Embedded Systems

Week 7: PL/PS CoProcessing

Fenerbahçe University

Professor & TAs

Prof: Dr. Vecdi Emre Levent

Office: 311

Email: emre.levent@fbu.edu.tr

TA: Arş. Gör. Uğur Özbalkan

Office: 311

Email: ugur.ozbalkan@fbu.edu.tr

Petalinux

- sudo apt install device-tree-compiler
- dtc -I dtb -O dts -o system.dts system.dtb --> Decompile
- dtc -I dts -O dtb -o system.dtb system.dts --> Compile

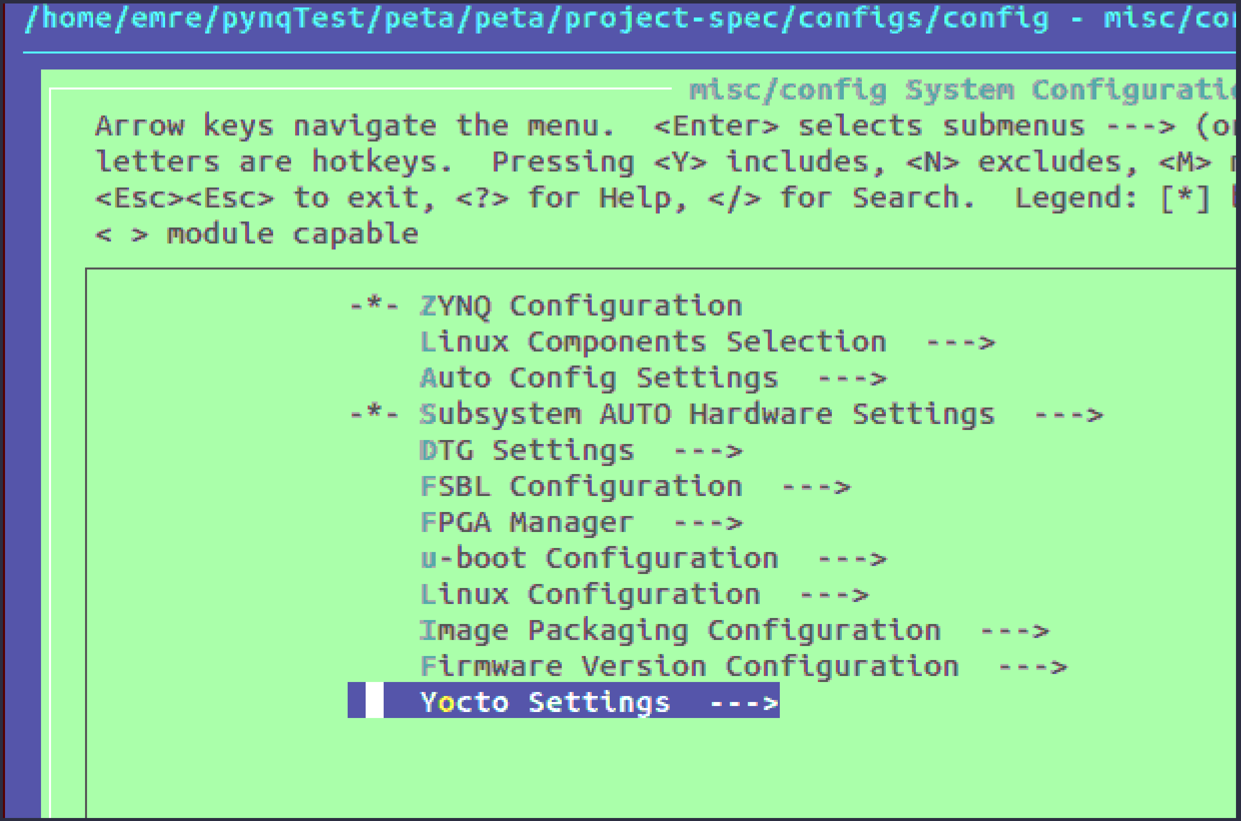
Petalinux

Buildin function memory test

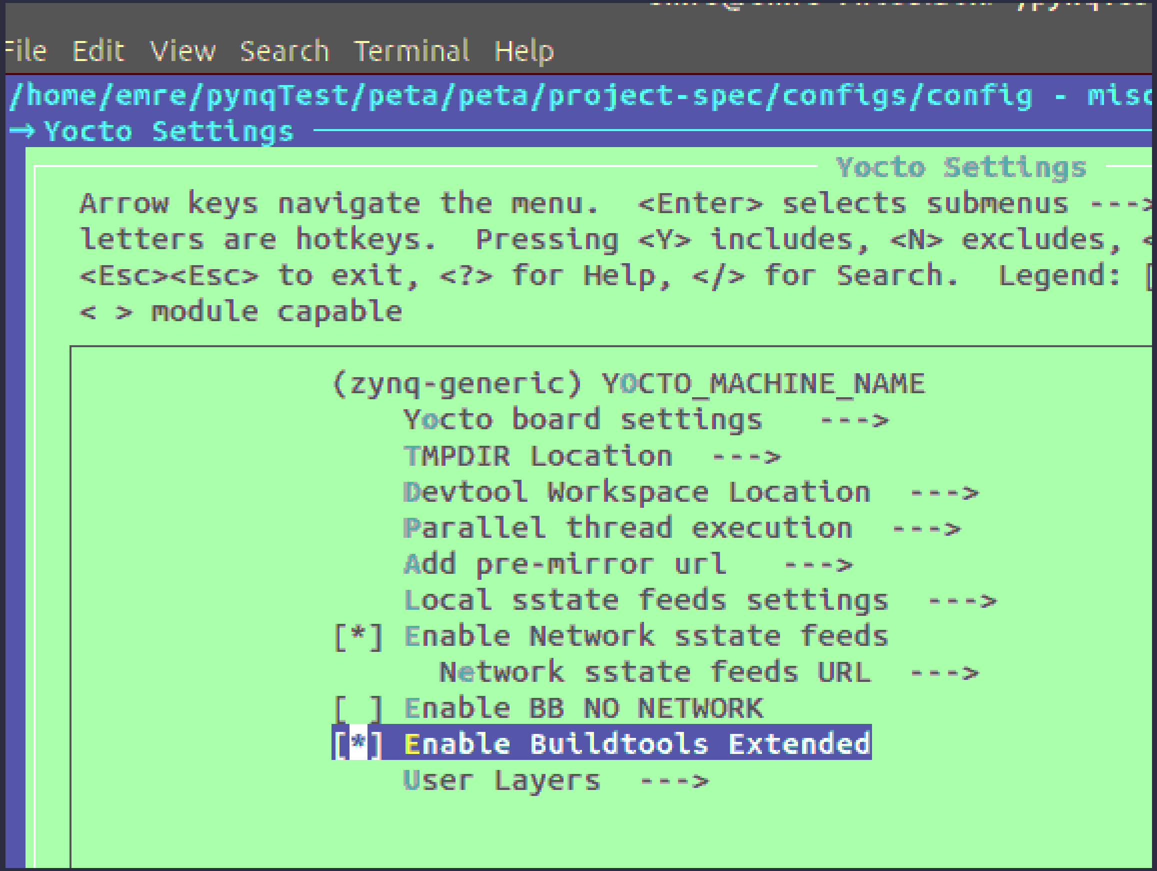
- devmem 0x40000000 64 --> reads address
- devmem 0x40000000 64 0xaaaaaaaaaaaaaaaa -> writes 64 bytes to address

Petalinux

Adding GCC Support to Yocto



```
/home/emre/pynqTest/peta/peta/project-spec/configs/config - misc/co
                                   misc/config System Configurati
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (o
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> 
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] 
  < > module capable

               -*- ZYNQ Configuration
                   Linux Components Selection  --->
                   Auto Config Settings  --->
               -*- Subsystem AUTO Hardware Settings  --->
                   DTG Settings  --->
                   FSBL Configuration  --->
                   FPGA Manager  --->
                   u-boot Configuration  --->
                   Linux Configuration  --->
                   Image Packaging Configuration  --->
                   Firmware Version Configuration  --->
                   Yocto Settings  --->
```

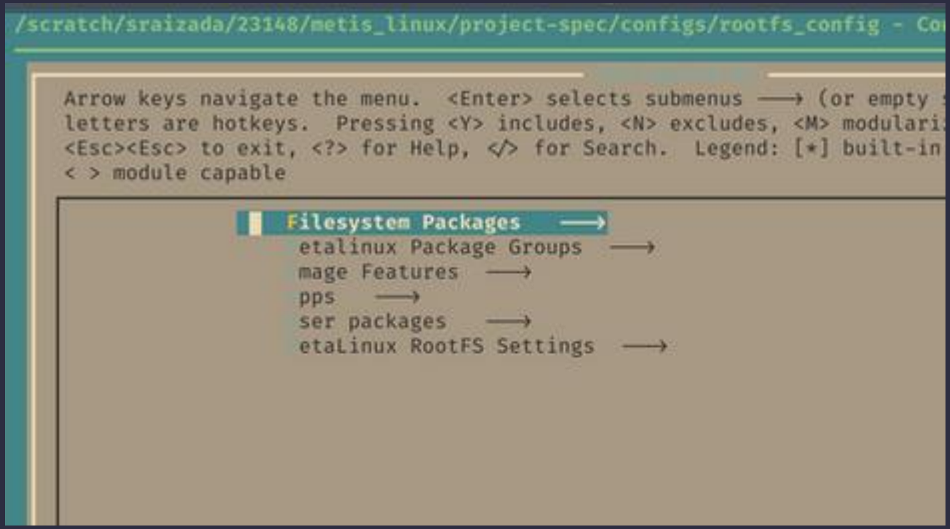


```
File  Edit  View  Search  Terminal  Help
/home/emre/pynqTest/peta/peta/project-spec/configs/config - misc
→ Yocto Settings
                                              Yocto Settings
  Arrow keys navigate the menu.  <Enter> selects submenus --->
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, 
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: 
  < > module capable

               (zynq-generic) YOCTO_MACHINE_NAME
                   Yocto board settings  --->
                   TMPDIR Location  --->
                   Devtool Workspace Location  --->
                   Parallel thread execution  --->
                   Add pre-mirror url  --->
                   Local sstate feeds settings  --->
               [*] Enable Network sstate feeds
                     Network sstate feeds URL  --->
               [ ] Enable BB NO NETWORK
               [*] Enable Buildtools Extended
                   User Layers  --->
```

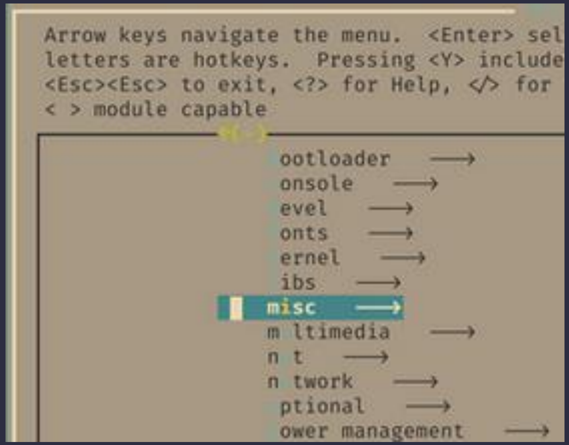
Petalinux

Adding GCC Support to Yocto

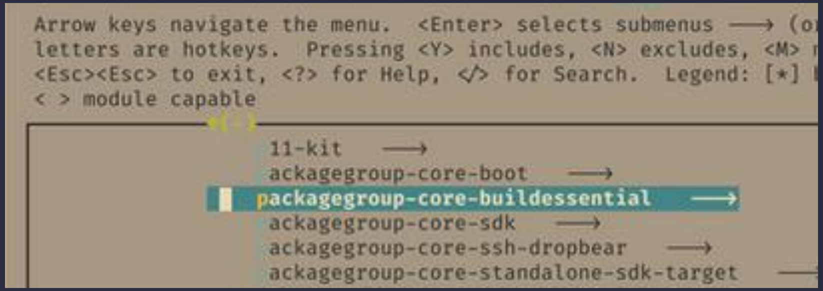
petalinux-config -c rootfs



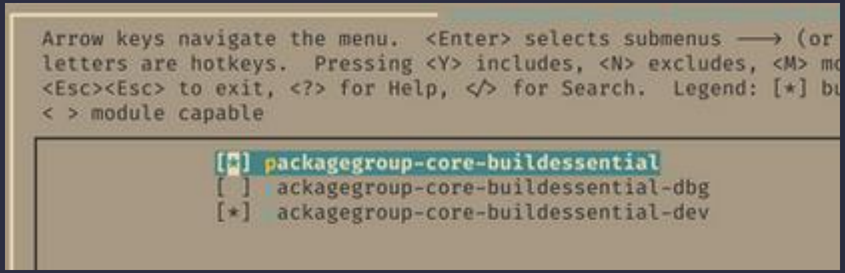
```
/scratch/sraizada/23148/metis_linux/project-spec/configs/rootfs_config - Co

  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modulari
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in
  < > module capable

              Filesystem Packages  --->
               etalinux Package Groups  --->
               mage Features  --->
               pps  --->
               ser packages  --->
               etaLinux RootFS Settings  --->
```



```
  Arrow keys navigate the menu.  <Enter> sel
  letters are hotkeys.  Pressing <Y> include
  <Esc><Esc> to exit, <?> for Help, </> for
  < > module capable

                 ootloader  --->
                 onsole  --->
                 evel  --->
                 onts  --->
                 ernel  --->
                 ibs  --->
                misc  --->
                m ltimedia  --->
                n t  --->
                n twork  --->
                 ptional  --->
                 ower management  --->
```



```
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (o
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> 
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] 
  < > module capable

          11-kit  --->
          ackagegroup-core-boot  --->
         packagegroup-core-buildessential  --->
          ackagegroup-core-sdk  --->
          ackagegroup-core-ssh-dropbear  --->
          ackagegroup-core-standalone-sdk-target  --->
```



```
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> mo
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] bu
  < > module capable

        [*] packagegroup-core-buildessential
        [ ]  ackagegroup-core-buildessential-dbg
        [*]  ackagegroup-core-buildessential-dev
```

run `petalinux-build`

Petalinux

Add a application to YoctoFS

petalinux-create -t apps --template c --name testapp --enable
petalinux-create -t apps --template c++ --name testapp –enable

Will create a app on

project-spec/meta-user/recipes-apps/testApp

under files you can find testapp.c file

For more source files edit .bb and make files

Clean up project
petalinux-build -x mrproper
petalinux-build -c testapp
petalinux-package --prebuilt --fpga images/linux/system.bit
After you can do petalinux-build
Then petalinux-boot --jtag --prebuilt 3 --hw_server-url 10.21.0.20:3121

Petalinux

Yocto GCC Compile

```
gcc main.c –o main
chmod u+x main
./main
```

Petalinux

Access MMAP function

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>

// Make the SDK console work in the debugger
#define printf(...) \
 fprintf(stdout, __VA_ARGS__); \
 fflush(stdout);

typedef long long int u64;
```

```
int main()
{
  unsigned int bram_size = 0x8000;
  off_t bram_pbase = 0x40000000; // physical base address
  u64 *bram64_vptr;
  int fd;
  printf("My test app\n");
  // Map the BRAM physical address into user space getting a virtual
address for it
  if ((fd = open("/dev/mem", O_RDWR | O_SYNC)) != -1) {
    printf("Opened\n");
    bram64_vptr = (u64 *)mmap(NULL, bram_size,
PROT_READ|PROT_WRITE, MAP_SHARED, fd, bram_pbase);
    printf("Opened %x\n", bram64_vptr);
    // Write to the memory that was mapped, use devmem from the
command line of Linux to verify it worked
    // it could be read back here also

    bram64_vptr[0] = 0xDEADBEEFFACEB00C;
    printf("Bye\n");
    close(fd);
  }
 }
```

Petalinux

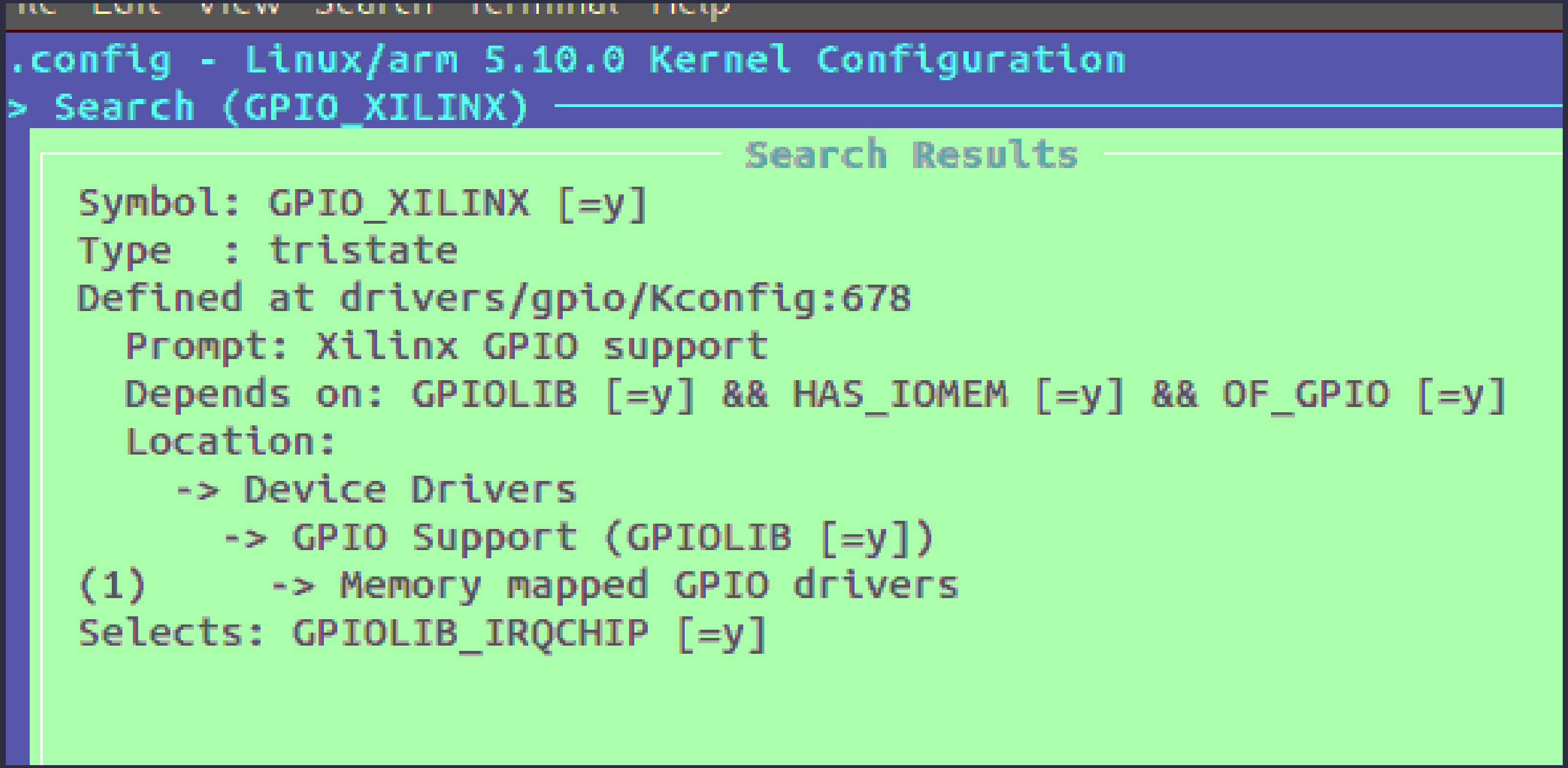
AXI GPIO

petalinux-config -c kernel

Make sure all enabled:

- CONFIG_GPIO_SYSFS=y
- CONFIG_SYSFS=y
- CONFIG_GPIO_XILINX=y

Check device tree



```
.config - Linux/arm 5.10.0 Kernel Configuration
> Search (GPIO_XILINX)
                         Search Results
  Symbol: GPIO_XILINX [=y]
  Type  : tristate
  Defined at drivers/gpio/Kconfig:678
    Prompt: Xilinx GPIO support
    Depends on: GPIOLIB [=y] && HAS_IOMEM [=y] && OF_GPIO [=y]
    Location:
      -> Device Drivers
        -> GPIO Support (GPIOLIB [=y])
  (1)     -> Memory mapped GPIO drivers
  Selects: GPIOLIB_IRQCHIP [=y]
```

Petalinux

AXI GPIO

The GPIO driver fits in the Linux GPIO framework.
It does provide access to the GPIO by user space through the sysfs filesystem.

- mkdir /sys
- mount -t sysfs sysfs /sys

Find AXI GPIO number

```
ls /sys/class/gpio
```

more /sys/class/gpio/gpiochip992/label -->Check others and find AXI GPIOs Address

```
echo 504 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio504/direction
echo 1 > /sys/class/gpio/gpio504/value

echo 496 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio496/direction
cat /sys/class/gpio/gpio496/value
```

Petalinux

AXI GPIO

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

int main()
{
   int valuefd, exportfd, directionfd;

   printf("GPIO test running...\n");


   exportfd = open("/sys/class/gpio/export", O_WRONLY);
   if (exportfd < 0)
   {
      printf("Cannot open GPIO to export it\n");
      exit(1);
   }

   write(exportfd, "992", 4);
   close(exportfd);
 printf("GPIO exported successfully\n");

   // Update the direction of the GPIO to be an output

   directionfd = open("/sys/class/gpio/gpio992/direction",
O_RDWR);
   if (directionfd < 0)
   {
      printf("Cannot open GPIO direction it\n");
      exit(1);
   }

   write(directionfd, "out", 4);
   close(directionfd);

   printf("GPIO direction set as output successfully\n");
                  // Get the GPIO value ready to be toggled

   valuefd = open("/sys/class/gpio/gpio992/value", O_RDWR);
   if (valuefd < 0)
   {
      printf("Cannot open GPIO value\n");
      exit(1);
   }
   printf("GPIO value opened, now
toggling...\n");

   // toggle the GPIO as fast a
possible forever, a control c is
needed
   // to stop it

   while (1)
   {
      write(valuefd,"1", 2);
      sleep(2);
      write(valuefd,"0", 2);
      sleep(2);
      printf("Toggle ok\n");
   }
}
```