Xilinx Tools on Linux

These notes are for ISE Design Suite 10.1 (applicable to ISE WebPack 10.1)... also, the later ones when implied.

General Stuff

Warning!

I found this - so remember... `libCurl_Curl.so`!

Information

I've also downloaded ISE 12.1 and found out that it is now using flexlm-based license. So, the easiest way to setup the license is to place it in `~/.Xilinx`. The license file name need to be exactly `Xilinx.inc` (case-sensitive, as you'd have received it from Xilinx).

The TCL script I created with 10.1 may still be usable after all in 12.1. The project name extension HAS to be `.xise` instead of the old `.ise` and the some of the old cores (LogiCore) are no longer supported - has to be recreated. There may be other problems that haven't surfaced yet - I'm still considering whether to move to 12.1 or just write custom VHDL codes to replace the LogiCores.

Also, the project option for simulation tool using Xilinx ISE Simulator gets a new label - “ISim (VHDL/Verilog)” instead of the old one “ISE Simulator (VHDL/Verilog)”.

Changing time resolution for ISIM - edit `ISim Properties→Advanced→Other Compiler Options` and type in *-timescale 10ns/10ns -override_timeprecision -override_timeunit* (timescale = unit/precision, default is 10ns/1ps). The *-override_** options are for overriding verilog's '*timescale* command?! For VHDL, use *-timeprecision_vhdl 10ns* (default is 1ps).

ISIM supports VHDL2000 and Verilog2001 (I don't care about other HDLs). Of course, it also supports a mixed VHDL/Verilog design.

So, the free ISIM Lite has a limitation after all (I think they don't have this in earlier versions - e.g. ISE10 and older). So Based on Xilinx ISIM user guide UG660 (version 12.4) dated 14/12/2010, when “*...design plus testbench exceeds 50,000 lines of HDL code, the simulator begins to derate the performance of the simulator for that vocation*”. Hmm.... I'll try to check if the `xst` synthesis tools on WebPack free license has similar limitations.

FPGA Design Flow

This is a summary for FPGA design flow using Xilinx tools that comes with ISE.

```
- Synthesis Tool (xst)

= [formal] creates a netlist from design entry
= [plain] identifies component (fsm, reg, mux, etc.) structure
= outputs ngc file (Xilinx netlist format)
= optionally outputs a log file
= creates other files like lso (library search order)
= requires temporary directory (xst - but can be set)
= need input project prj file (list of design files)
= need input command xst file (xst command options)

- Translate Tool (ngdbuild)

= [formal] generates a database of required resources on specified package
= [plain] translates the netlist into physical component in an fpga package
= will fail if the specified package cannot provide enough resources
= outputs ngd file (Xilinx data format)
= creates temporary bld file
= requires temporary directory (_ngo - but can be set)
= need input netlist ngc file
= need input contraints ucf file (includes pin assignments)

- Mapping Tool (map)

= [formal] maps requested resources to specific area on the fpga
= [plain] puts the physical component into their actual location on fpga
= can be time-driven (which puts connected component close to avoid delay)
= outputs an intermediate ncd file (placement file format) & pcf file
= optionally outputs a usage file
= creates mrp file (details), ngm file
= need input database ngd file

- Place & Route Tool (par)

= [formal] refine resource allocation to comply with timing constraits
= [plain] rearrange the resources so that the design performs better
= outputs final ncd file (placement file format)
= creates pad, par, unroutes xpi, _pad.csv, _pad.txt file
= need input intermediate placement ncd file
= need input physical constraints pcf file

- Bitstream Generation Tool (bitgen)

= [formal] creates final bitstream from placement file
= [plain] generate the bitstream file to be programmed onto the fpga
= outputs bit file (bitstream file)
= creates bgn, drc file
= need input final placement ncd file

- (Optional) Simulation File Creation Tool (trce)
```

```
= back-annotation?
= creates twr file from ncd & pcf
```

VHDL Library Selection

When working with VHDL for hardware logic design (i.e. for synthesis to be used on programmable logic hardware), using IEEE std_logic_1164 package is a must. This is the core package to work with low level signals. When you need arithmetic package there are 2 options:

1. The std_logic_arith package: this is an extension by Synopsys that was widely used by hardware vendors. However, there is a slight possibility that this will not be supported by tools that decided not to implement them since they are not actually a defined standard.
2. The numeric_std package: this is the standard defined by IEEE. However, it is relatively new compared to std_logic_arith - so, the possibility of this not being implemented by a tool is even greater. However, at the time of writing, it can be confirmed that Xilinx ISE suite and Modelsim simulator both support this library.

I was more familiar with the std_logic_arith package - but somehow I just experienced something that might just push me towards using the other package. Now, to use signed arithmetic, we need another package called std_logic_signed (surprise, surprise...). It turns out, conv_integer for both signed and unsigned type will yield signed results and adding std_logic_unsigned will create an error because it will try to define the same set of functions. Using numeric_std gave no problems at all. So, maybe it's time I get standardized.

Tcl Interface

It's an abbreviation for Tool Command Language - but became 'Tcl' (notice only first letter is in capital). It's a scripting language (has it's own shell environment through 'tclsh') mainly embedded into other applications. I mainly use it to automate build of Xilinx FPGA bitstream (without GUI).

Running the shell:

```
# tclsh
```

Getting the info:

```
% info tclversion
8.5
% info patchlevel
8.5.7
% package require Tcl
8.5.7
```

Read more @ wikipedia

Installation Issues

Linux USB Cable

Installing cable driver is a mess. There's this problem with the one supplied by Xilinx (proprietry kernel module *windrv* by Jungo?) that is said to be broken with every new release of ISE. So, luckily I found this website. Downloaded libusb driver and the cable gets detected. Phew! I thought I had to move back to Window$!

Enforced Path-name

ISE 12.1 (I guess starting from ISE 11) starts to impose ISE_DS in the installation path. This does not work well with my script. I know I can change my script, but then the installation path will be a bit long and this has, in the past, pose some problems. A solution is to edit all `settings*.sh` file (i.e. root, ISE, EDK and common folders) and change the path to something like what I have in my script. This way, I can install multiple version of ISE and run this script with `XILINX_VERS=xx.x xil-exec` (xx.x being the desired version).

chpath.sh

```
#!/bin/bash

CURRPATH=$(pwd)
CONFFILE=$(find $CURRPATH -maxdepth 1 -name "settings*.sh" -type f)

[[ "$CONFFILE" == "" ]] && echo "Cannot find expected files in
'$CURRPATH' (Not an ISE_DS folder?)!" && exit 1

for settfile in $(find $CURRPATH -name "settings*.sh" -type f); do
    sed -i 's@XIL_SCRIPT_LOC="\(.*\)"@XIL_SCRIPT_LOC="'"$CURRPATH"'"@'
$settfile
    echo "$settfile updated with new installation path [$CURRPATH]!"
done
```

Digilent USB Cable

To use Digilent USB JTAG cable, have this in `/etc/udev/rules.d/`

99-digilent-usb.rules

```
SUBSYSTEM=="usb", ACTION=="add", ATTRS{idVendor}=="1443",
ATTRS{manufacturer}=="Digilent", GROUP="users", MODE:="666",
```

```
RUN+="/usr/sbin/dftdrvdtch %s{busnum} %s{devnum}"
```

Of course, we need to install stuffs from digilent (Adept?).

Run-time Issues

Run-time issues running Xilinx tools on Slackware Linux.

Running ISE 12.1 on Slackware 14.1

When trying to run ISE 12.1 on Slackware 14.1, nothing comes up. Turns out an environment variable QT_PLUGIN_PATH is the reason for this. Unsetting this (or setting to 'null'@nothing) will make ISE 12.1 run normally.

```
QT_PLUGIN_PATH= ise
```

Note: *KDE_SESSION_VERSION variable is no longer an issue in Slackware 14.1!*

Running ISE 12.1 on KDE 4

When trying to run ISE 12.1 on KDE, it will give out a segmentation fault. Googling for it brings us to this forum. Apparently, an environment variable KDE_SESSION_VERSION is the culprit. Unsetting this will make ISE 12.1 run again.

```
unset KDE_SESSION_VERSION; ise
```

Conflicting TCL Library

Apparently running Xilinx's free ISE WebPACK tool on Slackware is not nice at first because the tcl library that comes with the tool conflicts with Slackware's tcl package. I got the solution from here, which basically sets an environment variable to override (preload) the tcl library. I just created a wrapper script to start up EDK/ISE and added these lines before actually calling xps or ise (after sourcing settings32.sh):

```
# added to override usage of system's tcl library
LD_PRELOAD=${XILINX}/lib/${PLATFORM}/libtcl8.4.so
export LD_PRELOAD
```

Update20100609 This is no longer needed for Slackware 13.0

Missing Library

I got myself the Video Starter Kit development board from Xilinx - which comes with ISE Design Suite 10.1 that includes a license for EDK and System Generator. I was trying to synthesize the demo project when it complains about a shared library *libdb-4.1.so* not found. On Slackware 12.2, I found *libdb-4.2.so* and *libdb-4.4.so* in */lib*. So, I just created a link to *libdb-4.2.so* and named it *libdb-4.1.so*.

Technical Issues

20150616 - Impact sometimes refuse to connect (USB cable)

On Linux, run this:

```
echo -e 'cleancablelock\nexit' | impact -batch
```

20121218 - Running Check Syntax in 64-bit ISE 12.1

I've just found out this issue when running the command using tcl script - I got a “undefined symbol: _ZN11xalanc_1_1016XalanTransformer9terminateEv” error message and apperently, this issue has somewhat escalated in version 13 and above. I'd say it has something to do with glibc not playing nice with static binaries, but who am I to say? :p Anyways, luckily the issue is NOT seen in the 32-bit version. So, the show must go on…

ADDED201212182133 Maybe… just maybe… it would work fine if I use multi-lib?

20100819 - GNU Cross-Compiler for Microblaze in 64-bit ISE 10.1

I found this out earlier but I just remembered to put this here. The GNU cross-compiler for Microblaze (at least some of the binaries) are actually 32-bit ELF binaries. Obviously, I can't run this on my 64-bit machine. For the record, I don't like having multi-lib configuration - my system is a pure 64-bit system. So, I have to split the bitstream and software development and synthesis/compile them separately, before merging them back into an ACE file.

20091221 - Creating SystemACE file

I managed to find out how I can create system.ace file from both executable.elf and download.bit - we need the genace.tcl script ($XILINX/EDK/data/xmd). The command is:

```
xmd -tcl genace.tcl -opt genace.opt
```

Running the command without *-opt* will give you a short description on how to use the script. I'd like to remind myself that this script requires a shell environment with Xilinx environment set (i.e. sourcing settings32.sh/settings64.sh). In addition to that, we need to include these path along (don't know why settingsxx.sh doesn't do this):

```
PATH=$XILINX/EDK/gnu/microblaze/lin/bin:$XILINX/EDK/gnu/powerpc-
eabi/lin/bin:$PATH
```

Without these paths (actually you only need one depending on which processor core you're using), the script will not be able to extract information like the start address from the ELF file. The path is where the related tools are located.

Unfortunately, this script only caters for only a few of the development board and the VSK is NOT one of them. It's usable as long as we know how to obtain the relevant information. So, after a few days of looking around, here's how the options file look like:

```
-jprog
-board user
-target mdm
-hw implementation/download.bit
-elf Camera_Frame_Buffer_Sw/executable.elf
-configdevice devicenr 1 idcode 0x05059093 irlength 16 partname xcf32p
-configdevice devicenr 2 idcode 0x06e59093 irlength 8 partname xc2c64a
-configdevice devicenr 3 idcode 0x0a001093 irlength 8 partname xccace
-configdevice devicenr 4 idcode 0x0384e093 irlength 6 partname xc3sd3400a
-debugdevice devicenr 4 cpunr 1
-ace system.ace
```

The hardest part was to find the IDcode/irlength/partname of each device in the JTAG chain. The `-hw`, `-elf` and `-ace` options are obviously the input/output file names (i.e. create ace from hw and elf). The `-jprog` option will clear the existing FPGA configuration. We will need this unless we're doing runtime (partial?) reconfiguration. The `-target` option specifies the target for downloading ELF/data file. We're using Xilinx's Microblaze softcore (`mdm`) and the default is `ppc_hw`. The `-board` option gives a few options (e.g. ML300, memec) but we're going to specify this manually (I just found out that we can specify `auto` here but we need to switch on and connect to the board for that). So, we'll be using `user` for this and we need to use `-configdevice` and `-debugdevice` directives.

We can get the chain information (device number) and part names by running Impact (with the board connected and switched on). We can also get the IDcodes here but somehow I can't get the one for `xccace` (ACECF). And, we actually need to specify this or the script WILL NOT work! I got that and the IRlength (INSTRUCTION_LENGTH) information from the respective BSD file that we need to look for in respective subdirectories in $XILINX/ISE/.

So, with all these, I managed to create a system.ace file - but I haven't actually test this. My card reader is not functioning and I need another way to copy the file to the CF card. I'll try to do that a.s.a.p.

Update20100119 Okay... it did not get me the desired results. The created ACE file was not loaded properly! Where did I do wrong??

Update20100121 Need to look into `startupclk = jtagclk` option for `bitgen`
Update20100124 It's already set that way!?!

Update20100520 Thanks to Vit Zikmund who emailed me a solution for this! I managed to get it working from the created ACE file! So... the final configuration file is as shown below.

```
-jprog
-board user
-target mdm
-hw download.bit
-elf executable.elf
-configdevice devicenr 1 idcode 0x0384e093 irlength 6 partname xc3sd3400a
-debugdevice devicenr 1 cpunr 0
-ace system.ace
```

20091217 - Softapps too big for bitstream

There are times (like what I'm facing while working on the Xilinx Spartan3A Video Starter Kit) when the software (e.g. executable.elf file) is too big to be embedded in the bitstream. As a solution, we can try to convert the application into a system ace file. I still can't do that but for the time being, here's a usable solution:

1. download the bitstream file as usual, but you most probably can't see any difference since the software part is still not running

2. open up XMD debugger interface that will take you to an XMD prompt

3. type these commands:

```
XMD% Stop
XMD% dow path/to/executable.elf
XMD% run
```

4. you should now have the application running as expected

note the XMD console is a PITA - the backspace needs <CTRL> key to be pushed as well and it's not aware of the filesystem at all! I want BASH! 

20091215 - Using Xilinx EDK Tool

Now I remember why I hate working with EDK. Synthesis took forever... and I can't create/modify TCL scripts for building the project (like the one we have on ISE). Well, at least the features in EDK 10.1 is a lot better compared to those in pre-8.0 versions. I'll just have to bear with it a little bit longer.

I need to work with the XtremeDSP Video Starter Kit (Spartan-3A DSP Edition). What makes it a little bit tricky is, I am on Slackware Linux - which is not directly supported by Xilinx and, most tools are developed for Windows (ironically enough, the EDK shell is using Cygwin). System Generator (for DSP-

based designs) does not run at all on Linux. Before I forget, I got the older 10.1 ISE Design Suite. Which is fine because I have been using ISE WebPack 10.1 for my previous work.

Custom Solutions

Stuffs to share...

Startup Script

Based on the issues discussed above, I have created a bash startup script for my Xilinx Tool.

Updated 20100715 This one adds support for 64-bit Linux.

xil-exec

```
#!/bin/bash

XILINX_VERS=${XILINX_VERS:="10.1"}
XILINX_PATH=${XILINX_PATH:="/home/ftp/software/xilinx-$XILINX_VERS"}
XILINX_ARCH="32"
[[ "$(uname -m)" == "x86_64" ]] && XILINX_ARCH="64"
XILINX_CONF=${XILINX_CONF:="$XILINX_PATH/ISE/settings$XILINX_ARCH.sh"}
XILEDK_CONF=${XILEDK_CONF:="$XILINX_PATH/EDK/settings$XILINX_ARCH.sh"}
PATH_ARCH=""
[[ "$XILINX_ARCH" == "64" ]] && PATH_ARCH=$XILINX_ARCH

if [ -r $XILINX_CONF ] ; then
    source $XILINX_CONF
else
    echo "Cannot find $XILINX_CONF!"
    exit 1
fi

if [ -r $XILEDK_CONF ] ; then
    source $XILEDK_CONF
else
    echo "Cannot find $XILEDK_CONF!"
    exit 1
fi

# allow ISE/EDK to use the USB Cable Driver
#export XIL_IMPACT_USE_LIBUSB=1
# added to override usage of system's tcl library
# may not need this on slack13.0 (tcl8.5.7)
#export LD_PRELOAD=${XILINX}/lib/${PLATFORM}/libtcl8.4.so
# added to use a working cable driver
```

```
#export LD_PRELOAD="$LD_PRELOAD /home/ftp/software/usb-driver/libusb-driver.so"

if [ "$1" == "" ] ; then
    # start a shell
THISPATH="$XILINX_PATH/EDK/gnu/microblaze/lin${PATH_ARCH}/bin:$XILINX_PATH/EDK/gnu/powerpc-eabi/lin${PATH_ARCH}/bin"
    export PATH=$THISPATH:$PATH
    exec env PS1="(XIL-EXEC) \u@\h:\w$ " $(which bash)
else
    target=$1
    shift
    [ "$target" == "edk" ] && xps $1
    [ "$target" == "ise" ] && ise $1
    [ "$target" == "imp" ] && impact $1
fi

exit 0
```

Bitstream File Info

Something I wrote based on some info I got on the internet... it has been a while since I last used this. It was working... I think.

xbitinfo.c

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

#define BIT_HEAD_LENGTH 0x0009
#define ONE_HEAD_LENGTH 0x0001
/* LITTLE-ENDIAN of 0xaa995566 */
#define DATA_SYNC 0x665599aa

typedef unsigned long kword;
typedef unsigned short hword;
typedef unsigned char kbyte;

typedef struct _bitsdata
{
    kword sizeup;
    hword length;
    kbyte *pdata;
}
bitsdata;

hword swapbyte(hword test)
{
```

```
    hword temp = test & 0xff;
    test >>= 8;
    temp <<= 8;
    temp |= test;
    return temp;
}

kword swapword(kword test)
{
    kword temp = test & 0xff;
    test >>= 8;
    temp <<= 8;
    temp |= test & 0xff;
    test >>= 8;
    temp <<= 8;
    temp |= test & 0xff;
    test >>= 8;
    temp <<= 8;
    temp |= test & 0xff;
    return temp;
}

int getalpha(FILE* pfile, char test)
{
    char tdata;
    void *ptest = (void*) &tdata;
    fread(ptest,1,sizeof(char),pfile);
    if(tdata==test) return 1;
    else return 0;
}

int getlength(FILE* pfile, hword* pdata)
{
    int retval;
    void *ptest = (void*) pdata;
    retval = fread(ptest,1,sizeof(hword),pfile);
    *pdata = swapbyte(*pdata);
    return retval;
}

int getsizeup(FILE* pfile, kword* pdata)
{
    int retval;
    void *ptest = (void*) pdata;
    retval = fread(ptest,1,sizeof(kword),pfile);
    *pdata = swapword(*pdata);
    return retval;
}

int createdata(FILE* pfile, bitsdata* pbits)
{
```

```
    kbyte* ptemp = 0x0;
    kword sizeup = 0x0;
    hword length = 0x0;
    void* ptest;
    int result;
    /* get data length */
    if(pbits->sizeup)
    {
        getsizeup(pfile,&sizeup);
        length = sizeup;
    }
    else
    {
        getlength(pfile,&length);
        sizeup = length;
    }
    /* create storage */
    ptemp = malloc(sizeup);
    if(!ptemp) return -1;
    /* setup structure */
    pbits->pdata = ptemp;
    pbits->length = length;
    pbits->sizeup = sizeup;
    /* read in data, return read count */
    ptest = (void*) pbits->pdata;
    result = fread(ptest,1,sizeup,pfile);
    if(result!=(int)sizeup)
    {
        printf("ERROR? Result: %d [%08x], Sizeup: %d [%08x]\n",
            result,result,sizeup,sizeup);
    }
    return result;
}

int deletedata(bitsdata* pbits)
{
    if(pbits->pdata) free(pbits->pdata);
    pbits->pdata = 0x0;
    pbits->length = 0;
    pbits->sizeup = 0;
    return 0;
}

int main(int argc, char *argv[])
{
    FILE *pfile;
    int count;
    bitsdata xdata = { 0,0,0x0 };
    kword* datasync;
    kbyte headmask = 0x0f;
    char protocol = 'a';
```

```
    if(argc<2)
    {
        printf("%s <bitfile>.bit\n",argv[0]);
        return -1;
    }

    pfile = fopen(argv[1],"rb");
    if(!pfile)
    {
        printf("Cannot open Xilinx bitstream file %s!\n",argv[1]);
        return -2;
    }

    /* get file header */
    if(createdata(pfile,&xdata)<0)
    {
        printf("error creating file header!\n");
        fclose(pfile);
        return -1;
    }
    /* check file header length */
    if(xdata.length!=BIT_HEAD_LENGTH)
    {
        printf("invalid %04x format?
[%04x]\n",BIT_HEAD_LENGTH,xdata.length);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
    /* check file header */
    for(count=0;count<BIT_HEAD_LENGTH-1;count++)
    {
        if(xdata.pdata[count] != headmask)
        {
            printf("invalid %02x format? [%02x]/[%d]\n",
                headmask,xdata.pdata[count],count);
            deletedata(&xdata);
            fclose(pfile);
            return -1;
        }
        headmask = ~headmask;
    }
    if(xdata.pdata[BIT_HEAD_LENGTH-1]!=0x00)
    {
        printf("invalid null tail? [%02x]\n",
            xdata.pdata[BIT_HEAD_LENGTH-1]);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
```

```
    /* data not used anymore! */
    deletedata(&xdata);
    /* check special length data? -> not in pattern? */
    getlength(pfile,&xdata.length);
    if(xdata.length!=ONE_HEAD_LENGTH)
    {
        printf("invalid %04x format? [%04x]\n",
            ONE_HEAD_LENGTH,xdata.length);
        fclose(pfile);
        return -1;
    }
    /* check protocol char */
    if(!getalpha(pfile,protocol))
    {
        printf("protocol char [%c] not found!\n", protocol);
        fclose(pfile);
        return -1;
    }
    /* check design name */
    if(createdata(pfile,&xdata)<0)
    {
        printf("error getting design name!\n");
        fclose(pfile);
        return -1;
    }
    /* validate data */
    if(xdata.pdata[xdata.length-1]!=0x00)
    {
        printf("invalid data terminator? [%02x]\n",
            xdata.pdata[xdata.length-1]);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
    /* print design name */
    printf("Design name: %s\n",(char*)xdata.pdata);
    deletedata(&xdata);
    /* check next protocol char */
    protocol++;
    if(!getalpha(pfile,protocol))
    {
        printf("protocol char [%c] not found!\n", protocol);
        fclose(pfile);
        return -1;
    }
    /* check part name */
    if(createdata(pfile,&xdata)<0)
    {
        printf("error getting part name!\n");
        fclose(pfile);
        return -1;
```

```
    }
    /* validate data */
    if(xdata.pdata[xdata.length-1]!=0x00)
    {
        printf("invalid data terminator? [%02x]\n",
            xdata.pdata[xdata.length-1]);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
    /* print part name */
    printf("Part name: %s\n",(char*)xdata.pdata);
    deletedata(&xdata);
    /* check next protocol char */
    protocol++;
    if(!getalpha(pfile,protocol))
    {
        printf("protocol char [%c] not found!\n", protocol);
        fclose(pfile);
        return -1;
    }
    /* check string date  */
    if(createdata(pfile,&xdata)<0)
    {
        printf("error getting string date!\n");
        fclose(pfile);
        return -1;
    }
    /* validate data */
    if(xdata.pdata[xdata.length-1]!=0x00)
    {
        printf("invalid data terminator? [%02x]\n",
            xdata.pdata[xdata.length-1]);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
    /* print string date */
    printf("Date created: %s\n",(char*)xdata.pdata);
    deletedata(&xdata);
    /* check next protocol char */
    protocol++;
    if(!getalpha(pfile,protocol))
    {
        printf("protocol char [%c] not found!\n", protocol);
        fclose(pfile);
        return -1;
    }
    /* check string time  */
    if(createdata(pfile,&xdata)<0)
    {
```

```
        printf("error getting string time!\n");
        fclose(pfile);
        return -1;
    }
    /* validate data */
    if(xdata.pdata[xdata.length-1]!=0x00)
    {
        printf("invalid data terminator? [%02x]\n",
            xdata.pdata[xdata.length-1]);
        deletedata(&xdata);
        fclose(pfile);
        return -1;
    }
    /* print string time */
    printf("Time created: %s\n",(char*)xdata.pdata);
    deletedata(&xdata);
    /* check next protocol char */
    protocol++;
    if(!getalpha(pfile,protocol))
    {
        printf("protocol char [%c] not found!\n", protocol);
        fclose(pfile);
        return -1;
    }
    /* check raw bits  */
    xdata.sizeup = 1;
    if(createdata(pfile,&xdata)<0)
    {
        printf("error getting raw bits!\n");
        fclose(pfile);
        return -1;
    }
    /* validate raw bit stream */
    datasync = (kword*) xdata.pdata;
    printf("Dummy word => [%08x]\n", *datasync);
    datasync++;
    printf("DSYNC word => [%08x]\n", *datasync);
    if(*datasync!=DATA_SYNC)
        printf("invalid raw data sync? [%08x]\n",DATA_SYNC);
    /* print raw bits */
    printf("Raw bitsize: %d bytes\n",xdata.sizeup);
    deletedata(&xdata);

    fclose(pfile);
    return 0;
}
```

Using Digilent JTAG-USB Cable

- install digilent adept runtime & utilities

```
= need to tweak udev rules a bit... (do i really have to?)
```

- install libusb - read the README file that comes with the plugin! how to select using that plugin!

to be continued...