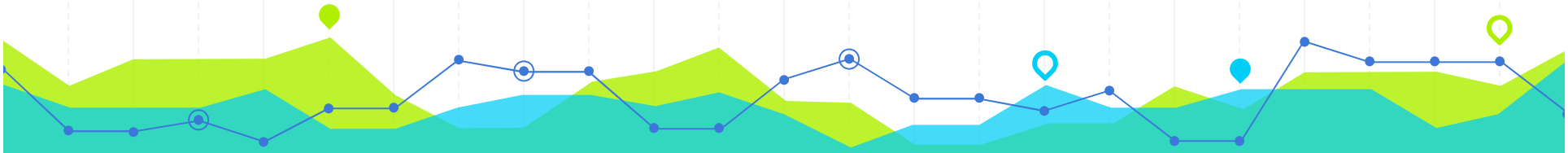
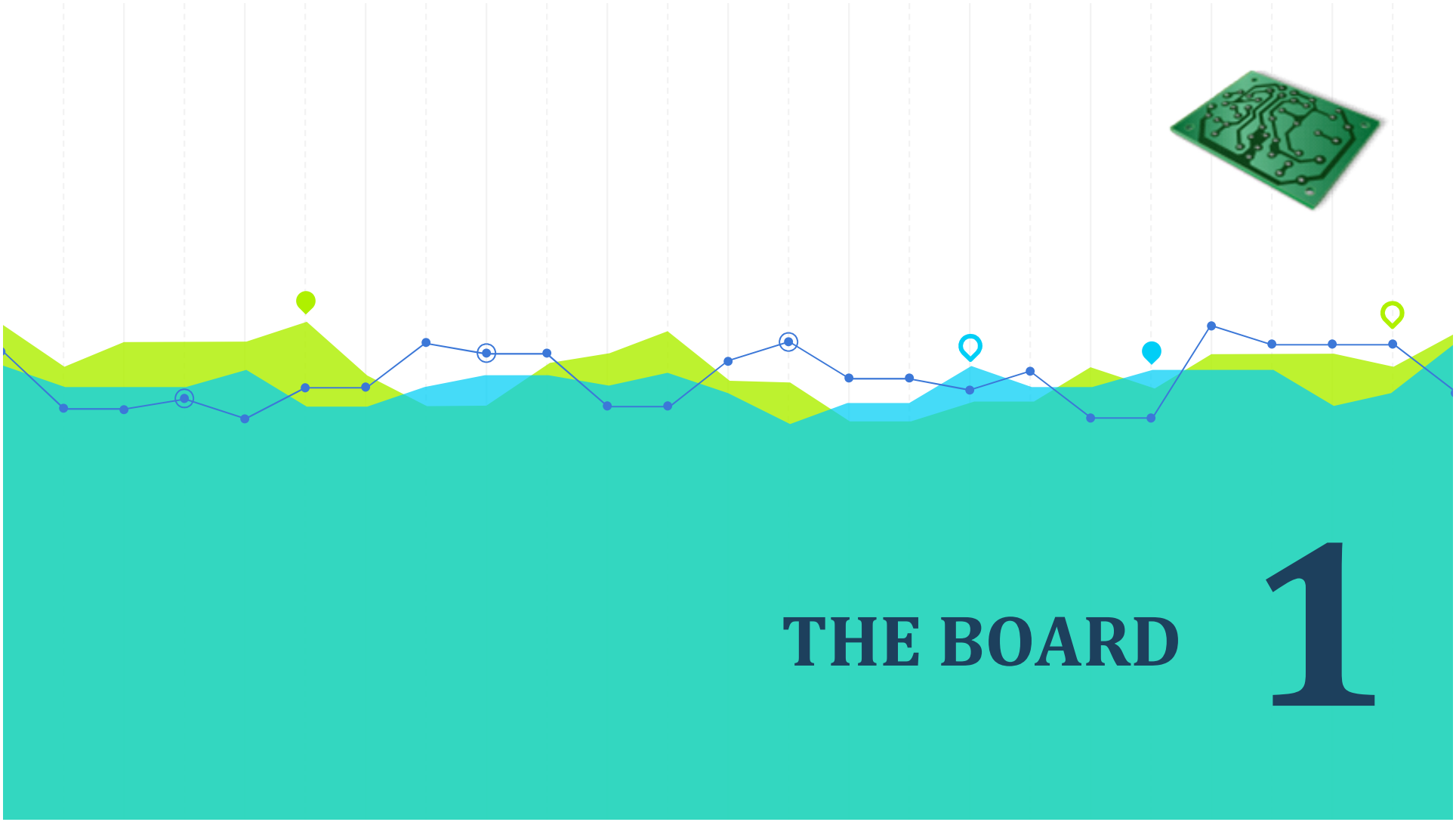
Department of Electrical & Computer Engineering
COE838 System-on-Chip Design

Ryerson University

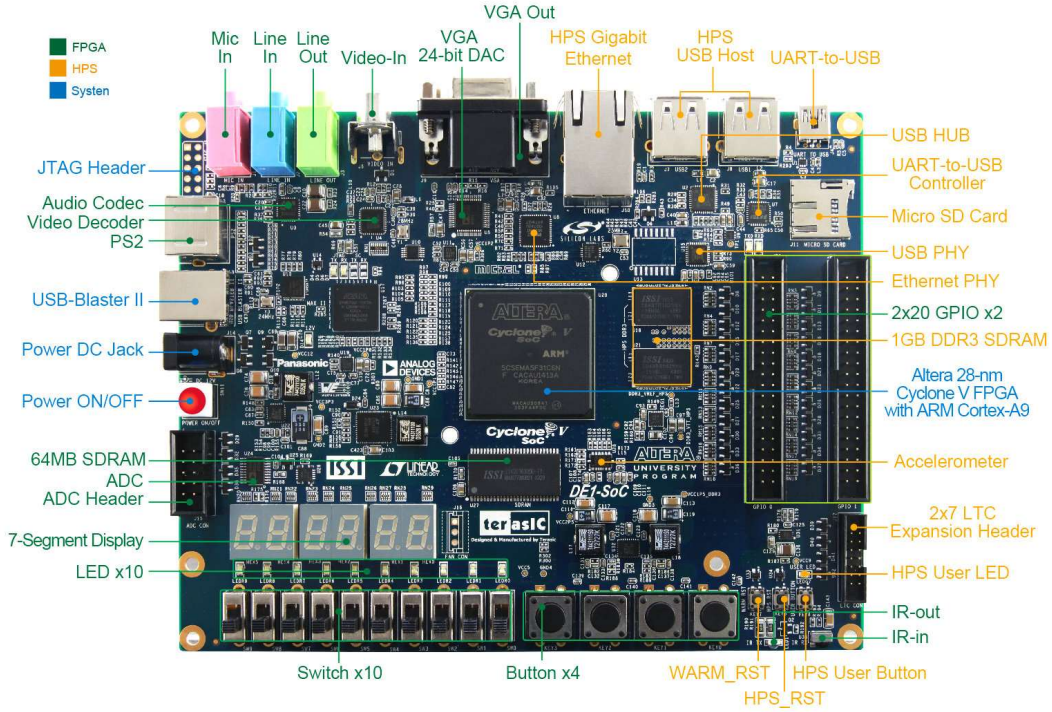


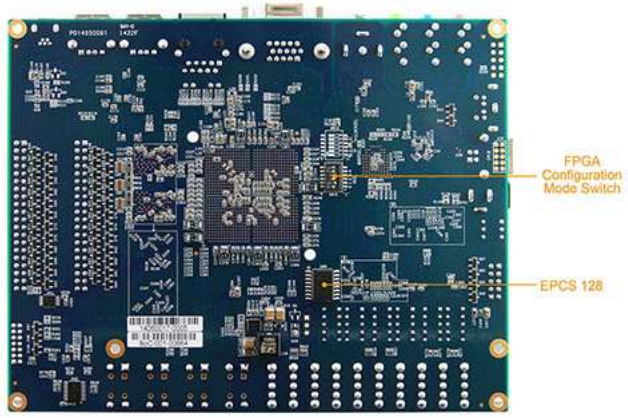
DE1 SoC Overview and Configuration of Android OS



THE BOARD 1

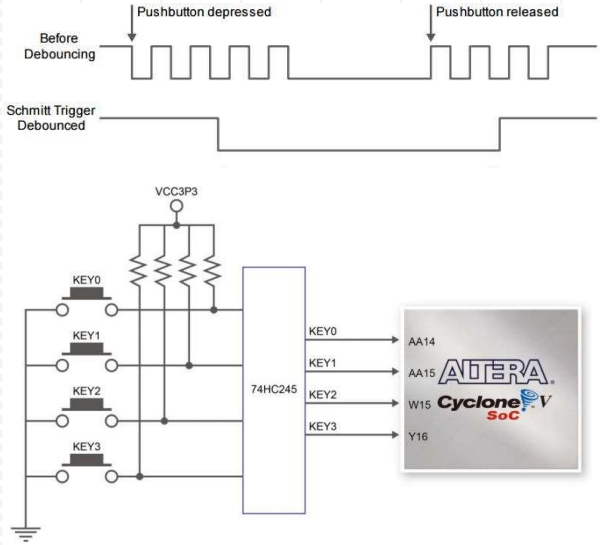
Board Layout





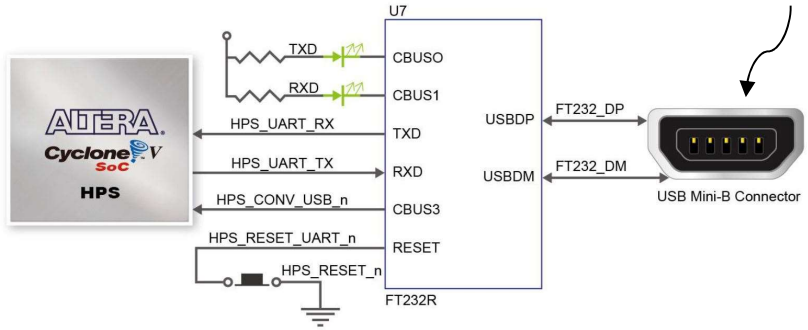
Why so many other chips ?

Switch bouncing problem

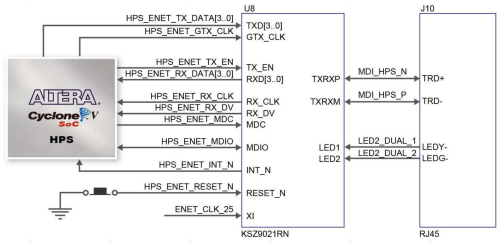


FTDI UART to USB

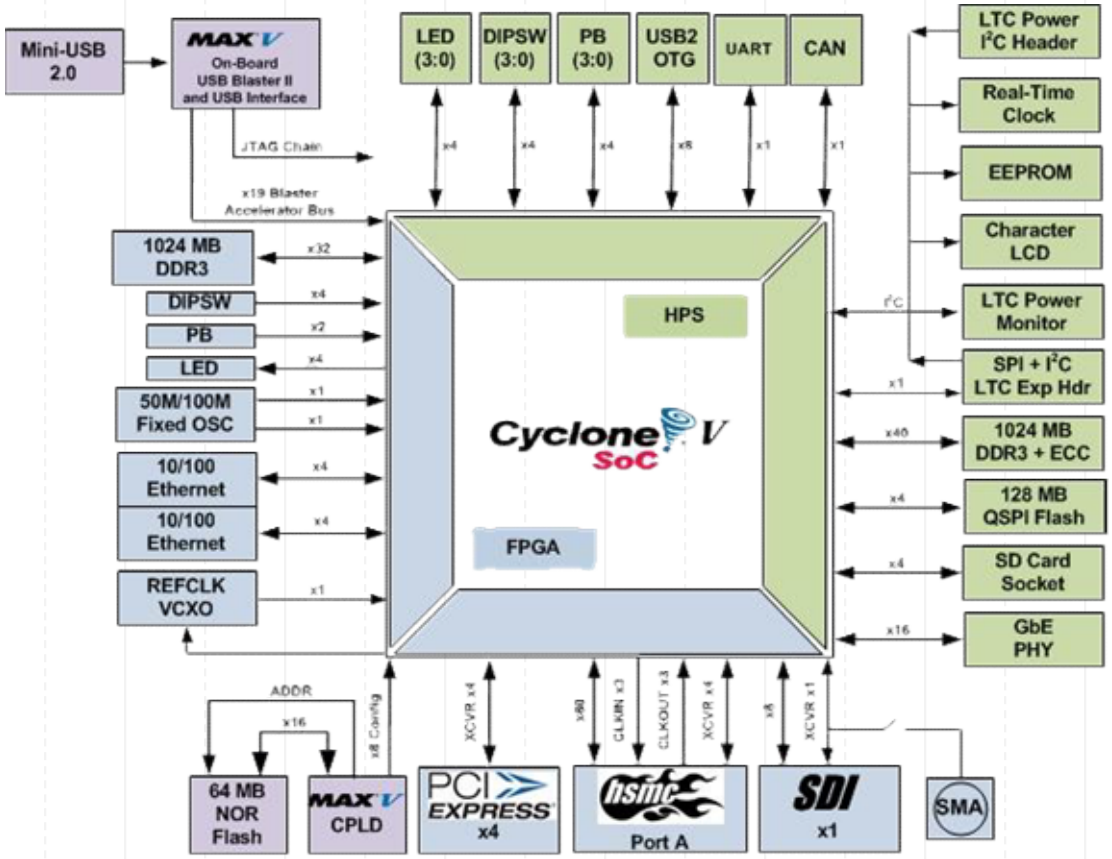
Provides important log data at boot up !!



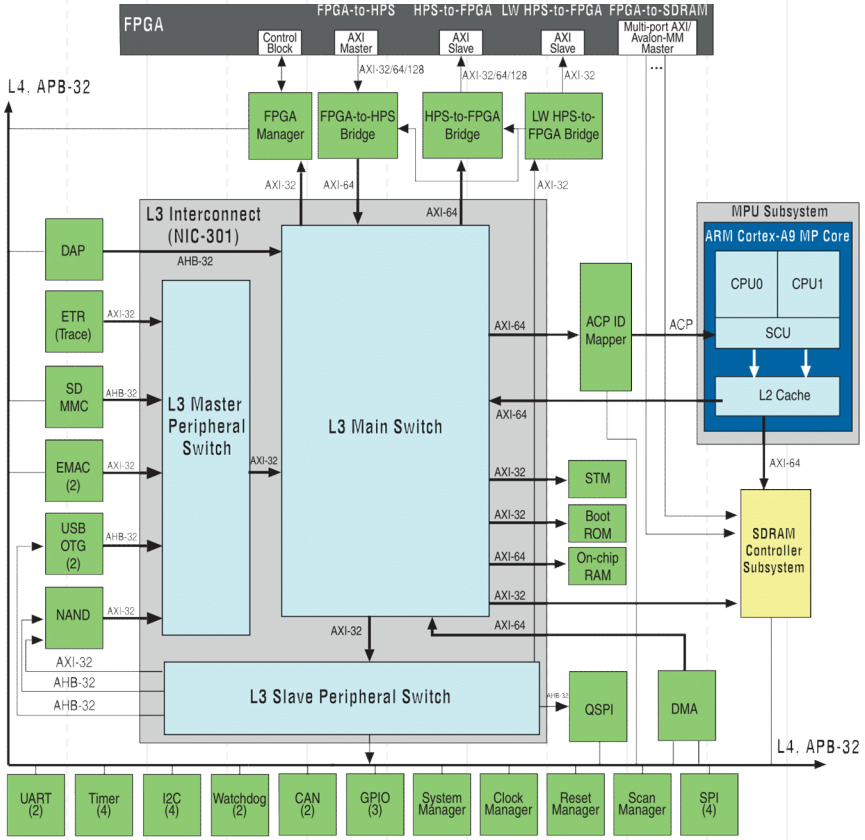
Ethernet Controller

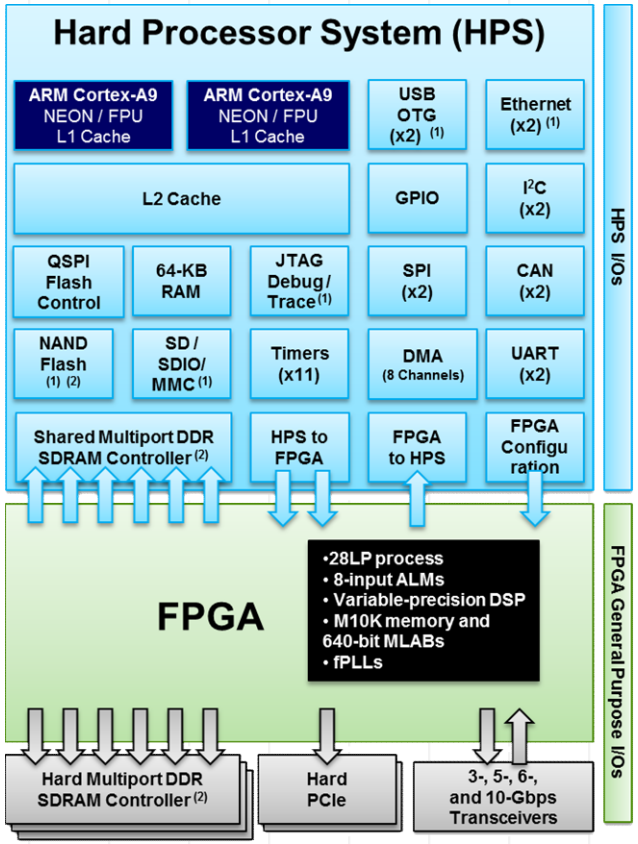


Cyclone V Peripheral Connections

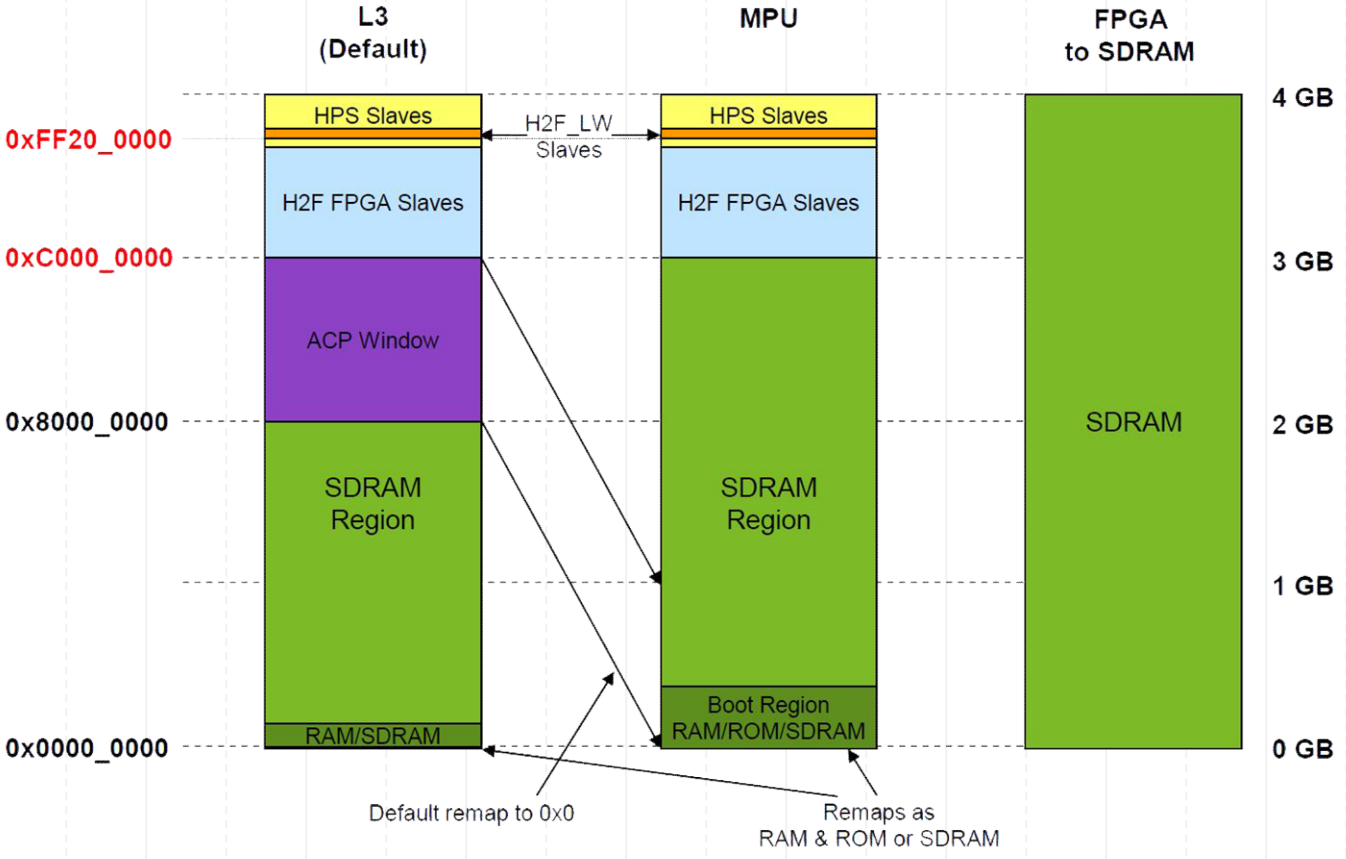


Inside SoC


FPGA
FPGA-to-HPS
HPS-to-FPGA
LW HPS-to-FPGA
FPGA-to-SDRAM
Control Block
AXI Master
AXI Slave
AXI Slave
Multi-port AXI/ Avalon-MM Master
AXI-32/64/128
AXI-32/64/128
AXI-32
L4, APB-32
FPGA Manager
FPGA-to-HPS Bridge
HPS-to-FPGA Bridge
LW HPS-to-FPGA Bridge
AXI-32
AXI-64
AXI-32
L3 Interconnect (NIC-301)
AXI-64
MPU Subsystem
ARM Cortex-A9 MP Core
CPU0
CPU1
SCU
L2 Cache
DAP
AHB-32
ETR (Trace)
AXI-32
SD MMC
AHB-32
EMAC (2)
AXI-32
USB OTG (2)
AHB-32
NAND
AXI-32
L3 Master Peripheral Switch
AXI-32
L3 Main Switch
AXI-64
ACP ID Mapper
ACP
AXI-64
AXI-64
AXI-64
STM
AXI-32
Boot ROM
AXI-32
On-chip RAM
AXI-64
SDRAM Controller Subsystem
AXI-32
AXI-64
AXI-32
AXI-32
AHB-32
AHB-32
L3 Slave Peripheral Switch
QSPI
DMA
L4, APB-32
UART (2)
Timer (4)
I2C (4)
Watchdog (2)
CAN (2)
GPIO (3)
System Manager
Clock Manager
Reset Manager
Scan Manager
SPI (4)


Hard Processor System (HPS)
ARM Cortex-A9 NEON / FPU L1 Cache
ARM Cortex-A9 NEON / FPU L1 Cache
USB OTG (x2) (1)
Ethernet (x2) (1)
L2 Cache
GPIO
I2C (x2)
QSPI Flash Control
64-KB RAM
JTAG Debug/ Trace (1)
SPI (x2)
CAN (x2)
NAND Flash (1) (2)
SD / SDIO/ MMC (1)
Timers (x11)
DMA (8 Channels)
UART (x2)
Shared Multiport DDR SDRAM Controller (2)
HPS to FPGA
FPGA to HPS
FPGA Configu ration
HPS I/Os
FPGA
•28LP process
• 8-input ALMs
• Variable-precision DSP
• M10K memory and 640-bit MLABs
• fPLLs
FPGA General Purpose I/Os
Hard Multiport DDR SDRAM Controller (2)
Hard PCIe
3-, 5-, 6-, and 10-Gbps Transceivers

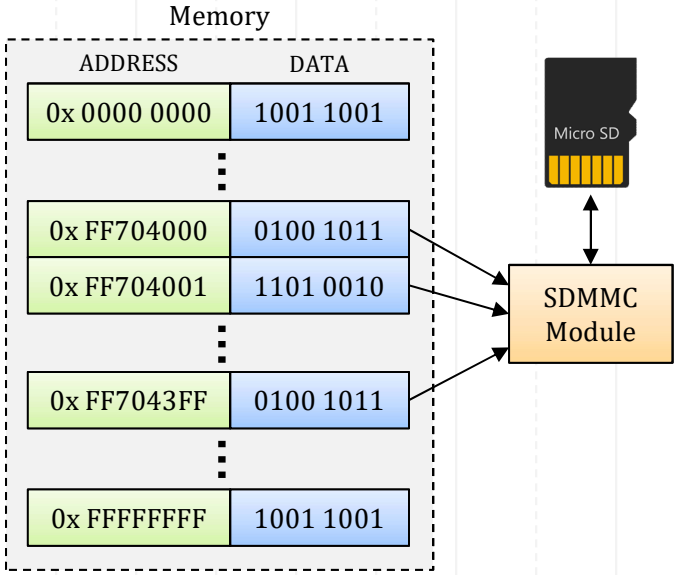
Physical Address Mapping



ACP: Accelerator Coherency Port

Cyclone V HPS Memory Map

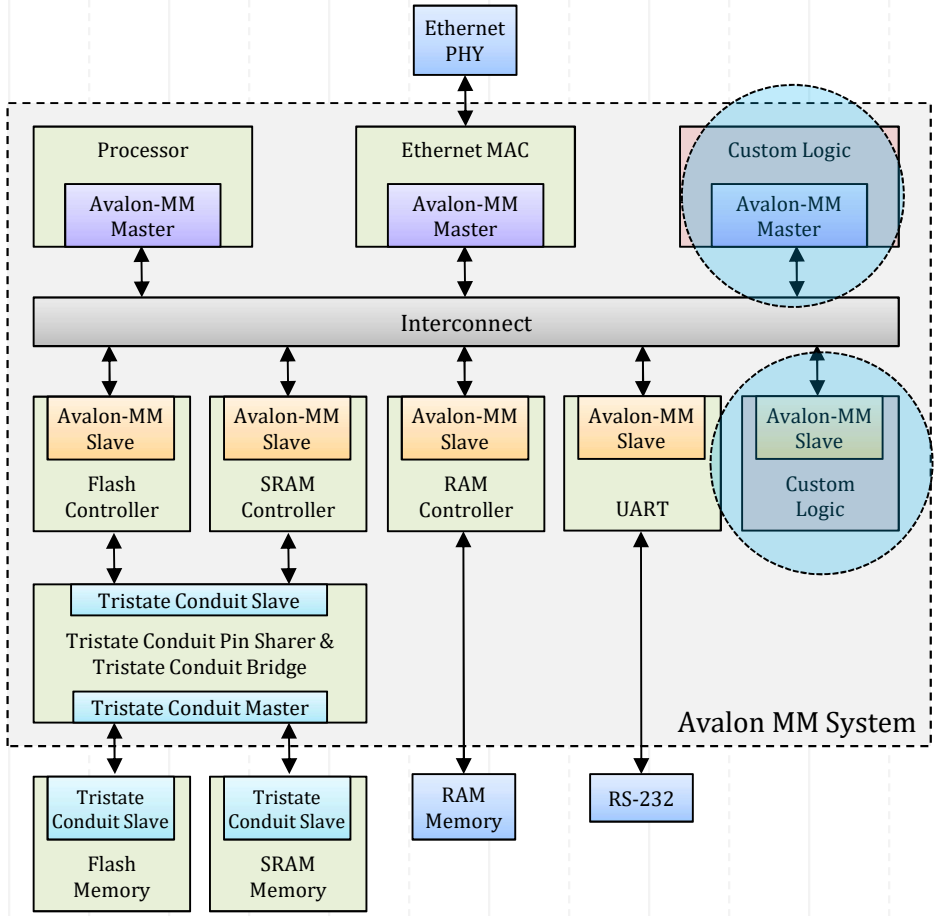
[5]



0x C0000000 - 0x FBFFFFFF	Slaves via HP AXI Bridge
0x FF200000 - 0x FF3FFFFF	Slaves via LW AXI Bridge
0x FF704000 - 0x FF7043FF	SDMMC Module
0x FF705000 - 0x FF7050FF	QSPI Flash Controller Module
0x FF708000 - 0x FF70A07F	GPIO Module
0x FFB00000 - 0x FFB7FFFF	USB OTG Controller
0x FFFE0000 - 0x FFE01FFF	DMA Module
0x FFFD0000 - 0x FFFDFFFF	Boot ROM Module
0x FFFD0000 - 0x FFFFFFFF	On-Chip RAM Module

Custom Logic Bindings

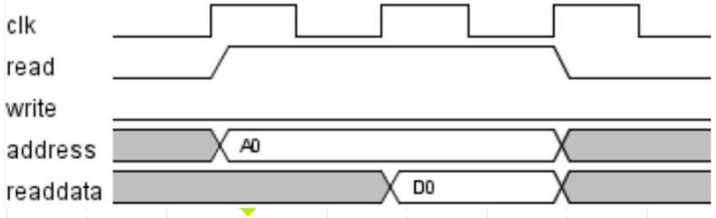
1. Custom Logic can be a master or slave
2. Usually NOIS CPU (Soft IP) is Avalon-MM Master
3. Usually any custom I/O is Avalon-MM Slave



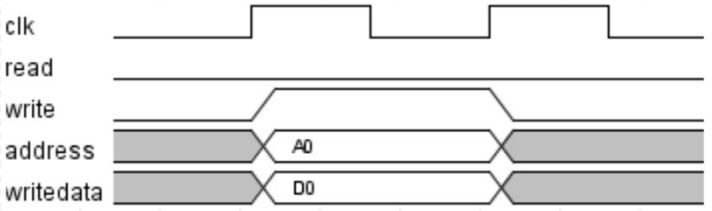
Avalon-MM Slave Interface

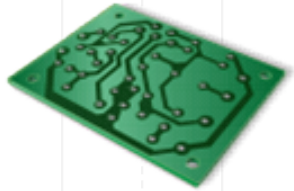
Name	Width	Direction	Comments
avs_address	64 bits	Input	Address of slave being accessed
avs_read	1 bit	Input	Read operation requested
avs_write	1 bit	Input	Write operation requested
avs_readdata	8, 16, 32, or 64 bits	Output	Data read from slave
avs_writedata	8, 16, 32, or 64 bits	Input	Data to be written to slave

Read Waveforms:



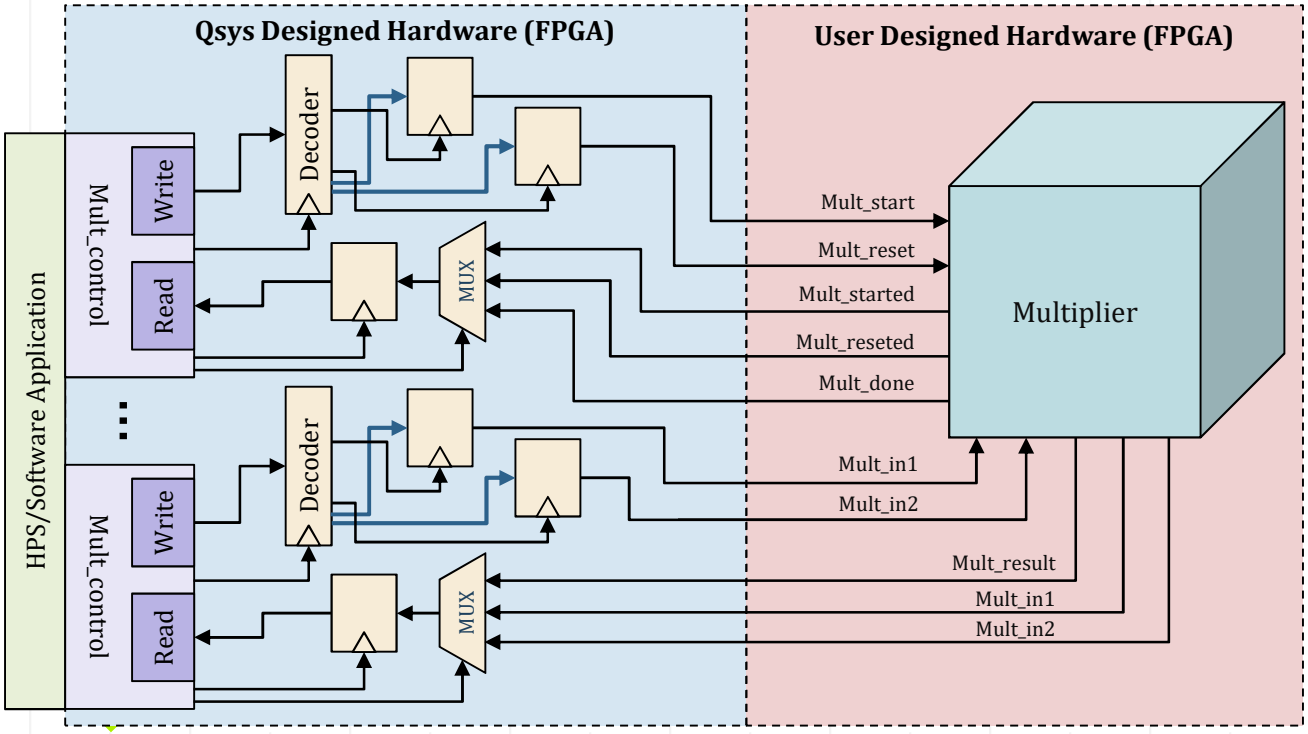
Write Waveforms:



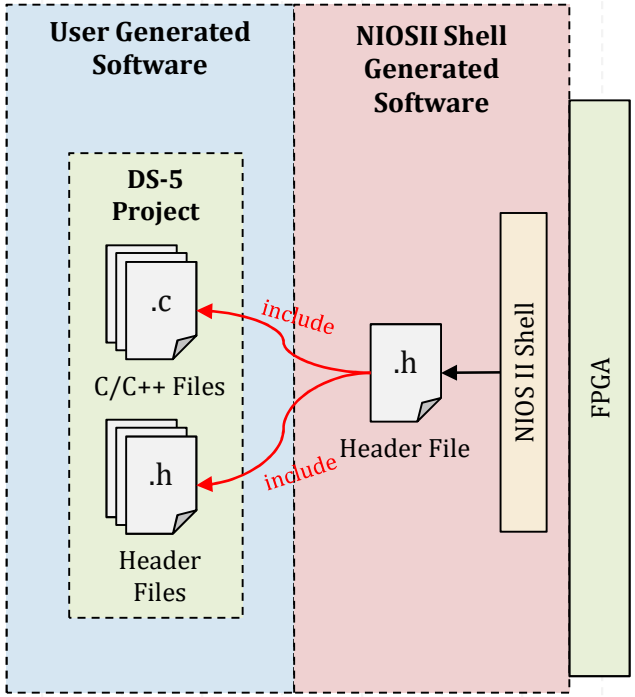


LAB 3 - MULTIPLIER 2

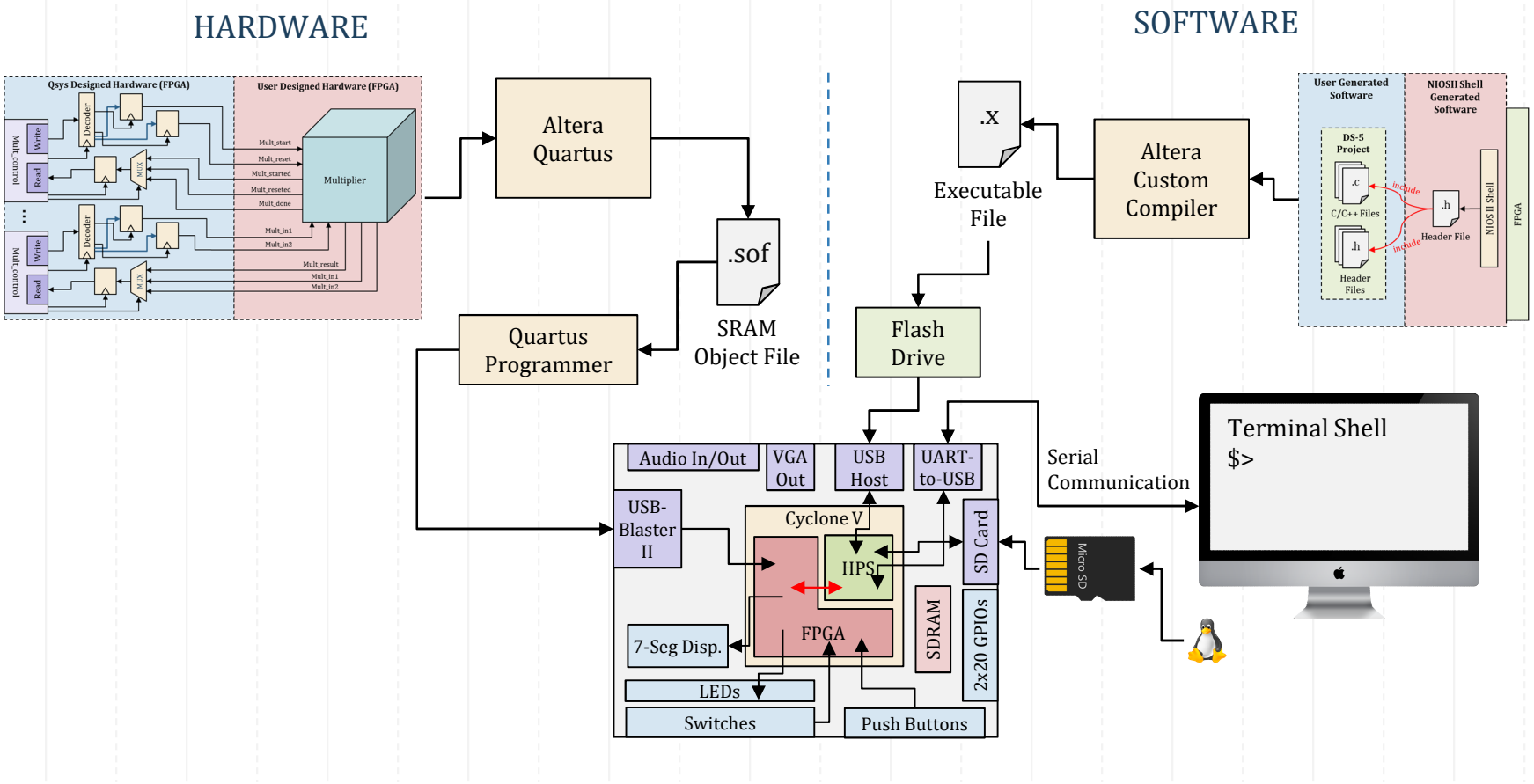
Avalon-MM Slave Interface – Hardware Side

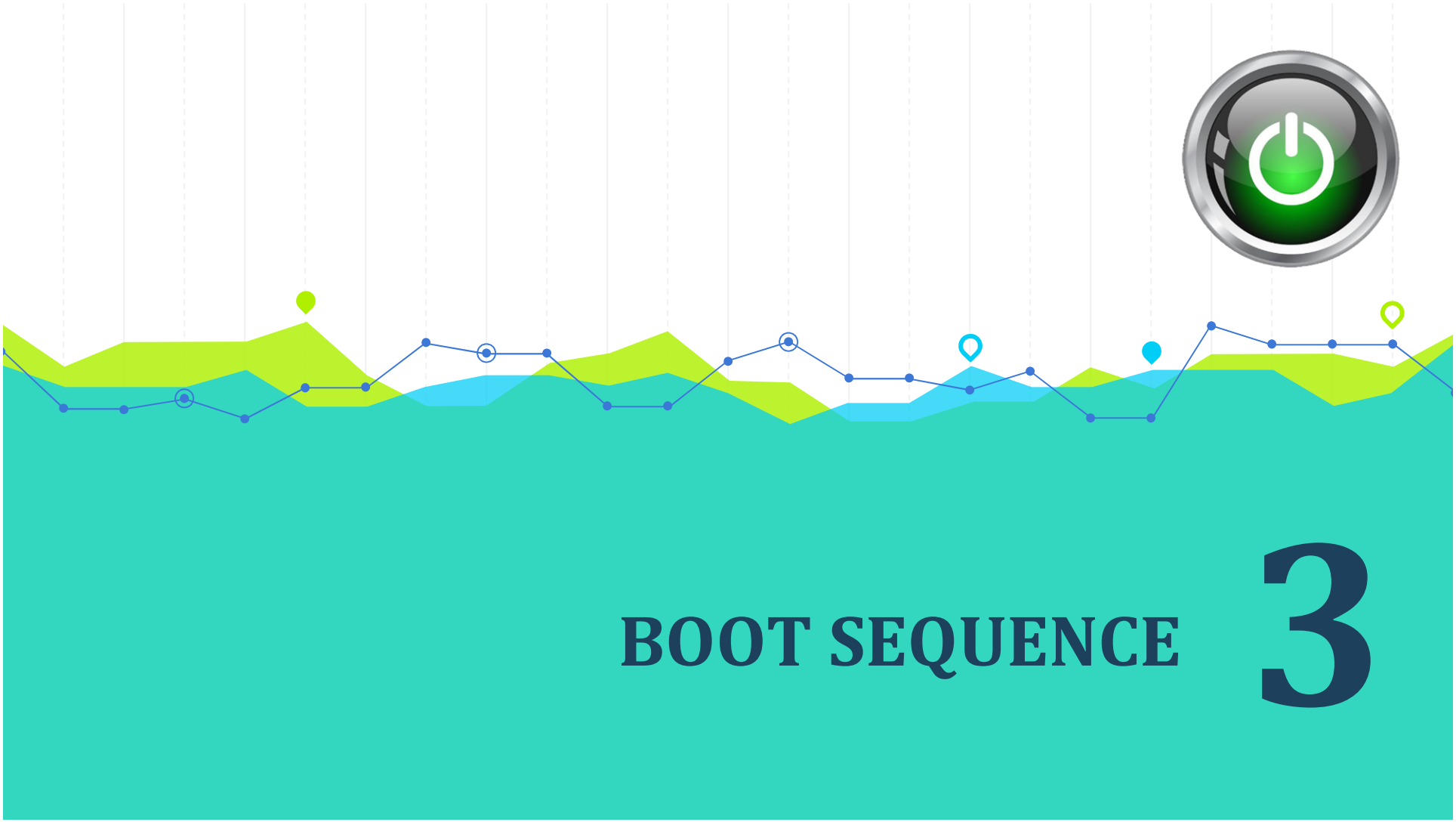


Avalon-MM Slave Interface – Software Side



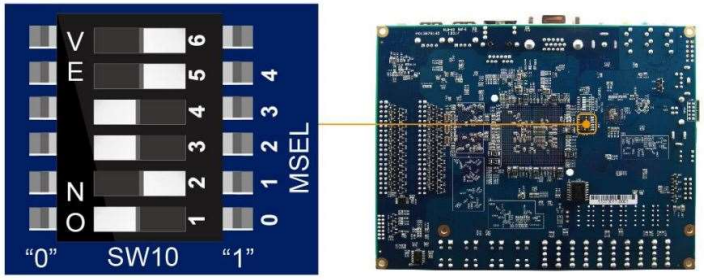
Putting it all together





BOOT SEQUENCE 3

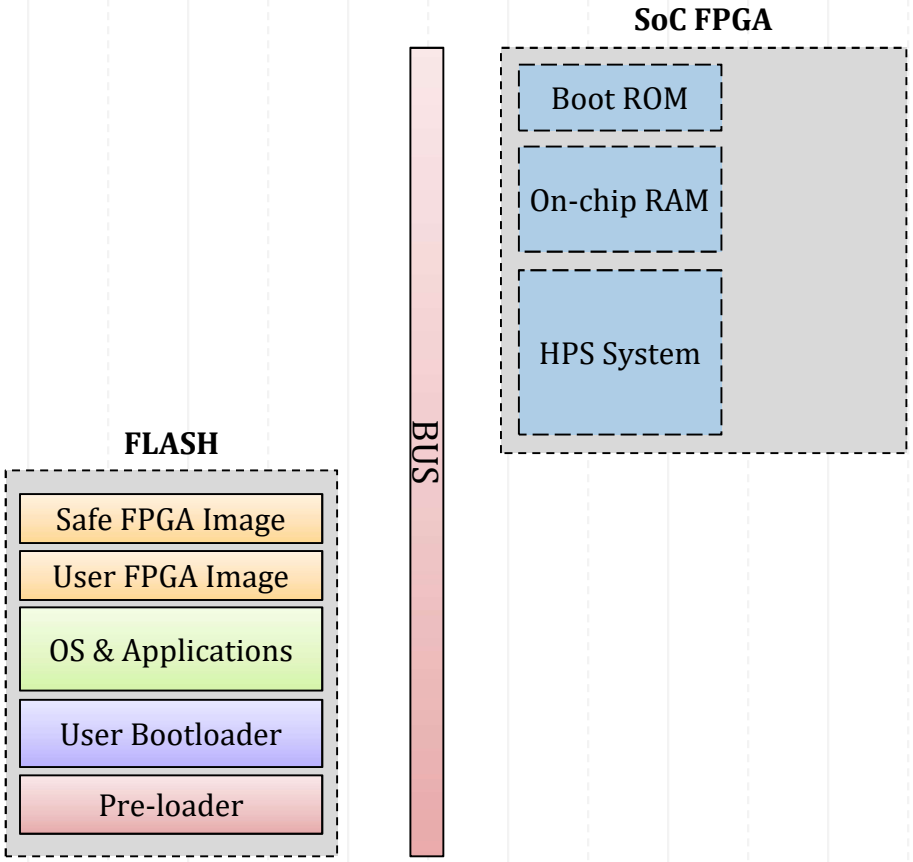
Internal ROM Configuration



MSEL[4:0]	Configuration Scheme	Description
10010	AS	FPGA configured from EPCQ (default)
01010	FPPx32	FPGA configured from HPS software: Linux
00000	FPPx16	FPGA configured from HPS software: U-Boot, with image stored on the SD card, like LXDE Desktop or console Linux with frame buffer edition

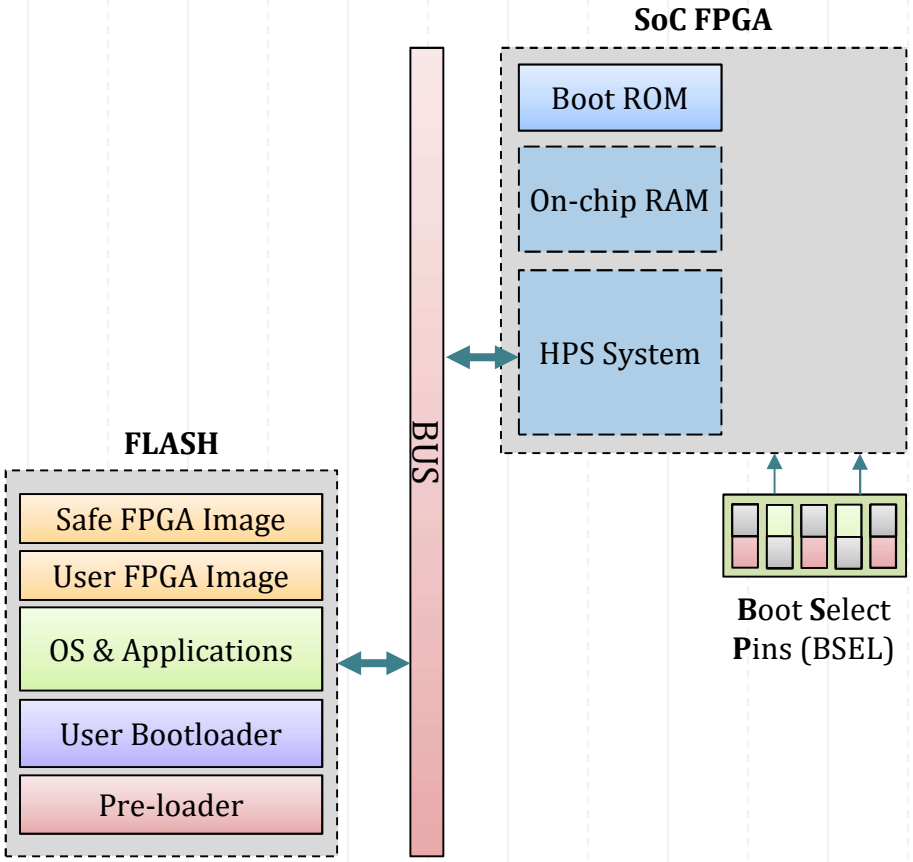
Boot Stages

1. Reset



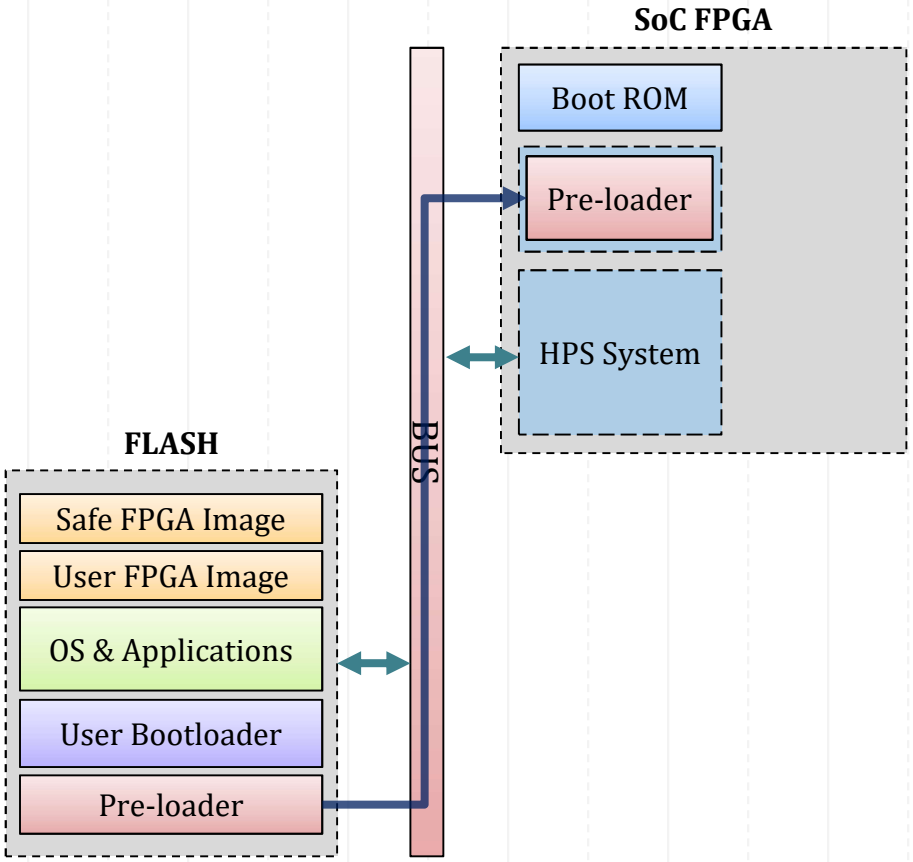
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins



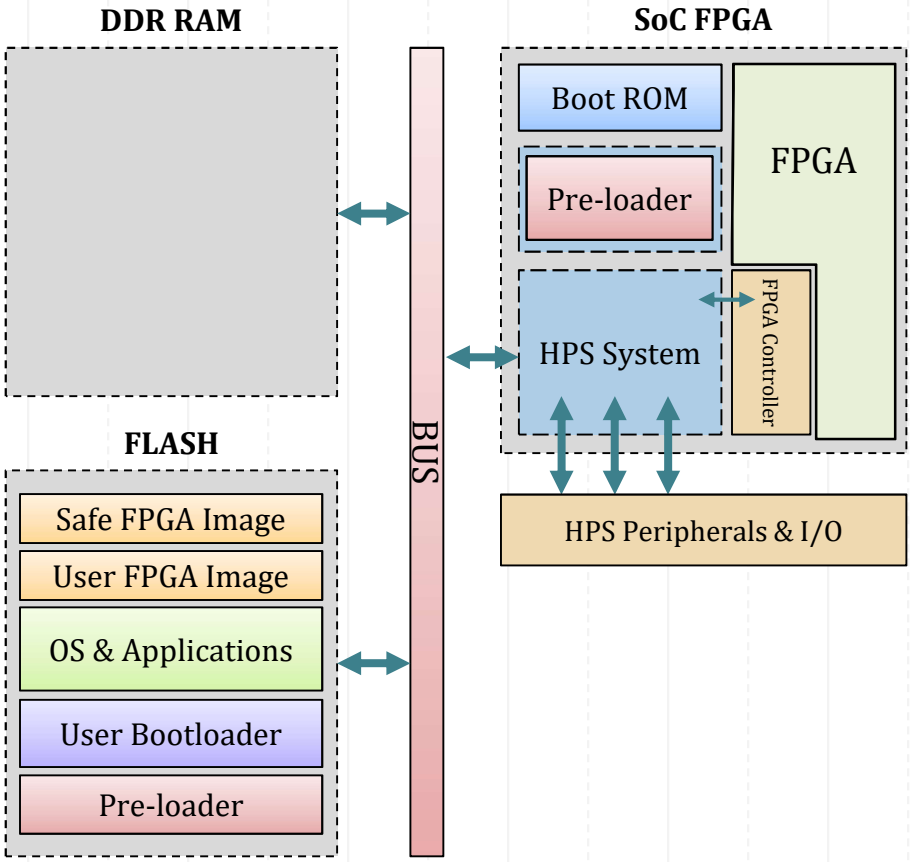
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM



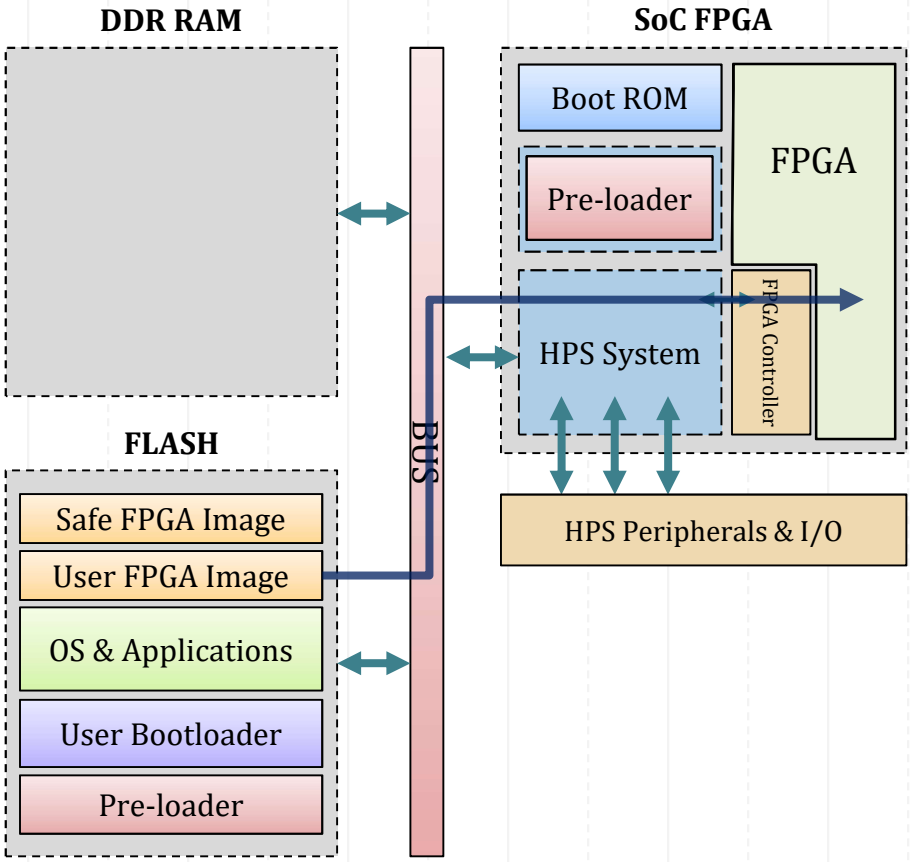
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR



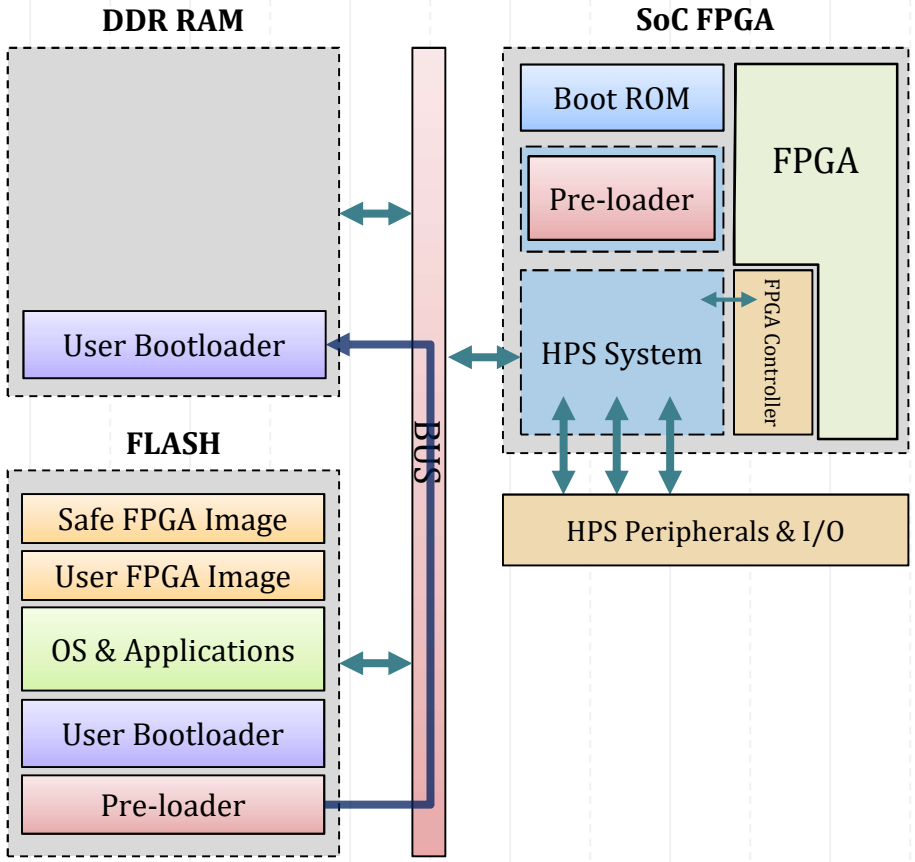
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)



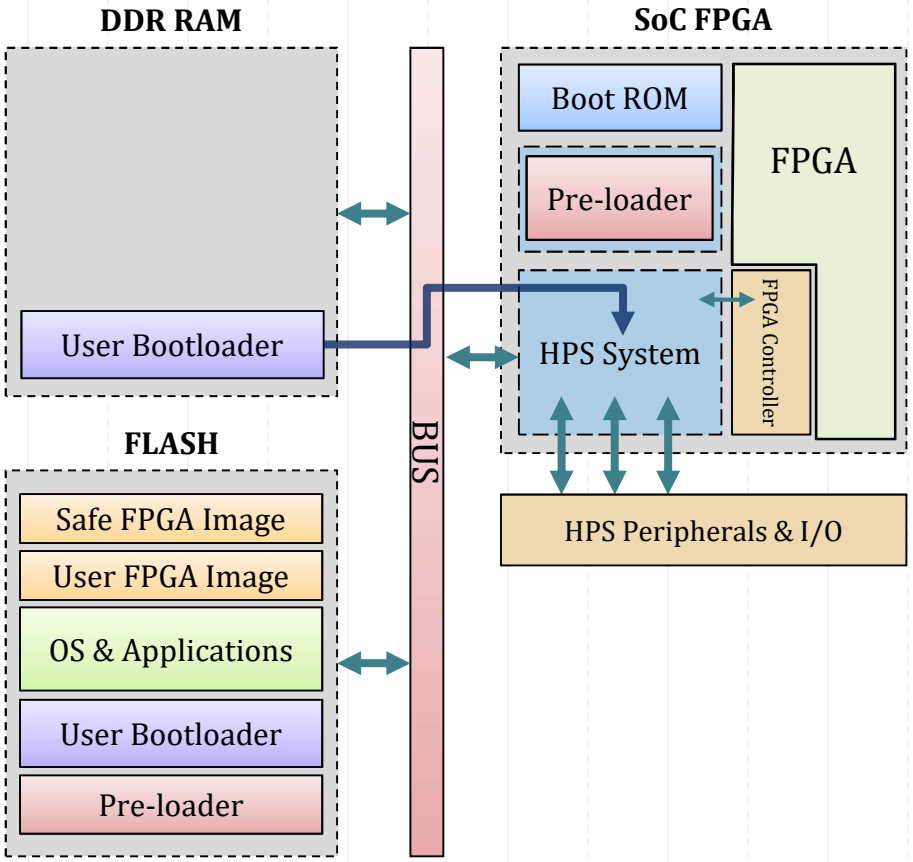
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM



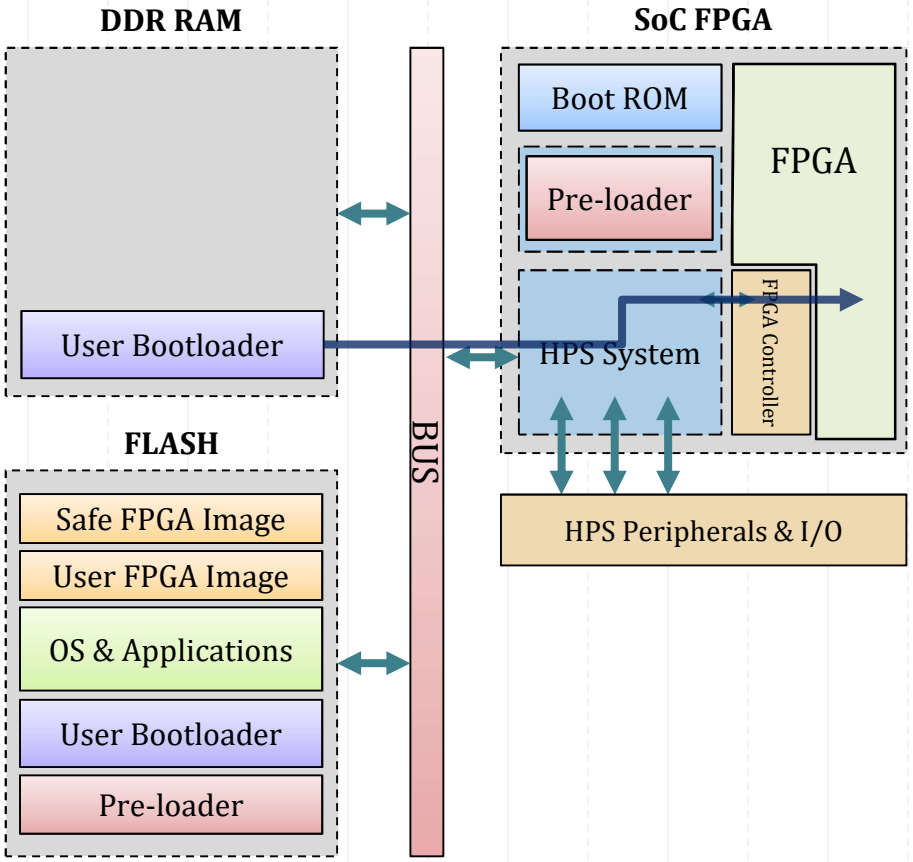
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader



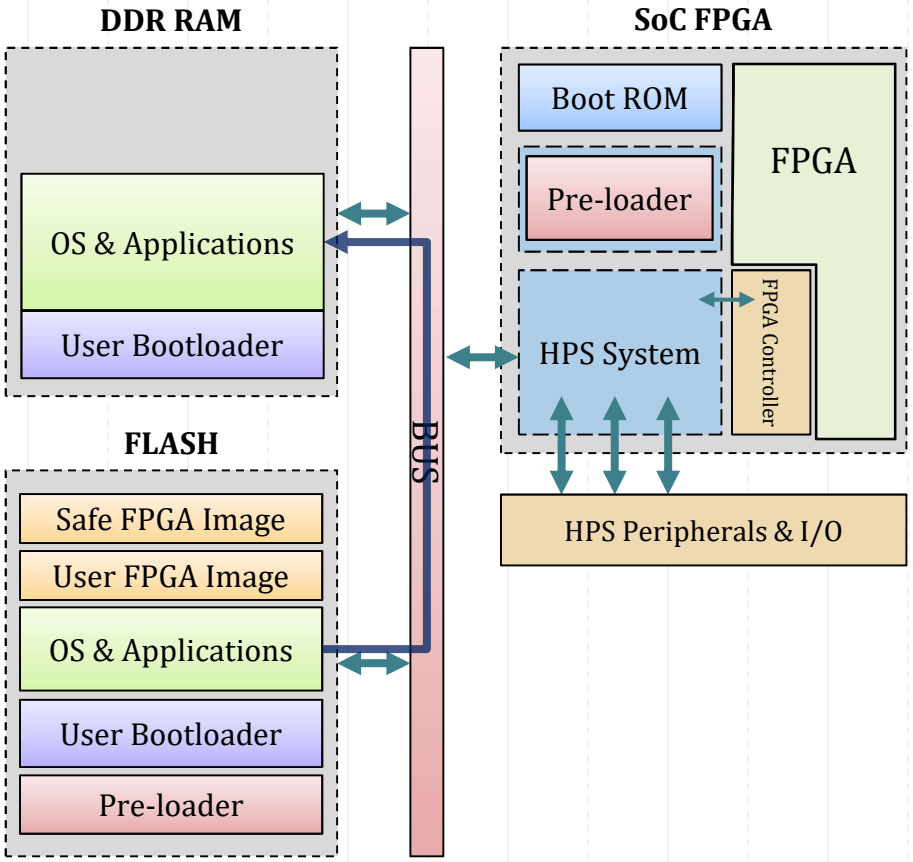
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)



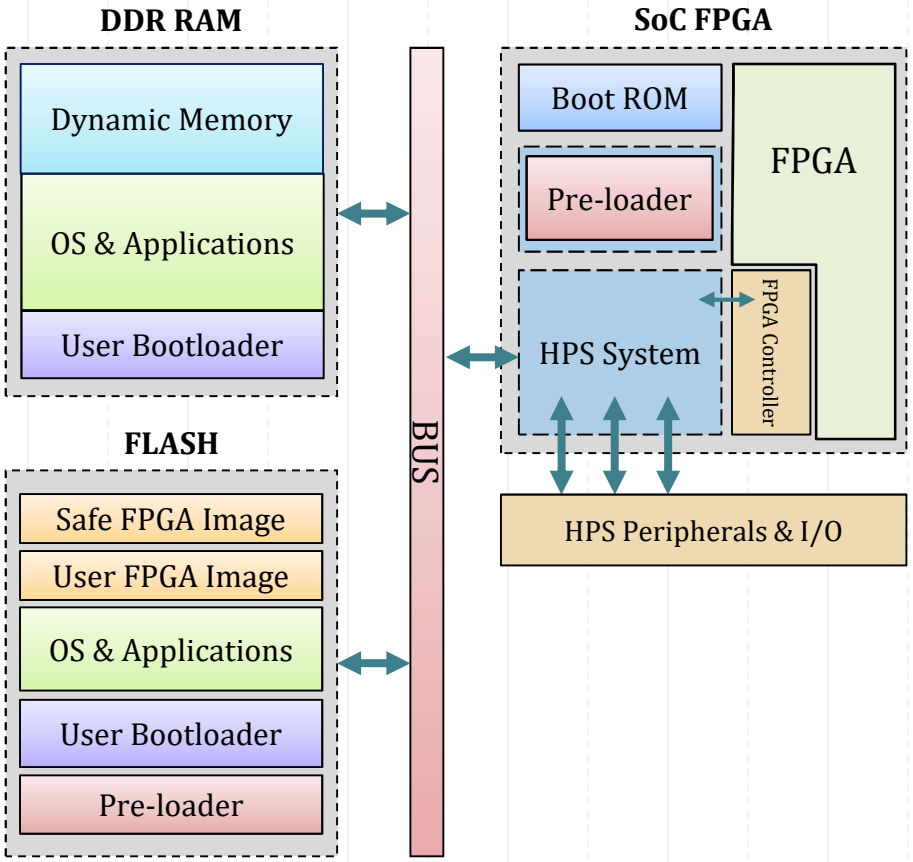
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM



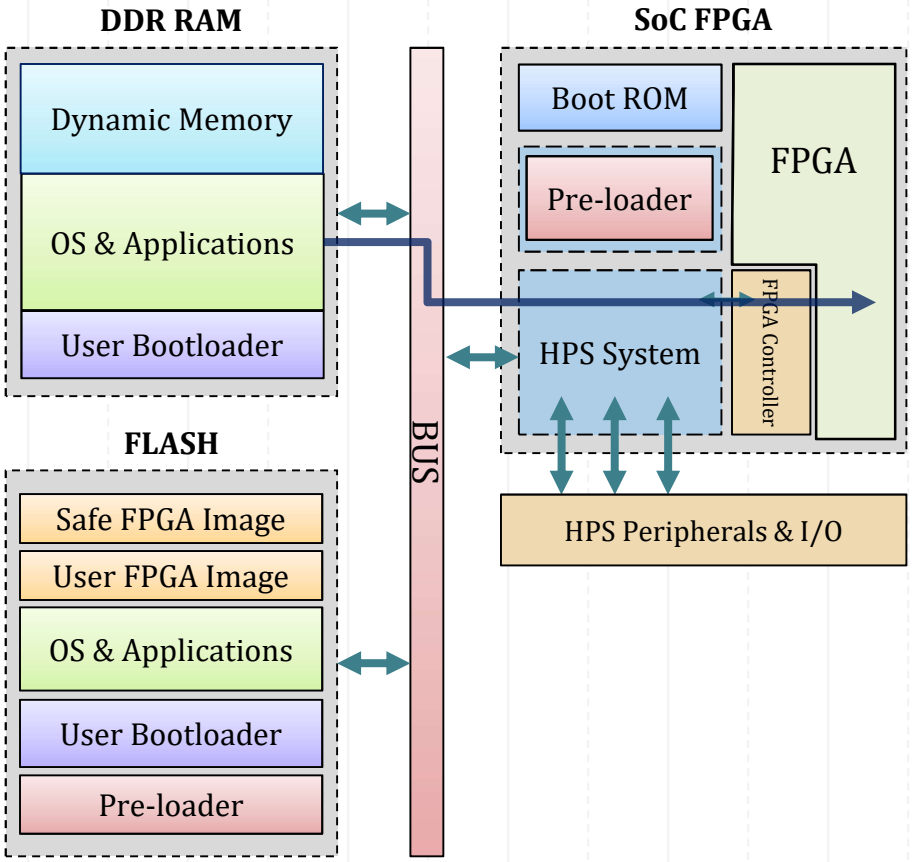
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM
12. Run OS
13. Run Applications



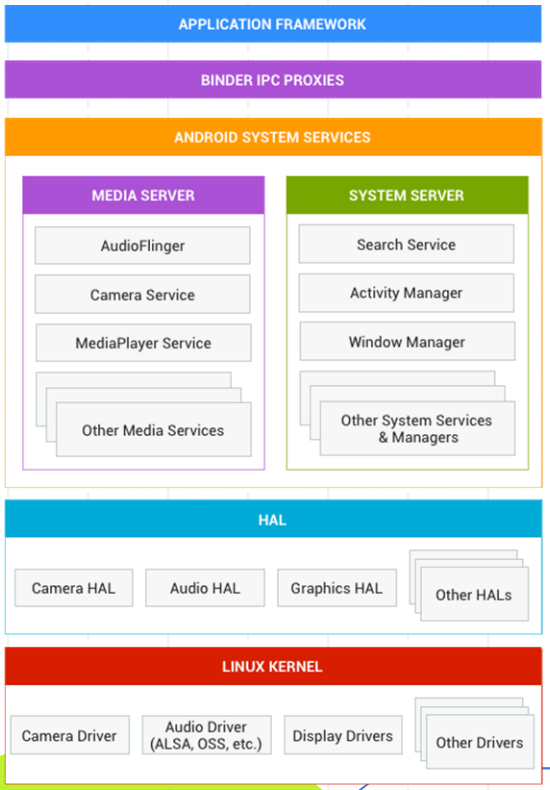
Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader into Onchip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Bootloader into DDR RAM
9. Run User Bootloader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM
12. Run OS
13. Run Applications
14. Configure FPGA (optional)

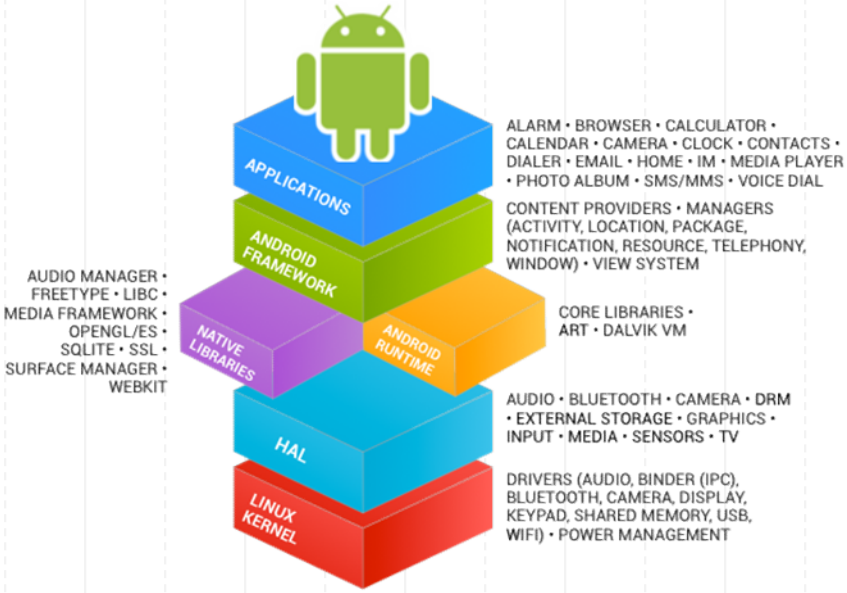


ANDROID ON DE1 SoC 3

What is Android ?

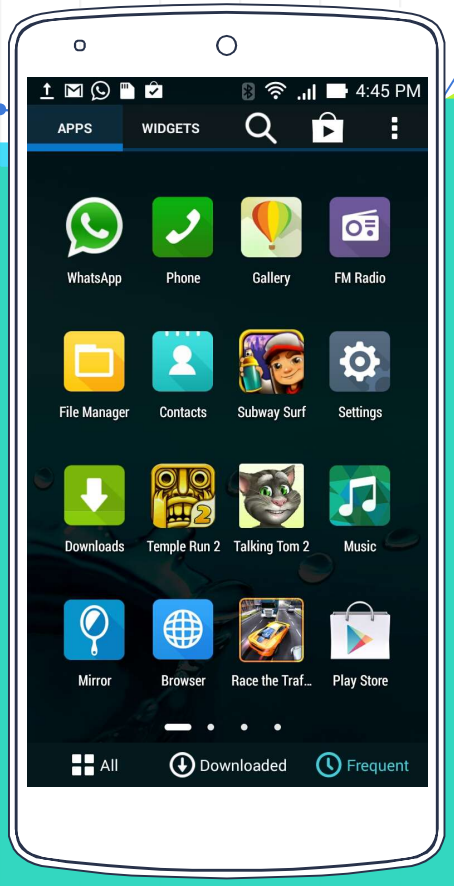


[1]

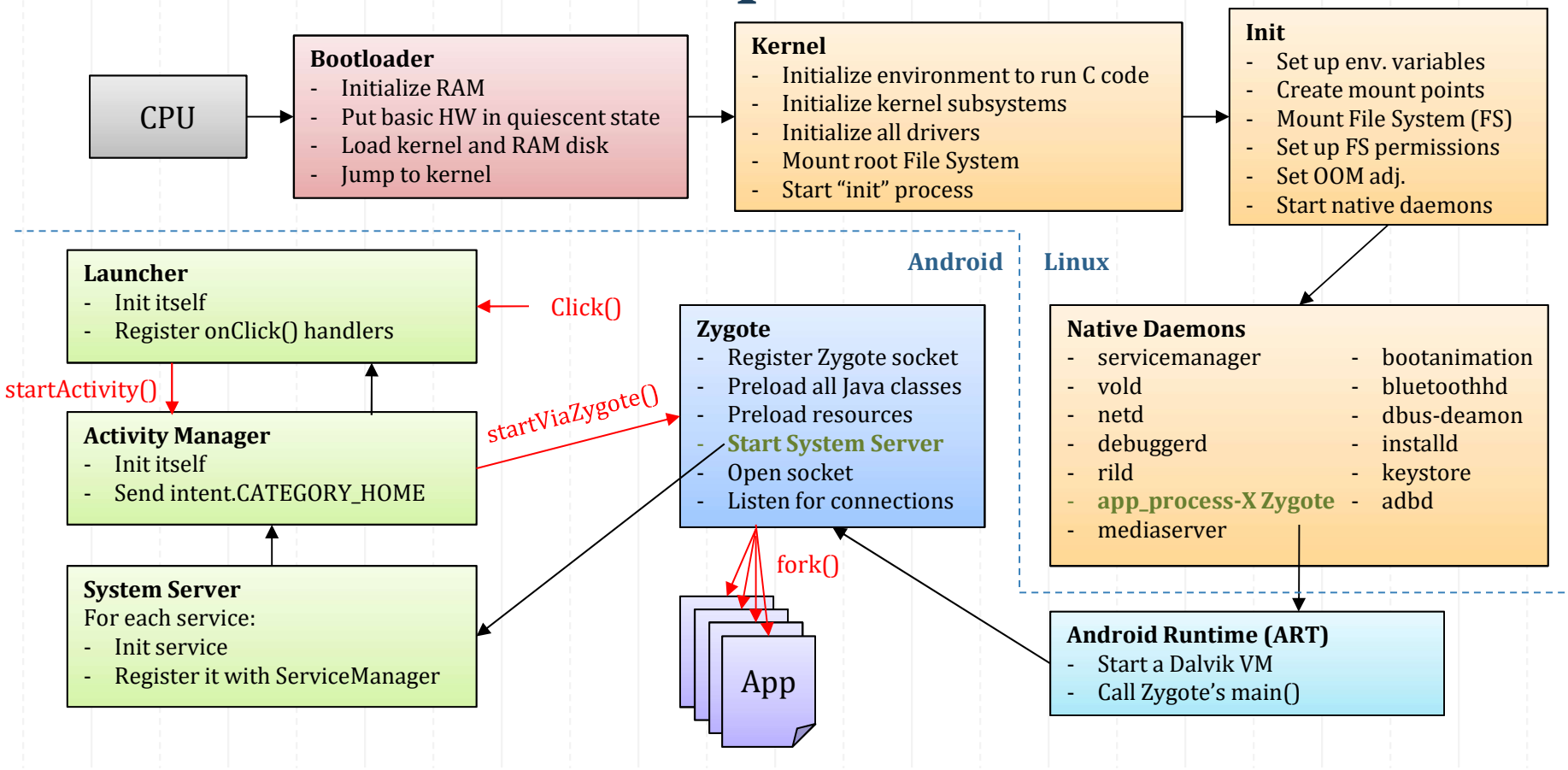


[1]

- Code is available online and managed by Google Android Open Source Project (AOSP).
- Based on Linux.
- Designed for embedded devices.
- Suited to low power devices
- Android is Everywhere

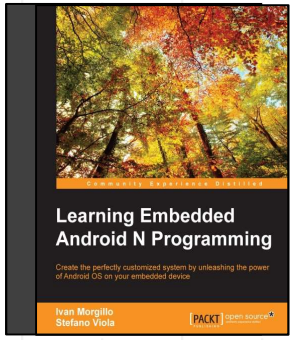


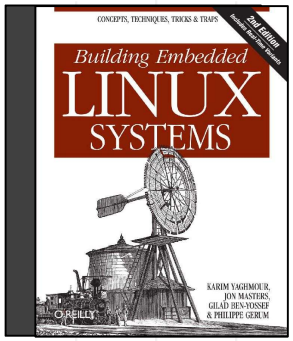
Android Boot Sequence

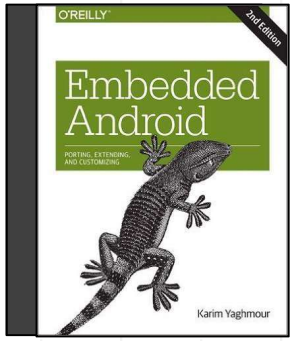


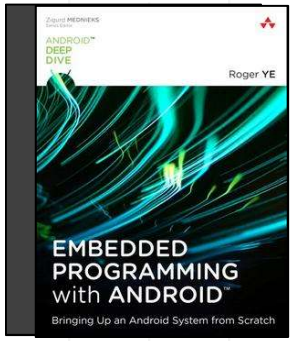
LITERATURE REVIEW 4

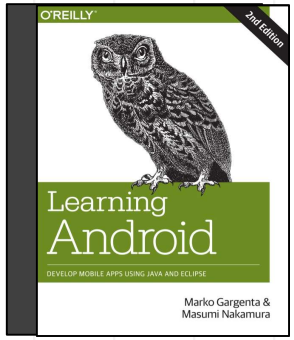
Books Used

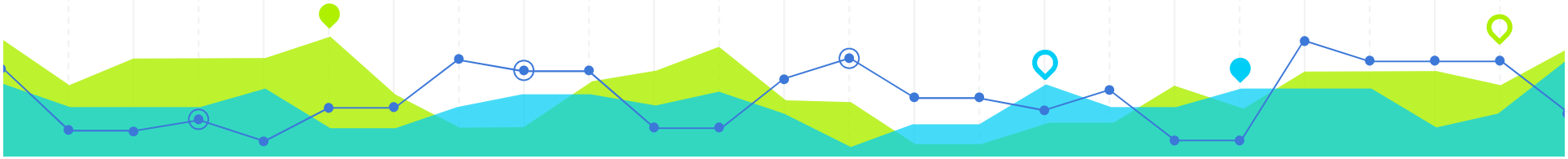

Learning Embedded
Android N Programming
Create the perfectly customized system by unleashing the power of Android OS on your embedded device
Ivan Morgillo
Stefano Viola
PACKT


CONCEPTS, TECHNIQUES, TRICKS & TRAPS
2nd Edition
Building Embedded
LINUX
SYSTEMS
KARIM YAGHMOUR,
JON MASTERS,
GILAD BEN-YOSSEF
& PHILIPPE GERUM
O'REILLY

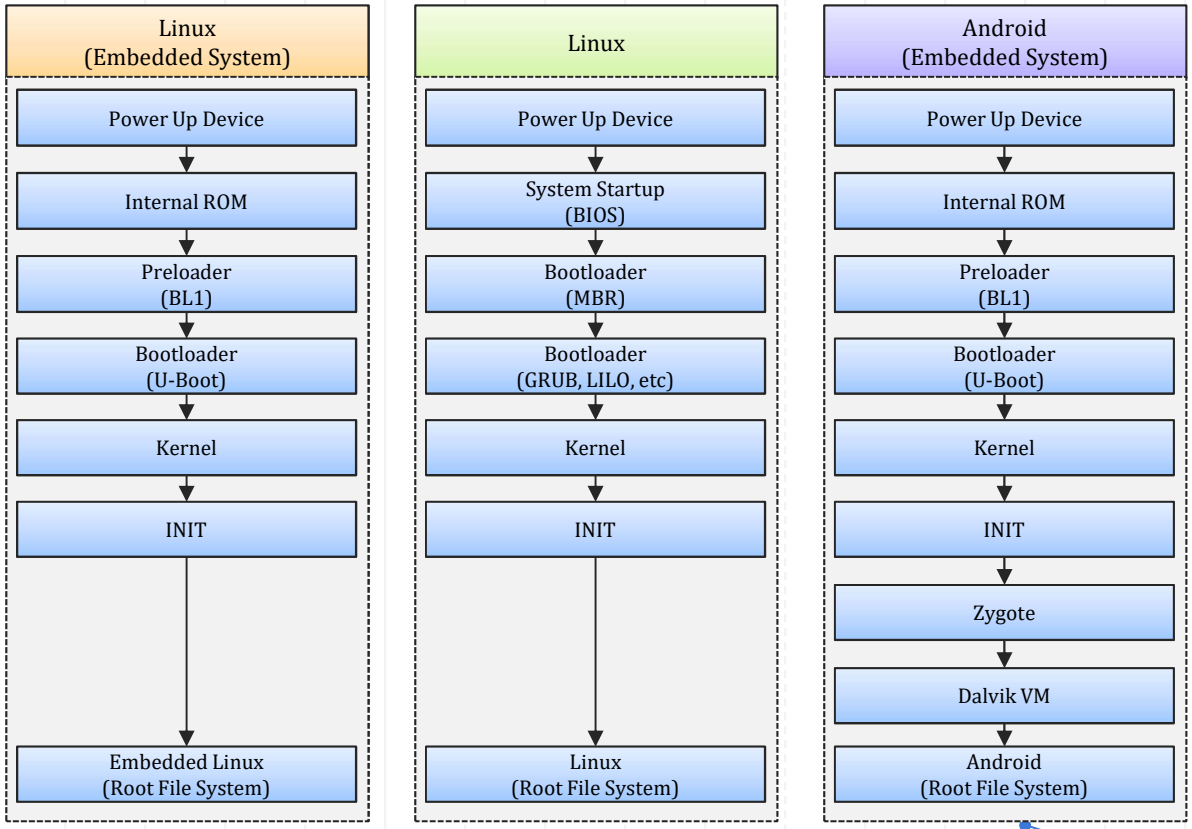

O'REILLY
2nd Edition
Embedded
Android
PORTING, EXTENDING, AND CUSTOMIZING
Karim Yaghmour


ANDROID
DEEP
DIVE
Roger YE
EMBEDDED
PROGRAMMING
with ANDROID
Bringing Up an Android System from Scratch

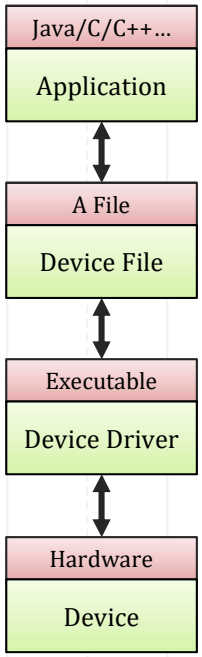

O'REILLY
2nd Edition
Learning
Android
DEVELOP MOBILE APPS USING JAVA AND ECLIPSE
Marko Gargenta &
Masumi Nakamura



Difference Between Linux and Android



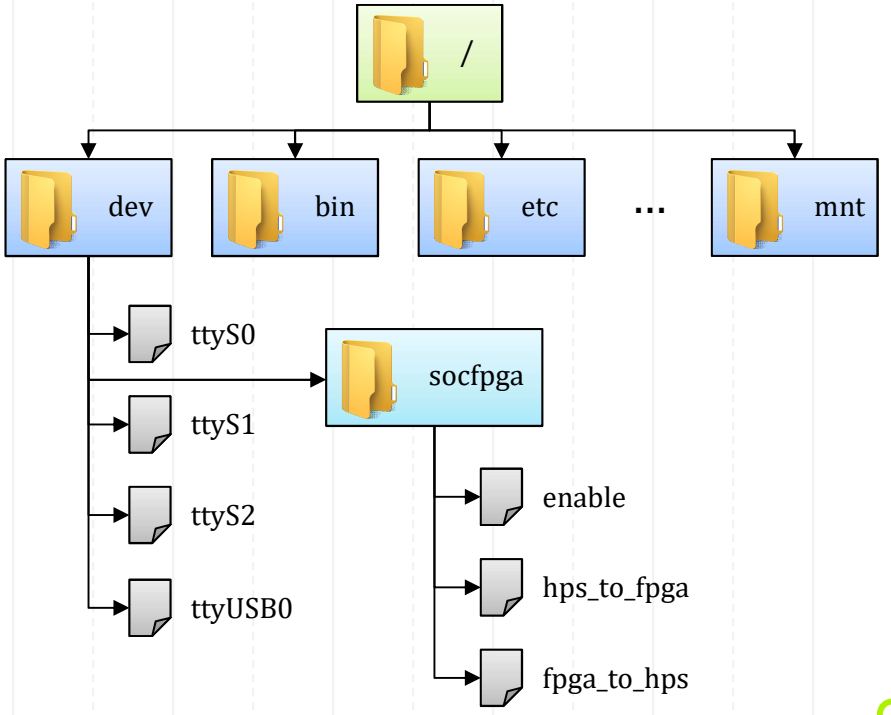
Linux Device Files



The application finds the device file based on device file name

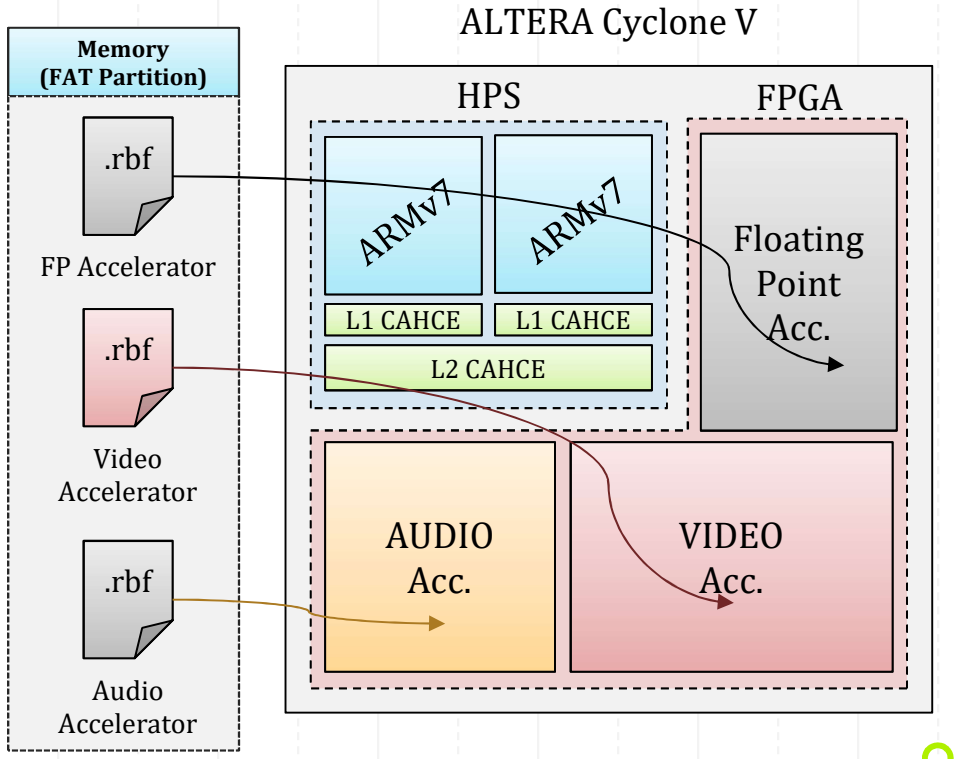
Device file finds the device driver based on the major number

Device driver finds device based on the minor number

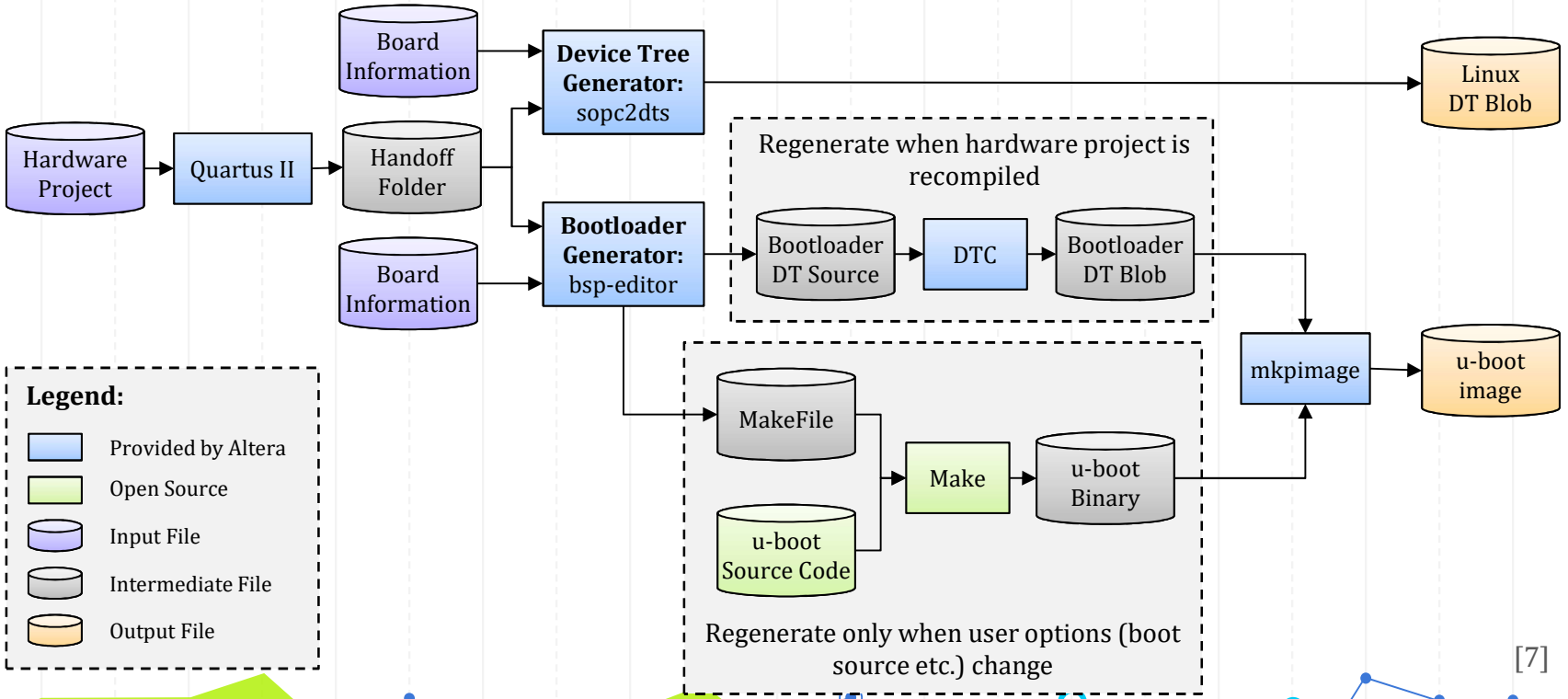


Raw Binary Files (.rbf)

- Write hardware code in VHDL/Verilog/System Verilog
- Compile and verify the design using Quartus to get SRAM Object File (.sof)
- Convert .sof file to .rbf
- Copy the generated .rbf file to SD card's FAT partition
- Decide on how to load the configuration bit stream (Choose from following)
  - Pre-loader script
  - U-Boot source code
  - U-Boot script
  - Linux init
  - Linux application (runtime)



Altera SoC Linux Hardware/Software Handoff

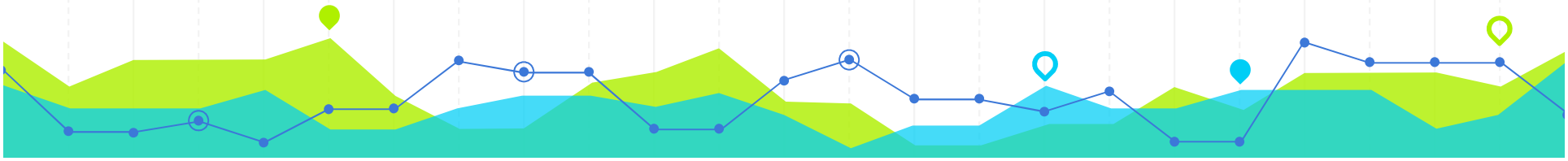


IMPLEMENTATION 4

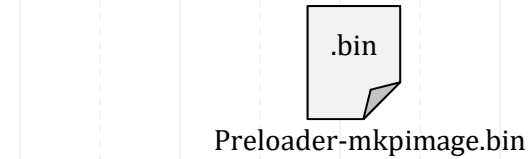
Step 2: Finding Resources

S.No.	Component	Repository Name	GitHub URL
1.	Angstrom Scripts	angstrom-socfpga	https://github.com/altera-opensource/angstrom-socfpga
2.	Boot Loader	u-Boot-socfpga	https://github.com/altera-opensource/u-boot-socfpga
3.	Device Tree Generator	sopc2dts	https://github.com/altera-opensource/sopc2dts
4.	Linux Kernel	linux-socfpga	https://github.com/altera-opensource/linux-socfpga
5.	Reference Designs	linux-refdesigns	https://github.com/altera-opensource/linux-refdesigns
6.	Yocoto Layer	meta-altera	https://github.com/altera-opensource/meta-altera

[6]

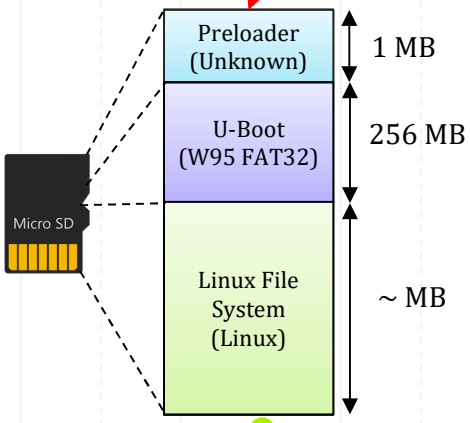


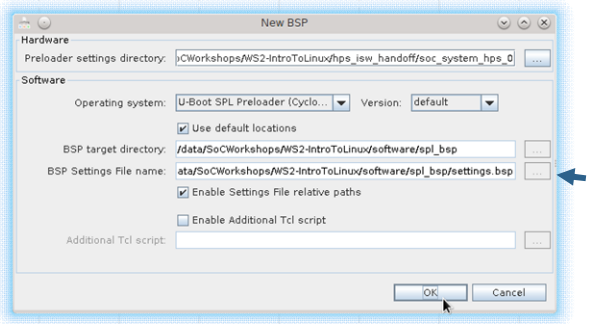
Step 4: Generating Preloader

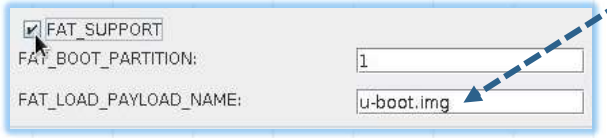


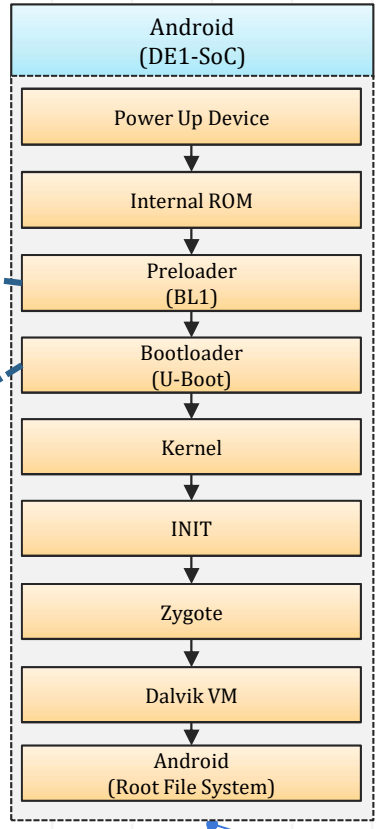
Preloader-mkpimage.bin

dd

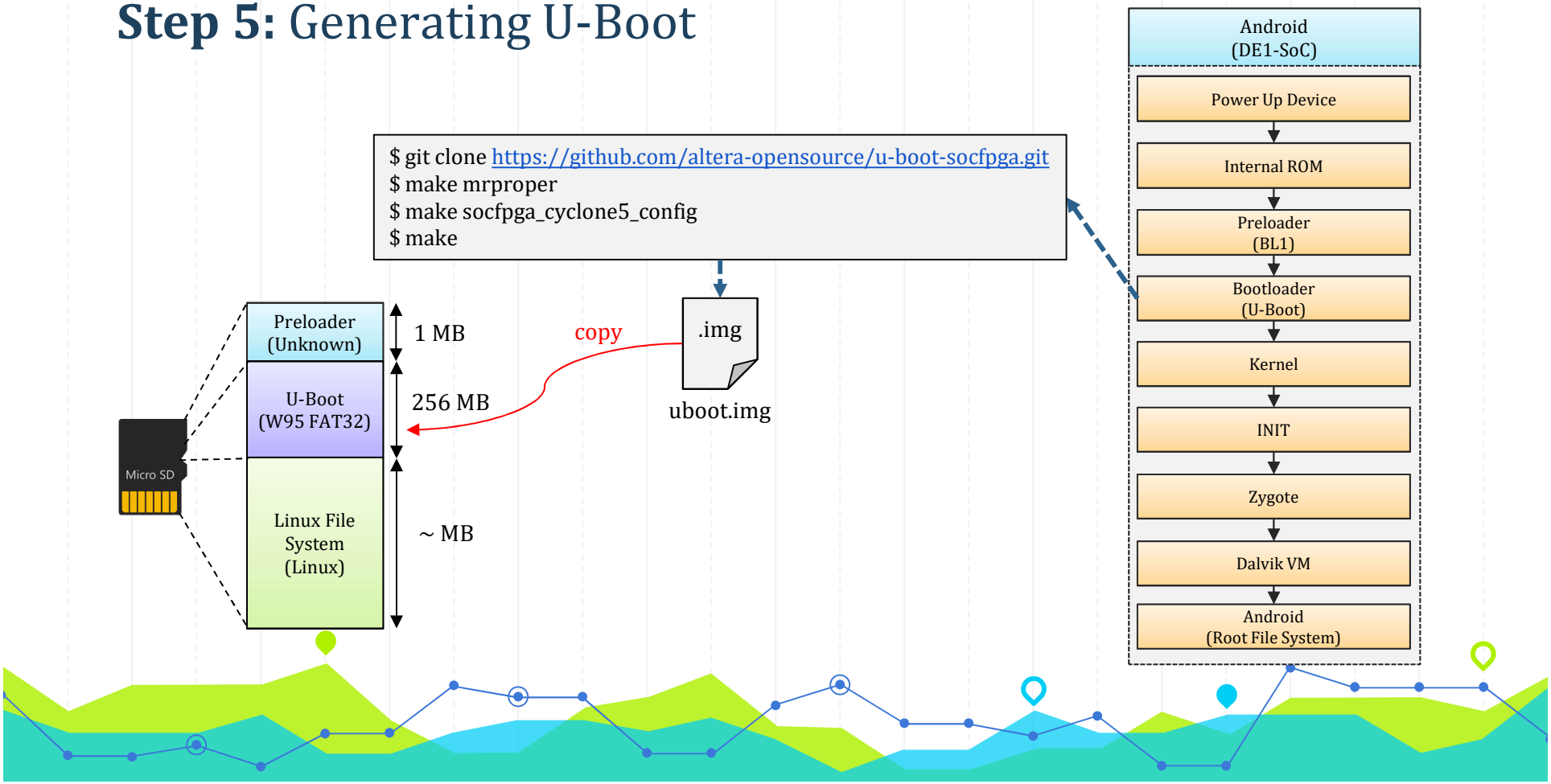




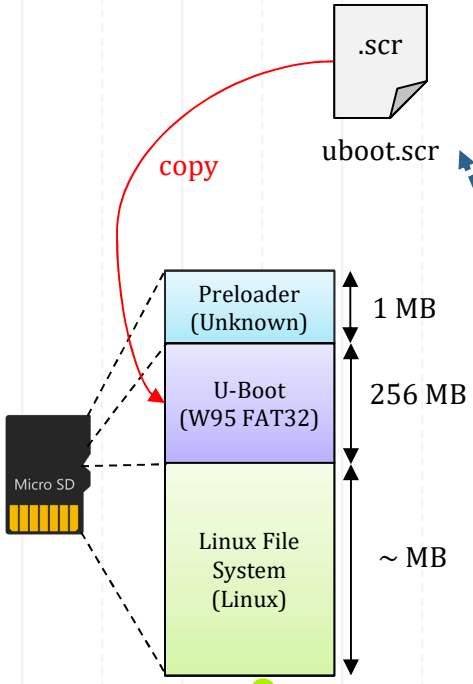




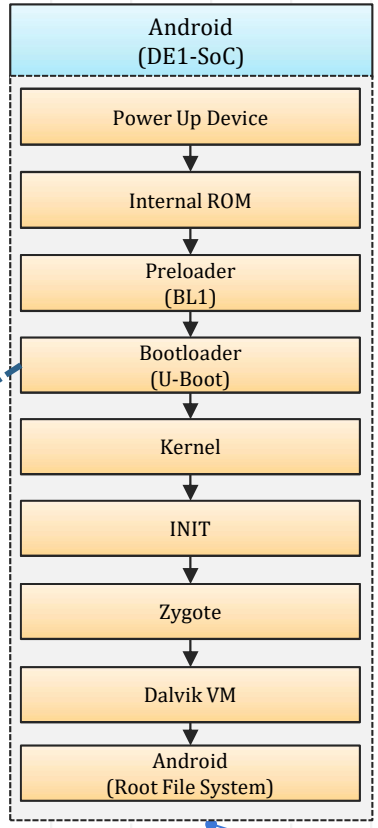
Step 5: Generating U-Boot



Step 6: Generating U-Boot Script

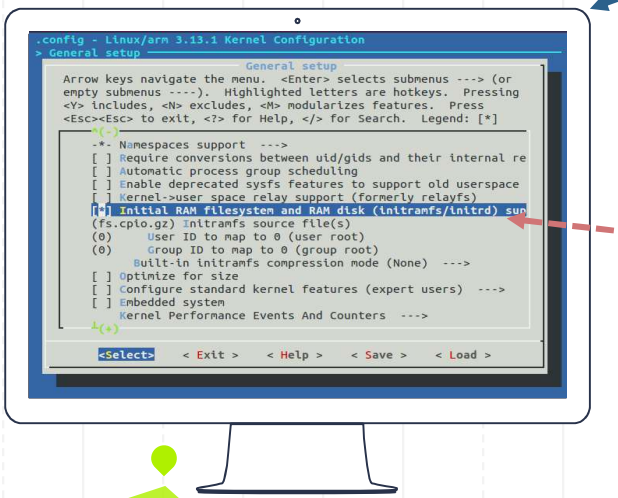



```
echo -- Programming FPGA –
fatload mmc 0:1 $fpgadata soc_config.rbf;
fpga load 0 $fpgadata $filesize;
run bridge_enable_handoff;

echo -- Setting Env Variables –
setenv fdtimage soc_system.dtb;
setenv mmcroot /dev/ram;
setenv mmcload 'mmc rescan;
${mmcloadcmd} mmc 0:${mmcloadpart} ${loadaddr}
${bootimage};
${mmcloadcmd} mmc 0:${mmcloadpart} ${fdtaddr}
${fdtimage};
setenv mmcboot 'setenv bootargs
console=ttyS0,115200 root=${mmcroot} rw
rootwait;
bootz ${loadaddr} - ${fdtaddr}';

run mmcload;
run mmcboot;
```

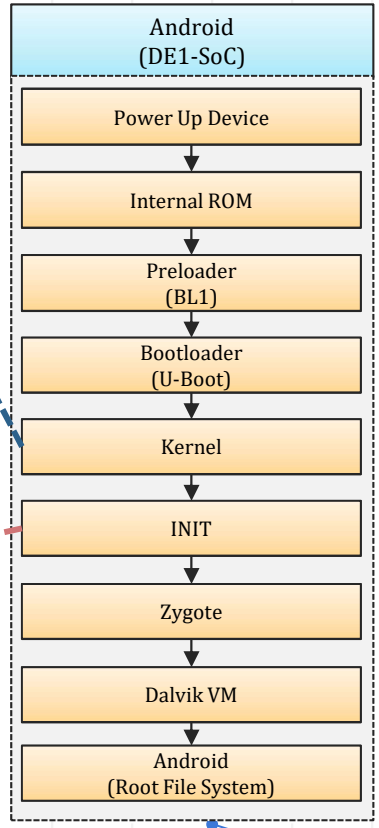


Step 7: Configuring Linux

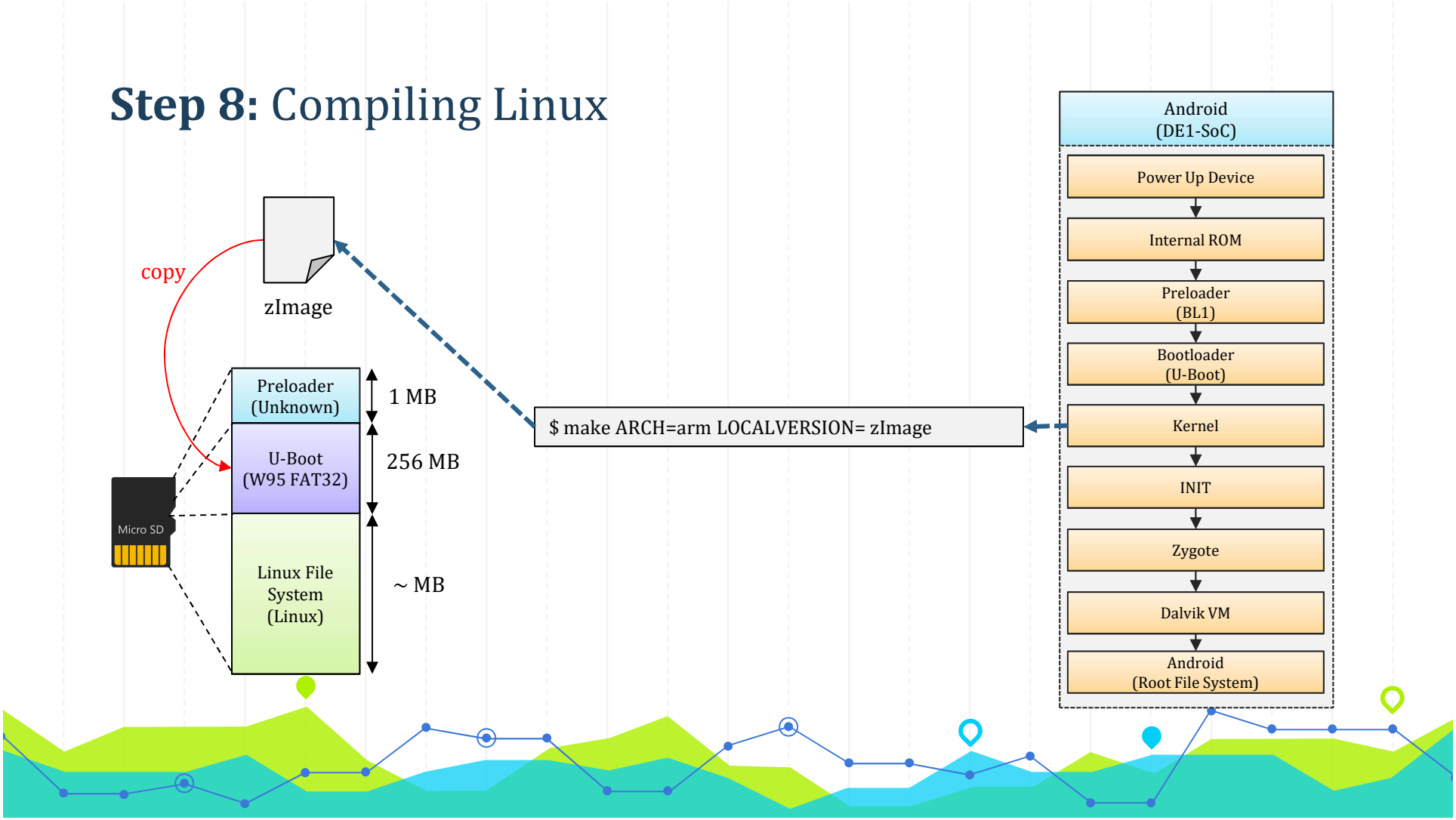
$ git clone https://github.com/altera-opensource/linux-socfpga.git
$ make ARCH=arm socfpga_custom_defconfig
$ make ARCH=arm menuconfig



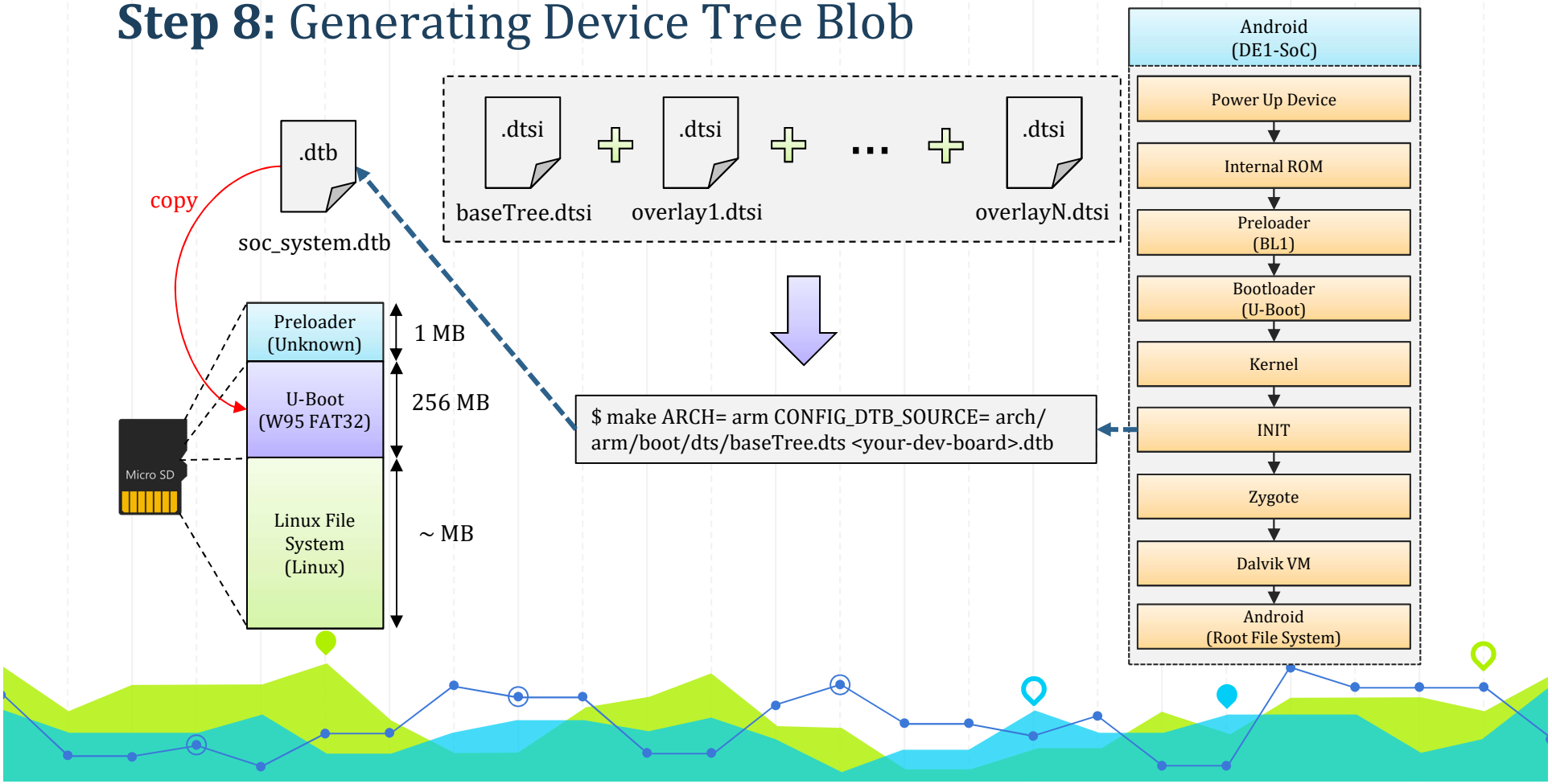
Specify the file system and RAM disk to load



Step 8: Compiling Linux



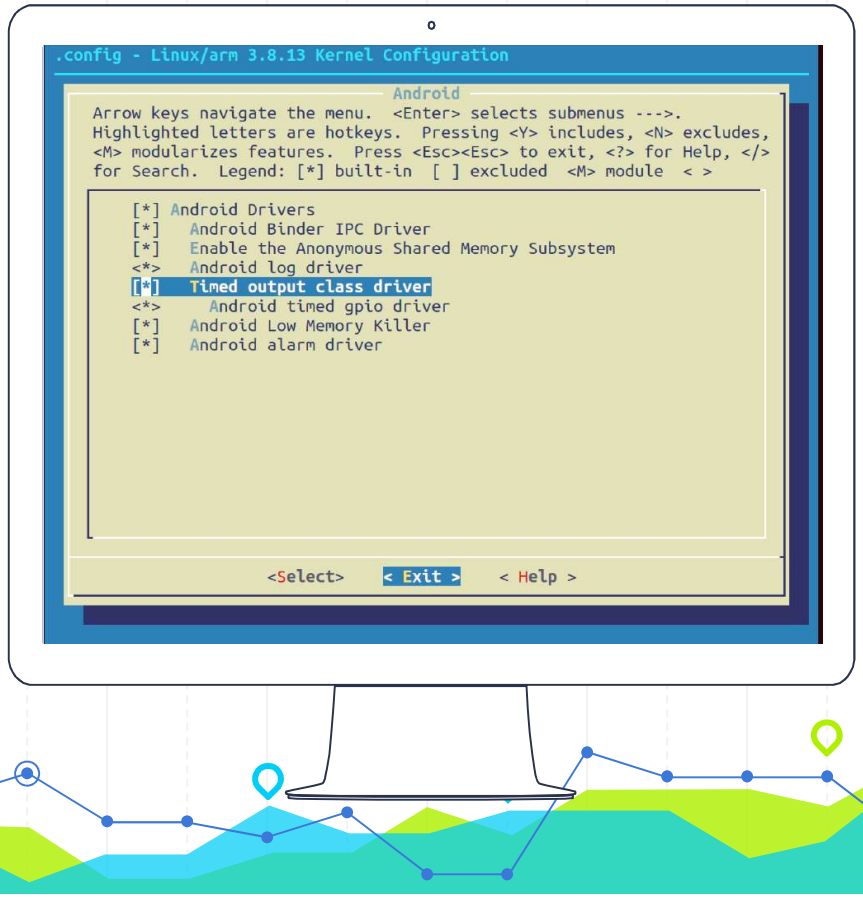
Step 8: Generating Device Tree Blob



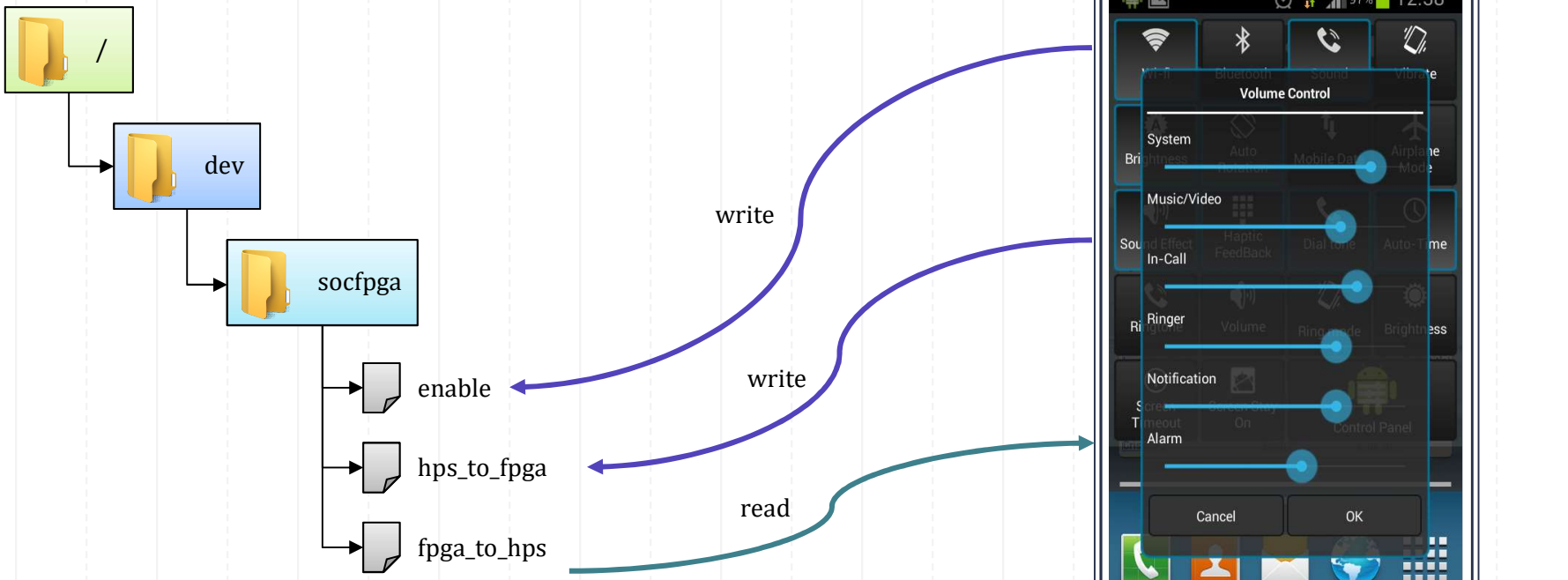
Step 9: Android Bindings

Developing your device drivers is similar to developing a typical Linux device driver. Android uses a version of the Linux kernel with a few special additions such as wake locks (a memory management system that is more aggressive in preserving memory), the Binder IPC driver, and other features important for a mobile embedded platform. These additions are primarily for system functionality and do not affect driver development.

You can use any version of the kernel as long as it supports the required features (such as the binder driver).



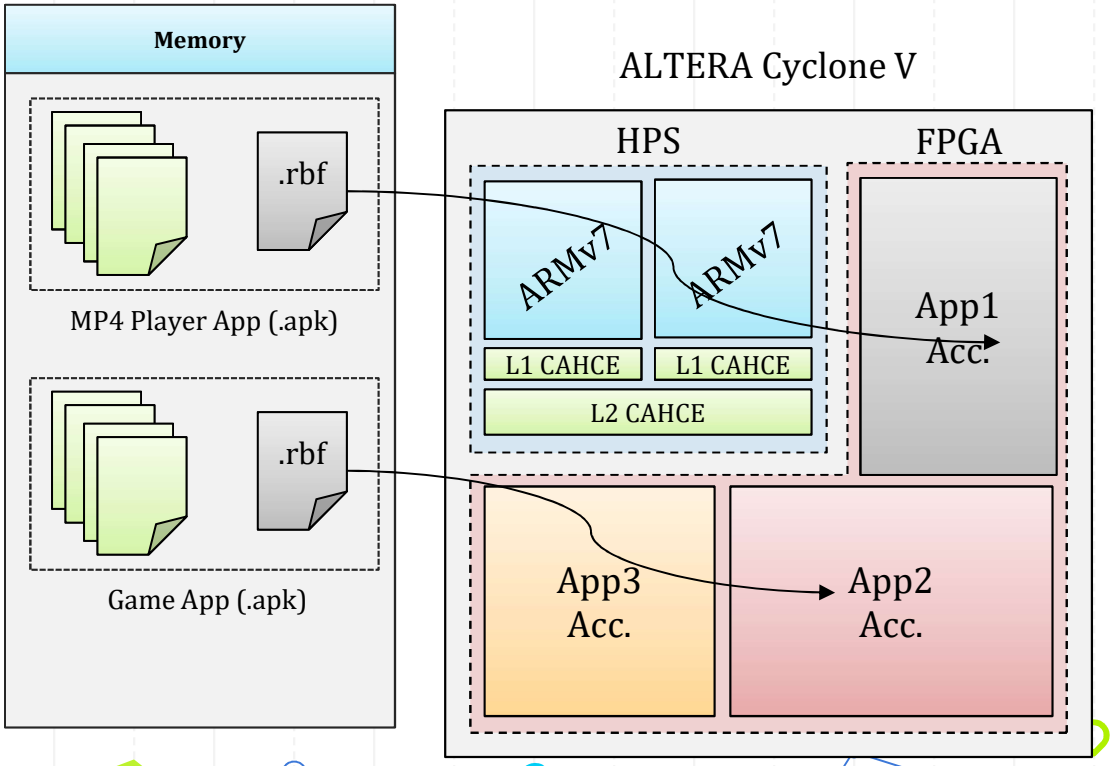
Step 10: Android Application



5 RESULTS

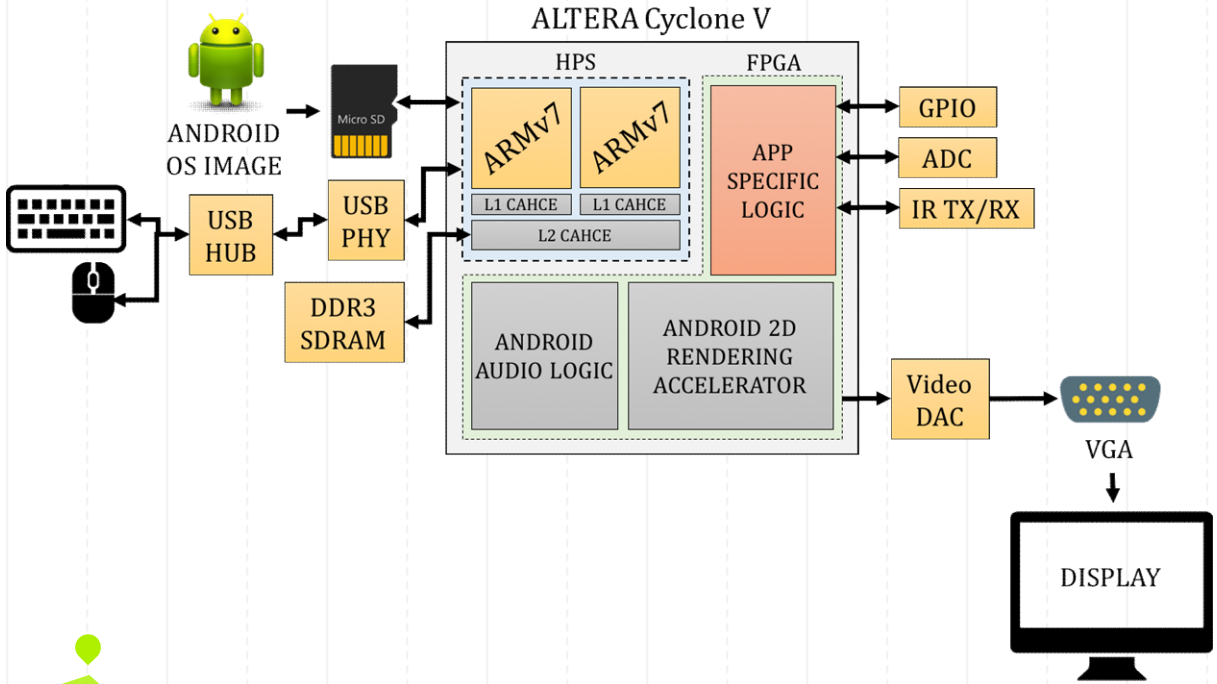
Raw Binary Files (.rbf) in Android Application Package (.apk)

- Load .rbf file on activity create event onto the fpga for acceleration
- Remove .rbf file on activity close event
- Android Init() should configure the fpga with base configuration file and several PRRs (Partial Reconfiguration Regions).



CONCLUSION 6

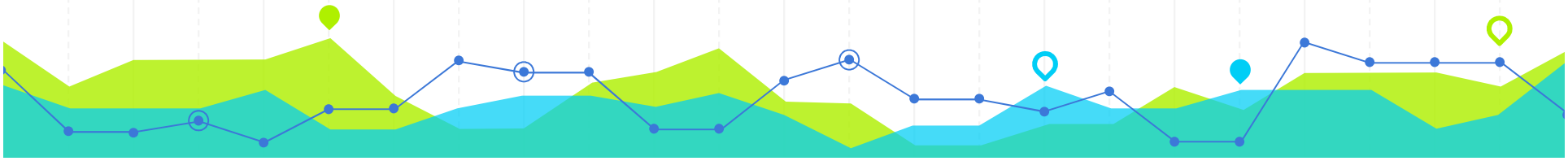
App-based Hardware Acceleration



“

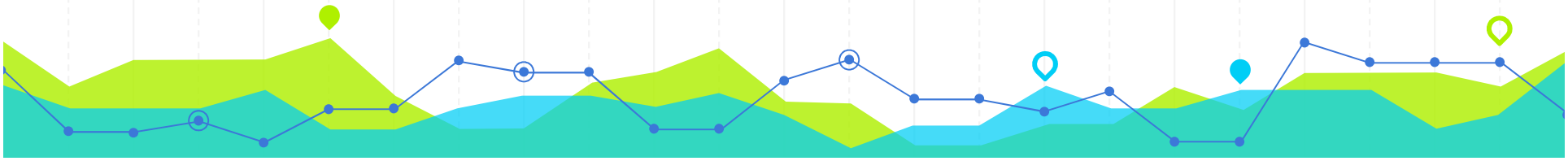
People who are really serious about software should make their own hardware.

~ Alan Kay (Computer Scientist)



References

1. Google Inc., "Android Open Source Project," Google, 23 August 2016. [Online]. Available: https://source.android.com. [Accessed 9 November 2016].
2. M. Daum, "Android for DE1-SoC Board," RocketBoards.org, 6 October 2016. [Online]. Available: https://rocketboards.org/foswiki/view/Projects/AndroidForDE1SoCBoard. [Accessed 9 November 2016].
3. "Graphics Accelerator for Android," FUJISOFT INCORPORATED, 15 November 2013. [Online]. Available: https://www.fsi-embedded.jp/e/_emb/gaforandroid_e/. [Accessed 9 November 2016].
4. Intel, "Implementation of an Android™ Operating System on an Altera SoC," Intel, 8 January 2014. [Online]. Available: https://youtu.be/zHqS_yWiMNI. [Accessed 9 November 2016].
5. Altera, "Cyclone V HPS Memory Map," Altera, 16 November 2016. [Online]. Available: https://www.altera.com/hps/en_us/cyclone-v/hps.html#topic/sfo1418687413697.html. [Accessed 16 November 2016].
6. Yves Vandervennet, "Move to GitHub," RocketBoards.org, 7 August 2015. [Online]. Available: https://rocketboards.org/foswiki/view/Documentation/MoveToGitHub. [Accessed 21 November 2016].
7. Altera, "Workshop #1 – Altera SoC Software Development Overview," RocketBoards.org, 23 April 2016. [Online]. Available: https://rocketboards.org/foswiki/pub/Documentation/WS1IntroToAlteraSoCDevices/WS_1_Intro_To_SoC_SW_Workshop.pdf. [Accessed 23 November 2016].



THANKS!

Any questions?

You can find me at

ENG 402 / mobaidullah@ryerson.ca