box for Windows Subsystem for Linux from the bottom of the list and press the OK button. Applying the changes may take a few minutes. Finally, press the Restart now button to reboot the computer.

Step 2: Install Linux distribution Ubuntu 18.04 LTS for WSL (not 24.04, which seems not to be supported by Intel Altera?)
Open PowerShell as Administrator and run:

```
>wsl --install --distribution Ubuntu-18.04
```

Launch Ubuntu 18.04 from Windows Start Menu and follow instructions.
Install additional packages:

```
>sudo apt update
>sudo apt upgrade
>sudo apt install wsl
>sudo apt install dos2unix
>sudo apt install make
>sudo apt install build-essential
```

Alternatively, you could install the Ubuntu-18.04 Linux distribution as follows. It also requires no administrative privileges.

1. Open the Start menu and select Microsoft Store. (If the Microsoft Store is not available, the WSL must be installed manually. After this proceed with 6.)
2. Search for Ubuntu.
3. Select Ubuntu-18.04 LTS among the list of presented apps, then click the Install (or Get) button to download this Ubuntu Linux distribution.

4. If a Microsoft Sign in dialog opens (without a user name), it is not necessary to sign in. Just close the dialog with the upper right window button and the installation continues. If instead a Microsoft dialog appears asking for password of your specific Microsoft account, you need to enter your password to continue with the installation.
5. After the download has finished press the Launch button to install and initialize the WSL.
6. A Linux terminal window is opened. After some minutes, the message "Enter new UNIX username" is shown. Type a user name and press the Return key. Next a message "Enter new UNIX password" is shown. Type a password (no characters are shown while typing) and press the Return key. Then, a message "Retype new UNIX password" is shown. Type the password again and press the Return key. The credentials should be memorized as they are needed to install further Linux components.
7. Finally, close the Linux terminal window or enter exit.

2.10: Download and unzip CD of DE1-SoC board

Finally, make sure you have downloaded the complete CD-ROM of the DE1-SoC board from Terasic's website. I will refer to some documents and examples from this CD. The boards that we use are the Revision E, so, please make sure you download the correct CD-ROM archive.
http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=836&PartNo=4
Install the whole thing let's say as:
C:\Terasic\DE1_SoC

3. Examples

EXAMPLE #1: Simply boot Linux on DE1-SoC board and connect to it using Putty from host PC

Read DE1-SoC Getting Started Guide (revision E Board) Chapter 5, and do all the steps therein:
C:\Terasic\DE1_SoC\UserManual\DE1-SoC_Getting_Started_Guide.pdf
(also included with the files provided with this Part 1 tutorial)

Download and burn the microSD card with the Linux Console image, which can be downloaded from Terasic:
http://download.terasic.com/downloads/cd-rom/de1-soc/linux_BSP/DE1_SoC_SD.zip
To create the microSD card image, you must download and use Win32 Disk Imager beta - a tool for writing images to USB sticks or SD/CF cards. Follow the steps in the above guide.

NOTE: As an alternative to Win32DiskImager, you can use the Etcher (https://www.balena.io/etcher) to burn the .iso image to the microSD card.

After that, insert the microSD card into the slot of the DE1-SoC board. Also, set the FPGA Configuration Mode Switch (MSEL) from the back of the DE1-SoC board to the setting “000000” as described in Chapter 3 of the same document **DE1-SoC_Getting_Started_Guide.pdf**. This needs to be done when we want to boot Linux from the micro SD card.

The example in this chapter of the guide simply starts a Putty terminal on the PC and connects to the Linux running on DE1 board. That is all is does! After a successful boot, the Linux will ask for the login name. Linux will ask for the login name. Use "root" and "terasic" as password.

EXAMPLE #2: Example from Terasic CD with HPS only

The files provided with this Part 1 tutorial include all the files needed for this example, in the sub-folder
my_first_hps/
Which is basically a slightly modified version of the example from the Terasic CD for DE1-SoC board:

C:\Terasic\DE1_SoC\Demonstrations\SoC\my_first_hps

Read and do the steps of the example described here:
C:\Terasic\DE1_SoC\UserManual\My_First_HPS.pdf

The example provides information on how to create a C language software design and run it on the ARM HPS of the DE1-SoC development board. This example, essentially creates a "Hello World!" design that displays a message on the Linux terminal, as you can see by opening and reading:
C:\Terasic\DE1_SoC\Demonstrations\SoC\my_first_hps\main.c

```
#include <stdio.h>
int main(int argc, char **argv) {
   printf("Hello World, from Cris!\r\n");
   return( 0 );
}
```

We use the Embedded Command Shell (ESC). Open an ESC terminal by executing:
C:\intelFPGA_lite\20.1\embedded\Embedded_Command_Shell.bat
Navigate to the location of the example.

```
>cd
M:/MARQUETTE/EECE4740_VHDL_FPGA/sandbox_quartus/quartus_projects/my_first_hps
```

Then, compile example **my_first_hps** by typing

```
>make
```

This create executable **my_first_hps**, which needs to be copied to the bootable SD card, to the **/home/root/** folder.
Keep the ESC shell open, we'll need it for the next steps.

Currently, there are several ways to copy the executable that we just created onto the microSD card.
See the following document for details:
http://dejazzer.com/eece4740/lectures/3_Copying_executables_to_SDCard.pdf

Next, we use Method #1 (doable if you are at home and have access to your router via an Ethernet cable, provided in your hardware kit). Otherwise, you can use Method #2 described in the above document.

Method #1: Use an Ethernet cable to connect the DE1-SoC to your Internet Router at home.
Connect an Ethernet cable between your Router and the DE1-SoC board.
Connect also the microUSB cable to JP4 of the board and open the Putty terminal and connect to the DE1-SoC board.
Reboot Linux on the DE1-SoC board by turning the power off and on again, then check you can connect to the Internet. Run in your Putty terminal that is connected to the DE1-SoC board via the USB cable to JP4 of the board:

```
>ifconfig
```

In my case, I got inet addr: 192.168.0.34 at home (in my office I got 134.48.91.26).
Note, that optionally you can change password (to say "terasic", like I did) on Linux using the command `passwd`.
Place copy of created executable **my_first_hps** on the SD card, in **/home/root** folder (create folder using mkdir if not already created).
Copying can be done Linux system command "**`scp`**" in the ESC shell on host PC (the same where we compiled the program earlier using make). So in the ESC shell, type:

```
>scp my_first_hps root@134.48.90.44:/home/root
```

to copy the executable file into the folder "/home/root" on the Linux running on the SD card on DE1-SoC board.

The SD card should contain the Linux image that can be downloaded from Terasic webpage (not LXDE or Ubuntu or the framebuffer)! Only in that case, the above "`scp`" command works; though, you may have to say "yes" to

agree with some authenticity failure. If I tried to use LXDE or Ubuntu, I get an error: "ERROR: no permission on Linux on board to scp".

At this time, **my_first_hps** executable is copied to the SD card on DE1 board via Ethernet. You should be able to see it by doing in the Putty terminal:

```
>ls
```

Run the executable on the Putty terminal as:

```
>./my_first_hps
```

You should see the "Hello World, from Cristinel!" message now. Now, this is cool. :)

EXAMPLE #3: Example from Terasic CD with HPS-FPGA

The files provided with this Part 1 tutorial include all the files needed for this example: **my_first_hps-fpga/** provided on the website for this class on dejazzer.

*NOTE: In recent versions of Quartus Prime, **Qsys** subtol has been replaced/renamed actually by/as **Platform Designer**. So, wherever you read **Qsys** in Terasic's not-updated documentation, you should actually use the new **Platform Designer**. You can read some info about it here:*
https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-platform-designer.html
https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/an/an812.pdf

This project is another example provided by Terasic; it includes everything already. You can also find it inside the Terasic CD: **C:\Terasic\DE1_SoC\Demonstrations\SOC_FPGA\my_first_hps-fpga**, but, please use the files from the website of my class on dejazzer.
The example's folder contains two folders: **fpga-rtl/** and **hps-c/**.
In doing this example, you must read the documentation here:
C:\Terasic\DE1_SoC\UserManual\My_First_HPS-Fpga.pdf

In the Intel Altera SoC FPGA chip, the HPS logic and the FPGA fabric are connected through the so-called AXI (Advanced eXtensible Interface) bridge. For HPS logic to communicate with FPGA fabric, Altera system integration tool **Qsys** should be used to design the system. From the AXI master port of the HPS component, HPS can access those HPS components whose memory-mapped slave ports are connected to the master port.
The HPS contains the following HPS-FPGA AXI bridges:

- FPGA-to-HPS Bridge
- HPS-to-FPGA Bridge
- Lightweight HPS-to-FPGA Bridge

Summary of this example: The Lightweight HPS-to-FPGA bridge is used for HPS to control the LEDs connected to FPGA. The lightweight HPS-to-FPGA AXI Master port of the HPS component is connected to the memory-mapped slave port of a **Parallel IO (PIO)** component, which we wish to access from HPS side.
The PIO component needs to be added to the system in Platform Designer. Then, connect the PIO component to the HPS component. The PIO component is used to control the ten red LEDs connected to the FPGA side. This is done by doing in the top-level Verilog (in this example) design something like this:

```
      .pio_led_external_connection_export     (LEDR),
```

Where "pio_led_external_connection_export" is the name of the exported from Platform Designer – see column Export, and "LEDR" is the name of the output port in the top-level entity used to drive the ten LEDs on the board.
Read **C:\Terasic\DE1_SoC\UserManual\My_First_HPS-Fpga.pdf**
and follow the steps to do this example, which has basically two parts:

FPGA part:
Synthesize the Quartus project from folder **fpga-rtl/** and program the FPGA. Note that this example uses Verilog for the top-level entity and not VHDL.

Note: To read more on the PIO Core, see Chapter 22 of this document:
https://www.altera.com/en_US/pdfs/literature/ug/ug_embedded_ip.pdf

HPS part:
As done in Example 2, launch:
C:\altera\17.1\embedded\Embedded_Command_Shell.bat
Navigate to the location of the example, in my case it is.

```
>cd
M:/MARQUETTE/EECE4740_VHDL_FPGA/sandbox_quartus/quartus_projects/my_first_hps
-fpga/hps-c
```

Then, compile it and then set permission to all:

```
>make
>chmod 777 *
```

This creates executable **my_first_hps-fpga**, which again needs to be copied to the bootable SD card, to the **/home/root/** folder.
As done in the previous example, connect Ethernet cable between board and your Router first, then, in the ESC terminal do:

```
>scp my_first_hps-fpga root@134.48.90.44:/home/root
```

After copying the executable to the micro SD card, finally, from within the Putty terminal that connects via the micro USB to JP4 of the board run the example:

```
>./my_first_hps-fpga
```

If all ok, you should see the whole thing working! LEDs are all on and are set off and then on from left to right and vice-versa, 60 times because that's how it's programmed inside **main.c**. This is cool too! :) When done shutdown the Linux, by typing in the Putty terminal:

```
>shutdown -t 3 -h now
```

Now it's a good time to read a bot the main.c:
C:\Terasic\DE1_SoC\Demonstrations\SOC_FPGA\my_first_hps-fpga\hps-c\main.c
It should help understand how the LEDs are controlled from within the C program executed on the HPS side.

*NOTE: Another thing that is incomplete as help from Terasic is that this HPS+FPGA example comes with the top-level design **ghrd_top.v** as only Verilog and not as VHDL too. This may be annoying if one wants to use VHDL out of the box and not mix VHDL with Verilog; but we can live with that...*

4. Summary

After going through the above steps and examples, you should hopefully have a bit better idea about HPS+FPGA systems.

However, the above steps put together based only what's available from Intel Altera and Terasic is obviously not enough to get a good grasp of such type of systems. The good news is that some very nice people out there looked into this and worked out some examples, which they described and shared online - thank you!. In the next part, we will look at several such examples.

5. Additional Resources

Recent documentation:

[1] Platform Designer (formerly Qsys): https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-platform-designer.html
[2] Creating a System with Platform Designer: https://www.altera.com/documentation/jbr1437426657605.html#mwh1409958596582
[3] Platform Designer overview video that walks through how to use the tool: https://www.altera.com/products/design-software/fpga-design/quartus-prime/features/qts-platform-designer.html

Older documentation, but still relevant and useful (Platform Designer looks a lot like the old Qsys):

[1] Introduction to Altera's Qsys system integration tool: ftp://ftp.altera.com/up/pub/Altera_Material/15.1/Tutorials/Introduction_to_the_Altera_Qsys_Tool.pdf
[2] Quartus II Handbook (read Chapter 5: Creating a wystem With Qsys): https://www.altera.com/en_US/pdfs/literature/hb/qts/qts-qps-handbook.pdf
[3] In addition, check out some of the tutorials and examples available here (those related to DE1-SoC board). Some of them can be helpful: https://www.altera.com/support/training/university/materials-tutorials.html