HLS AXI master and Yocto kernel device driver module for Zynq

A complete flow example

Stefano Tabanelli

Digital FAE & Software Specialist

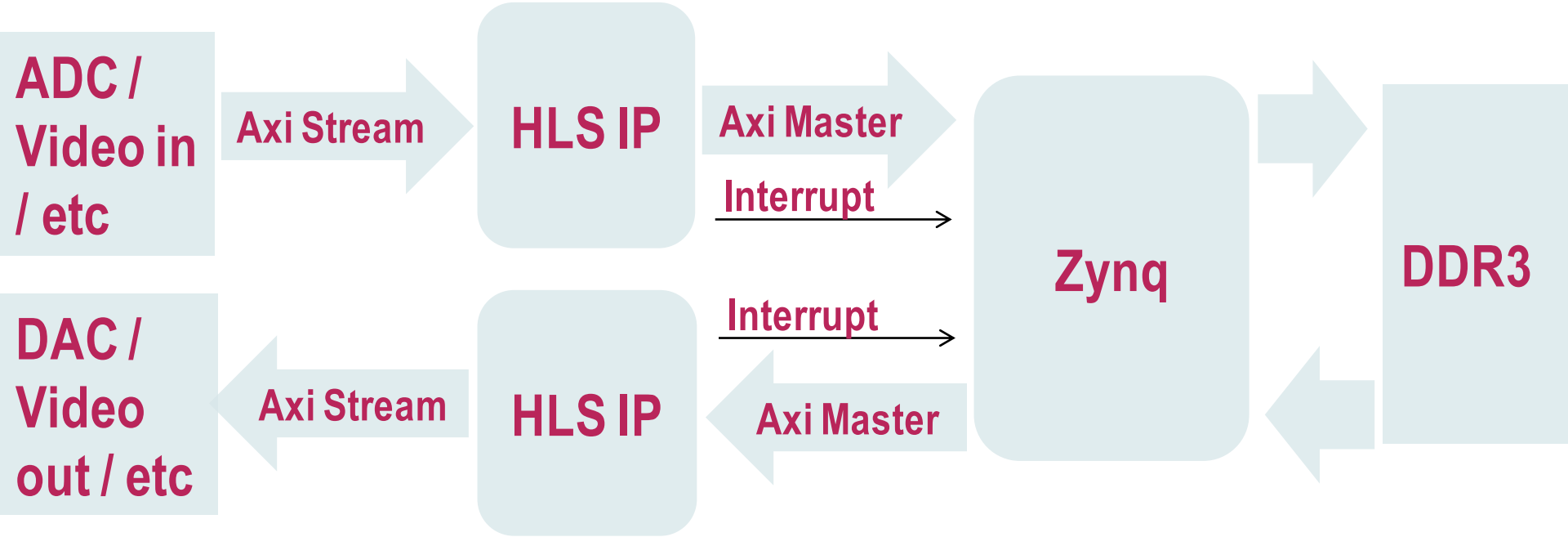
Agenda

- The need for Axi master
- A template for multiple applications
- HLS Axi master project
- Vivado IPIntegrator project
- Yocto Kernel Module
- Demo

The need for Axi master

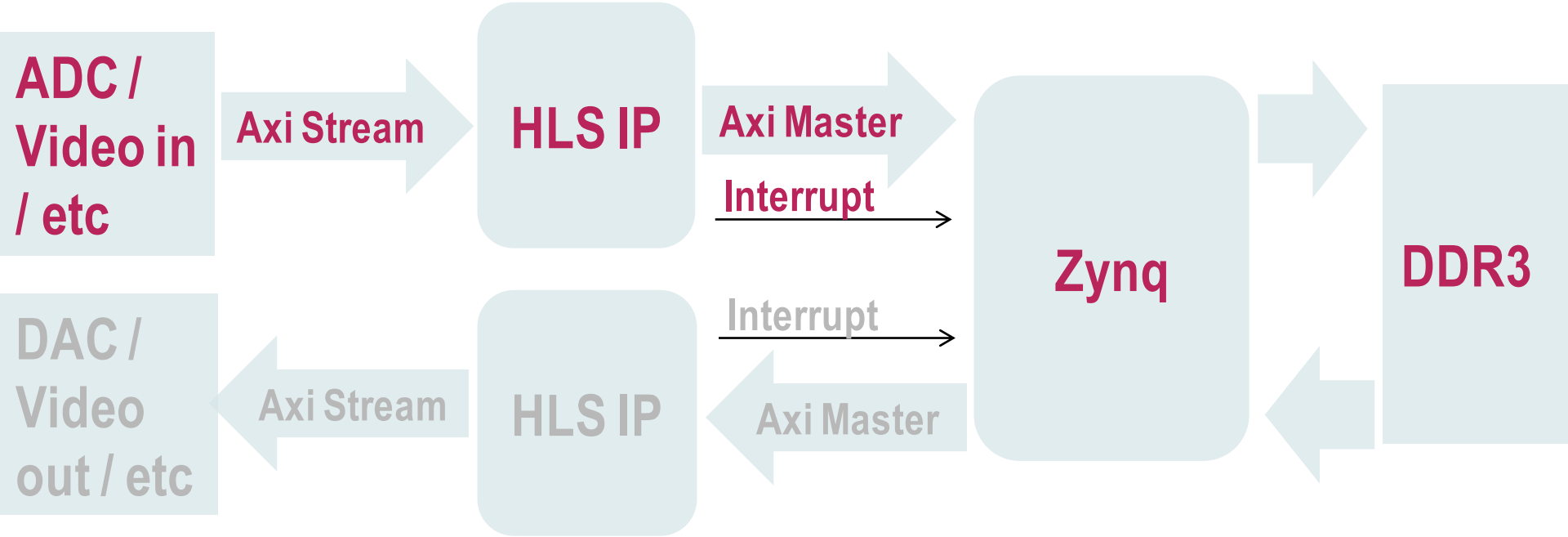
- **Frequent request from customers**
- **Axi master allows a data elaboration flow in parallel to CortexA9**
  - Video acquisition to memory / Video output from memory
  - ADC data acquisiton to memory / DAC output driving from memory
  - Data filtering / elaboration on the fly or directly on memory
- **Need a sync method to coordinate with CortexA9**
  - Interrupt generation / handling
- **Sw Device Driver recommended**

A template for multiple applications



A template for multiple applications

What is available now



HLS Axi master project

The ANSI C source code

Vivado HLS - proj_axi_master (C:\tmp\axi_master_and_stream\axi_master\proj_axi_master)

File Edit Project Solution Window Help

Explorer

- proj_axi_master
  - Includes
  - Source
    - example.cpp
  - Test Bench
  - *solution1*
    - constraints
      - directives.tcl
      - script.tcl
    - csim
      - build
      - report
    - impl
      - ip
      - vhdl
    - syn

example.cpp | example_csynth.rpt

```
1  #include <stdio.h>
2  #include <string.h>
3  #define N 128
4  void example(volatile int *a, int stream_in[N], unsigned int byte_wroffset,
5                  unsigned int ctrl_reg, unsigned int *status_reg, bool *interrupt )
6  {
7      #pragma HLS INTERFACE ap_bus port=a
8      #pragma HLS RESOURCE variable=a core=AXI4M
9      #pragma HLS INTERFACE axis port=stream_in
10     #pragma HLS STREAM variable=stream_in
11     #pragma HLS INTERFACE ap_none register     port=byte_wroffset
12     #pragma HLS RESOURCE core=AXI4LiteS    variable=byte_wroffset metadata="-bus_bundle MYAXIL"
13     #pragma HLS INTERFACE ap_none register     port=ctrl_reg
14     #pragma HLS RESOURCE core=AXI4LiteS    variable=ctrl_reg metadata="-bus_bundle MYAXIL"
15     #pragma HLS INTERFACE ap_ovld register     port=status_reg
16     #pragma HLS RESOURCE core=AXI4LiteS    variable=status_reg metadata="-bus_bundle MYAXIL"
17     #pragma HLS INTERFACE ap_ovld port=interrupt
18     int i;
19     int buff[N];
20     static unsigned int count=0;
21     static bool int_pending = false;
22     static unsigned int local_ctrl_reg=0;
23
24     local_ctrl_reg = ctrl_reg;
25     if ((local_ctrl_reg & 1) && (int_pending == false)){
26       for(i=0; i < N; i++)
27           buff[i] = stream_in[i];
28       memcpy((int *)(a+byte_wroffset/4),buff,N*sizeof(int));
29       count += N;
30       *status_reg = count;
31       int_pending = true;
32     }
33     if (local_ctrl_reg & 2) {
34      int_pending=false;
35     }
36     *interrupt=int_pending;
37 }
```

HLS Axi master project

HLS compilation to VHDL (/Verilog) => 333Mhz estimated IP clock

Vivado HLS - proj_axi_master (C:\tmp\axi_master_and_stream\axi_master\proj_axi_master)

File Edit Project Solution Window Help

Explorer

- proj_axi_master
  - Includes
  - Source
    - example.cpp
  - Test Bench
  - *solution1*
    - constraints
      - directives.tcl
      - script.tcl
    - csim
      - build
      - report
    - impl
      - ip
      - vhdl
    - syn

example.cpp | example_csynth.rpt

Product family: zynq

Target device: xc7z010clg400-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
default	3.00	2.63	0.38

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
5	265	6	266	none

Detail

Instance

Loop

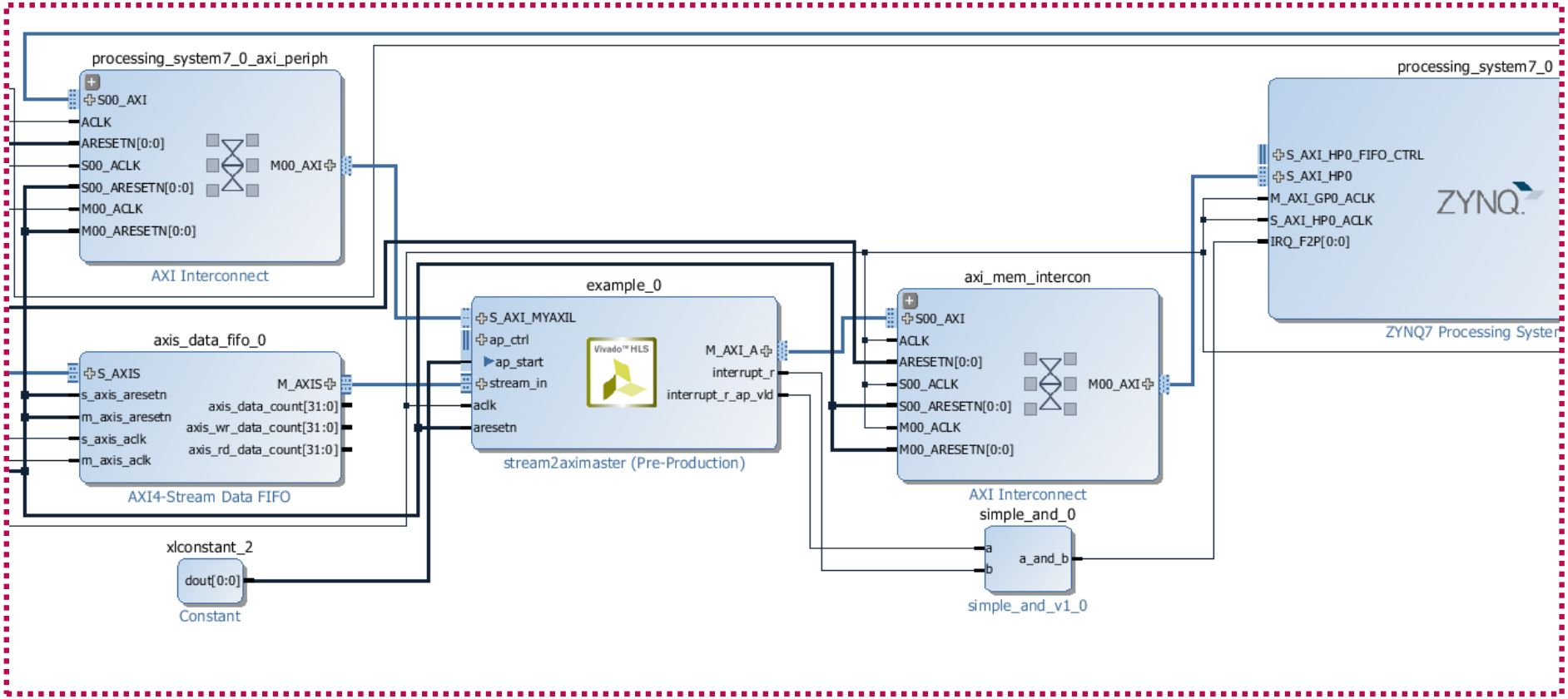
Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	0	83
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	1	-	0	0
Multiplexer	-	-	-	29
Register	-	-	231	-
Total	1	0	231	112
Available	120	80	35200	17600
Utilization (%)	~0	0	~0	~0

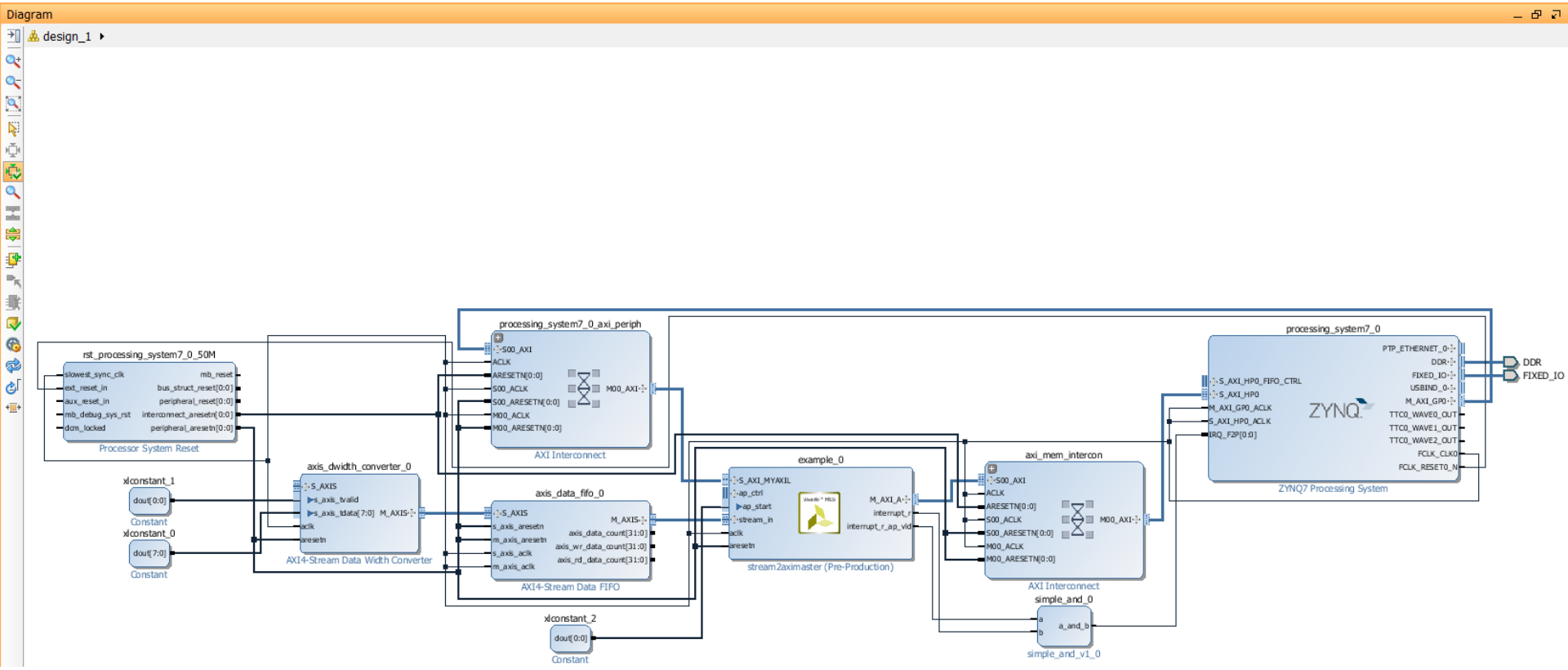
Vivado IPIntegrator project

Targeting Microzed board



Vivado IPIntegrator project

Targeting Microzed board



Yocto Kernel Module

A code sample to start from

- **There is a lack of complete flow (HLS->Vivado->Linux->Yocto->Zynq eval board) on Web resources**
- **The supplied driver is**
  - A Linux Kernel Module (vs User Space approach)
  - Simplified, bare minimal to addess open/read/write/close and single interrupt
  - Provided as Yocto recipe
  - Added to the same Architech VM used for Yocto trainings
  - Tested on Microzed

Yocto Kernel Module

A code sample to start from

- **The supplied driver is NOT**
  - Using Device Tree model
    - It will map to a fixed 22 major number device , easier to read
    - A Device Tree based version will be available later
  - Allocating dynamic kernel memory
    - It will write to absolute address 0x1001 0000, easier to read, must instruct kernel to avoid this memory area at boot time
    - It's the right approach when big memory space is required
    - A dynamic allocation version will be available later
  - Including the header files automatically generated by HLS/Vivado
    - Registers and interrupt are hard coded in the source code
    - Easier to read, a cleaner version will be available later

Yocto Kernel Module

Addresses, offsets, interrupt number definition

hello.c (~/yocto/meta-xilinx/recipes-myboard/hello-mod/files) - gedit

Open Save Undo

hello.c

```c
MODULE_LICENSE("GPL");

#define AMASTER_BASE    0x43C00000      // copy paste from Vivado address editor
#define XEXAMPLE_MYAXIL_ADDR_BYTE_WROFFSET_DATA 0x14    // copy paste from xexample_hw.h, HLS generated
#define XEXAMPLE_MYAXIL_BITS_BYTE_WROFFSET_DATA 32
#define XEXAMPLE_MYAXIL_ADDR_CTRL_REG_I_DATA    0x1c
#define XEXAMPLE_MYAXIL_BITS_CTRL_REG_I_DATA    32
#define XEXAMPLE_MYAXIL_ADDR_STATUS_REG_O_DATA  0x24
#define XEXAMPLE_MYAXIL_BITS_STATUS_REG_O_DATA  32
#define XEXAMPLE_MYAXIL_ADDR_STATUS_REG_O_CTRL  0x20

#define AMASTER_OFFSET_W        (AMASTER_BASE+XEXAMPLE_MYAXIL_ADDR_BYTE_WROFFSET_DATA)
#define AMASTER_CTRL_W          (AMASTER_BASE+XEXAMPLE_MYAXIL_ADDR_CTRL_REG_I_DATA)
#define AMASTER_STATUS_R        (AMASTER_BASE+XEXAMPLE_MYAXIL_ADDR_STATUS_REG_O_DATA)

#define LOAD_VAL        0x10010000      // HLS IP will write to this address. Warning: no mem allocation done
#define START           0x00000001
#define STOP            0x00000000

#define IRQ_NUM         61              // interrupt line as per IPI Vivado connection to Zynq IRQ_F2P[0:0]

#define DEVICE_NAME "syscall"           // device name
#define SYSCALL_MAJOR 22                // device major number
#define BUF_LEN 80                      // max buffer length
#define SUCCESS 0                       // success return value

unsigned long *pAMASTER_OFFSET_W;
unsigned long *pAMASTER_CTRL_W;
unsigned long *pAMASTER_STATUS_R;
```

Yocto Kernel Module

Write syscall: allow to enable/disable interrupt generation from HLS IP

hello.c (~/yocto/meta-xilinx/recipes-myboard/hello-mod/files) - gedit

Open Save Undo

hello.c

```c
// write routine (called when write() is used in user-space)
ssize_t syscall_write(struct file *flip, const char *buf, size_t length, loff_t *offset)
{
        printk("syscall_write.\n");                             // debug: procedure call message
        if (length > BUF_LEN)
                {
                        length = BUF_LEN;
                        printk ("length cut to %d \n", BUF_LEN);
                }
        if (copy_from_user(msg, buf, length) != 0)              // read buffer from user space
                return -EFAULT;                                 // return error if it failed
        printk("Received: %s \n",msg);                          // debug: what string is received

        if (msg[0] == '1') //(strcmp(msg,"1") == 0)                           // enable IP interrupt
        {
                printk("Driver enables master.\n");
                iowrite32(LOAD_VAL,pAMASTER_OFFSET_W);
                iowrite32(START,pAMASTER_CTRL_W);
                return length;
        }
        else if (msg[0] == '0') // (strcmp(msg,"0") == 0)                             // disable IP interrupt
        {
                printk("Driver disables master.\n");
                iowrite32(STOP,pAMASTER_CTRL_W);
                return length;
        }
        else
        {
                printk("Driver received wrong value.\n");       // Print error message
                return -EFAULT;                                 // unknown value received
        }
}
```

C Tab Width: 8 Ln 29, Col 1

Yocto Kernel Module

Read syscall: return total write transactions done by HLS IP to memory

```
// read routine (called when read() is used in user-space)
ssize_t syscall_read(struct file *flip, char *buf, size_t length, loff_t *offset)
{
        unsigned long tval;
        unsigned int msg_len;
        // read status value

        printk("syscall_read.\n");                          // debug: procedure call message
        tval = ioread32(pAMASTER_STATUS_R);                 // read status register from HLS IP = transactions count
        printk("Read() call value : %lu Cycles\n",tval);    // display status register value
        printk("length=%d\n",length);
        sprintf(msg,"%d",tval);                             //return tval as a string
        msg_len=strlen(msg);
        if (length < msg_len)
                msg_len = length;
        /*
         * If file position is non-zero, then assume the string has
         * been read and indicate there is no more data to be read.
         */
        if (*offset != 0)
                return 0;
        /*
         * Tell the user how much data we wrote.
         */
        *offset = msg_len;
        if (copy_to_user(buf, &msg, msg_len) != 0)          // send counter value
                return -EFAULT;
        else
                return msg_len;
}
```

Yocto Kernel Module

Irq handler routine: clear HLS IP pending interrupt and disable it after 100 calls

hello.c (~/yocto/meta-xilinx/recipes-myboard/hello-mod/files) - gedit

Open Save Undo

hello.c

```c
//  interrupt handler
static irqreturn_t irq_handler(int irq,void*dev_id)
{
        unsigned long temp;
        unsigned long statusvalue;

        statusvalue = ioread32(pAMASTER_STATUS_R);
        printk("Interrupt! Status counter value : %lu Cycles\n",statusvalue );
        iowrite32(2 ,pAMASTER_CTRL_W); //clear int while int gen disabled

        intcount++;
        if (intcount>=100)                                      // after 100 interrupts
        {
                printk("100 interrupts have been registered.\nDisabling master");// print status message
                iowrite32(STOP,pAMASTER_CTRL_W);
        }
        else
                iowrite32(START,pAMASTER_CTRL_W);

        return IRQ_HANDLED;
}
```

Yocto Kernel Module

Module init:link ISR to interrupt, map IP registers,write dest address and create device num. 22

hello.c (~/yocto/meta-xilinx/recipes-myboard/hello-mod/files) - gedit

Open Save Undo

hello.c

```c
static int __init mod_init(void)
{
        unsigned long temp;
        printk(KERN_ERR "Init syscall module. \n");
        if (request_irq(IRQ_NUM,irq_handler,IRQF_DISABLED, DEVICE_NAME, NULL))  //request interrupt
        {
                printk(KERN_ERR "Not Registered IRQ. \n");
                return -EBUSY;
        }
        printk(KERN_ERR "Registered IRQ. \n");
        pAMASTER_STATUS_R = ioremap_nocache(AMASTER_STATUS_R,0x4);
        pAMASTER_OFFSET_W = ioremap_nocache(AMASTER_OFFSET_W,0x4);
        pAMASTER_CTRL_W = ioremap_nocache(AMASTER_CTRL_W,0x4);
        iowrite32(LOAD_VAL,pAMASTER_OFFSET_W);
        iowrite32(START,pAMASTER_CTRL_W);       //uncomment for HLS IP transations autostart
        intcount = 0;                                   // set interrupt count to 0, driver will unload on 100 interrupts
        /* => to be completed, allocation of memory, Here's the allocation of a single quantum */
        /*      if (! PAGE_SIZE<<log2()) {
            dptr->data[s_pos] =
              (void *)__get_free_pages(GFP_KERNEL, dptr->order);
            if (!dptr->data[s_pos])
                goto nomem;
            memset(dptr->data[s_pos], 0, PAGE_SIZE << dptr->order);
        }*/
        // manual node creation => easier to understand, but in future adopt device tree model
        if (register_chrdev(SYSCALL_MAJOR, DEVICE_NAME, &syscall_fops))
                printk("Error: cannot register to major device 22.\n");

        if (irq_set_irq_type(IRQ_NUM, IRQ_TYPE_EDGE_RISING))
                printk("Error: cannot enable rising edged on interrupt %d\n",IRQ_NUM);
        else    printk("Rising edge enabled on interrupt %d\n",IRQ_NUM);

        printk("Type: mknod /dev/%s c %d 0\n",DEVICE_NAME, SYSCALL_MAJOR);
        printk("And remove it after unloading the module.\n");

        return SUCCESS;
}
```

C ▾ Tab Width: 8 ▾ Ln 169, Col 51

Yocto Kernel Module

Recompiling hello-mod recipe

- **bitbake hello-mod**
  - Will create $HOME/yocto/build/tmp/sysroots/zedboard-zynq7/lib/modules/3.8.11-xilinx/extra/hello.ko
- **bitbake core-image-minimal**
  - Will create the sysroot image containing the hello.ko module at /lib/modules/3.8.11-xilinx/extra

Demo

Microzed: Architech Yocto booting from SD card, kernel module already included by recipe

COM23:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
Sending discover...
No lease, failing
hwclock: can't open '/dev/misc/rtc': No such file or directory
Mon Jan 12 10:49:00 UTC 2015
hwclock: can't open '/dev/misc/rtc': No such file or directory
INIT: Entering runlevel: 5
hwclock: can't open '/dev/misc/rtc': No such file or directory
Starting syslogd/klogd: done
Stopping Bootlog daemon: bootlogd.
Starting tcf-agent: OK

Poky (Yocto Project Reference Distro) 1.5 zedboard-zynq7 /dev/ttyPS0

zedboard-zynq7 login: root
root@zedboard-zynq7:~# ls /lib/
ld-2.18.so                libdl.so.2                libresolv.so.2
ld-linux.so.3             libm-2.18.so              librt-2.18.so
libBrokenLocale-2.18.so   libm.so.6                 librt.so.1
libBrokenLocale.so.1      libnsl-2.18.so            libthread_db-1.0.so
libanl-2.18.so            libnsl.so.1               libthread_db.so.1
libanl.so.1               libnss_compat-2.18.so     libtinfo.so.5
libblkid.so.1             libnss_compat.so.2        libtinfo.so.5.9
libblkid.so.1.1.0         libnss_dns-2.18.so        libutil-2.18.so
libc-2.18.so              libnss_dns.so.2           libutil.so.1
libc.so.6                 libnss_files-2.18.so      libuuid.so.1
libcrypt-2.18.so          libnss_files.so.2         libuuid.so.1.3.0
libcrypt.so.1             libpthread-2.18.so        modules/
libcrypto.so.1.0.0        libpthread.so.0           udev/
libdl-2.18.so             libresolv-2.18.so
root@zedboard-zynq7:~# ls /lib/modules/3.8.11-xilinx/extra/
hello.ko
root@zedboard-zynq7:~#
```

Demo

Microzed: loading kernel module, the IP transactions and interrupt generation are triggered

COM23:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
root@zedboard-zynq7:~# modprobe hello
Init syscall module.
Registered IRQ.
Rising edge enabled on interrupt 61
Interrupt! Status counter value : 128 Cycles
Interrupt! Status counter value : 256 Cycles
Interrupt! Status counter value : 256 Cycles
Interrupt! Status counter value : 384 Cycles
Interrupt! Status counter value : 512 Cycles
Interrupt! Status counter value : 640 Cycles
Interrupt! Status counter value : 768 Cycles
Interrupt! Status counter value : 896 Cycles
Interrupt! Status counter value : 1152 Cycles
Interrupt! Status counter value : 1152 Cycles
Interrupt! Status counter value : 1280 Cycles
Interrupt! Status counter value : 1408 Cycles
Interrupt! Status counter value : 1536 Cycles
Interrupt! Status counter value : 1664 Cycles
Interrupt! Status counter value : 1792 Cycles
Interrupt! Status counter value : 1920 Cycles
Interrupt! Status counter value : 2048 Cycles
Interrupt! Status counter value : 2176 Cycles
Interrupt! Status counter value : 2304 Cycles
Interrupt! Status counter value : 2432 Cycles
Interrupt! Status counter value : 2560 Cycles
Interrupt! Status counter value : 2688 Cycles
Interrupt! Status counter value : 2816 Cycles
Interrupt! Status counter value : 2944 Cycles
```

```
Interrupt! Status counter value : 10240 Cycles
Interrupt! Status counter value : 10368 Cycles
Interrupt! Status counter value : 10496 Cycles
Interrupt! Status counter value : 10624 Cycles
Interrupt! Status counter value : 10752 Cycles
Interrupt! Status counter value : 10880 Cycles
Interrupt! Status counter value : 11008 Cycles
Interrupt! Status counter value : 11136 Cycles
Interrupt! Status counter value : 11264 Cycles
Interrupt! Status counter value : 11392 Cycles
Interrupt! Status counter value : 11520 Cycles
Interrupt! Status counter value : 11648 Cycles
Interrupt! Status counter value : 11776 Cycles
Interrupt! Status counter value : 11904 Cycles
Interrupt! Status counter value : 12032 Cycles
Interrupt! Status counter value : 12160 Cycles
Interrupt! Status counter value : 12288 Cycles
Interrupt! Status counter value : 12416 Cycles
Interrupt! Status counter value : 12544 Cycles
Interrupt! Status counter value : 12672 Cycles
100 interrupts have been registered.
Disabling masterInterrupt! Status counter value : 12800 Cycles
Type: mknod /dev/syscall c 22 0
And remove it after unloading the module.
root@zedboard-zynq7:~#
```

Demo

Microzed: making node and sending '1' to re-trigger 128 transactions and interrupt IP

```
root@zedboard-zynq7:~# mknod /dev/syscall c 22 0
root@zedboard-zynq7:~# echo 1 > /dev/syscall
syscall_write.
Received: 1
u tried to open the syscall module.

Driver enables master.
Interrupt! Status counter value : 12928 Cycles
100 interrupts have been registered.
Disabling master
```

```
Disabling masterroot@zedboard-zynq7:~# echo 1 > /dev/syscall
syscall_write.
Received: 1
u tried to open the syscall module.

Driver enables master.
Interrupt! Status counter value : 13056 Cycles

100 interrupts have been registered.
Disabling masterInterrupt! Status counter value : 13056 Cycles
100 interrupts have been registered.
Disabling masterroot@zedboard-zynq7:~# echo 1 > /dev/syscall
syscall_write.
Received: 1
u tried to open the syscall module.

Driver enables master.
Interrupt! Status counter value : 13184 Cycles

100 interrupts have been registered.
Disabling masterroot@zedboard-zynq7:~# Interrupt! Status counter value : 13184 Cycles
100 interrupts have been registered.
Disabling master
```

Demo

Microzed: making sure expected addresses have been written

- The HLS IP has been sw initialized to write DDR from address 0x10010000
- When triggered, the IP will write N=128 words (=32bit) from Axi Stream to DDR
- The Axi Stream is fed by a const value, 0xCA = 0xb11001010
- We expect address range 0x10010000 – 0x100101FF filled by 0xCA
- We'll use rwmem utility (see Architech Yocto Training) to dump absolute memory address

```
root@zedboard-zynq7:~#
root@zedboard-zynq7:~# rwmem 0x10010000
0x0000000010010000 = 0xcacacaca
root@zedboard-zynq7:~# rwmem 0x100101f0
0x00000000100101f0 = 0xcacacaca
root@zedboard-zynq7:~# rwmem 0x10010200
0x0000000010010200 = 0xf77ffdff
root@zedboard-zynq7:~#
```