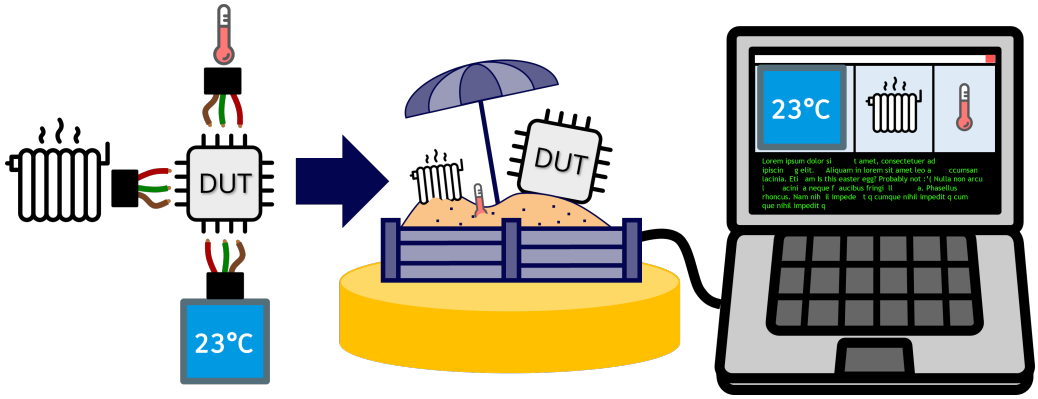
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2022
http://excel.fit.vutbr.cz

Emulace periférií vestavěných systémů pro rychlé prototypování

Bc. Dominik Müller



Abstrakt

Tato práce se zabývá návrhem a realizací platformy, umožňující vývojáři definovat simulované prostředí vestavěného systému. Platforma umožňuje vývojáři vybrat nebo vytvořit zařízení, vyskytující se v okolí vyvíjeného systému a detailně definovat jejich parametry a chování. Simulace okolí mikrokontroléru je rozdělena na harwarovou úroveň a na úroveň deterministického real-time prostředí. Odpadají tak některé limitace softwarových řešení. Navrhovaná platforma představuje pohled na testování vestavěných systémů během ranných fazí vývoje. Tento pohled je odlišný od běžných řešení a snaží se tak zvýšit efektivitu vývojáře. Návrh platformy cílí na její snadnou rozšířitelnost, konfigurovatelnost a univerzalitu. Výstupem této práce je platforma demonstrující myšlenky uvedené v tomto článku, která slouží jako vhodný základ pro budoucí rozšíření.

Klíčová slova: Hardware-in-the-loop — Simulace vestavěných systémů — Akcelerace vývoje

Přiložené materiály: *N/A*

* xmulle26@fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Vestavěné systémy se vyskytují všude kolem nás a staly se nezbytnou a užitečnou součástí našich životů. Tyto systémy představují jednoúčelová zařízení komunikující s vnějším prostředím s použitím elektronických analogových a digitálních signálů. Jako příklady lze uvést např. řídící systém robota, různé druhy směrovačů nebo elektronický termostat. Jejich komplexnost však může komplikovat vývoj a testování firmware. Například při paralelním vývoji několika částí zároveň potřebujeme nahradit chybějící části funkčně podobným či ekvivalentním protikusem. Takovéto problémy typicky řešíme simulací některých částí. Tento přístup se také používá pro testování zařízení, která by při testování svým nesprávným chováním mohla způsobit újmu na zdraví, financích nebo poškodit životní prostředí (např. hlásič požáru, naváděcí systém rakety, ...). V rámci vývoje využíváme simulaci také pro navození zřídka vyskytovaných jevů nebo urychlujeme dlouho trvající děje (např. porucha, ohřívání, ...).

Nástroje pro simulaci existují a jsou běžně využívány. Čistě softwarová simulace však vyžaduje

velkou míru aproximace a tím vznikají nepřesnosti. Na druhou stranu běžně využívané testovací systémy HIL (hardware-in-the-loop), které simulují pouze okolí testovaného zařízení, jsou komplexní a finančně náročné. Proto jsem se rozhodl navrhnout vlastní platformu, která bude umožňovat snadnou a rychlou simulaci vnějšího prostředí vestavěného systému za přijatelnou cenu. Cílem není nahradit dostupné nástroje, ale představit odlišný pohled a společně s existujícími nástroji vylepšit vývojový proces. Cílem navrhované platformy je vytvořit pro uživatele snadno konfigurovatelné a rozšiřitelné okolí procesoru, se kterým může interagovat. A to tak, aby mohl ověřit funkčnost či realizovatelnost své implementace na fyzické úrovni přímo s vybraným procesorem a bez zásahu do způsobu vývoje na tomto procesoru. Dalším cílem platformy je, aby ji mohl mít každý vývojář u sebe na stole a mohl ji snadno a rychle začlenit do svého pracovního procesu. Výsledné zařízení je kompaktní a zároveň se snaží integrovat nejčastěji využívané laboratorní přístoje při vývoji vestavěných systémů. Jmenovitě: zdroj napájení, ampérmetr a logický analyzátor.

Navržená platforma se dá využít ve všech fázích vývoje vestavěného zařízení. Na počátku vývoje slouží platforma jako pomocník při evaluaci - spouštění benchmarků, průzkumu práce s periferiemi atd. Po specifikaci požadavků platforma umožňuje simulaci zařízení v okolí mikrokontroléru a umožňuje tak vývojáři pohodlný vývoj a testovaní do doby, než má k dispozici první prototyp. Po obdržení fyzického prototypu platforma vývojáři slouží hlavně pro testování hraničních situací, ladění chyb a jako základ pro automatické testování HIL. Díky tomu můžeme i ve vestavěném zařízení a jeho fyzickém rozhraní využívat koncepty známé z klasického softwarového vývoje, jako jsou automatické unit-testy a podobně. Tyto testy mohou být automatizovány pro všechny další případné revize vyvíjeného systému.

Pro evaluaci testovaného zařízení, typicky mikrokontroléru, využíváme dostupné vývojové a evaluační kity. Platforma se snaží představit co nejobecnější rozhraní pro testovaný systém, volí tak právě takový kit jako své rozhraní a řídí přímo vstupy a výstupy mikrokontroléru. Vývojář implementuje chování vyvíjeného systému ve své režii a pro jeho otestování připojí evaluační kit k platformě. Následně vybere či implementuje zařízení k simulaci (teploměr, paměť, akcelerometr, ...) a jejich funkcionality přiřadí na pin evaluovaného mikrokontroléru. Pomocí API pak může generovat testovací stimuly, měnit konfiguraci připojených zařízení a integrovaným logickým analyzátorem sledovat chování na fyzické vrstvě.

Navržená platforma je připravena na všechny výše zmíněné fáze vývoje. Při návrhu jsem se ale soustředil zejména na prvotní simulaci zařízení, kdy ještě nejsou k dispozici všechny vnější periférie. Platforma se zaměřuje na malé až střední vestavěné systémy, s rychlostmi signálů do 10 MHz. Pro analogové signály jsem se rozhodl tolerovat menší nepřesnosti v signálu za cenu levnější vývojové platformy. Implementovaná platforma je postavená na vývojové desce Pynq, založené okolo 2 jádrového procesorem ARM, který je přímo propojený s FPGA. Uživatel má k dispozici až 32 pinů s libovolnou digitální funkcionalitou, kde 8 z nich poskytuje i analogové funkcionality (ADC, DAC). Digitální rozhraní (I2C, SPI, UART, ...) jsou implementována v FPGA a chování zařízení na nich postavených je popsáno jako jedna z úloh real-time operačního systému FreeRTOS, který běží na jednom jádru procesoru. Na druhém jádru pak běží operační systém Linux v režimu master, obsluhující vše mimo simulaci. Součástí digitálních rozhraní je i periférie PIO (Programmable I/O), umožňující vytvoření vlastního digitálního rozhraní pomocí skriptu, která přidává rozšířitelnost uživatelem i na ”hardware” úrovni. Mimo I/O rozhraní platforma umožňuje simulovat i libovolný zdroj napájení. Ten je užitečný při vývoji vestavěných systémů napájených z baterie.

2. Přístupy k simulaci

Simulaci můžeme s větší mírou aproximace docílit čistě softwarově nebo přesněji za pomoci hardware, kdy softwarově aproximujeme pouze chybějící části. Oba přístupy mají své výhody a nevýhody, při testování tak volíme jejich vhodné rozložení.

2.1 Software-in-the-loop

Vynecháním jakéhokoliv hardware a přesunutí simulace čistě do softwarového prostředí umožníme spouštění testů kdekoliv a zvýšíme rychlost jejich exekuce. Tento přístup je tak velmi vhodný pro testování bussiness logiky systému. Pokud však chceme testovat kód závislý na procesoru, musíme najít vhodný emulátor např. QEMU, Renode. Žádný emulátor ale nepodporuje všechny architektury procesorů a minimum z nich implementuje i periférie uvnitř mikrokontroléru. Správné nastavení takového prostředí tak není triviální úkon a může být těžko přenositelné.

2.2 Hardware-in-the-loop

Abychom otestovali reálné chování systému, spouštíme kód přímo na vyvíjeném systému a

simulujeme pouze jeho okolí. Pro tento přístup lze jít do různé úrovně detailu a komplexity simulovaného systému. Obvykle se však setkáme s velkými a komplexními systémy specializovanými pro vyvíjený systém, jelikož nám musí zaručit, že se daný systém chová dle specifikace. Výsledný systém je tak těžko přenositelný a přidaná komplexita se promítne do ceny takového systému.

2.3 Podobná řešení návrhu

Nejblíže navrhovanou platformu připomíná open-source projekt Glasgow [1]. Jedná se o flexibilní nástroj pro průzkum digitálních rozhraní, který umožňuje uživateli vybrat z předpřipravených rozhraní a případně je i rozšířit o vlastní. Nad vybranými rozhraními uživatel implementuje své případy užití. Platforma se skládá z podpůrné desky s FPGA a pomocnými periferiemi a konzolovou aplikací pro její obsluhu. Digitální rozhraní jsou implementovány v FPGA a komunikají s PC pomocí USB FIFO rozhraní. Platforma se zaměřuje hlavně na digitální rozhraní, ale umožňuje také zpracování analogových signálů pomocí přídavných A/D a D/A převodníků. Celý projekt je postavený na open-source nástrojích, které jsou velmi dobře optimalizované. Glasgow si tak může dovolit sestavovat výsledný bitstream on-demand na straně uživatele. Platforma však pro snadné použití předpokládá, že využijete předpřipravené případy užití. Pro implementaci vlastního případu užití uživatel musí implementovat chování v jazyce Python a také definovat komunikaci s rozhraním v FPGA. Pro implementaci vlastního rozhraní musí mít uživatel znalosti návrhu číslicových obvodů a znalost knihovny Amaranth (knihovna pro popis RTL obvodů v jazyce Python). Hlavní myšlenka projektu je vytvořit nástroj pro snadné ladění rozhraní, nezabývá se přímo podporou vývoje.

V komerčním prostoru existuje například řešení od společnosti hitex: miniHIL [2]. Jedná se o platformu s výkoným mikrokontrolérem, který simuluje okolí testovaného zařízení. Uživatel může simulovat běžně používáná digitální a analogová rozhraní a zařízení je využívající. Ovládání platformy je integrováno do vývojářské platformy TrueStudio IDE nebo pomocí konzolové či grafické aplikace. Motivací platformy je mít kompaktní HIL systém pro automatické testování. Celá platforma je však uzavřená a vlastní rozšíření musí uživatel poptat u výrobce.

3. Představení platformy

Vzhledem k výše uvedeným omezením je mým cílem vytvořit platformu, která by byla kompaktní a cenově odpovídala ostatním laboratorním zařízením na trhu, tedy okolo 300 $. Tato platforma by měla mít FPGA pro implementaci konfigurovatelných rozhraní a vymezit tak výkon procesoru čistě pro simulaci a řízení. Dále by měla disponovat real-time operačním systémem (RTOS), který by umožnil deterministickou a opakovatelnou simulaci zařízení s nízkou latencí reakce. Platformu by měl být schopný rozšířit běžný vývojář vestavených systému.

Obrázek 1. Jednotlivé vrstvy simulace.

3.1 Rozšířitelnost

Platforma je rozšířitelná na třech úrovních. Na nejvyšší úrovni máme simulovaná zařízení (např. teploměr, paměť), která vývojář implementuje na úrovni real-time operačního systému. Zařízení má vyhrazenou úlohu operačního systému a vývojář může ovlivnit prioritu jejího vykonání. Zároveň využívá synchronizačních primitiv operačního systému pro komunikaci mezi zařízeními.

Při implementaci zařízení si vývojář vybírá, jaké periférie jeho zařízení používá. Pro analogvé signály vybírá mezi D/A převodníkem a A/D převodníkem. Pro digitální signály si volí z předpřipravených periférií nebo implementuje vlastní pomocí skriptu pro periférii PIO. PIO je jednoduchý stavový automat s přechodovými pravidly (instrukční sadou). V rámci mého návrhu jsou tato pravidla inspirována těmi, které jsou použité v mikrokontroléru RP2040 od společnosti Raspberry Pi Foundation [3]. Ačkoliv se jedná pouze o 8 instrukcí, stavový automat je výpočetně úplný a lze pomocí něho implementovat periférie jako (UART, SPI, I2C a další).

Ukázka 1. Jednoduchá ukázka skriptu pro přijmací část UART periférie. Na počátku skriptu je konfigurace periférie, nastavíme automatické vložení dat to výstupní FIFO fronty po 8 bitech a odsazení prvního čteného pinu. Běh stavového automatu je následující: Počkáme na stop bit na pinu 0. Počkáme 10 hod. cyklů a nastavíme registr X na 7. Přečteme jeden bit. Počkáme 6 hodinových cyklů, pokud je $X > 0$ dekrementujeme registr X a skočíme na návěští readloop. V opačném případě pokračujeme další instrukcí, kterou skočíme na začátek programu (přijali jsme jedno slovo).

```
.config:
    autopush 8
    read_pin_offset 1

.program:
    uart_rx:
        wait PINS, False, 0
        set[10] X, 7,
    readloop:
        in PINS, 1
        jmp[6] readloop, X--
        jmp uart_rx
```

Implementace vlastních periférií popisem číslicových obvodů je možná, ale z důvodu časové náročnosti nepředpokladám její využití koncovým uživatelem. Vlastní implementaci periférií ale můžeme využít v případě, že chceme vytvářet vlastní složitější komponentu (např. RMII rozhraní Ethernetu) nebo chceme využít existující IP core. Periférie přímo interagují s testovaným mikrokontrolérem. Pokud by vývojář potřeboval periférie mimo digitální sféru, může si vytvořit fyzický převodník. Převodník je na jedné straně připojený k platformě přes jednu z dostupných digitálních periférií a na druhé pak k testovanému systému. Můžeme tak snadno přidat bezdrátová rozhraní (WiFi, Bluetooth, ...) nebo rozhraní se speciální fyzickou vrstvou (Ethernet, RS485, ...).

3.2 Konfigurovatelnost

Během testování potřebujeme simulované prostředí měnit a vytvořit tak co možná největší testovací scénář. Například při testování elektronického termostatu, chceme simulovat různé komunikační rozhraní s topným systémem, různé teplotní křivky a kombinace uživatelských vstupů. Platforma tak přes API umožňuje konfigurovat jak simulované prostředí (teplota, čas, ...), tak i použité periférie, pomocí kterých testovaný mikrokontrolér tyto informace získává, a také jejich parametry (baudrate, šířka slova, ...).

3.3 Znovupoužitelnost

Definováním pinu mikrokontroléru jako rozhraní platformy, zavedeme co možná nejobecnější rozhraní, na kterém stavíme všechny další elementy testovaného systému. Dostaneme se tak rychle k prvnímu funkčnímu konceptu a rychle se adaptujeme na změny které nás potkají při vývoji. Navíc jsme schopni inkrementálními úpravami pokrýt testování širokého spektra vestavěných systémů.

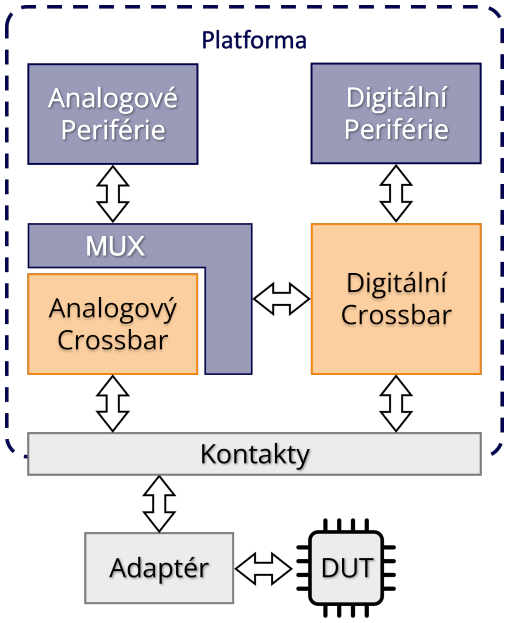
Piny mikrokontroléru mohou nabývat více funkcionalit (GPIO, SPI, ADC, ...). K pohodlnému vývoji, bez nutnosti fyzicky měnit zapojení při každé změně rozložení pinů, podporuje platforma mapování jakékoliv digitální funkcionality na jakýkoliv pin. Analogové funkcionality lze mapovat na omezený výběr pinů, obdobně jak tomu je u mikrokontrolérů zvykem.

3.4 Univerzalita

Vývojář vestavěných systémů využívá při vývoji mnoho nástrojů (např. nástroje pro zprovoznění systému, hledání chyb a validaci jeho chování). Platforma se tak snaží identifikovat ty nejčastější a nejužitečnější nástroje, které vývojář využívá. Tyto nástroje pak integruje do jednoho celku, tak aby vývojáři nabídla univerzální platformu pro vývoj.

4. Architektura

4.1 Propojení s testovacím systémem



Obrázek 2. Diagram propojení platformy s testovaným systémem. Ukazuje způsob jakým je docíleno mapování periférie na jakýkoliv pin.

Jak bylo zmíněno v sekci 3.3, platforma umožňuje připojení periférie na jakýkoliv pin. Tato funkce je

docílena pomocí řad přepínačů v maticovém zapojení tzv. crossbar. Pro digitální periférie lze implementovat crossbar uvnitř FPGA, avšak pro analogové musíme dodat dedikovaný obvod. Abychom docílili jak analogové tak i digitální funkcionality na jednom pinu, zapojíme analogový crossbar za ten digitální. A to takovým způsobem, že před analogový crossbar umístíme řadu fyzických přepínačů, které volí, jestli je ke vstupu crossbaru připojená digitální nebo analogová periferie. Řada přepínačů navíc je zvolena z úsporných opatření. Jelikož je úspornější zapojit $N \times N$ crossbar a N přepínačů než zapojení crossbaru $N \times 2N$. Z hlediska funkcionality nás toto omezí pouze tak, že nelze zaráz přímo číst hodnotu pinu digitálně a analogově (nepřímo stále lze), což je přijatelné. Funkci zápisu to neovlivní vůbec - blokovanou digitální periferii přesuneme digitálním crossbarem na jiný pin. Výstup analogového crossbaru je zapojen na vybrané kontakty a zbytek kontaktů nabízí jen digitální periférie.

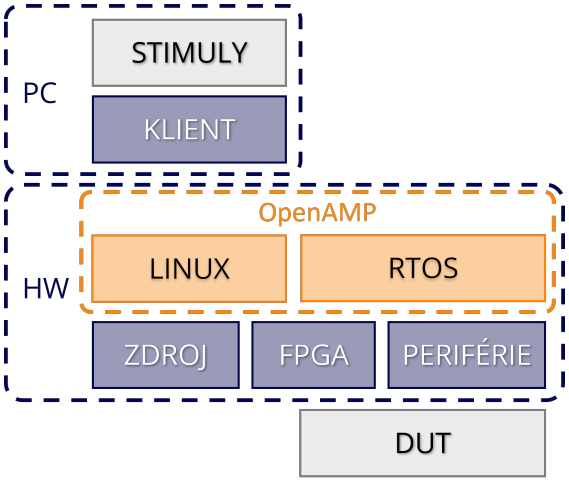
Jelikož je analogový crossbar zapojený jako matice přepínačů, umožňuje nám jakékoliv propojení vstupů a výstupů. Z hlediska simulace toho můžeme využít tak, že například výstup jednoho D/A převodníku napojíme na více výstupů. Zároveň je možné zapojit výstup integrovaného D/A převodník na vstup integrovaného A/D převodníku a otestovat tak jejich funkčnost. Je však potřeba speciální validátor, který zajistí, aby uživatelem zvolené zapojení nezpůsobilo poškození platformy.

Ačkoliv existují standardizované formáty evaluačních kitů (např. formáty kitů od společnosti Arduino či Raspberry), nelze zaručit fyzickou kompatibilitu s každým kitem. Je tak ponecháno na uživateli, aby vytvořil jednoduchý adaptér, který pouze navede piny do kompatibilního rozpoložení.

4.2 Systém

Celá platforma se skládá ze 2 částí. Klientské aplikace, běžící na počítači uživatele, a fyzického zařízení připojeného k aplikaci. Klientská aplikace vytváří API pro testovací stimuly a zároveň obsahuje konzolovou aplikaci, která umožňuje s daným API přímo interagovat. Pomocí API je uživatel schopen dotazovat se na stav různých částí platformy a měnit jejich stav. Dále je také schopen aktualizovat běžící systém, real-time operační systém a také FPGA.

Fyzické zařízení je rozděleno na další části. Celý systém je řízen z operačního systému Linux, na kterém je spuštěna obslužná rutina. Ta se stará o obsluhu API a řízení platformy. Sleduje stav připojených USB zařízení a komunikuje s RTOS. RTOS pak slouží jako prostředí pro implementaci simulace. Obsahuje tak



Obrázek 3. Ukázka z jakých vrstev se navrhovaná platforma skládá.

chování jednotlivých zařízení, implementovány jako jedna z úloh systému, ovladače pro periférie a případně i řízení průběhu simulace.

Pro rozdělení systému Linux a RTOS je použitý framework OpenAMP [4]. OpenAMP je framework umožňující asymetrický multi-processing, tedy souběžnou činnost více systémů nezávisle na sobě. Toto rozložení jsem zvolil, protože operační systém Linux poskytuje potřebné technologické zázemí pro rychlý vývoj, snadnou práci se sítí a USB zařízeními. RTOS pak nabízí prostředí známé vývojáři vestavěných systémů s determenistickým a předvídatelným chováním. Oddělení dvou systémů je dále vhodné kvůli stabilitě platformy. Operační systém Linux je v tomto rozdělení v pozici master, na počátku tak má k dispozici obě jádra procesoru a pouze když má dostupný kód pro RTOS, tak přesune jeho kód do vyhrazené paměti, uvolní jedno z jader a spustí na něm RTOS. Operační systém Linux tak může simulaci monitorovat a v případě chybného chování simulaci zastavit.

4.3 FPGA

Na nejnižší konfigurovatelné vrstvě platformy, FPGA, jsou implementovány všechny digitální periférie, logický analyzátor a digitální crossbar.

Digitální crossbar přímo interaguje a směruje všechny příchozí/odchozí signály do/z FPGA. Uživateli umožňuje zvolit, jaký kanál periférie má přístup k zápisu na zvolený pin a také z jakého pinu zvolený kanál periférie čte. Můžeme tak přiřadit jednomu pinu jeden zapisovací kanál a N čtecích kanálů. Díky těmto vlastnostem tak lze zpracovávat jeden signál více perifériemi nebo interně propojit periférie mezi sebou (např. vytvořit loopback pro UART periférii). Důležitým bezpečnostním prvkem

je také ochrana proti zkratu. Uživatel na počátku definuje rozložení zařízení. Z tohoto rozložení jsou vyextrahována data o vstupech a výstupech. Tato informace je vložena do crossbaru, který zaručí, aby pin označený jako vstup, nebyl nikdy aktivován jako výstup (např. chybnou implementací periférie).

Logický analyzátor je připojen přímo na výstup digitálního crossbaru a umožňuje tak vývojáři analyzovat komunikaci mezi platformou a testovaným systémem. Uživatel může nastavit jednoduchou aktivační podmínku (např. rostoucí hrana pinu 1) a délku záznamu. Po aktivaci podmínky je nahraný záznam exportován jako soubor ve formátu vcd (value change dump). Tento soubor pak může být otevřen v nějakém externím nástroji (GtkWave, PulseView).

4.4 Periférie

Dostupné periférie nabízené navrhovanou platformou můžeme dělit na interní (uvnitř FPGA) a na externí (mimo FPGA). Typicky interní periférie implementují nějaké digitální rozhraní a externí periférie nějaké analogové rozhraní.

Obrázek 4. Rozhraní interní periférie

Interní periférie mají zavedené rozhraní (znázorněno na obrázku 4) pro připojení k ostaním částem systému. Jednotlivé periférie jsou tak dobře modularizovatelné a mohou být snadno vyměněny kus za kus. Vstupem periférie je AXI4-lite rozhraní, přes které processor konfiguruje danou periférii a přesouvá data. Výstupem periférie je interrupt a piny, které se připojují k digitálnímu crossbaru. Piny periférie jsou třístavové - sama periférie si volí, jestli daný pin je vstup nebo výstup (toto je důležité například pro rozhraní I2C, kde linka SDA mění směr signálu). Interrupt notifikuje processor o nějaké akci uvnitř periférie (např. prázdná fronta, chyba, ...). Rozhraní také definuje AXI-Stream rozhraní a to jak pro vstup tak i výstup. Tato rozhraní jsou zamýšlena pro připojení k DMA kontroléru, pro přenos dat do paměti bez zásahu procesoru, ale v rámci této práce nejsou využita a jsou ponechána jako možné rozšíření.

Externí periférie pro svoji funkci využívají některé z digitálních periférií. Jejich použití je však od uživatele abstrahováno pomocí ovladačů.

4.5 Komunikace uvnitř platformy

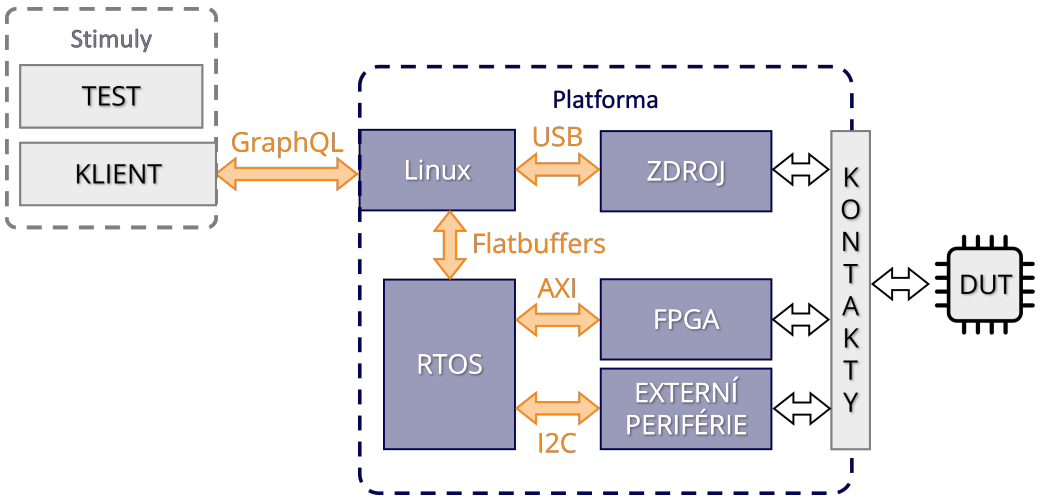
Napříč platformou je použito mnoho rozhraní. Pro uživatele je nejdůležitější rozhraní GraphQL mezi klientskou aplikací a fyzickým zařízením, s kterým přijde do styku přímo. GraphQL je dotazovací jazyk pro vytváření API a prostředí pro plnění těchto dotazů. GraphQL poskytuje úplný a srozumitelný popis dat a dává uživateli možnost dotazovat přesně ty data, která potřebuje. [5] Toto rozhraní jsem zvolil, jelikož snižuje potřebnou implementaci na straně fyzického zařízení a tím se urychlil jeho vývoj. Zároveň umožňuje dynamické rozšíření protokolu, což je potřebné pro uživatelsky přidané periférie. Pro komunikaci mezi operačními systémy se využívá knihovna rpmsg (poskytovaná frameworkem OpenAMP) a nad ní je přidaná knihovna flatbuffers [6]. Knihovna flatbuffers je serializační knihovna a byla zvolena pro rychlou deserializaci bez nutnosti kopírování. RTOS a FPGA spolu komunikují pomocí rozhraní AXI, předávají si pomocí něho data a konfiguraci. S externími periférie komunikuje RTOS nepřímo přes vestavěnou digitální periférii I2C.

5. Realizace

Navržená platforma byla realizována na vývojové desce Pynq. založené okolo SoC Zynq7020 s 2 jádrovým procesorem ARM, který je přímo propojený s FPGA. K vývojové desce pak byly připojeny externí zařízení, tak aby platforma byla schopna splnit stanovené cíle. Pro napájení a měření spotřeby testovaného systému byla zvolena jednotka Power Profiler Kit II (PPK2) [7] od firmy Nordic Semiconductors. Převod analogových a digitálních signálů je realizován pomocí rozšířených hotových modulů založených na MCP4725 (DAC) a ADS1115 (ADC). Převodníky jsou připojeny k analogovému crossbaru ADG2188 a všechna tato zařízení komunikují s vývojovou deskou pomocí I2C rozhraní. Všechny vstupně-výstupní signály platformy jsou sdruženy do předpřipraveného adaptéru, který umožňuje připojení vývojových kitů kompatibilních s jedním z rozložení: Arduino Nano, Arduino Uno, Raspberry Pi Pico nebo ESP32 Wemos.

Jako hlavní operační systém byla zvolena linuxová distribuce Petalinux a jako vedlejší real-time operační systém, byl zvolen FreeRTOS. Hlavní řídící rutina obsluhuje GraphQL server dostupný přes ethernetový port vývojové desky. Součástí hlavní rutiny je periodicky volaná úloha, která ověřuje validní stav platformy a kontroluje připojená USB zařízení. Mezi podporovaná USB zařízení patří pouze napájecí zdroj PPK2, pro který byl napsán speciální ovladač.

Klientská aplikace je implementována jako



Obrázek 5. Diagram ukazující komunikační rozhraní mezi jednotlivými bloky systému.

jednoduchá Python3 konzolová aplikace, se kterou může uživatel přímo interagovat. Dostupné je ale i API pro použití při automatickém testování. V interaktivním módu má uživatel přístupný objekt *com*, přes který vytváří dotazy a mění stav platformy. Pro vytvoření dotazu uživatel využije metodu *query*, do které předá string obsahující validní dotaz v jazyce GraphQL.

Ukázka 2. Dotaz na stav pinu 0. Pin může být ovládán různými druhy kanálů, explicitně tak specifikujeme jaké informace pro daný typ kanálu chceme získat. Můžeme také omezit výběr pinů pomocí parametru *id*, jeho syntax odpovídá té použité pro práci se seznamem v jazyce Python3.

```
>>> com.query(
    '''pin(id: "0:3:2") {
            id,
            channel {
                ... on gpioChannel {
                    name, level
                }
            },
            direction
    }''')

Response: {
    pin: [{
        id: 0,
        channel: {
            name: "gpio0",
            level: true
        },
        direction: "out"
    },
    {
        id: 2,
        channel: {
            name: "gpio2",
            level: false
        },
        direction: "in"
    }]
}
```

Pro změnu platformy uživatel použije metodu *mutate*. Změna platformy se nemusí vždy povést (např. zadání nesmyslné hodnoty, neexistujícího atributu, ...). Uživatel tak musí zkontrolovat stav návratové hodnoty pole *ok* a pro více informací o chybě specifikovat i pole *info*.

Ukázka 3. Příkaz ke změně teploty simulovaného zařízení *heater0*. V části příkazu, kde definujeme návratové hodnoty uvedeme pole *ok* pro validaci vykonání operace a pole *info* pro více informací o případné chybě.

```
>>> com.mutate(
    '''setAttribute(
        device: "heater0",
        attr: "temperature",
        value: 66.6
    ) { ok, info } ''')

Response: {
    ok: true,
    info: ""
}
```

Pro konkrétní jazyk pak lze vytvořit přehlednější API, např. pro Python3 implementované v CLI aplikaci:

Ukázka 4. Ukázka API pro konkrétní jazyk, která nás provede vytvořením zařízení a jeho mutací. 1. vytvoříme periférii I2C, 2. přiřadíme konkrétnímu pinu kanál zařízení, 3. vytvoříme teploměr komunikující přes dříve vytvořenou periférií, 4. nastavíme jeho snímanou teplotu. 5. spustíme akci, která lineárně změní teplotu z 20 °C na 30 °C za 600 s.

```
1) i2c0 = com.create_peripheral(
    periheral="i2c"
   )
```

```
2) i2c0["sda"] = 0
   i2c0["scl"] = 1
3) t0 = com.create_device(
       device="thermometer",
       peripherals=[i2c0]
   )
4) t0["temperature"] = 23.2
5) t0["linearTransition"](20, 30, 600)
```

API nabízí různé pohledy na poskytovanou funkcionalitu. Mezi hlavní pohledy patří pohledy z hlediska pinů a zařízení. Pohled z hlediska pinů ukazuje jaký kanál na daném pinu je, jeho nastavený směr a případné informace o kanálu. Druhý pohled z hlediska dostupných zařízení, ukazuje jaké zařízení jsou dostupné, jaké jsou jejich kanály, atributy, akce a stav. Dostupnost zařízení je detekována automaticky po nahraní simulace. Ostatní pohledy jsou více specifické a lze je nalézt v dokumentaci API. Jedná se např. o stav FPGA, stav simulace atd. Každý pohled má i své mutace. Mutací uživatel například přiřazuje pinu kanál, který ho řídí (tím se určuje i jeho směr), nebo kanály, které z daného pinu čtou. Dále mutací může také měnit stav a hodnotu atributů vybraného zařízení (viz. ukázky 3, 4).

6. Budoucí rozšíření

Budoucí rozšíření, které by umožnilo větší propustnost je podpora streamování dat z periférií. Jako vhodný případ užití lze uvést simulace displeje. Implementované periférie na to jsou již připraveny. Jelikož platforma Zynq disponuje pouze dvěma porty pro stream dat, bylo by nutné navrhnout jednotku, která by řídila přepínání kanálů DMA mezi periferiemi. Dále pak navrhnout vhodný způsob přenosu těchto dat do klientské aplikace.

Další možné rozšíření je přidání parciální rekonfigurace. Ta by se uplatnila u volby periférií uživatelem, v FPGA by se přeprogramovala pouze část vyhrazená pro danou periférii a nebylo by nutné přeprogramovat celé FPGA. Pro její přidání by bylo potřeba správně oddělit rozhraní periférie od rekonfigurovatelného regionu a správně nastavit vývojové prostředí Vivado.

Pro lepší integraci vývojářem často používaných zařízení, by se mohla implementovat podpora programování a ladění procesoru pomocí JTAG nebo SWD. Toto rozšíření by také umožnilo lepší automatické testování, jelikož by k naprogramování testovaného systému nebyl potřeba počítač.

7. Závěr

Cílem této práce bylo navrhnout a realizovat snadno konfigurovatelnou a rozšířitelnou platformu, která umožní vývojáři vestavěných systému definovat simulované okolí mikrokontroléru. Tím tak urychlit vývoj vestavěného systému a to vše v dostupné a kompaktní podobě.

Podařilo se vytvořit platformu, která představuje přístup k vývoji a testování vestavěných sytému odlišný od běžných řešení. Realizovaná platforma splňuje stanovené cíle a slouží jako dobrý základ pro budoucí rozšíření.

Poděkování

Rád bych poděkoval mému vedoucímu Ing. Vojtěchovi Mrázkovi Ph.D. za cenné rady a za jeho čas strávený na konzultacích.

Literatura

[1] Glasgow debug tool, Jan 2018. https://github.com/GlasgowEmbedded/glasgow.

[2] Hardware-in-the-loop testing. online. https://www.hitex.com/tools-components/test-tools/minihil.

[3] Rp2040 datasheet, Apr 2021. https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf.

[4] open-amp, Oct 2014. https://github.com/OpenAMP/open-amp.

[5] Graphql. https://graphql.org/.

[6] Flatbuffers, Jan 2014. https://github.com/google/flatbuffers.

[7] Power profiler kit ii. https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2.