Mikrorendszerek Linux (Wacha Gábor-féle kérdések)

December 20, 2017

1 Milyen hardverelemeket kell tartalmaznia egy Linuxot futtató MicroBlaze alapú rendszernek? Mire szükségesek ezek? (Józan ésszel kitalálható, a "miért" a fontos)

- MMU
  - Virtuális memóra kezelés
  - Memória védelem
- Külső RAM
  - BRAM-ban nem férünk el
  - Általában oda a bootloader-t helyezzük el
- Megszíktás vezérlő
  - A legtöbb periféria IT-osan van kezelve Linux alatt (és minden normális rendszer alatt)
- Timer
  - Ütemezéshez
- Soros port
  - Hogy lássuk mi történik a rendszerben
  - Interfészt biztosít a paraméterezéshez

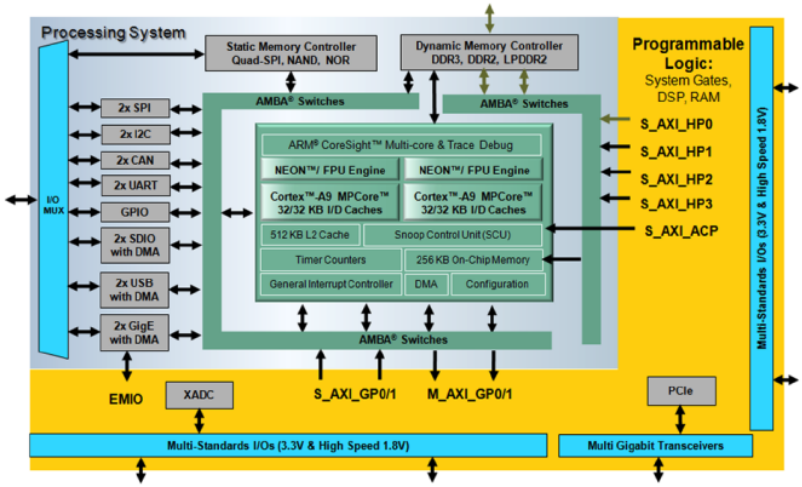
2 Mire szolgál a device tree leírás? Hogyan kapcsolódik össze a device tree és a platform driver?

- AXI, PLB, IIC stb buszokon nincsen eszközdetektálás funkció megvalósítva (ellentétben a PCI buszokkal)
- Így a kernel nem tudhatná milyen eszközeink vannak csatlakoztatva
- A device tree írja le a kernel számára, milyen busz rendszerek vannak a rendszerben, és azokra milyen eszközöket illesztettünk
  - Felsorolja milyen buszra, milyen típusú eszközök csatlakoznak, milyen címen
    - *compatible* - Milyen driver segítségével kezelhető
    - *reg* - Milyen címen érhető el az eszköz a buszon
    - *interrupts* - Milyen IT vonalakra csatlakozik
- Platform driver esetén a is a device tree leírásból olvassuk ki az illesztendő eszköz címeit stb. Valamint ez alapján töltődik be a driverünk (compatible)

3 Operációs rendszereknél mit jelent a hardver absztrakció? Mutass rá példát a Linux kernelen belül (pl. labor alapján)!

- Az operációs rendszer egy egységes interfészt biztosít a hardver konkrét típusától függetlenül. Linux alatt a driverek HW függő részei külön definiálva vannak, így azok már egységes interfészt mutatnak a felső rétegeknek.
- A <linux/asm.h> tartalmazza a HW függő részeket. A többi magasabb szintű függvény már a <linux/asm.h>-ban elkészített alacsony szintű függvények interfészét használja.

4 Xilinx Zynq esetén milyen lehetőségeid vannak a processzoros rendszer és a programozható logika közötti kommunikációra? Melyiket mire alkalmaznád?



- EMIO (Extended Multiplexed IO)
  - IO priférіák csatlakoztatására a PL-hez
- AXI high speed ports (HP0-HP3 Slave)
  - Nagy mennnyiségű adat mozgatására a PS memóriája (külső RAM) és a PL között (nagy mennyiségű adatokon számítást végző speciális periféria PL-ben megvalósítva)
  - Ezekre a portokra illeszthető DMA verézlő (ezen a switch-en vannak a külső DDR vezérlővel, és a On Chip Memóroával (OCM))
- Accelerator Coherence Port (ACP Slave)
  - Processzor cache-ének elérésére. Periféria ami a processzor cache-éből, regsztereiből veszi fel az adatokat. Kissebb mennyiségű adatok feldolgozására
- AXI General purpose (GP0-GP1 Master és Slave)
  - Általános célú perifériák illesztésére
  - Axi és AXI Lite vonalak
    - RD Addres
    - WR Address
    - RD Data

        * WR Data
        * WR response
    - Full AXI
        * Nagyobb erőforrásigényű 256-os burst adat átvitelre képes
            · Állítható adatszélesség (1024 bit)
            · Full duplex
        * AXI-lite
            · Nincs burst
            · 32-64 bit adatszélesség
            · Kicsi erőforrásigény
            · Pont-Pont kapcsolat
            · Full duplex
        * AXI Stream
            · Adatfolyam (FIFO) jellegű összeköttetés
            · Nincs limitálva a burst mérete
            · Master -> Slave adatfolyam (simplex)
            · Nincs címzés

- CLK és RST vonalak (csak a PS szolgáltathat órajelet a PL-nek)
- Interruupt vonalak

5 Mutasd be a Xilinx Zynq bootolási folyamatát bare metal és Linux operációs rendszer esetén! Mire szolgálnak az egyes boot fokozatok? Ki és mikor programozhatja fel a programozható logikát egy kész (JTAG nélküli) rendszerben?

5.1 Bare Metal

- Először a FSBL (Fist Stage Boot Loader) indul el, a processzoros rendszeren.
    - Felkonfigurálja a PL-t
    - Bemásolja az alkalmazásunk a DDR memóriába
    - Elindítja a bare-metal alkalmazásunkat
        * Ez inicializálja a HW perifériákat (regiszterek beállítása, működési mód meghatározása stb.)

5.2 Linux

- FSBL lefut, mint az előbb, de most a boot loaderünket hívja meg
- Valamilyen bootloader (pl.: UBOOT) beolvassa a Linux kernelt (SD kártyáról, hálózatról, flashből, stb)
- Linux kernel elindul, és inicializálja a HW-t
- Linux kernel átadja a vezérlést az init-nek (ez más user space-ben fut)
- init inicializálja a user space-t
- init elindítja az általunk megadott alkalamzást

Az FSBL felelős a PL felprogramozásáért (tehát a processzor programozza fel), indításkor. Az első program ami lefut.

6 Mi az a TCL? Mutasd be felhasználási lehetőségeit a Xilinx Vivado környezetben!

Tool Command Language

- Egy interpreteres program nyelv, mely segítségével automatizálhatjuk a Xilinx Vivado környezet működését.
- Példák:
  - Szintézis indítható automatikusan
  - Fájlok írása, olvasása
  - Dokumentáció generálás indítható
- Egyszóval teljes körű automatizálást biztosít

7 Mire szolgál az IPXACT leírás? Az EDK melyik funkcióit váltja ki?

Egy szabványos IP formátum. Vivado alatt ezzel tudunk exportálni általunk készített HW-t.

EDK által biztosított IPIF interfészét váltja fel.

8 (Linux) operációs rendszer használatakor mi nehezíti meg a kommunikációt egy felhasználói alkalmazás és egy hardveres gyorsító között? Hogyan küszöbölhető ki? (Többféle értelmes válasz létezhet)

A modern operációs rendszerek nem engedik a HW-t elérni a user space programoknak, erre szolgálnak a kernel driver-ek, hogy egységes interfészt biztosítsanak.

A virtuális címzés miatt virtuális címen folytonosan elhelyezkedő adatok a fizikai memóriában töredezetten helyezkedhetnek el. Az is elképzelhető, hogy más-más HW-en (egyik fele DDR RAM a másik része ki swappelődött a HDD-re stb.)

A scatter-gather DMA megoldást nyújthat, ez képes a fizikai memóriában töredezetten lévő adatokat megfelelően írni, illetve olvasni.

Tudsz mér értlemes választ?

9 Hogy néz ki egy HLS gyorsítót tartalmazó processzoros rendszer fejlesztése SDSoC nélkül? Mi automatizálható ebből (= mit csinál meg helyettünk az SDSoC)?

Alapvető kiindulás

- Megvizsgáljuk a SW-t melyik részét érdemes, és lehetséges HW-ben megvalósítani
- Megvalósítjuk
- Teszteljük
- Ha az eredmény még továbbra is lassú, repeat

9.1 SDSoC nélkül

- Elkészítjük a processzoros rendszert, majd felélesztjük.
- Analizáljuk azokat a részeket, melyeket érdemes HW-ben megvalósítani.
- A processzoros rendszer szoftverőből eltávolítjuk a HW-be átültetendő részt
- Megvalósítjuk HW-ben
- Kialakítjuk a megfelelő interface-eket
  - Megfelelő buszra illesztjük a HW-es gyorsítónkat
  - Elkészítjük a drivert
- Módosítjuk a SW-t az új HW-nek megfelelően

9.2 SDSoC

- Elkészítjük a processzoros rendszert, majd felélesztjük.
- Analizáljuk azokat a részeket, melyeket érdemes HW-ben megvalósítani.
- A többbi elvileg automatikusan működik

Megjegyzés: Az ilyen generált kódok, midig elmaradnak a kézzel készített kódoktól, mind sebesség, mind erőforrás tekintetében. Természetesen csak megfelelő programozó mellett.