Cd Pinout: Meir Fogel <[meir.f@horizon-pss.com](mailto:meir.f@horizon-pss.com)>

R&d manager **Bar Shpilman** <[Bar.s@horizon-pss.com](mailto:Bar.s@horizon-pss.com)>

AI: prepare a excel with flow from me to client

Questions for Kickoff:

* Who is in charge of pinout?
* We need to construct the pinout table = what to do with each pin.
* We need to verify the pinout. Using Vivado
* Is there a requirement doc for the FPGA? The full product specification is very large we need to break it down for the FPGA.
* MS1: What do you need in the PDR? (PPT?, more docs?)
* MS2: Method to prove the design.
* MS3: what do you need for documentation? Method to prove the integration.
* Contact points (HW? SW delivery? Requirements and protocols?)
* Log file not defined yet
* I need go over all requirements with someone and designate how it is done HW/SW
* Define power up bit
* If Periodic bit part of the repeatable status 100 Hz?
* List of documents required (we can deliver version doc, register list)
* We need 2 boards
* Log file trailer has checksum this is redundant and very hard to implement.

Microzed board

Add to ps pins for UART PS

Installing:

(updating ) Virtual box (to version) 7.0.12

Adding a new Ubuntu machine to VBOX from:

The following in red was deleted but saved here for reference

<https://www.releases.ubuntu.com/jammy/> => ubuntu-22.04.3-desktop-amd64.iso

usr: mati pswd: mati

Automatic install: I gave it 32 G ram 14 cpus, and 25G HD

Internet is working fine. Screen resolution: could only set to FHD.

Moved VM from C: to D: (next time check why it has not put it on D…)

Enabled shared clipboard bidirectional

Install hebrew

Changed to automatic login

Shard a folder at: D:\Mati\vms\Ubuntu\shared into /media/sf\_shared

mati was not part of sudoers

Open file

su root

nano /etc/sudoers

Then add the user below admin user like below syntax.

user\_name ALL=(ALL) ALL

sudo apt-get update

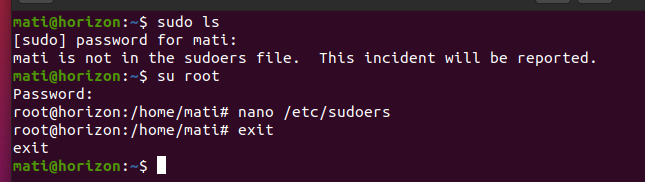
sudo apt-get update

sudo apt-get install build-essentials

sudo apt-get install dkms

installed a new machine from -> ubuntu-20.04.6-desktop-amd64.iso

then add user to sudoers:



Than:

sudo apt update

sudo apt install build-essential dkms linux-headers-$(uname -r)

sudo apt update

sudo apt install samba

sudo apt install nfs-kernel-server

in vbox -> devices -> update guest additions

when installed and ask to run, approve.

In settings -> users change to automatic login

sudo Nano /etc/init.d/mati.sh

then add to file :

**#!/bin/sh**

**/usr/bin/VBoxClient --clipboard**

**/usr/bin/VBoxClient --draganddrop**

**/usr/bin/VBoxClient --checkhostversion**

**/usr/bin/VBoxClient --seamless**

**/usr/bin/VBoxClient --vmsvga**

**/usr/bin/VBoxClient --display**

Then reboot

This also helped:

sudo apt-get install virtualbox

put the video memory to high values to allow grater screen sizes

more installations: (7.1.23)

TFTP

1. **sudo apt-get install xinetd tftpd tftp**

   edit **/etc/xinetd.d/tftp**

**paset :**

**service tftp**

**{**

**protocol = udp**

**port = 69**

**socket\_type = dgram**

**wait = yes**

**user = nobody**

**server = /usr/sbin/in.tftpd**

**server\_args = /tftpboot**

**disable = no**

**}**

**sudo mkdir /tftpboot**

**sudo chmod -R 777 /tftpboot**

**sudo chown -R nobody /tftpboot**

**sudo service xinetd restart**

NFS

2.

1. sudo apt update
2. sudo apt install nfs-kernel-server

edit /etc/exports

/home /home \*(rw,sync,no\_subtree\_check,no\_root\_squash,insecure)

SAMBA

3.

**sudo apt install samba -y**

**edit the file** etc/samba/smb.conf

**and replace it with the attached file smb.conf**

**run:** sudo smbd restart

change the network to bridge

type : **ifconfig  to see you ip address**

**then you can map this address from windows as windows drive**

**4. create directory /home/mati/develop**

**download buildroot-2023.11.tar.gz**

**tar xf  buildroot-2023.11.tar.gz**

cd buildroot-2023.11

**make  zynq\_zed\_defconfig**

**make**

download petallinux from Xilinx site (2023.2)

prepare installation:

sudo apt-get -y install iproute2 \

gcc \

g++ \

net-tools \

libncurses5-dev \

zlib1g:i386 \

libssl-dev \

flex \

bison \

libselinux1 \

xterm \

autoconf \

libtool \

texinfo \

zlib1g-dev \

gcc-multilib \

build-essential \

screen \

pax \

gawk \

python3 \

python3-pexpect \

python3-pip \

python3-git \

python3-jinja2 \

xz-utils \

debianutils \

iputils-ping \

libegl1-mesa \

libsdl1.2-dev \

pylint3 \

cpio

now install (from the downloads folder):

(or better download the script and run it from <https://support.xilinx.com/s/article/73296?language=en_US>

Before you try to install petalinux)

sudo apt-get install libtinfo5

./petalinux-v2023.2-10121855-installer.run --dir /home/mati/peta2023.2

**source  /**home/mati/peta2023.2/settings.sh

cd /home/mati/develop

**petalinux-create -t project -n horizon --template zynq**

**mkdir -p /**home/mati/develop/horizon/**xsa**

**copy the xsa to /**home/develop/horizon/**xsa**

**cd /**home/mati/develop/horizon/

petalinux-config   --get-hw-description=**./xsa**

**petalinux-build**

to prepare for burning sdcard

תכניס תאפשר לראות אותו מצד הלינוקס ubuntu

תריץ sudo fdisk /dev/sdxx

תימחק את הפרטישן הגדול ותעשה פרטישן שהיה קטן מ32G פ

תפרמט אותן fat32

The size + offset I used for 31G is 65,013,760

Use samba on windows:

[\\192.168.1.61](file:///\\192.168.1.61)

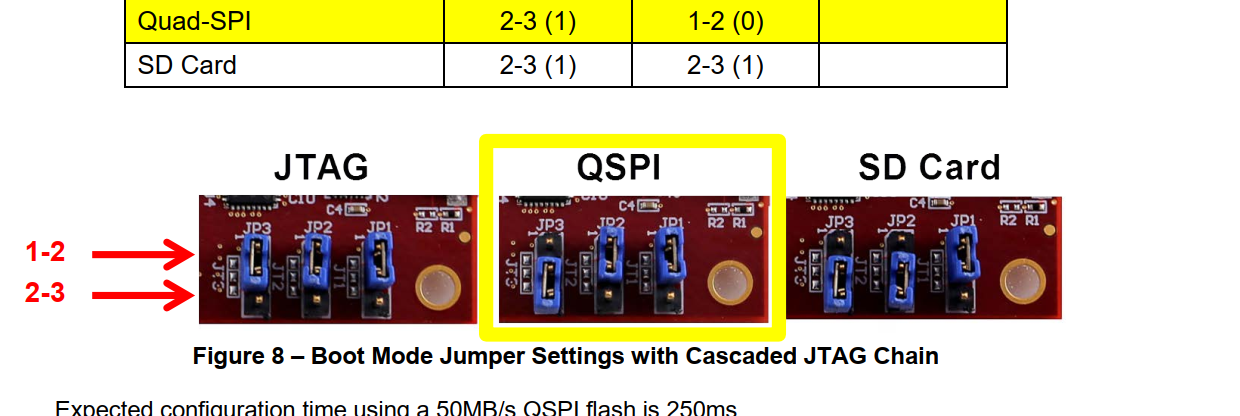
Map this to a network drive

Prepare sd card:

Can be done in vitis

Or use bootgen stand alone

Plug in sdcard set the according to data sheet for sdcard and turn on the board.



The \*.bif file should look like this:

the\_ROM\_image:

{

[bootloader] /home/mati/develop/horizon/images/linux/zynq\_fsbl.elf

/home/mati/develop/horizon/images/linux/system.bit

/home/mati/develop/horizon/images/linux/u-boot.elf

[load=0x00100000] /home/mati/develop/horizon/images/linux/system.dtb

}

Since it is not working (ethernet is not booting)

I will try to do it with petalinux 2022.1 as we have a bsp ready from Avnet for this version.

Once I had error with mz7020\_som\_base\_2022\_1

petalinux-build -c avnet-image-minimal

this is a script Itamar sais we need to run after u-boot starts:

**setenv serverip 192.168.1.61**  
setenv ethaddr 00:06:9F:00:e0:19  
setenv fdt\_addr 0x02000000  
setenv loadaddr 0x03000000  
**setenv ipaddr 192.168.1.25**  
**setenv kernel uImage  
setenv ftd system.dtb**setenv bootcmd 'run bootcmd\_net'  
setenv nfsroot '/home/develop/rootfs'  
setenv bootargs\_base 'setenv bootargs  console=ttyPS0,115200 earlycon'   
setenv bootargs\_nfs 'setenv bootargs ${bootargs} root=/dev/nfs ip=${ipaddr} nfsroot=${serverip}:${nfsroot},v3,tcp'  
setenv bootcmd\_net 'run bootargs\_base bootargs\_nfs;tftpboot ${loadaddr} ${serverip}:${kernel}; tftpboot ${fdt\_addr} ${serverip}:${ftd}; bootm ${loadaddr} - ${fdt\_addr};'  
saveenv

some holp on using time:

<https://www.guyrutenberg.com/2013/01/27/using-stdchronohigh_resolution_clock-example/>

zynq ram

setenv fdt\_addr 0x02000000

setenv kernel\_addr 0x03000000

setenv rootfs\_addr 0x04000000

setenv rootfs rootfs.cpio.uboot

setenv kernel uImage

setenv ftd zynq-zed.dtb

setenv bootargs\_ram 'setenv bootargs ${bootargs} root=/dev/ram rw'

setenv bootcmd\_ram 'run bootargs\_base bootargs\_ram; fatload mmc 0 ${kernel\_addr} ${kernel}; fatload mmc 0 ${fdt\_addr} ${ftd}; fatload mmc 0 ${rootfs\_addr} ${rootfs};'

setenv bootargs\_base 'setenv bootargs console=ttyPS0,115200 '

setenv bootcmd 'run bootcmd\_ram; bootm ${kernel\_addr} ${rootfs\_addr} ${fdt\_addr};'

saveenv