OPEN SOURCE INDIA

11th - 13th Nov 2013
NIMHANS Convention Centre, Bengaluru

denx software engineering

XILINX
ALL PROGRAMMABLE™

U-Boot – Multi image booting scenarios

JagannadhaSutradharudu Teki
Suneel Garapati

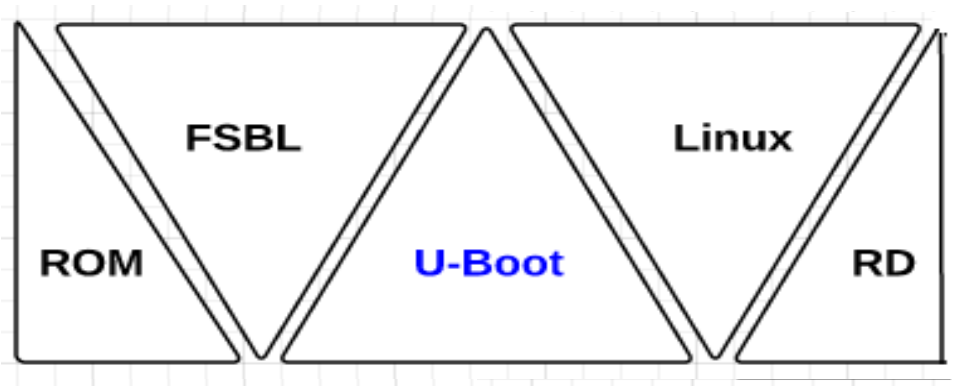
Me

- **Jagan Teki**
  - Currently working for Xilinx in System Software – handling u-boot and Linux
  - Almost 5+ years of experience in embedded domain (LDD, Android BSP) – Sasken/Veda Solutions
  - *Professional Interest:* Opensource
  - Dell Streak Android kernel - http://opensource.dell.com/releases/streak/4.05_and_4.07/#!
  - U-Boot SPI Custodian - http://git.denx.de/?p=u-boot/u-boot-spi.git;a=summary
  - Tech talk (ELCE,2013 – Edinburgh) - http://www.denx.de/wiki/pub/U-Boot/MiniSummitELCE2013/U-Boot_verified_RSA_boot_flow_on_arm_target.pdf

Agenda

- **U-Boot Overview**
- **Features**
- **Community**
- **Multi image booting**
  - Legacy image
  - Single component uImage
  - Monolithic image
  - FIT image
  - Verified RSA image
  - U-boot FDT
- **Demo run**
- **References**

U-Boot Overview



- Universally **Configurable** bootloader
- Robust, flexible
- 3 to 4 releases per year
- 38+ Custodians
- 134 developers
- 1165+ supported boards
- 70K lines of code added for each release
- Most of sub-systems are fully featured

U-Boot Features

Source tree

- Well structured tree as like Linus’ tree, called ad Denx tree

Autoboot

- Will automatically boot the system on power up or reset of the board

Environment variables

- Env. variables can set, save and even print with respective commands.

Networking

- Supports all possible n/w commands like ping, dhcp, tftp and nfs

O/S loading

- Supports variety of commands to load an O/S (legacy, multi image)

Flash support

- parallel NOR, NAND, SD/MMC, serial NOR, USB etc

Serial download

- Files can be loaded through serial via loady, loadb

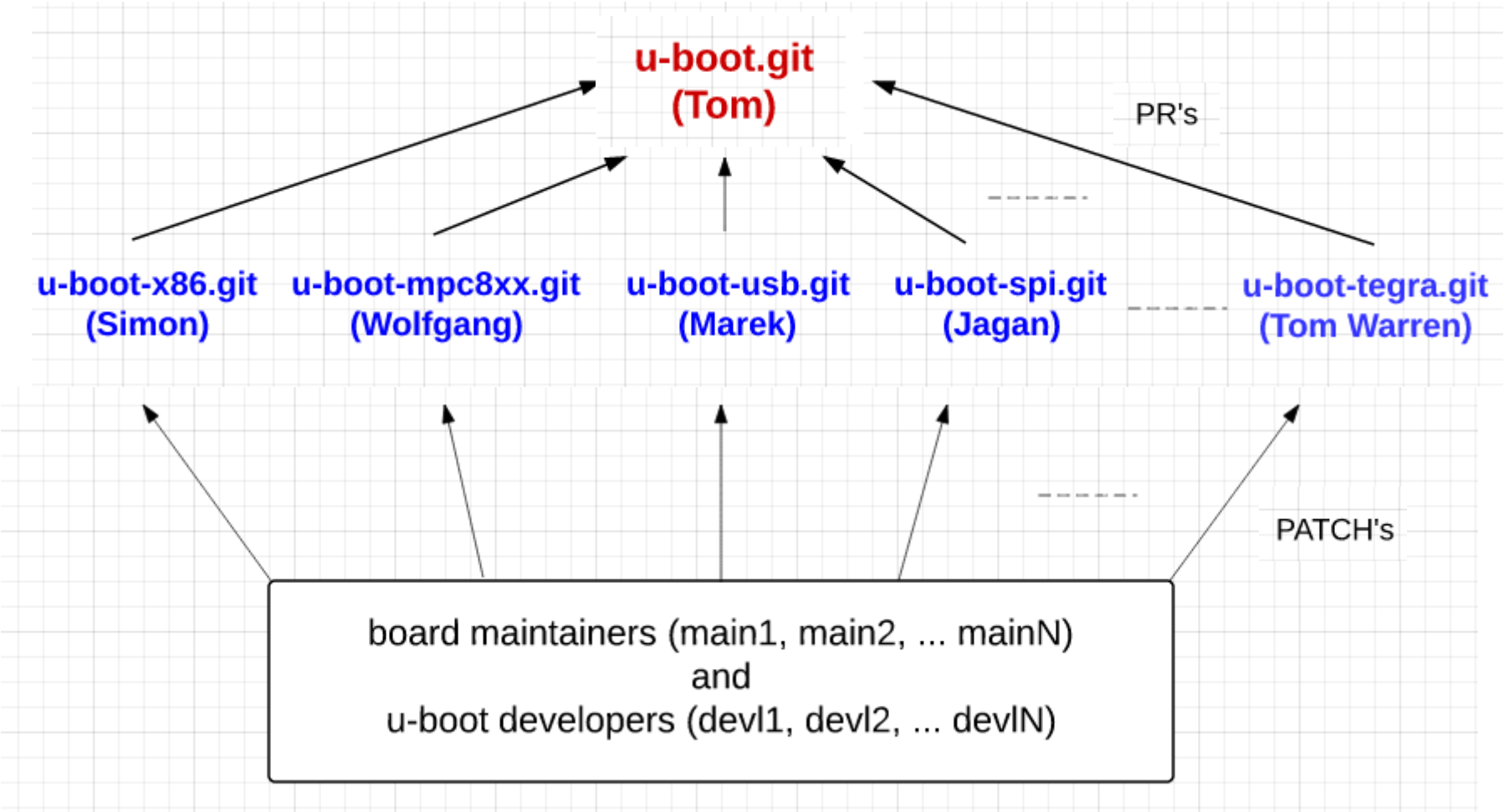
Miscellaneous

- Filesystems, SPL, DFU, Falcon boot, Secure boot, FDT, tools(mkimage, buildman, patman) - http://www.denx.de/wiki/U-Boot/MiniSummitELCE2013

U-Boot Community

- Work similar to Linux community – With some naming notations
- Every year - 3 or 4 releases
- 19 days of MW (new features) after every release
- Once MW closed – then release candidate phase(bug fixes)

Community Workflow

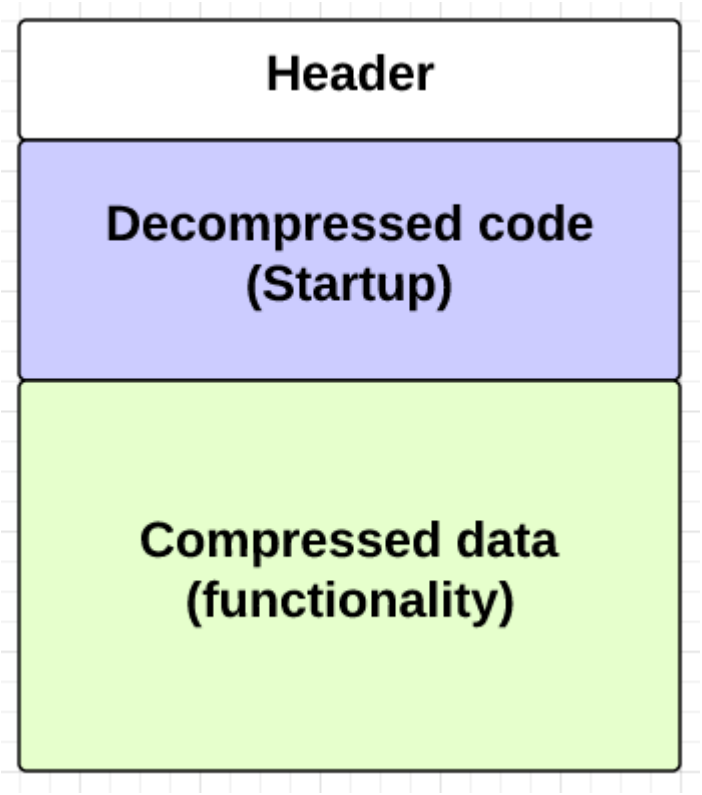


Master - Custodians - Maintainers - Developers

Legacy Image

- Legacy image – typically with decompressed kernel
- Legacy image format – quite normal, with startup and functionality

- zImage format



head. S
+
Some C code to unzip data

Legacy image boot

- **go addr [arg ...]** - start application at address 'addr'

```
zynq-uboot> fatload mmc 0 0x8000 zImage
reading zImage

2298328 bytes read
zynq-uboot> fatload mmc 0 0x1000000 devicetree.dtb
reading devicetree.dtb

5460 bytes read
zynq-uboot> fatload mmc 0 0x800000 ramdisk.image.gz
reading ramdisk.image.gz

2500546 bytes read
zynq-uboot> go 0x8000
## Starting application at 0x00008000 ...
Uncompressing Linux... done, booting the kernel.
Linux version 3.7.0 (jaganna@xhdrdevl6) (gcc version 4.5.2 (Sourcery CodeBench Lite 2011.07-57) )
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
CPU: VIPT nonaliasing data cache, VIPT aliasing instruction cache
Machine: Xilinx Zynq Platform, model: Xilinx Zynq ZC702
bootconsole [earlycon0] enabled
.....................
..............
.....
zynq>
```

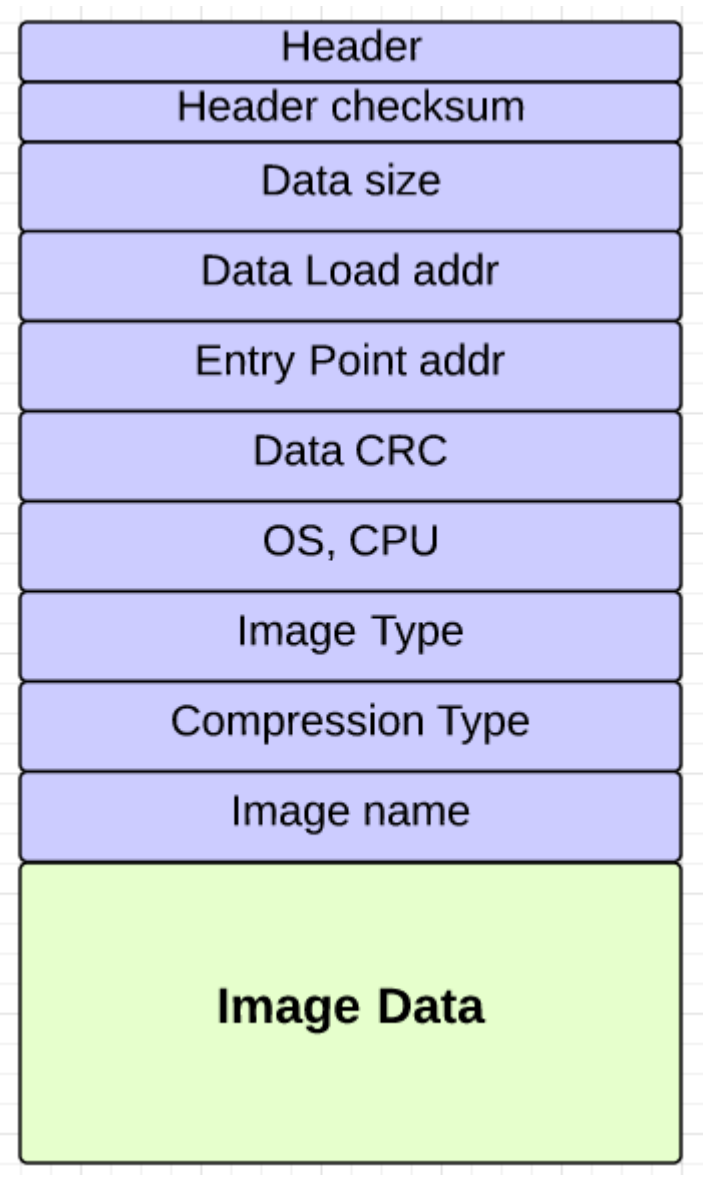
? Fixed offset
? No checksum integrity
? No Multi image component
? No hash integrity
? No security

Single component uImage

- **bootm [addr [arg ...]]** - boot application image stored in memory

```
zynq-uboot> fatload mmc 0 0x3000000 uImage
reading uImage
3005632 bytes read in 472 ms (6.1 MiB/s)
zynq-uboot> fatload mmc 0 0x2000000 uramdisk.image.gz
reading uramdisk.image.gz
2500610 bytes read in 400 ms (6 MiB/s)
zynq-uboot> fatload mmc 0 0x2A00000 zynq-microzed.dtb
reading zynq-microzed.dtb
7581 bytes read in 17 ms (434.6 KiB/s)
zynq-uboot> bootm 0x3000000 0x2000000 0x2A00000
## Booting kernel from Legacy Image at 03000000 ...
   Image Name:   Linux-3.10.0-xilinx
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3005568 Bytes = 2.9 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 02000000 ...
   Image Name:
   Image Type:   ARM Linux RAMDisk Image (uncompressed)
   Data Size:    2500546 Bytes = 2.4 MiB
   Load Address: 00800000
   Entry Point:  00800000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 02a00000
   Booting using the fdt blob at 0x2a00000
   Loading Kernel Image ... OK
   Loading Ramdisk to 1fd9d000, end 1ffff7c2 ... OK
   Loading Device Tree to 1fd98000, end 1fd9cd9c ... OK
Starting kernel ...
```

uImage format

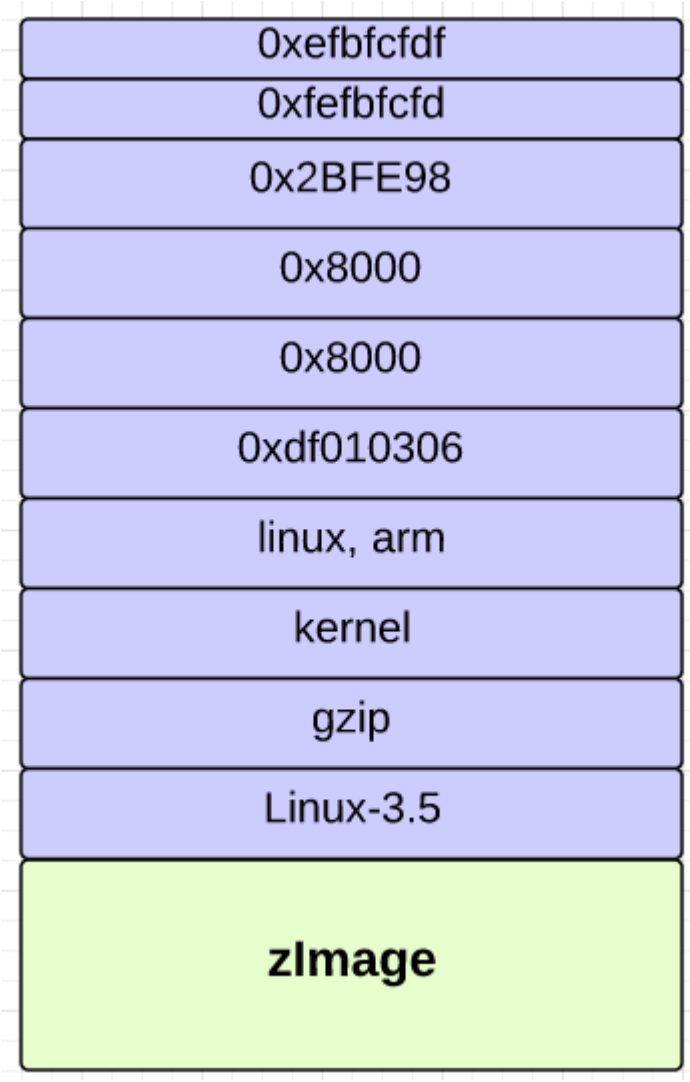



```
bash> mkimage -A arm -O linux -T kernel -c gzip -a 0x8000 -e 0x8000 -n "Linux-3.5" -d zImage uImage
Image Name:   Linux-3.5
Created:      Sun Nov 10 00:58:36 2013
Image Type:   ARM Linux Kernel Image (gzip compressed)
Data Size:    2883224 Bytes = 2815.65 kB = 2.75 MB
Load Address: 00008000
Entry Point:  00008000
```

? No Multi image component

? No hash integrity

? No security



Monolithic image

- Single image – with combination of multiple images

```
zynq-uboot> fatload 0 mmc 0 0x2000000 uMulti
reading uMulti
6691301 bytes read in 1027 ms (6.2 MiB/s)
zynq-uboot> bootm 0x2000000
## Booting kernel from Legacy Image at 02000000 ...
   Image Name:   Multi image
   Image Type:   ARM Linux Multi-File Image (gzip compressed)
   Data Size:    6691237 Bytes = 6.4 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Contents:
      Image 0: 2994675 Bytes = 2.9 MiB
      Image 1: 3688961 Bytes = 3.5 MiB
      Image 2: 7581 Bytes = 7.4 KiB
   Verifying Checksum ... OK
## Loading init Ramdisk from multi component Legacy Image at 02000000 ...
## Flattened Device Tree from multi component Image at 02000000
   Booting using the fdt at 0x0265fc48
   Uncompressing Multi-File Image ... OK
   Loading Ramdisk to 1fc7b000, end 1ffffa01 ... OK
   Loading Device Tree to 1fc76000, end 1fc7ad9c ... OK

Starting kernel ...

Booting Linux on physical CPU 0x0
Linux version 3.10.0-xilinx (jaganna@xhdrdevl6) (gcc version 4.7.2 (Sourcery CodeBench Lite 2012.09-104) )
CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
.................................
................
zynq>
```

Header
Header checksum
Data size
Data Load addr
Entry Point addr
Data CRC
OS, CPU
Image Type
Compression Type
Image name
Image0 size
Image1 size
Image2 size
Image0 Data
Image1 Data
Image2 Data

```
bash> mkimage -A arm -O linux -T multi -a 0x8000 -e 0x8000 -C gzip -n 'Multi image' -d vmlinux.bin.gz:ramdisk.image.gz:zynq-microzed-linux.dtb uMulti
Image Name:   Multi image
Created:      Sun Nov 10 01:40:55 2013
Image Type:   ARM Linux Multi-File Image (gzip compressed)
Data Size:    6691237 Bytes = 6534.41 kB = 6.38 MB
Load Address: 00008000
Entry Point:  00008000
Contents:
   Image 0: 2994675 Bytes = 2924.49 kB = 2.86 MB
   Image 1: 3688961 Bytes = 3602.50 kB = 3.52 MB
   Image 2: 7581 Bytes = 7.40 kB = 0.01 MB
```

? Not flexible, indexing

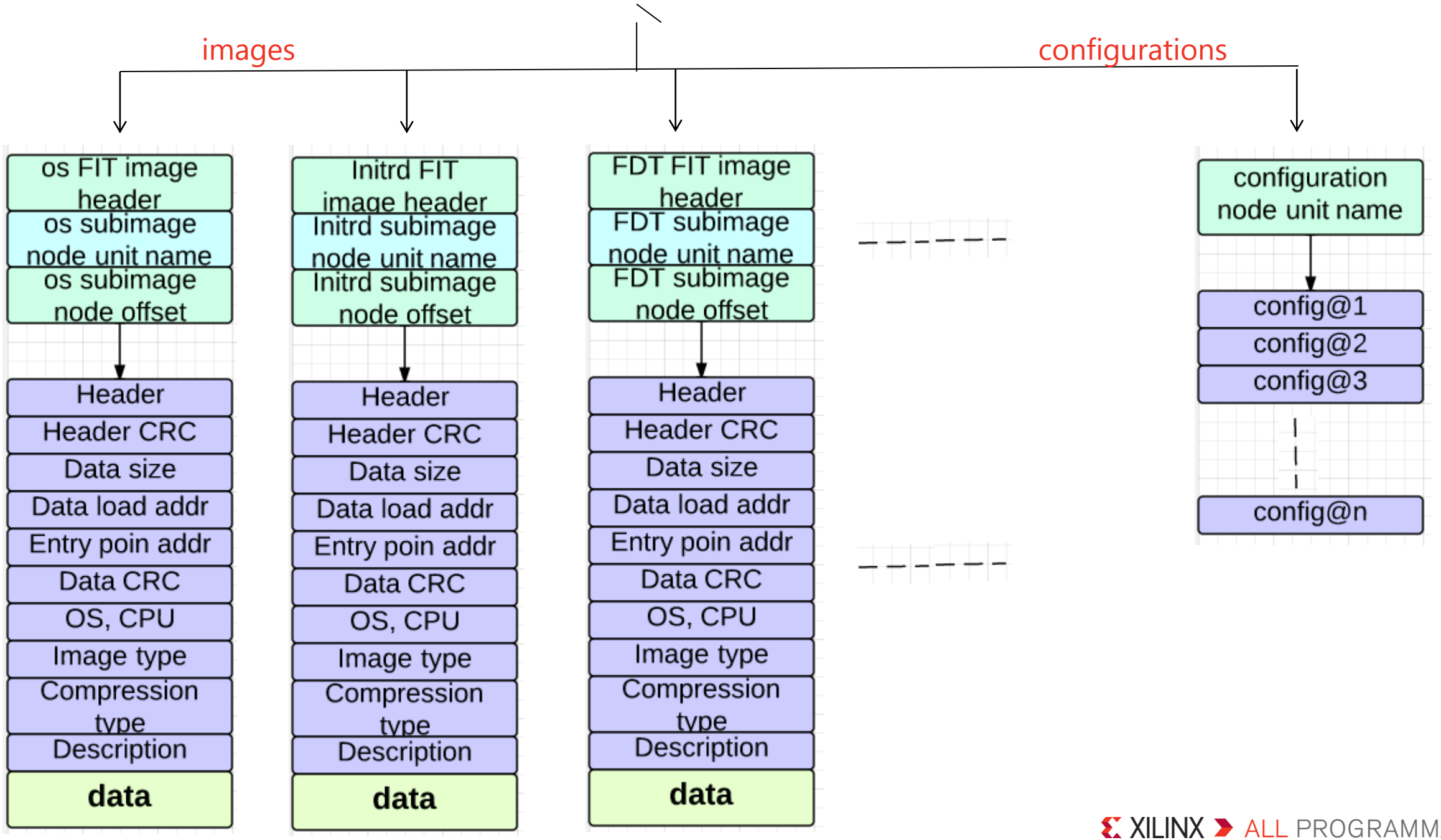
? No hash integrity

? No security

0xefbfcfdf
0xfefbfcfd
0x6619A5
0x8000
0x8000
0xdfa0b3c4
linux, arm
multi
gzip
Multi image
0x2DB1F3
0x384A01
0x1D9D
vmlinux.bin.gz
ramdisk.image.gz
zynq-microzed-linux.dtb

FIT image

- FIT - Flattened Image Tree
- Tree like structure and more flexibility in handling images of various types (kernel, ramdisk, dtb etc.)
- Tree structure is same like DT.
- image node have a details of image’s – each image node has sub nodes like kernel, ramdisk and fdt.
- Each sub node has a property attributes like data, type, os, arch, entry, load, algo etc.
- configuration node has default and list of configuration to pick-up while booting.


images
configurations
os FIT image header
os subimage node unit name
os subimage node offset
Header
Header CRC
Data size
Data load addr
Entry poin addr
Data CRC
OS, CPU
Image type
Compression type
Description
data
Initrd FIT image header
Initrd subimage node unit name
Initrd subimage node offset
Header
Header CRC
Data size
Data load addr
Entry poin addr
Data CRC
OS, CPU
Image type
Compression type
Description
data
FDT FIT image header
FDT subimage node unit name
FDT subimage node offset
Header
Header CRC
Data size
Data load addr
Entry poin addr
Data CRC
OS, CPU
Image type
Compression type
Description
data
configuration node unit name
config@1
config@2
config@3
config@n

FIT input

```
/dts-v1/;
/ {
    description = "Simple image with single Linux kernel, FDT blob and ramdisk";
    #address-cells = <0x1>;

    images {
        kernel@1 {
            description = "Zynq Linux kernel";
            data = /incbin/("./vmlinux.bin.gz");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "gzip";
            load = <0x8000>;
            entry = <0x8000>;
            hash@1 {
                algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
        };
        fdt@1 {
            description = "ZED board Flattened Device Tree blob";
            data = /incbin/("./zynq-microzed.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            hash@1 {
                algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
        };
        ramdisk@1 {
            description = "Ramdisk Image";
            data = /incbin/("./ramdisk.image.gz");
            type = "ramdisk";
            arch = "arm";
            os = "linux";
            compression = "gzip";
            load = <0x00800000>;
            entry = <0x00800000>;
            hash@1 {
            algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
        };
    };
    configurations {
        default = "conf@1";
        conf@1 {
            description = "Boot Linux kernel, FDT blob and ramdisk";
            kernel = "kernel@1";
            fdt = "fdt@1";
            ramdisk = "ramdisk@1";
        };
    };
};
```

FIT output

```
zynq-uboot>   fatload mmc 0 0x2000000 fit.itb
reading fit.itb 6692860 bytes read in 903 ms (7.1 MiB/s)
zynq-uboot> bootm 0x2000000
## Booting kernel from FIT Image at 02000000 ...
   Using 'conf@1' configuration
   Trying 'kernel@1' kernel subimage
     Description:  Zynq Linux kernel
     Type:         Kernel Image
     Compression:  gzip compressed
     Data Start:   0x020000f0
     Data Size:    2994675 Bytes = 2.9 MiB
     Architecture: ARM
     OS:           Linux
     Load Address: 0x00008000
     Entry Point:  0x00008000
     Hash algo:    md5
     Hash value:   13f4314bdb90fb7dfd6aca876ad169ef
     Hash algo:    sha1
     Hash value:   c5a93bb01325fbe12888d4a485f778fe3f871a0b
   Verifying Hash Integrity ... md5+ sha1+ OK
## Loading init Ramdisk from FIT Image at 02000000 ...
   Using 'conf@1' configuration
   Trying 'ramdisk@1' ramdisk subimage
     Description:  Ramdisk Image
     Type:         RAMDisk Image
     Compression:  gzip compressed
     Data Start:   0x022dd2b4
     Data Size:    3688961 Bytes = 3.5 MiB
     Architecture: ARM
```

? Try Secure

```
     OS:           Linux
     Load Address: 0x00800000
     Entry Point:  0x00800000
     Hash algo:    md5
     Hash value:   f47ab43c4b5a452b34142e8cf93da2d4
     Hash algo:    sha1
     Hash value:   99e48d770444bed51b0825ff88832feb097e370e
   Verifying Hash Integrity ... md5+ sha1+ OK
## Flattened Device Tree from FIT Image at 02000000
   Using 'conf@1' configuration
   Trying 'fdt@1' FDT blob subimage
     Description:  ZED board Flattened Device Tree blob
     Type:         Flat Device Tree
     Compression:  uncompressed
     Data Start:   0x022db420
     Data Size:    7581 Bytes = 7.4 KiB
     Architecture: ARM
     Hash algo:    md5
     Hash value:   e130ba9ac6a27f1e46edbff64377e453
     Hash algo:    sha1
     Hash value:   da7bb3a255847571e0e46c1f10750af362a0c128
   Verifying Hash Integrity ... md5+ sha1+ OK
   Booting using the fdt blob at 0x022db420
   Uncompressing Kernel Image ... OK
   Loading Ramdisk to 1f7d1000, end 1fb55a01 ... OK
   Loading Device Tree to 1f7cc000, end 1f7d0d9c ... OK

Starting kernel ...
```

FIT uses

- Enhances hash integrity of images with sha1, md5, etc.
- Better solution for multi component images.
  - Multiple Kernel’s (production vs. debug)
  - Multiple DT’s (SOC kernel vs. board specific DT’s)
  - bootm addr:<subimg_uname> - direct component image specification
  - bootm addr#<conf_uname> - configuration specification
- Good for embedding a new booting features.
- Most of the architecture – PowerPC, ARM, x86, Microblaze.

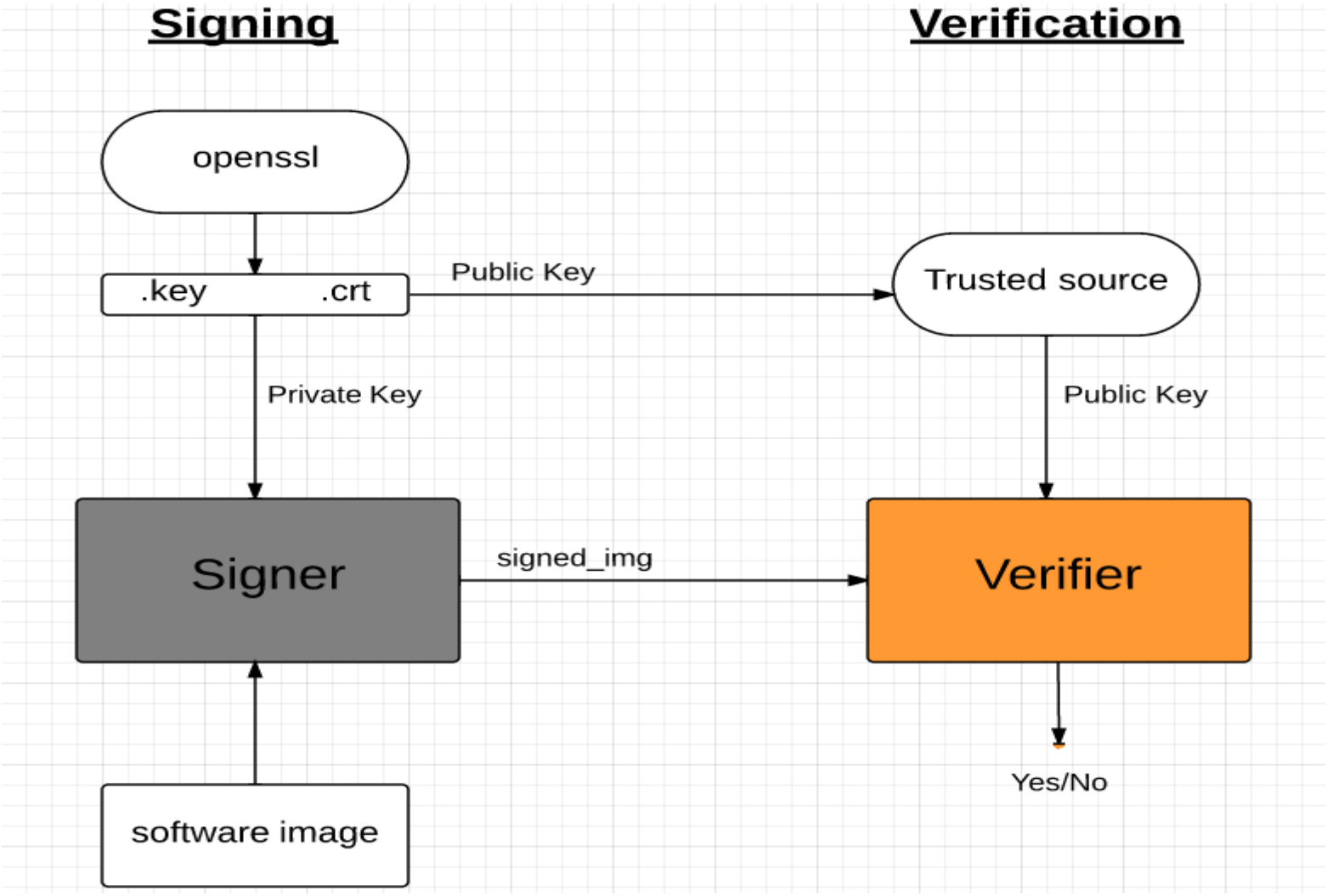
Suneel

- **Suneel Garapati**
  - Currently working for Xilinx in System Software drivers – Linux device drivers
  - Almost 5+ years of experience in embedded domain (LDD, Android BSP) – Sasken/Xilinx

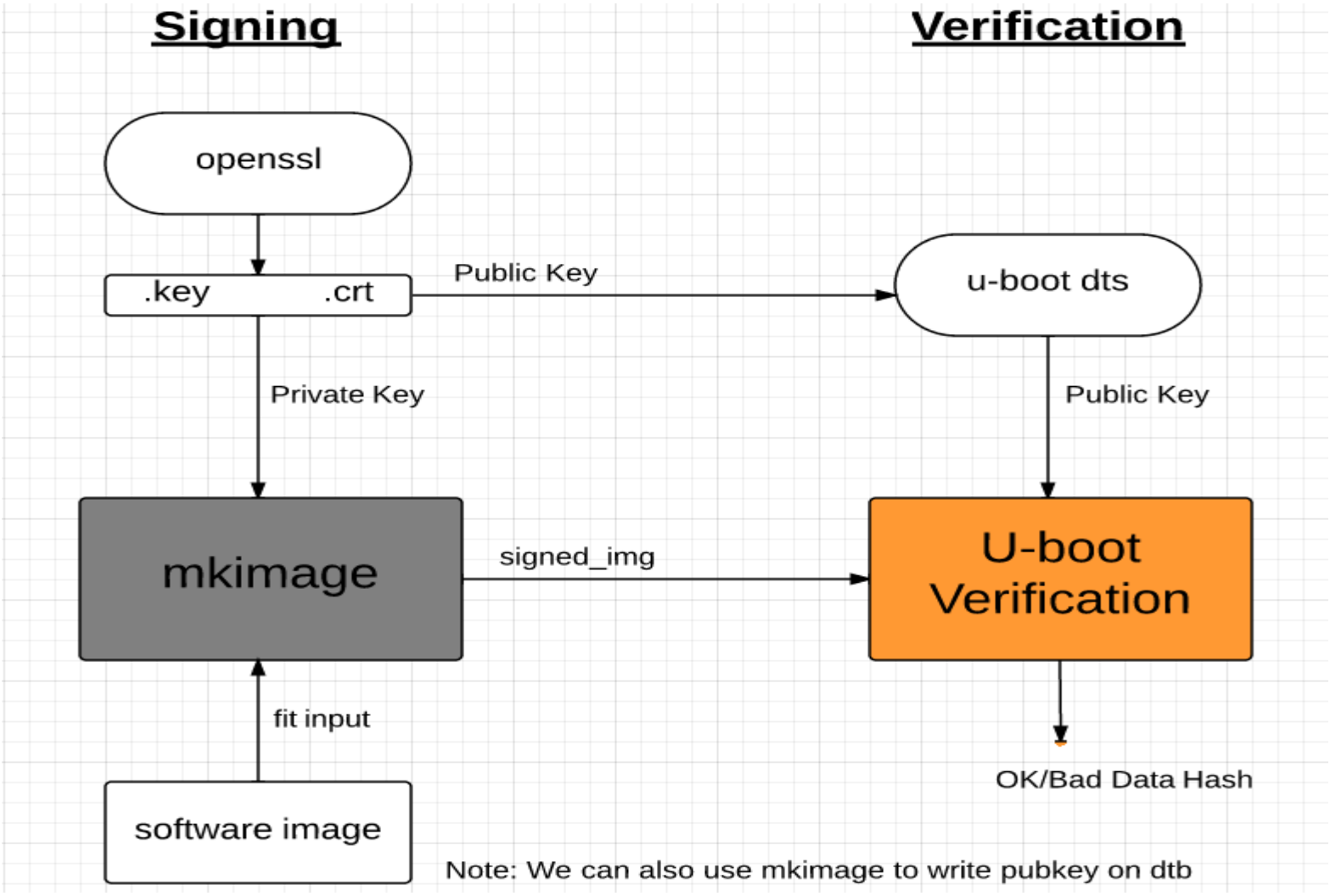
Verified image

- Verified – Secure – Trusted boot
- Verify the loaded software to ensure that it is authorized during boot.
- Benefits:
  - Prevent from malware
  - Provide authorized read access
  - Machine safe – runs only signed software
  - Possible to field-upgrade the software

RSA Concept


Signing
Verification
openssl
.key
.crt
Public Key
Trusted source
Private Key
Public Key
Signer
signed_img
Verifier
Yes/No
software image

U-Boot RSA


Signing
Verification
openssl
.key
.crt
Public Key
u-boot dts
Private Key
Public Key
mkimage
signed_img
U-boot Verification
fit input
OK/Bad Data Hash
software image
Note: We can also use mkimage to write pubkey on dtb

RSA support

2013-06-26	Dirk Behme	spi: mxc_spi: Fix pre and post divider calculation	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add verified boot information and test	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	sandbox: config: Enable FIT signatures with RSA	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	image: Add support for signing of FIT configurations	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	libfdt: Add fdt_find_regions()	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	mkimage: Add -r option to specify keys that must be...	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	mkimage: Add -c option to specify a comment for key...	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	mkimage: Add -F option to modify an existing .fit file	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	mkimage: Add -K to write public keys to an FDT blob	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	mkimage: Add -k option to specify key directory	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	image: Add RSA support for image signing	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	image: Support signing of images	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	image: Add signing infrastructure	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	x86: config: Add tracing options	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	x86: Support tracing function	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	exynos: config: Add tracing options	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	exynos: Avoid function instrumentation for microsecond...	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	arm: Implement the 'fake' go command	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add a 'fake' go command to the bootm command	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Refactor the bootm command to reduce code duplication	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Clarify bootm OS arguments	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add a simple test for sandbox trace	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	sandbox: Support trace feature	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add proftool to decode profile data	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add trace support to generic board	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Support tracing in config.mk when enabled	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add a trace command	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add trace library	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Add function to print a number with grouped digits	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	bootstage: Correct printf types	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Show stdout on error in fit-test	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)
2013-06-26	Simon Glass	Fix missing return in do_mem_loop()	commit | commitdiff | tree | snapshot (tar.gz tar.bz2)

RSA configurations

- Enable FIT
  - CONFIG_FIT - enable support for the FIT uImage format
- Enable FDT
  - CONFIG_OF_CONTROL
  - CONFIG_OF_SEPARATE
- *Enable verified boot*
  - *CONFIG_FIT_SIGNATURE - enables signature verification of FIT images*
  - *CONFIG_RSA - enables the RSA algorithm used for FIT image verification*

RSA input

```
/dts-v1/;
/ {
    description = "Veried_rsa image with single Linux kernel, FDT blob and ramdisk";
    #address-cells = <0x1>;

    images {
        kernel@1 {
            description = "Zynq Linux kernel";
            data = /incbin/("./vmlinux.bin.gz");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "gzip";
            load = <0x8000>;
            entry = <0x8000>;
            hash@1 {
                algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
            signature@1 {
                algo = "sha1,rsa2048";
                key-name-hint = "dev";
            };
        };
        fdt@1 {
            description = "ZED board Flattened Device Tree blob";
            data = /incbin/("./zynq-microzed-linux.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            hash@1 {
                algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
            signature@1 {
                algo = "sha1,rsa2048";
                key-name-hint = "dev";
            };
        };
        ramdisk@1 {
            description = "Ramdisk Image";
            data = /incbin/("./ramdisk.image.gz");
            type = "ramdisk";
            arch = "arm";
            os = "linux";
            compression = "gzip";
            load = <0x00800000>;
            entry = <0x00800000>;
            hash@1 {
                algo = "md5";
            };
            hash@2 {
                algo = "sha1";
            };
            signature@1 {
                algo = "sha1,rsa2048";
                key-name-hint = "dev";
            };
        };
    };
    configurations {
        default = "conf@1";
        conf@1 {
            description = "Boot Linux kernel, FDT blob and ramdisk";
            kernel = "kernel@1";
            fdt = "fdt@1";
            ramdisk = "ramdisk@1";
        };
    };
};
```

RSA output

```
zynq-uboot> fatload mmc 0 0x2000000 rsa_signed.img
reading rsa_signed.img
6694570 bytes read in 1027 ms (6.2 MiB/s)
zynq-uboot> bootm 0x2000000
## Loading kernel from FIT Image at 02000000 ...
   Using 'conf@1' configuration
   Verifying Hash Integrity ... OK
   Trying 'kernel@1' kernel subimage
     Description:  Zynq Linux kernel
     Type:         Kernel Image
     Compression:  gzip compressed
     Data Start:   0x020000f4
     Data Size:    2994675 Bytes = 2.9 MiB
     Architecture: ARM
     OS:           Linux
     Load Address: 0x00008000
     Entry Point:  0x00008000
     Hash algo:    md5
     Hash value:   13f4314bdb90fb7dfd6aca876ad169ef
     Hash algo:    sha1
     Hash value:   c5a93bb01325fbe12888d4a485f778fe3f871a0b
     Sign algo:    sha1,rsa2048:dev
     Sign value:  531d822a89d9f07f881feea0f0ab137543e34dbf18a67b1890acfc5e0fa8c9f69fd4
778b780dd3f6628290b904c3228740f5e84856eadce6f48c1cffd265aa223adefcdd57edc27c2b359a7e56b
a1719a6b57cbfe2fa3e47bd6bef64c3c42202740679fcd5317ab73b2221b87e4fea3485c6298f96284243d1
34fb13928df7515e4adb6fbbdad079c690f027f1752705aeb04e1c0873f7c2c84883ad4383a5a407a95c2c9
e75406fa599f9a19cfb72d2533d2df6eedc8519b4ac6e814ddaee3705727bab614e865209d078514e31b4a2
6b24eaf2fc85ef1e1415a5597d48773de4e726e27a583df1e5cc22752fe60386b0799a0432de25096c71314f149f8133
   Verifying Hash Integrity ... md5+ sha1+ sha1,rsa2048:dev- OK
```

```
## Loading ramdisk from FIT Image at 02000000 ...
   Using 'conf@1' configuration
   Trying 'ramdisk@1' ramdisk subimage
     Description:  Ramdisk Image
     Type:         RAMDisk Image
     Compression:  gzip compressed
     Data Start:   0x022dd5f0
     Data Size:    3688961 Bytes = 3.5 MiB
     Architecture: ARM
     OS:           Linux
     Load Address: 0x00800000
     Entry Point:  0x00800000
     Hash algo:    md5
     Hash value:   f47ab43c4b5a452b34142e8cf93da2d4
     Hash algo:    sha1
     Hash value:   99e48d770444bed51b0825ff88832feb097e370e
     Sign algo:    sha1,rsa2048:dev
     Sign value:   1db3dcdaae7d1ac16e39012ba86a0f1c2389130e16494869c23cacf9af2eccb9e9db52
c2aabc0d02c9d4b81a2faaf0d3690d34c45bfab3c5c0757fd0820d9d0b260fba4e977268a3b51b5ff61e3821ef
39488b7c6f48b4e42ef628ee8d5ae58f57e050d628281bc09a21ee4311206820b8f5dddd37e93d3e6a39068a29
1eabe3e230a55b5738619f55df424ec83c63821e73ff49b79ff9f6059a9ffe9bfe294c2e69f26ebe17c0b474d9
e88e09aa7badb2f45684f5f676d5f8c03e67ea21a17bbb6bc1fb011e917b8ba457bb34c253a497e51076cf991f
1f3958c2488bd9fec8c0f2d0be0d36d25683762a49c573cf7557e38056d8a9fed40518508044075cfa
   Verifying Hash Integrity ... md5+ sha1+ sha1,rsa2048:dev- OK
```

RSA output (cont…)

```
## Loading fdt from FIT Image at 02000000 ...
   Using 'conf@1' configuration
   Trying 'fdt@1' fdt subimage
     Description:  ZED board Flattened Device Tree blob
     Type:         Flat Device Tree
     Compression:  uncompressed
     Data Start:   0x022db5c0
     Data Size:    7581 Bytes = 7.4 KiB
     Architecture: ARM
     Hash algo:    md5
     Hash value:   e130ba9ac6a27f1e46edbff64377e453
     Hash algo:    sha1
     Hash value:   da7bb3a255847571e0e46c1f10750af362a0c128
     Sign algo:    sha1,rsa2048:dev
     Sign value:   368fca300e965d5ef6f84a407851b6adbb10574362c5f21406258ae99d6e4466aad064cc076b84e65dab1f85f
14a0ba1b64ceaf6dde610cfdbd6bf083a22380a5d1b35347db562f1934f464a2e2507383a32b18377934d13f3a999517d153a2fe
54e02d5befd816e1281f94363f507b8b6ed51a23c8ab7d82c68623d49a3da364acbceb1f2999abc0f6840ef48c8620417409e775
7284dabf6d45a5f40d91c9c867fc0bb03ee01fa3a90c249bd5bb939ea0848435b7ace4d4d6a4f4539ec3b6c53c5b2da028d46384
6ffbbe096c38a9e84ffcc1395bb30a9f42553ec4916765340b58c0743547906ced7c8ce2efad9def4ccf37e37a492e6c119a3e608fc197c
   Verifying Hash Integrity ... md5+ sha1+ sha1,rsa2048:dev- OK
   Booting using the fdt blob at 0x22db5c0
   Uncompressing Kernel Image ... OK
   Loading Ramdisk to 1fc7b000, end 1ffffa01 ... OK
   Loading Device Tree to 1fc76000, end 1fc7ad9c ... OK

Starting kernel ...
```

??? Try for encryption - over an Image

U-Boot FDT

- Similar to Linux DT – Tree structure defines hardware configurations on board
- U-boot FDT configurations

```
/* FDT support */
#define CONFIG_OF_CONTROL
#define CONFIG_OF_SEPARATE
#define CONFIG_DISPLAY_BOARDINFO_LATE
```

- Building FDT u-boot

```
$ make zynq_microzed_config
$ make DEVICE_TREE=zynq-microzed -j4
```

DTS files - samples

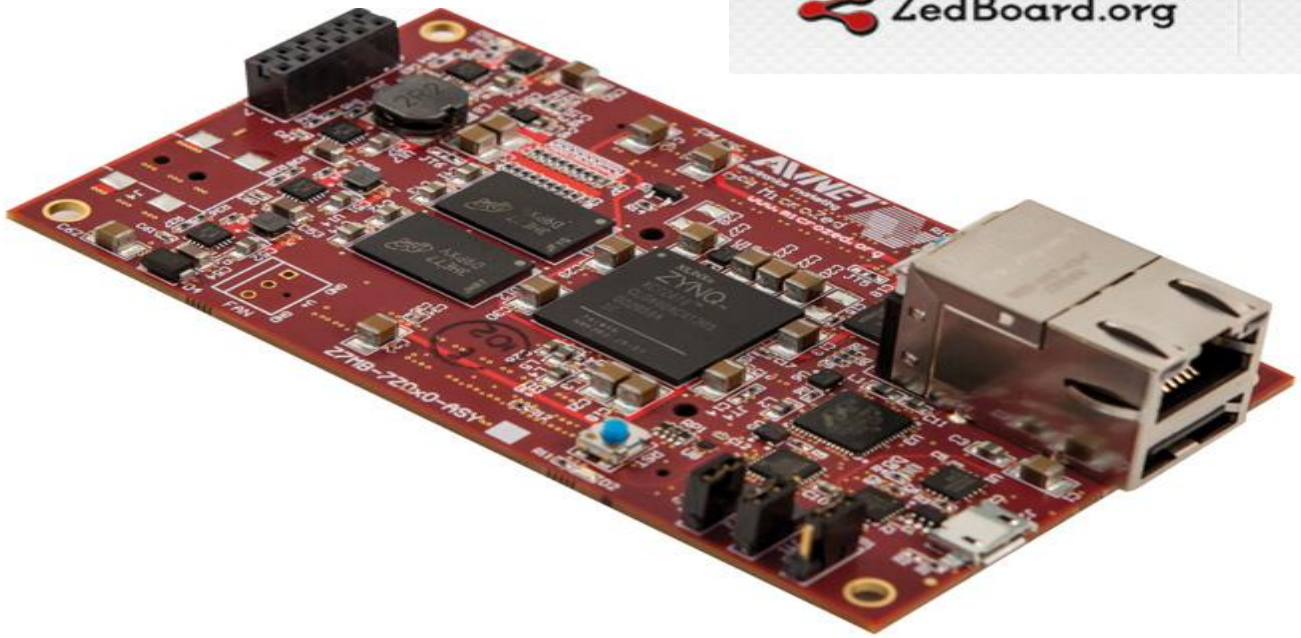
```
/*
 * Xilinx MicroZED board DTS
 *
 * Copyright (C) 2013 Xilinx, Inc.
 *
 * SPDX-License-Identifier:    GPL-2.0+
 */
/dts-v1/;
#include "zynq-7000.dtsi"

/ {
        model = "Zynq MicroZED Board";
        compatible = "xlnx,zynq-microzed", "xlnx,zynq-7000";
};
```

```
/dts-v1/;
#include "exynos5250.dtsi"

/ {
    model = "SAMSUNG Arndale board based on EXYNOS5250";
    compatible = "samsung,arndale", "samsung,exynos5250";

    aliases {
        serial0 = "/serial@12C20000";
        console = "/serial@12C20000";
    };

    mmc@12200000 {
        samsung,bus-width = <8>;
        samsung,timing = <1 3 3>;
    };

    mmc@12210000 {
        status = "disabled";
    };

    mmc@12220000 {
        samsung,bus-width = <4>;
        samsung,timing = <1 2 3>;
    };
```

Demo run

- Legacy boot
- Single image boot
- Monolithic image
- FIT image
- Verified RSA image

ZedBoard.org™ | microZed



Build FDT u-boot

All the respective patch set for this demo:

- Clone the source

```
$ git clone git://git.denx.de/u-boot-spi.git
$ cd u-boot-spi
$ git checkout -b master-xlnx origin/master-xlnx
```

- Toolchain setup

http://www.wiki.xilinx.com/Install+Xilinx+Tools

- Configure for microzed board

```
$ make zynq_microzed_config
```

- Build u-boot

```
$ make DEVICE_TREE=zynq-microzed -j4
<<< you can find the u-boot-dtb.bin >>>
```

Build rsa_signed

- RSA key generation:
  - Create RSA key pair

```
$ openssl genrsa -F4 -out mykeys/dev.key 2048
```

  - Create a certificate contains public key

```
$ openssl req -batch -new -x509 -key mykeys/dev.key -out mykeys/dev.crt
```

- Create dtb for existing u-boot dts

```
$ dtc -p 0x1000 board/xilinx/dts/zynq-zed.dts -O dtb -o zynq-zed.dtb
$ cp zynq-zed.dtb zynq-zed-pubkey.dtb
```

- Sign the images with mykeys

```
$ DTC_OPS="-I dts -O dtb -p 2000"
$ mkimage –D “${DTC_OPS}” –f rsa.its –K zynq-zed-pubkey.dtb –k mykeys –r rsa_signed.img
```

Build FDT u-boot with public key

- For building FDT u-boot with public key- externally

```
$ make DEV_TREE_BIN=./zynq-zed-pubkey.dtb
```

u-boot-dtb.bin -> Is final FDT u-boot image with public key on it, hence the pubkey will used in verification process.

References

- U-Boot wiki http://www.denx.de/wiki/U-Boot
- u-boot.git (master) repo http://git.denx.de/?p=u-boot.git;a=summary
- u-boot-spi.git - http://git.denx.de/?p=u-boot/u-boot-spi.git;a=summary
- Release details http://www.denx.de/wiki/U-Boot/ReleaseCycle
- For FIT details: doc/uImage.FIT/howto.txt
- For verified: doc/uImage.FIT/verified-boot.txt
- For signature: doc/uImage.FIT/signature.txt
- For demo run - http://jagannadhteki.blog.com/2013/11/07/u-boot-mibs/
- For verified RSA - http://jagannadhteki.blog.com/2013/11/08/u-boot-verified-rsa/
- Any questions - mail to jagannadh.teki@gmail.com CC u-boot@lists.denx.de

U-Boot TODO:

- As u-boot is an active community – works many of developers from the globe, we have some sort of TODO' which were discussed on ELCE-2013, Edinburgh.
- Interested developer's welcome to contribute
  - Kbuild
  - Driver Model
  - FDT (trying for flexible)
  - Kbuild, etc.
  - U-Boot wiki http://www.denx.de/wiki/U-Boot
- Try to read below mailing list for more info:

http://u-boot.10912.n7.nabble.com/Minutes-from-the-U-Boot-Mini-Summit-2013-td166175.html

Thanks, Q & A