Virtuální laboratoř

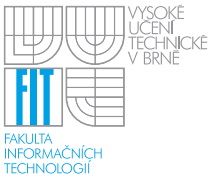
pro vývoj aplikací s mikroprocesory a FPGA

Autorský kolektiv

Michal Bližňák / Věra Budíková / Tomáš Dulík
Ota Jirák / Jiří Kadlec / Zbyněk Křivka
Nela Olšarová / Josef Trbušek / Zdeněk Vašíček

BRNO / 2011

CERM®
AKADEMICKÉ NAKLADATELSTVÍ



Poděkování:

Tato publikace byla finančně podpořena grantem MŠMT 2C06008 „Virtuální laboratoř aplikace mikroprocesorové techniky", jehož poskytovatelem je Ministerstvo školství, mládeže a tělovýchovy České republiky.

Další informace o projektu najdete na stránkách http://www.vlam.cz

Recenzent: doc. RNDr. Petr Šaloun, Ph.D.

ISBN 978-80-7204-754-3

Abstrakt:

Tato kniha popisuje implementaci výukové laboratoře VLAM (Virtuální laboratoř aplikace mikroprocesorové techniky), která kromě klasického lokálního přístupu k jednotlivým pracovištím poskytuje také možnost vzdálené práce. Pro realizaci vzdáleného přístupu jsou v roli pracovních stanic použity virtuální stroje na serveru, což přináší mnohé výhody a významné provozní úspory. Připojení virtuálních pracovních stanic k laboratornímu vybavení je realizováno pomocí rozhraní Ethernet a mostů „USB over Ethernet".

Prototyp laboratoře je určen pro výuku a vývoj aplikací s mikrokontroléry a programovatelnými hradlovými poli. Součástí našeho řešení je několik softwarových aplikací a hardwarových přípravků, které vývoj v této oblasti zjednodušují – např. Processor Expert™ ve formě modulu pro Eclipse (nyní integrováno ve Freescale CodeWarrior), VLAM IDE pro Eclipse, překladač jazyka C pro PicoBlaze atd.

Při vývoji laboratoře byl kladen velký důraz na dostupnost použitých komponent. Většina námi vyvinutých aplikací je proto uvolněna pod freeware nebo open source licencí a všechny HW komponenty spadají cenou do kategorie neinvestičních nákladů. Funkční prototyp laboratoře je díky tomu snadno opakovatelný i v jiných aplikačních oblastech.

Abstract:

This book describes implementation of Virtual Laboratory of Microprocessor Technology Application (VLAM) that allows students to work remotely with the equipment located in a school laboratory the similar way as they work in normal full-presence mode. For the remote work scenario, the VLAM infrastructure provides access to the laboratory by virtual desktop machines that are connected to the laboratory equipment through Ethernet or "USB over Ethernet" bridges.

The laboratory prototype targets the development of applications with microcontrollers and programmable gate arrays. To make the development and education easier, we have implemented several software tools and hardware gadgets – e.g., Processor Expert™ plug-in for Eclipse (now integrated into Freescale CodeWarrior suite), VLAM IDE for Eclipse, C compiler for PicoBlaze, etc.

The VLAM laboratory infrastructure is generic enough in order to be used in other application areas. For the reproducibility of our laboratory prototype, we have emphasized the general availability and/or low cost of all components. Moreover, most of the applications we have developed are licensed as freeware or even open source and the cost of hardware components we have used does not exceed the capital expenditure limit.

Obsah

1/ Úvod

Technologický pokrok ve všech oblastech informatiky umožňuje realizovat aplikace, které by před několika málo lety nebyly myslitelné. Kolektiv autorů této knihy v uplynulých pěti letech přispěl k tomuto pokroku řešeními, která jsou unikátní nebo mohou být užitečná pro vývojáře zabývající se aplikacemi s mikroprocesory a programovatelnými hradlovými poli.

V této publikaci chceme tato řešení představit a uvést do kontextu aktuálního stavu oboru.

Naše řešení reflektují dva důležité inovační směry:

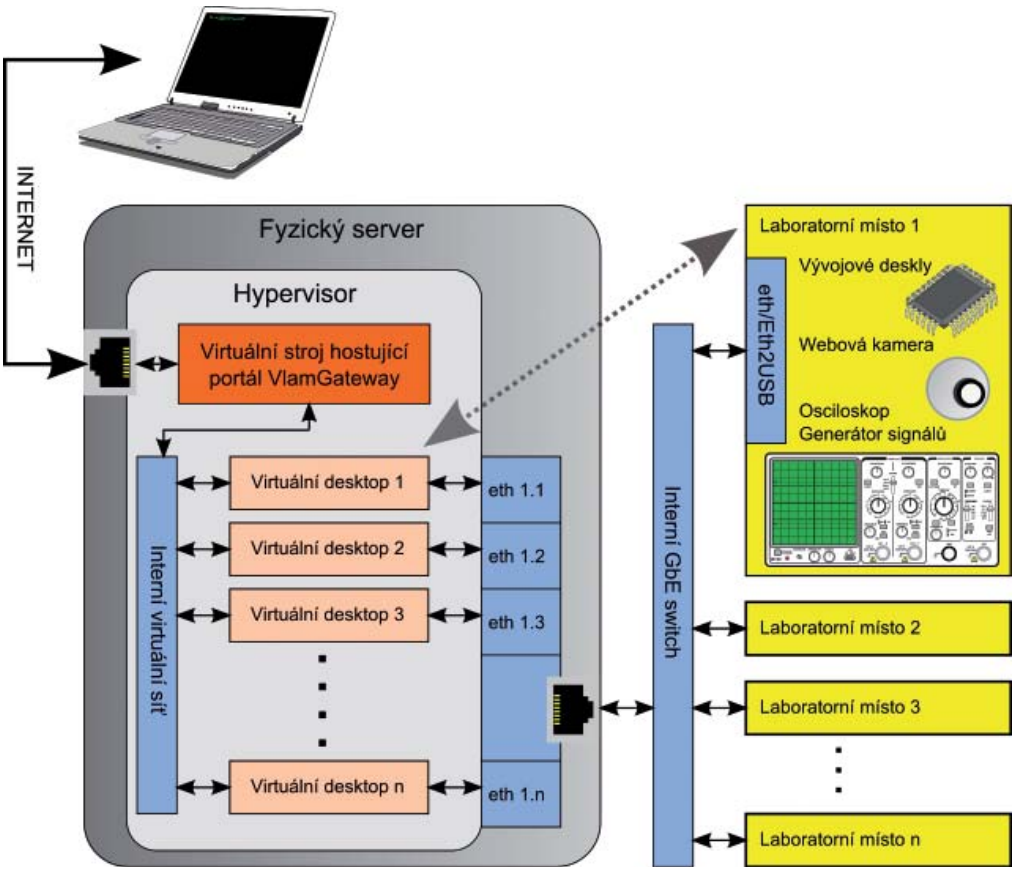
1. Implementaci vzdáleného přístupu s virtualizací pracovišť vývojářů se věnují kapitoly 2, 3 a 4.

2. Softwarové nástroje implementující nové metodologie návrhu mikroprocesorových a FPGA aplikací jsou popsány v kapitole 5, zatímco kapitola 6 obsahuje příklady použití těchto nástrojů ve virtuálních laboratořích.

Závěrečná kapitola naznačuje plány dalšího využití našich řešení v kontextu současného stavu technologií a prognóz jejich budoucího vývoje.

2/ Architektura laboratoře

V rámci řešení projektu MŠMT č. 2C06008 jsme navrhli novou architekturu pro vzdálené zpřístupnění a virtualizaci libovolné laboratoře obsahující libovolný počet pracovišť se stolními počítači a libovolnými zařízeními schopnými komunikace přes USB nebo Ethernet – např. vývojovými deskami, měřicími přístroji, laboratorními modely atd. – viz Obr. 1.



Obr. 1: Architektura laboratoře VLAM

Tato architektura splňuje následující požadavky:

1. Pro vzdálenou práci nemusí být zapnuté fyzické počítače v laboratoři. Pokud je laboratoř nebo část jejích pracovišť určena jenom pro vzdálenou práci, nemusí být fyzické počítače instalovány vůbec – jejich funkci zastávají virtuální stroje na serveru.

2. Virtualizační server nemusí být přímo v laboratoři, ale může být umístěn v libovolné stávající serverové místnosti s UPS, klimatizací a zabezpečením fyzického přístupu. Za tímto účelem jsou všechny přístroje a vývojové desky v laboratoři vybaveny rozhraním Ethernet nebo jsou připojeny k převodníku na Ethernet rozhraní.

3. Jednotlivá pracoviště laboratoře jsou/mohou být od sebe oddělena tak, že uživatel pracující na pracovišti A nemá možnost přístupu k přístrojům pracoviště B ani na nejnižší úrovni (např. na linkové vrstvě Ethernet a USB protokolů).

4. Jednotlivá pracoviště laboratoře lze vzdáleně zapínat a vypínat.

5. Uživatelé k laboratoři přistupují prostřednictvím webové aplikace, která zajišťuje všechny potřebné funkce laboratoře: autentizaci a autorizaci, rezervaci času, zobrazení obrazu webové kamery, vzdálenou plochu virtualizovaného stolního počítače atd.

Navrženou architekturu jsme realizovali ve dvou prototypech. První prototyp obsahuje 22 pracovišť s vývojovými deskami FIT Flexis pro mikrokontroléry Freescale a FITkit s FPGA Spartan XC3S50 a CPU MSP430. Druhý prototyp obsahuje 12 pracovišť s vývojovými deskami Freescale M68EVB08GB60 a Digilent Spartan 3E-1600.

Oba realizované prototypy prokázaly, že námi navržená a realizovaná architektura VLAM je dostatečně obecná, jednoduše konfigurovatelná a snadno i levně opakovatelná v kterékoli jiné laboratoři nebo vývojovém pracovišti. V této kapitole se pokusíme architekturu VLAM představit tak, aby čtenář pochopil její výhody a principy implementace.

2.1 / Výhody vzdáleného přístupu a virtualizace

Při vývoji aplikací s mikroprocesory a mikrokontroléry, které mohou být implementovány v jediném čipu programovatelného hradlového pole (PLD nebo častěji FPGA), se studenti a začínající vývojáři potýkají s mnoha problémy. Prvním z nich je bariéra vysokých pořizovacích nákladů vývojového pracoviště, které v minimální variantě zahrnují vývojové prostředí, vývojovou desku a osciloskop, popř. i generátor signálů.

Jedním ze základních cílů projektu VLAM proto bylo zvýšení přístupnosti a atraktivity vývoje aplikací pro studenty a začínající vývojáře. Vzdálené zpřístupnění laboratorního pracoviště je přímočarým řešením tohoto problému.

2.1.1/ Historické kontexty a virtualizace

Virtualizace stolních počítačů je znovuobjevením principu „tenký klient a tlustý server“, který byl ve světě UNIX systémů široce využíván nejdříve v podobě znakových terminálů připojených k terminálovým serverům, poté v podobě X Window System (dále jen X11) terminálů, jejichž zdařilou obdobou ve světě Microsoft (MS) Windows je systém vzdálené plochy (angl. Remote Desktop), který byl poprvé obsažen již v operačním systému (OS) Windows NT 4.0 Server, Terminal Server Edition.

Na rozdíl od systému X11 se ale princip víceuživatelského terminálového přístupu pomocí vzdálené plochy k serveru s MS Windows v praxi příliš neuchytil z následujících důvodů:

1. Nevhodná licenční politika firmy Microsoft: náklady na server s MS Windows pro více vzdálených uživatelů zahrnují licenci OS MS Windows Server a licence terminálových sezení pro každého uživatele. V přepočtu na jednoho uživatele jsou pak náklady na licence vyšší než náklady na OEM verzi MS Windows pro stolní počítač, kterou většina výrobců do prodávaných počítačů instaluje.

2. Důsledkem nevhodné licenční politiky je malé rozšíření terminálových nasazení MS Windows Server, které způsobilo, že mnoho výrobců software (SW) nepočítá s tím, že by jejich produkty někdo používal na MS Windows Server. Zájemci o terminálové nasazení pak s překvapením zjišťují, že mnoho aplikací třetích stran na MS Windows Server nefunguje z důvodu nekompatibility se stolními verzemi MS Windows.

   Důsledek předchozích bodů je takový, že pro smysluplné terminálové nasazení MS Windows jsou použitelné především tyto dvě cesty:

   a) Odstranění omezení jediného terminálového sezení u stolních MS Windows, což lze provést:

      - Úpravou MS Windows XP popsanou např. na stránkách [18], avšak s velkou pravděpodobností se jedná o porušení licenčních podmínek.

      - Instalací komerčního řešení např. [37] nebo [16].

   b) Virtualizace stolních verzí MS Windows. V takovém případě je pro každého uživatele vytvořena nová instance virtuálního stroje. Výhodou je úplné vzájemné oddělení uživatelů, díky němuž nemohou akce jednotlivých uživatelů negativně ovlivnit práci ostatních. Nevýhodou je větší náročnost na výpočetní a paměťovou kapacitu virtualizačního serveru.

V současné době hlavní proud komerčních i open source řešení směřuje k virtualizaci stolních verzí MS Windows. Komerční produkty jako VMware View™ nebo Red Hat Enterprise Virtualization for Servers (RHEV) poskytují kromě základních virtualizačních funkcí také šablony obrazů virtuálních stanic, podporu migrace, vyvažování zátěže mezi několika fyzickými servery atd. Podobné funkce jsou vyvíjeny i v několika open source projektech, např. oVirt, Cobbler, Deltacloud, AbiCloud, OpenNebula a další (viz [32] a [34]). Projekt VLAM využívá tohoto hlavního proudu open source řešení a doplňuje je o vlastní software specifický pro nasazení ve výukových laboratořích.

2.1.2 Ekonomické aspekty vzdáleného přístupu a virtualizace

Vytvoření jednoho laboratorního pracoviště se vzdáleným přístupem bez virtualizace je v dnešní době triviálním problémem a bylo jím i na začátku projektu VLAM. Spočívá v přidělení veřejné IP adresy jednomu počítači v laboratoři, který prostřednictvím vzdálené plochy popř. webové kamery zpřístupní jedno laboratorní pracoviště kterémukoli uživateli na internetu.

Pokud ale potřebujeme vytvořit fyzických pracovišť několik tak, aby mohly obsloužit více vzdálených uživatelů, zjistíme, že u běžných počítačů v běžné laboratoři je obtížné zajistit trvalý provoz a správu v režimu 24 hodin denně/365 dní v roce, protože:

- Laboratoře/učebny by musely mít velkokapacitní záložní zdroje (UPS), ty jsou ale velmi drahé - UPS pro 24 počítačů svojí cenou většinou přesáhne hranici pro investiční nákup. Bez UPS počítače po nekorektním vypnutí nemusí naběhnout, navíc proudový ráz při obnovení dodávky energie často vypne jističe, takže je nutný fyzický zásah správce.

- Počítač nechtěně vypnutý fyzickým uživatelem lze sice zapnout pomocí mechanismu WakeOnLan, pokud se ale uživateli podaří vypnout jej ze zásuvky, je opět potřeba fyzický zásah správce.

- „Zatuhlý“ počítač lze resetovat pouze manuálně.

- Laboratoře v menších místnostech mohou mít problém s nárůstem teploty nad rozumné meze, což se nejčastěji řeší instalací klimatizace, a tedy dalším skokovým nárůstem pořizovacích a provozních nákladů.

V rámci dvou prototypů jsme vytvořili 34 vzdálených pracovišť, které sice představují zlomek všech laboratorních kapacit dané vysoké školy, přesto již u tohoto množství můžeme porovnat pořizovací a provozní náklady běžné laboratoře s virtualizovanou laboratoří s tenkými klienty. Toto srovnání hovoří výrazně ve prospěch virtualizovaného řešení – viz následující příklad:

Tab. 1: Cena a příkon běžných počítačů a tenkých klientů

	Tenký klient	Běžné PC
Cena laboratorního počítače	5 000 Kč	15 000 Kč
Příkon laboratorního počítače	25 W	100 W

Úspora pořizovacích nákladů a elektřiny u virtualizované laboratoře spočívá v tom, že laboratorní počítače u virtualizované učebny mohou mít jen tak malý výpočetní výkon, jaký stačí pro připojení ke vzdálené ploše na serveru. Tomuto požadavku vyhovují již stroje s CPU AMD Geode (i586, 500MHz) za 3000 Kč, jejichž celkový příkon je 5W. V tabulce Tab. 1 je ale uvedena cena a příkon počítače s CPU Intel Atom D525 1.6 GHz s 1GB RAM a pevným diskem (HDD) s kapacitou 250 GB, který v případě výpadku serveru může provozovat většinu běžných aplikací také lokálně.

U 24 počítačů je tedy úspora pořizovacích nákladů 240 000 Kč, pro 300 počítačů pak 3 000 000 Kč. Z takto uspořených peněz lze snadno zakoupit dostatečně výkonný virtualizační server pro provozování aplikací náročných na výpočetní výkon, např. SuperMicro A+ 1042 (32 jader, 32GB RAM, HDD 1TB).

Roční úspora elektřiny závisí na využití laboratoře, tj. na počtu hodin, kdy jsou počítače zapnuté. Běžný provozní režim počítačových laboratoří na školách s kombinovaným studiem je 6 dnů v týdnu, 10 hodin denně 9 měsíců v roce, tj. např. 2160 hodin ročně. Pokud by ale měla být laboratoř přístupná i vzdáleně a bez virtualizace, musí být pracovní stanice zapnuté 24 hodin 365 dnů v roce, tj. 8760 hodin ročně. U řešení s virtualizací je samozřejmě nutné přičíst spotřebu serveru, který pracuje neustále, ale jeho příkon 200W je ekvivalentní dvěma běžným počítačům.

Při odběrném tarifu 4 Kč/kWh je roční úspora nákladů na elektřinu pro učebnu s 24 PC následující:

Tab. 2: Úspora nákladů na elektřinu pro 24 PC

	stálý provoz	běžný provoz
24 ks běžných PC	84 096,00 Kč	20 736,00 Kč
24 ks úsporných PC + virtualizační server	28 032,00 Kč	12 192,00 Kč
Úspora	56 064,00 Kč	8 544,00 Kč

Zajímavější čísla dostaneme pro větší počet počítačů. Např. u 300 úsporných PC v běžném provozu můžeme za peníze na elektřinu nakoupit každý rok 37 nových úsporných PC:

Tab. 3: Úspora nákladů na elektřinu pro 300 PC

	stálý provoz	běžný provoz
300 ks běžných PC	1 051 200,00 Kč	259 200,00 Kč
300 ks úsporných PC + virtualizační server	269 808,00 Kč	71 808,00 Kč
Úspora	781 392,00 Kč	187 392,00 Kč

Je jasné, že režim stálého (nonstop) provozu bude na většině škol využit jen u několika specializovaných učeben, pravděpodobně bude tedy větší část počítačů provozována běžným způsobem. Reálnou úsporu elektřiny proto můžeme vyjádřit intervalem, jehož spodní hranici určuje úspora pro běžný provoz a horní hranici úspora pro stálý provoz.

2.2/ Existující podobné projekty – studie „State of the art“

V celosvětovém i českém kontextu lze samozřejmě najít projekty se stejným nebo podobným zaměřením. Pokud je nám však známo, žádný z těchto projektů nevyužívá k realizaci vzdáleného přístupu virtualizaci. V rámci řešení VLAM bylo navíc vyvinuto několik dalších SW a HW nástrojů, které jsou ve své oblasti unikátní.

2.2.1/ 80C537 Microcontroller Remote Lab for E-Learning Teaching

Projekt Katalánské technické univerzity a Koruntanské univerzity aplikovaných věd umožňuje studentům práci z domova. Základem je mikrokontrolér µdee537 a kompletní profesionální vývojové prostředí pro jednočipové mikroprocesory Keil µVision2. Mikrokontrolér je spojen se serverem sériovou linkou RS-232. K nahrávání programů do mikrokontroléru a ladění je potřeba Keil µVision2. Ke vzdálenému přístupu k tomuto softwaru je využit Citrix Application Server software, který poskytuje rozsáhlému počtu klientů bezpečný přístup k aplikacím a programům nainstalovaným na serveru, proto studenti nemusí mít Keil µVision2 instalovaný na svém domácím počítači. K monitorování LCD a LED se využívá webová kamera. Tzv. „Human board interface“ slouží ke kontrole, monitorování, správě pracoviště a umožňuje uživateli chovat se tak, jako by byl přítomný přímo v laboratoři. Vyvinut byl pomocí software LabVIEW. Projekt byl realizován v roce 2006. [17]

2.2.2/ Internet School Experimental System - iSES

Měřicí a laboratorní studio iSES je široká otevřená platforma, která umožňuje měření a řízení experimentů. Systém iSES je otevřený modulární univerzální měřicí systém pro fyziku, chemii, biologii, elektrotechniku, elektroniku, automatizaci, měření aj. Celkem je k dispozici 20 čidel (teploměr, ampérmetr, voltmetr, siloměr, snímač polohy, optická závora, mikrofon, manometr, pH metr, konduktometr, relé, reproduktor, sonar, ohmmetr, měřič kapacit, snímač srdečního tepu aj.). Čidla se při připojení automaticky detekují, mají lineární charakteristiku, a pokud je to možné, tak nabízí diferenciální vstupy. ISES WEB Control je software podporující vzdálené měření a kontrolu iSES měřicího systému.

Souprava iSES je český výrobek, který se vyrábí a postupně vyvíjí již 25 let. [27]

2.2.3/ Vzdálená laboratoř NetLab pro vyučování inženýrských kurzů

Vzdálená laboratoř NetLab byla vyvinuta a implementována na Univerzitě jižní Austrálie v Adelaide. Reálný systém umístěný v laboratoři je připojený k NetLab serveru pomocí rozhraní GPIB (General Purpose Interface Bus). Server je připojen k internetu, fyzická zařízení jsou uživatelem vzdáleně ovládána přes internet. Zařízení lze sledovat pomocí webové kamery a ovládat s využitím animovaných displejů. NetLab server je připojen k univerzitnímu serveru, kde je databáze studentů, která slouží k autentizaci a autorizaci uživatelů. Pokud se student přihlásí do systému, je přesměrován na stránku experimentu a může začít provádět pokus. V tuto chvíli uživatel komunikuje pomocí klientského softwaru napsaného v Javě, jenž umožňuje vykonávat experimenty vzdáleně. NetLab klient komunikuje s NetLab serverem přímo posíláním odpovídajících příkazů. Software serveru je napsán v programovacím jazyce LabView. Laboratoř NetLab byla realizována v roce 2006. [43]

2.2.4/ Laboratoř integrované automatizace

Laboratoř integrované automatizace (LABI) je vybudována na Univerzitě Tomáše Bati ve Zlíně. Jde o sestavu laboratoří s reálnými fyzickými modely přístupnými přes internet. Laboratoř obsahuje devět základních řídících úloh: Regulace teploty, Teplárenská soustava, Měření průtoku, Řízení otáček motorů, Biochemické procesy, Propojení RS-232-Labi, Propojení Profinet – AS – Interface, Řízení soustavy hladin, Řízení zdrojů světla.

Systém řízení laboratoře zahrnuje:

- server Labi na bázi PC s uživatelským softwarem založeným na produktu Control Web,
- průmyslové počítače (IPC) s aplikačními programy Detlab v prostředí Control Web,
- programovatelné automaty (PLC) se softwarem vyvinutým v prostředí Step7,
- měřicí podsystém se snímači technologických veličin,
- podsystém ovládání s regulačními ventily a s fázovým řízením toku elektrické energie.

Řídicí systém je určen pro řízení a vizualizaci systému LABI, tj. funkci modelů technologických procesů a přístup k nim. Prostředí PC server je propojeno přes síť LAN s řídícími jednotkami DATALAB a SIEMENS. Uživatelé systému mohou pracovat s konkrétní úlohou a mohou ji řídit. Laboratoř vznikla v roce 2006. [21]

2.2.5/ FPGA Virtual Lab

FPGA laboratoř byla zrealizována na Technické univerzitě v Košicích. Cílem projektu bylo zpřístupnit špičkové softwarové a hardwarové nástroje pro vývoj a analýzu vestavěných systémů na bázi obvodů FPGA.

Pracoviště je vybaveno výkonným logickým analyzátorem a digitálním osciloskopem. Přístup do virtuální laboratoře je realizován pomocí serveru DELL 690, který je přístupný prostřednictvím přihlašovacího jména a hesla, jež je přiděleno správcem serveru. Pro konfiguraci FPGA na vývojové desce je použit program LabView firmy National Instruments. Reakce vývojové desky snímá webová kamera.

Součástí projektu je řešení ovládání osciloskopu Tektronix TDS 2024B prostřednictvím internetu. Využito bylo prostředí LabView a standardu VISA. Projekt byl vytvořen v letech 2007-2009. [11]

2.2.6/ Shrnutí

Ve srovnání s výše uvedenými projekty je laboratoř VLAM výjimečná mimo jiné tím, že je implementována čistě pomocí open source technologií, díky čemuž je snadno zopakovatelná v kterékoli další organizaci s minimálními náklady.

Řešení 80C537 Microcontroller Remote Lab for E-Learning Teaching využívá ke své realizaci prostředí National Instruments LabView a také Citrix Application Server [17]. O těchto produktech jsme na začátku uvažovali i v projektu VLAM. Citrix Application Server ale svojí cenou spadá do kategorie investic, což limituje snadnou opakovatelnost tohoto řešení. Mikrokontrolér 80C537 komunikuje se serverem sériovou linkou RS232, což omezuje vzájemné umístění serveru a laboratorního pracoviště – vzdálenost mezi serverem a mikrokontrolérem může být pouze několik málo metrů. To lze samozřejmě řešit pomocí převodníku RS232-Ethernet, ale protože většina současných vývojových desek namísto rozhraní RS232 využívá USB, řešili jsme v projektu VLAM raději způsob přenosu USB po Ethernetu.

Projekty laboratoře 80C537, NetLab a FPGA VirtualLab využívají pro ovládání měřicích přístrojů komerční produkt LabView, který není dostupný zdarma. Proto jsme v rámci řešení projektu VLAM vyvinuli jeho jednoduchý ekvivalent – SW Vlaxicon (**VLA**M L**XI** **Con**sole), jehož prostřednictvím lze ovládat libovolný měřicí přístroj dostupný přes rozhraní LXI nebo VISA. Podobný SW byl implementován i v rámci projektu VirtualLab, avšak na rozdíl od programu Vlaxicon je tento SW zřejmě určen pouze pro jeden konkrétní typ osciloskopu.

Kromě výše uvedených výhod jsme v rámci projektu VLAM vyvinuli i další unikátní SW a HW nástroje. Jejich popis je obsažen v následujících kapitolách.

2.3/ Softwarové vybavení laboratoře

Základní SW komponenty použité v rámci systému VLAM lze z hlediska jejich použití rozdělit do tří oblastí:

- SW nutný pro zajištění virtualizace pracovních stanic a přístupové brány systému,
- SW vybavení přístupové brány systému (VlamGateway),
- SW vybavení virtuálních pracovních stanic (VPS).

Při výběru vhodné virtualizační technologie jsme vycházeli ze specifických požadavků, jako jsou funkcionalita a výkon hypervizoru. Mezi požadované vlastnosti patřila možnost instalace hypervizoru na server bez nutnosti využití licencovaného hostujícího OS, možnost vzdálené správy virtuálních strojů a možnost připojení USB zařízení k hypervizoru i k virtuálním strojům. Dalším požadavkem byla schopnost efektivně virtualizovat operační systémy MS Windows i Linux a poskytnout dostatek výkonu pro běh desítek virtuálních strojů současně. V neposlední řadě byl kladen důraz také na cenu a přijatelné licencování zvoleného řešení.

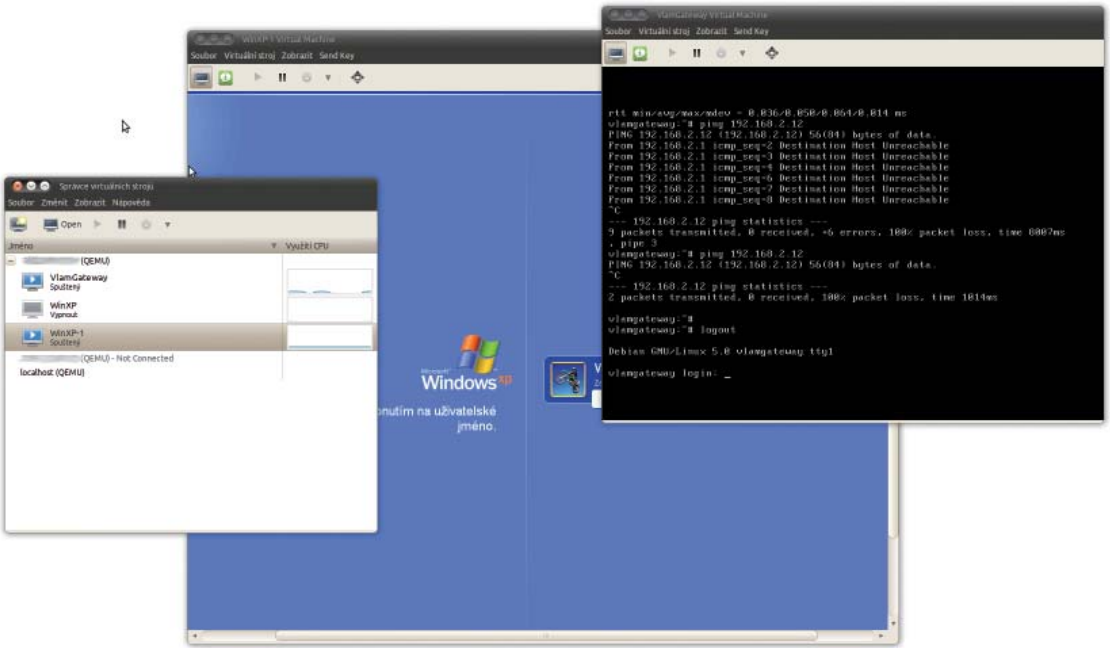
Na základě testů [10], které byly provedeny za účelem ověření funkcionality a výkonu dostupných řešení, byly vybrány následující vhodné virtualizační technologie:

Jako základ virtualizačního řešení byl v prvotní konfiguraci systému nasazen produkt VMware Server 2.0.2 běžící na OS Linux (Debian 5 Lenny). V průběhu testování se bohužel vyskytly problémy s nestabilitou hypervizoru a jeho administračního rozhraní, a proto byl jako hypervizor vybrán obdobný vhodný produkt VMware ESXi 4.

VMware ESXi 4 je další v řadě virtualizačních řešení, které umožňuje běh na „holém železe“ (anglicky bare-metal), tedy bez nutnosti použití hostujícího OS. Čtvrtá řada tohoto systému navíc umožňuje připojení USB zařízení k virtuálním strojům. Produkty řady ESX/ESXi jsou však citelně licenčně omezeny, což znesnadňuje, či kompletně znemožňuje efektivní správu systému bez zakoupení finančně drahých nástrojů pro management virtuálních strojů [40]. Navíc tyto systémy neumožňují připojení USB zařízení k hostujícímu systému (hypervizoru), čímž byla v našem případě znemožněna instalace některých důležitých komponent (např. instalace UPS). Z tohoto důvodu a také díky nestabilitě některých hostovaných OS bylo po dalších testech kompletně upuštěno od řešení postaveného na platformě VMware. Jako další vhodné virtualizační řešení byla zvolena technologie KVM [33] ve spolupráci s OS Ubuntu Server 10.04 LTS.

Technologie KVM umožňuje efektivně virtualizovat širokou škálu OS a je dostupná v celé řadě moderních Linuxových distribucí. Hypervizor umožňuje připojit libovolné HW zařízení dostupné na PCI nebo USB sběrnici, a to jak k hostujícímu systému, tak i k virtuálním strojům. Zároveň je možné celý systém spravovat jak pomocí skriptů, tak také prostřednictvím GUI, či webových nástrojů [34], jak je vidět na Obr. 2. Díky virtualizační technologii KVM se podařilo zkonfigurovat celý systém tak, aby zajišťoval všechnu potřebnou funkcionalitu a také spolehlivý běh virtuálních strojů.

Detailní popis instalace a konfigurace hypervizoru je uveden v dokumentu „Poznámky k instalaci fyzického serveru systému VLAM“ [4].



Obr. 2: Správce virtuálních strojů Virtual Machine Manager a hypervizor KVM

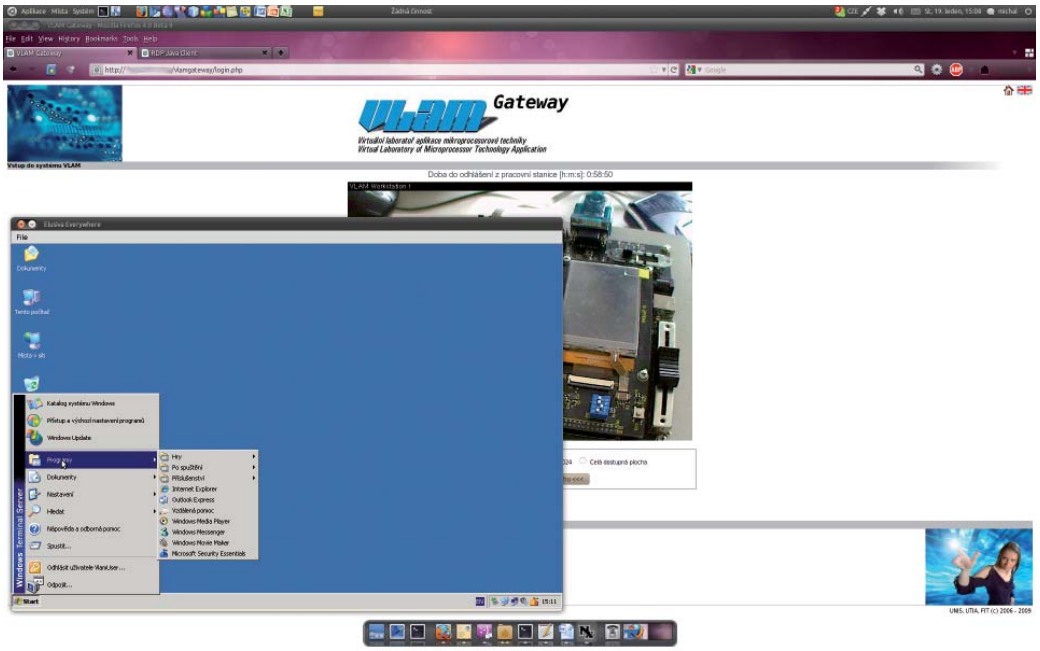
SW vybavení přístupové brány systému (VlamGateway) slouží k zajištění vzdáleného přístupu uživatelů k virtualizovaným pracovním stanicím.

Jedná se o virtuální serverové řešení založené na OS Linux (Debian 5 Lenny) poskytující následující služby:

- webový a databázový server (Apache 2 a MySQL 5) hostující webový portál projektu VLAM – VlamGateway,
- SW router založený na firewallu Shorewall sloužící k přesměrování datového spojení mezi virtuálními pracovními stanicemi a jejich uživateli.

Webový portál projektu VlamGateway v současné době poskytuje centralizované přihlašování do systému, jeho správu, přístup k publikovaným vzdáleným aplikacím a v neposlední řadě také monitoring pracoviště prostřednictvím kamery a specializovaného nástroje Vlaxicon [39], [38] určeného pro vzdálený přístup k měřicím přístrojům, jimiž je pracoviště vybaveno (viz Obr. 3).

Detailní popis instalace a konfigurace virtuální přístupové brány VlamGateway je uveden v dokumentu „Poznámky k instalaci virtuálního serveru webové aplikace VlamGateway“ [6].



Obr. 3: Přístup na VPS z prostředí aplikace VlamGateway pomocí Elusiva Java RDP Client

SW vybavení virtuálních pracovních stanic (VPS) umožňuje efektivní práci s HW prostředky dostupnými na vzdáleném pracovišti.

Jako OS pro VPS byl v prvotní fázi implementace systému zvolen MS Windows Server 2003, a to zejména díky možnosti publikovat aplikace prostřednictvím Terminal Services (TS) a 2X ApplicationServer [1]. V průběhu testování však bylo zjištěno, že některé důležité SW komponenty (např. IDE CodeWarrior) systému VLAM neumožňují běh společně s TS, a tudíž bylo nutné zvolit jiný způsob jejich publikace.

Problém s licenčními omezeními běhu některých SW nástrojů byl vyřešen umožněním přístupu ke vzdálené pracovní ploše VPS také pomocí klienta protokolu RDP. Také odpadla nutnost instalovat klienta pro 2X ApplicationServer na straně uživatele, a to díky tomu, že vzdálená plocha může být zobrazena buď pomocí nativního klienta protokolu RDP (je součástí MS Windows a také většiny Linuxových distribucí), nebo pomocí klienta Elusiva Open Source Java RDP Client (Elusiva Everywhere) postaveného na technologii Java, který je integrován přímo do webového portálu systému VLAM. Další výhodou je, že jako operační systém plně postačí MS Windows XP, jelikož technologie publikace vzdálené plochy pomocí protokolu RDP nevyžaduje plnohodnotný TS.

Podrobnou specifikaci softwarových nástrojů instalovaných na VPS lze nalézt v kapitole 5 a popis instalace a konfigurace VPS je uveden v dokumentu „Poznámky k instalaci virtuální pracovní stanice“ [5].

3/ VlamGateway

Jednou ze základních komponent systému VLAM je webový portál VlamGateway. Jeho účel a vlastnosti jsou popsány v následujících podkapitolách.

3.1/ Specifikace a stávající řešení

Požadavky kladené na webový portál systému VLAM vyplývají z jeho úzkého specifického zaměření. Jeho hlavní požadovanou funkcionalitou je umožnění přístupu k virtualizovaným pracovním stanicím prostřednictvím jednotného aplikačního rozhraní. Dílčí funkcionalitou portálu je kromě možnosti správy přihlašování k VPS a jejich rezervace také směrování datové komunikace mezi portálem a jednotlivými VPS a zprostředkování přístupu k publikovaným aplikacím uživatelům systému a vzdálený vizuální monitoring pracoviště pomocí kamerového systému. Z uvedeného je patrné, že se jedná o vysoce specifické řešení, pro které nebylo možno nalézt existující volně dostupný ekvivalent. Z tohoto důvodu bylo přikročeno k vytvoření proprietárního řešení s využitím některých již existujících a volně dostupných komponent.

Mezi proprietární komponenty systému patří:

- základní UI uživatelské rozhraní webové aplikace,
- přihlašovací systém a sekce pro správu webového portálu,
- systém pro monitoring vzdáleného pracoviště,
- systém pro směrování komunikace mezi portálem VlamGateway a VPS.

Mezi existující, integrované, komponenty využité při vývoji portálu patří:

- rezervační systém phpBookingCalendar [22] použitý pro implementaci rezervačního systému
- a zabudovaný klient pro přístup k publikované vzdálené ploše Elusiva Open Source Java RDP Client [15].

3.2/ Popis implementace

Webová aplikace VlamGateway představuje základní softwarové vybavení projektu VLAM a umožňuje práci na vzdálených virtuálních pracovních stanicích. Jedná se o aplikaci postavenou na technologiích Apache + MySQL + PHP + Java poskytující funkcionalitu nutnou pro správu a publikaci VPS.

Pro plnohodnotnou práci s aplikací musí uživatel disponovat:

- operačním systémem podporovaným aplikačním serverem (z důvodu nutnosti instalace klienta aplikačního serveru):
  - MS Windows XP, Vista, 7,
  - Linux (Red Hat, OpenSUSE, Debian, Ubuntu, ...),
  - OS X.
- a podporovaným internetovým prohlížečem:
  - MS Internet Explorer 6 a vyšší,
  - Mozilla Firefox 3.x a vyšší,
  - Google Chrome/Chromium.

Dále je vyžadována instalace běhového prostředí jazyka Java (Java Runtime Environment) na lokální stanici uživatele a v případě použití prohlížečů MS Internet Explorer také povolení spouštění prvků ActiveX.

Samotná webová aplikace se skládá ze dvou základních sekcí:

- uživatelské sekce určené pro běžnou práci se systémem,
- administrační sekce určené pro správu webové aplikace (definici uživatelských účtů, pracovních stanic a publikovaných aplikací).

3.2.1/ Společné ovládací prvky aplikace

Celé prostředí webové aplikace bylo vytvořeno tak, aby její ovládání bylo přehledné a intuitivní. Aplikace je navíc plně vícejazyčná a mezi dostupnými překlady lze přepínat kdykoliv během práce s aplikací (a to jak v rámci uživatelské, tak i administrační sekce aplikace). V současné době aplikace podporuje český a anglický jazyk.

Základní společné ovládací prvky aplikace jsou po celou dobu práce s ní umístěny v pravém horním rohu klientské oblasti okna prohlížeče a mají tuto podobu:

Ikona s podobou domu představuje tlačítko pro odhlášení z dané sekce (platí jak pro uživatelskou, tak i administrační sekci) a návrat k uvítací obrazovce.

Ikona s podobou vlajky slouží k přepnutí na jazyk právě zobrazované země. Toto přepnutí lze provést kdykoliv během práce s aplikací.

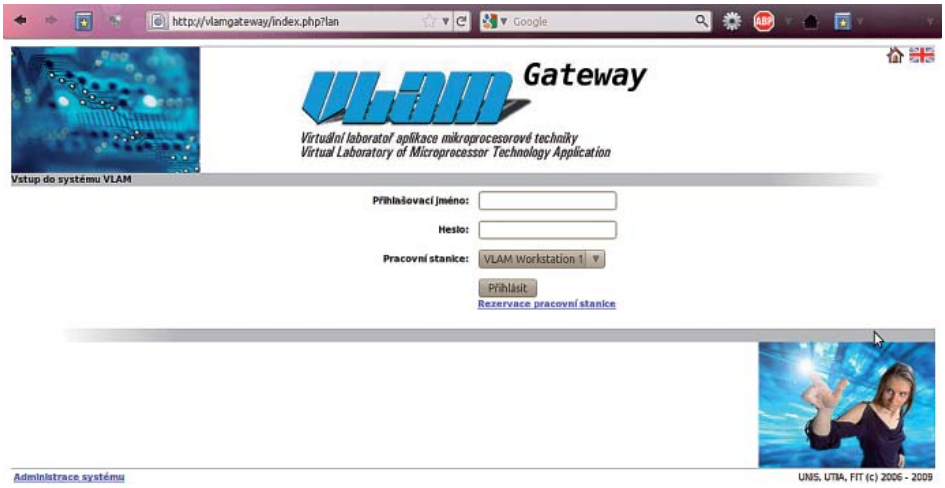
3.2.2/ Uživatelská sekce

Uživatelská sekce webové aplikace VlamGateway slouží pro práci běžných uživatelů se systémem. Nabízí možnost rezervace pracovní stanice, spouštění publikovaných aplikací a sledování vzdáleného pracoviště.

3.2.2.1/ Přihlášení do systému

K zahájení práce s publikovanými aplikacemi a sledování pracoviště je zapotřebí přihlášení do systému. Přihlásit se může pouze uživatel, který je v systému zaregistrován (tuto registraci musí provést administrátor systému) a má práva pro práci s publikovanými aplikacemi.

Přihlašovací obrazovka je uvedena na Obr. 4:



Obr. 4: Přihlášení do uživatelské sekce systému VlamGateway

Pro přihlášení se do uživatelské sekce aplikace je zapotřebí:

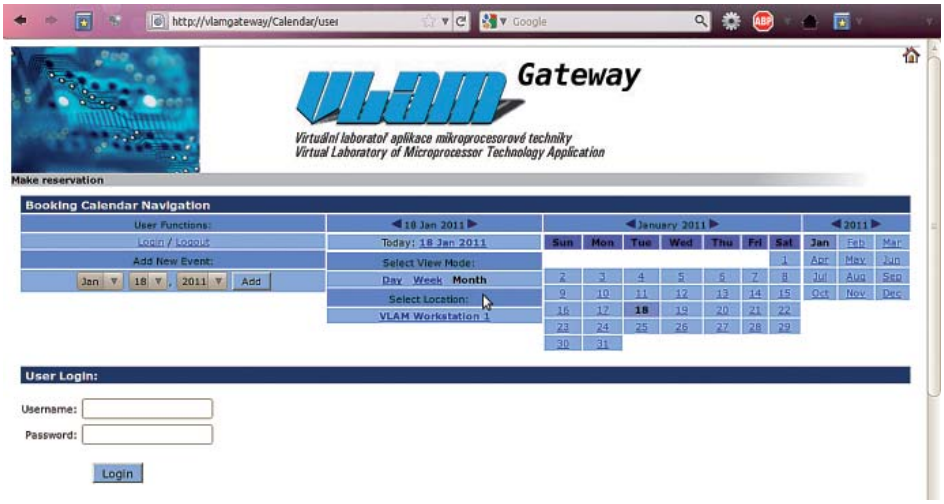
- vyplnit uživatelské jméno,
- vyplnit heslo uživatele,
- zvolit pracovní stanici,
- stisknout tlačítko „Přihlásit“.

Poznámka: V nabídce pracovních stanic se zobrazují pouze aktuálně neobsazené pracovní stanice, takže její obsah se může měnit.

Přihlášení se k VPS může selhat tehdy, je-li VPS na daný termín rezervována jiným uživatelem.

3.2.2.2/ Rezervace virtuální pracovní stanice

Přístup k VPS je možné dopředu rezervovat a tím se vyhnout situaci, kdy se nelze na konkrétní stanici přihlásit z důvodu jejího obsazení jiným uživatelem. K rezervačnímu systému lze přistoupit pomocí odkazu „**Rezervace pracovní stanice**“ zobrazenému pod tlačítkem „**Přihlásit**“ na úvodní stránce webové aplikace VlamGateway. Po vstupu do rezervační sekce (viz Obr. 5) a po přihlášení (**uživatelské jméno a heslo je totožné s údaji použitými pro přihlášení se k VPS**) je možné provést rezervaci vybraného pracoviště na konkrétní termín.



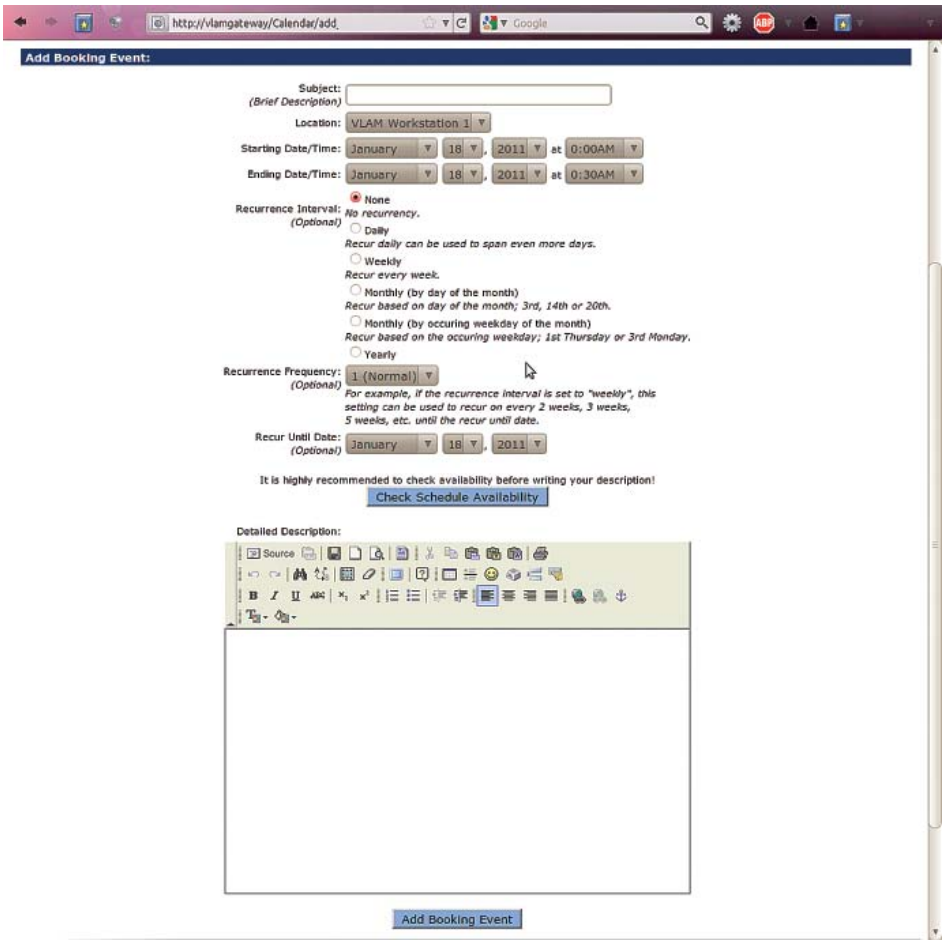
Obr. 5: Rezervační systém aplikace VlamGateway

Po přihlášení se k rezervačnímu systému je nutné ze seznamu „**Select Location**“ zvolit požadovanou VPS a v kalendáři vybrat první den rezervace.

Nový rezervační záznam lze přidat kliknutím na odkaz „(+)“ zobrazený v kalendáři u každého dne. Po kliknutí na tento odkaz bude zobrazen formulář s následujícími položkami (viz Obr. 6):

- Subject: Krátký název rezervace,
- Location: Rezervovaná VPS,
- Starting Date/Time: Počáteční čas rezervace,
- Ending Date/Time: Koncový čas rezervace,
- Recurrence Interval: Opakovaná rezervace,
  - None: bez opakování,
  - Daily: každodenní opakování,
  - Weekly: týdenní opakování,
  - Monthly: měsíční opakování,
  - Yearly: roční opakování,
- Recurrence Frequency: Frekvence opakování – násobek opakovacího intervalu,
- Recur Until Day: Datum ukončení opakované rezervace,
- Detailed Description: Nepovinné upřesnění důvodů rezervace.

Po vyplnění všech požadovaných údajů je vhodné ověřit, není-li již VPS v některém rezervovaném časovém slotu obsazena jiným uživatelem. To lze provést stisknutím tlačítka „**Check Schedule Availability**“. Je-li vše v pořádku, lze rezervaci potvrdit stiskem tlačítka „**Add Booking Event**“.



Obr. 6: Rezervace VPS

3.2.2.3/ Práce s virtuální pracovní stanicí

Po úspěšném přihlášení k VPS je zobrazena pracovní plocha obsahující:

- informaci o zbývajícím čase přiděleném pro práci s VPS,
- okno kamery monitorující ovládané HW zařízení,
- ovládací prvky pro přístup k publikovaným aplikacím.

Přístup k publikovaným aplikacím se liší v závislosti na použitém aplikačním serveru. Systém VlamGateway obecně umožňuje využít klasické připojení ke vzdálené ploše pomocí vestavěného klienta protokolu RDP (pro jeho funkci je vyžadována instalace běhového prostředí jazyka Java včetně zásuvného modulu pro internetový prohlížeč) nebo pomocí klienta aplikačního serveru 2X ApplicationServer. Výběr publikační technologie se provádí při vytváření VPS v administrační sekci a musí korespondovat s nastavením a konfigurací VPS.

3.2.3/ Připojení ke vzdálené ploše

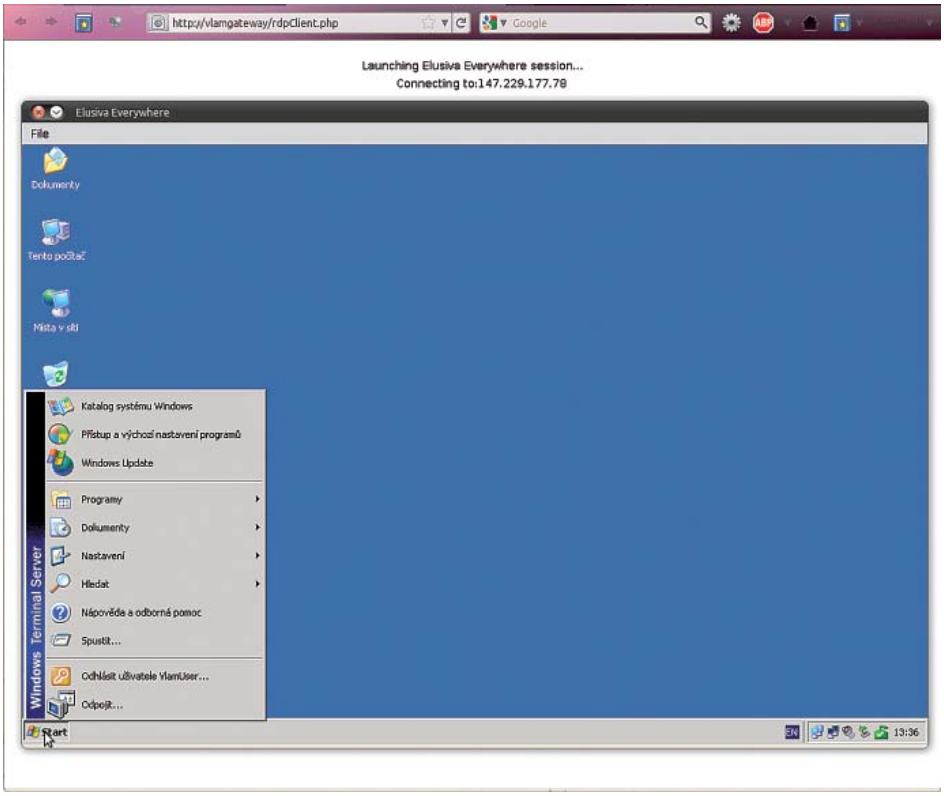
V tomto módu je pomocí protokolu RDP vzdáleně publikována celá pracovní plocha VPS.

K přístupu ke vzdálené pracovní ploše lze použít vestavěný RDP klient implementovaný v jazyce Java nebo nativní klient protokolu RDP dostupný na lokální stanici (přihlašovací údaje, tj. IP adresa, TCP port, jméno a heslo lze zobrazit přímo v portálu VlamGateway).

V tomto pracovním módu stačí jednoduše zvolit požadovanou velikost virtuální pracovní plochy a kliknout na tlačítko „>>> **Zobraz vzdálenou pracovní plochu** <<<“ (viz Obr. 7). Po stisknutí tlačítka bude v prohlížeči otevřen druhý panel, v němž bude zobrazeno informační okno RDP klienta, a samotná vzdálená pracovní plocha bude otevřena v samostatném okně (viz Obr. 8). Otevřený panel RDP klienta je možné kdykoliv zavřít (v tomto případě dojde k odpojení, nikoliv k odhlášení se z pracovní relace; běžící aplikace zůstanou otevřeny a budou k dispozici po opětovném zobrazení vzdálené plochy) a novou pracovní relaci lze opět zahájit stiskem tlačítka pro zobrazení vzdálené pracovní plochy.

Okno vzdálené pracovní plochy obsahuje vše potřebné pro práci na VPS.

Obr. 7: Práce s VPS v režimu RDP



Obr. 8: Vzdálená plocha VPS publikovaná pomocí vestavěného klienta protokolu RDP

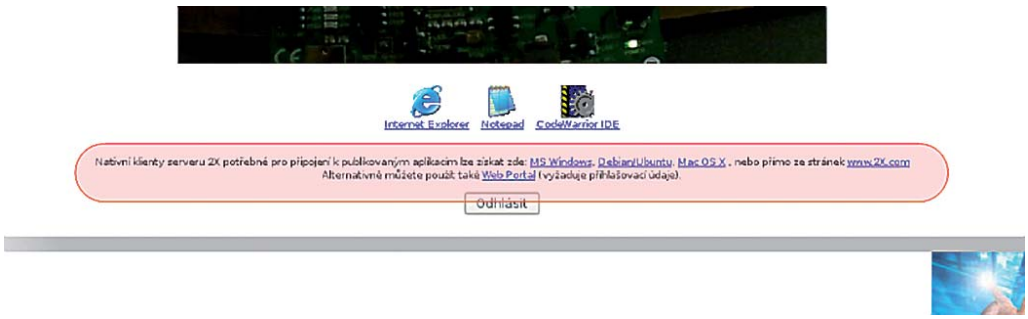
3.2.4/ Přístup pomocí 2X ApplicationServer

Aplikační server 2X ApplicationServer je určen pro publikování aplikací pomocí MS Windows Terminal Services (TS) a pro svůj běh vyžaduje OS MS Windows Server 2003 a vyšší s IIS a technologií .NET. Na straně uživatele vyžaduje také instalaci klientského SW. Na rozdíl od publikace vzdálené plochy pomocí RDP umožňuje publikovat sadu vybraných aplikací, nebo celý desktop. Nevýhodou z hlediska potřeb projektu VLAM je, že klient protokolu 2X není přenosný (vyžaduje instalaci) a že některé aplikace, které budou provozovány na VPS, neumožňují běh prostřednictvím TS. Z tohoto důvodu je v současné době v systému VLAM použito publikování aplikací pomocí klasického/Java RDP klienta.

3.2.4.1/ Stažení klienta pro 2X ApplicationServer

Pro práci s publikovanými aplikacemi je nutné mít nainstalovanou klientskou aplikaci aplikačního serveru. Klienta je možné stáhnout několika způsoby (viz Obr. 9):

- po přihlášení se do systému VlamGateway jsou na pracovní ploše umístěny odkazy na instalátory pro nejpoužívanější SW platformy (Windows, Debian/Ubuntu, OS X),
- v tomtéž místě je umístěn také odkaz na stránku pro stažení klientských aplikací přímo z domovských stránek pro 2X ApplicationServer (http://www.2x.com/applicationserver/downloadlinks.html).



Obr. 9: Odkazy pro stažení klienta pro 2X ApplicationServer

3.2.4.2/ Práce se systémem

Po úspěšném přihlášení do uživatelské sekce aplikace bude na webové stránce zobrazena pracovní plocha (viz Obr. 10), která obsahuje následující ovládací a informační prvky:

- záhlaví a logo systému aplikace VlamGateway se společnými ovládacími prvky (viz kapitola „Společné ovládací prvky“),
- informaci o době, za kterou bude uživatel automaticky odhlášen z pracovní stanice,
- výstup z kamery monitorující vzdálené pracoviště,
- seznam publikovaných aplikací,
- odkazy pro stažení klientské aplikace pro 2X ApplicationServer,
- odkaz na webový portál pro 2X ApplicationServer (vyžaduje speciální přihlašovací údaje)
- a tlačítko pro bezpečné odhlášení se z uživatelské části aplikace.

Nejdůležitějšími ovládacími prvky jsou výstup z kamery sledující programovaný vývojový kit, odkazy pro spuštění publikované aplikace a tlačítko pro odhlášení se z pracovní stanice.

Každý uživatel může na pracovní stanici strávit pouze předem definovanou dobu. Tuto dobu lze změnit v obecných nastaveních v administrační části aplikace nebo je dána následující rezervací.

Samotná práce pak probíhá prostým spouštěním publikovaných aplikací a prací s nimi. Odezvu ovládaných HW zařízení lze poté sledovat na výstupu kamery.

Poznámka: Z licenčních důvodů je možné v jeden okamžik spustit a používat pouze 3 publikované aplikace.

Obr. 10: Rozložení ovládacích prvků na pracovní ploše webové aplikace

3.2.4.3/ Spuštění publikované aplikace

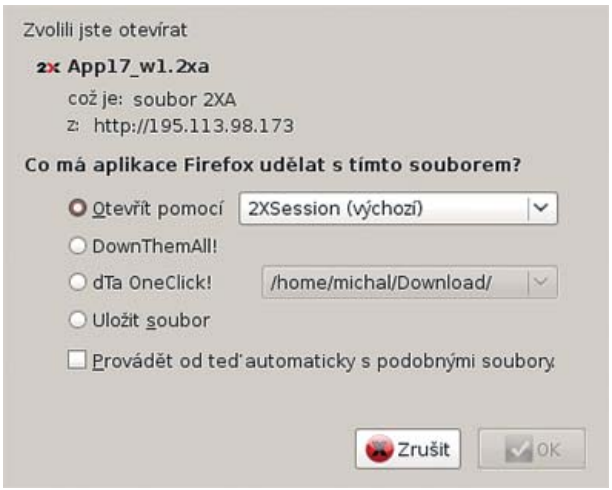
Po úspěšné instalaci klienta pro 2X ApplicationServer je možné spustit publikovanou aplikaci dvojím způsobem:

- v případě, že v průběhu instalace klienta došlo ke korektní registraci přípony zástupce aplikace *.2xa, bude u většiny prohlížečů zobrazeno okno nabízející stažení odkazu, nebo jeho spuštění klientskou aplikací (toto chování bylo ověřeno u MS Windows, Debian/Ubuntu Linux 32-bit s prohlížeči Internet Explorer, Mozilla Firefox a Google Chrome/Chromium, viz Obr. 11),

- v případě, že nedošlo k registraci přípony (což je možné u některých Linuxových distribucí), lze aplikaci spustit ručně tak, že stažený zástupce bude použit jako parametr klientské aplikace spouštěné z příkazové řádky: `./appserverclient App17_w1.2xa`.

V obou případech musí být v době spouštění a práce s aplikací uživatel stále přihlášen na pracovní stanici, aby bylo aktivní přesměrování datové komunikace mezi jeho lokální stanicí a vzdálenou pracovní stanicí.

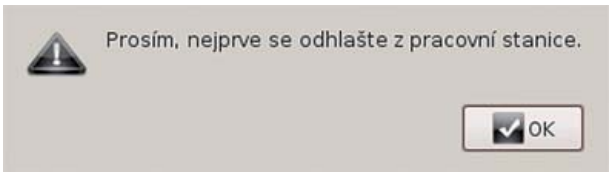
Ukončení práce s publikovanou aplikací probíhá standardním způsobem, tj. zavřením okna aplikace, nebo pomocí jiného vhodného ovládacího prvku dané aplikace (položka menu, tlačítko na panelu nástrojů atd.).



Obr. 11: Nabídka spuštění, nebo stažení zástupce publikované aplikace v OS Ubuntu

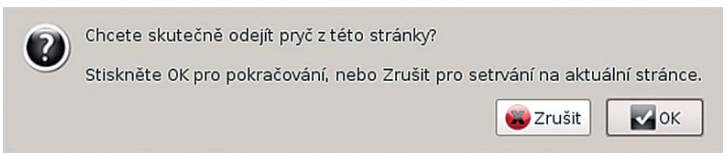
3.2.4.4/ Odhlášení ze systému

Odhlášení z webové aplikace musí probíhat zásadně pomocí tlačítka „Odhlásit“ umístěného na pracovní ploše pod seznamem publikovaných aplikací. V opačném případě nedojde ke korektnímu ukončení přesměrování komunikace mezi lokální stanicí uživatele a vzdálenou pracovní stanicí až do doby, než se na (libovolnou) pracovní stanici pokusí přihlásit někdo jiný. Pokud se uživatel pokusí zavřít okno prohlížeče nebo přejít na jinou stránku, bude na to upozorněn, jak demonstruje Obr. 12:



Obr. 12: Upozornění na nekorektní odchod z pracovní plochy webové aplikace

a bude mít možnost v aplikaci VlamGateway setrvat (stisknutím tlačítka „Zrušit“), nebo pokračovat na jinou stránku (tlačítko „OK“), tak, jak je uvedeno na Obr. 13.



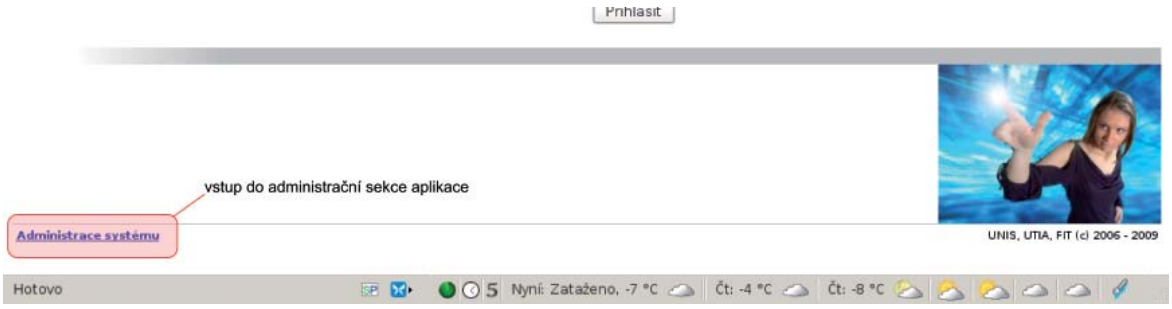
Obr. 13: Opětovný dotaz na opuštění pracovní plochy webové aplikace

3.2.5/ Administrační sekce

Administrační sekce webové aplikace VlamGateway umožňuje:

- spravovat účty uživatelů systému,
- upravovat obecná nastavení aplikace,
- spravovat pracovní stanice a publikované aplikace.

Do administrační sekce lze vstoupit pomocí odkazu v levém dolním rohu uvítací/přihlašovací obrazovky aplikace VlamGateway (viz Obr. 14).



Obr. 14: Vstup do administrační sekce aplikace

Pro přihlášení je vyžadováno jméno a heslo uživatele. Tento uživatel musí mít práva pro administraci systému.

Po úspěšném přihlášení jsou na základě rozsahu práv uživatele zobrazeny další možnosti správy aplikace. Jednotlivé administrační kategorie jsou uvedeny na záložkách (viz Obr. 15), pomocí nichž lze mezi danými kategoriemi přecházet.

Pro odhlášení se z administrační sekce aplikace slouží odkaz umístěný v pravém horním rohu záložek kategorií.

Obr. 15: Záložky administračních kategorií a odhlášení z administrace

Jednotlivé administrační kategorie dostupné na záložkách jsou následující:

- Uživatelé:
  - Nový uživatelský účet,
  - Správa uživatelů.

- Pracovní stanice:
  - Obecná nastavení,
  - Nastavení firewallu,
  - Nová pracovní stanice,
  - Správa pracovních stanic,
  - Nová aplikace,
  - Správa aplikací.

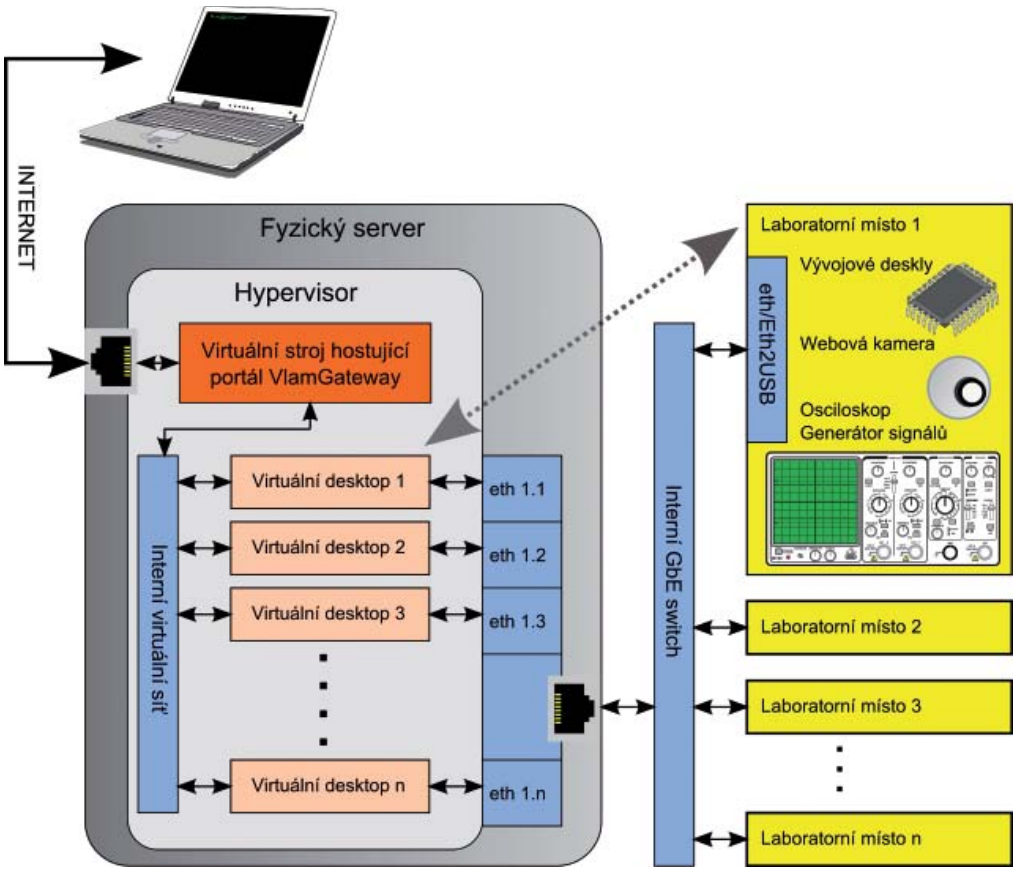
Bližší informace o možnostech administrace lze získat v dokumentu „Uživatelská příručka webové aplikace VlamGateway“ [6].

4/ Hardwarové vybavení VLAM pracovišť

V této kapitole představíme infrastrukturu virtuální laboratoře, podporované vývojové platformy (Flexis, FITkit a Spartan 3E) a měřicí přístroje. Jedná se o popis prostředků využitých při realizaci prototypu laboratoře s tím, že výše popsaný koncept lze jednoduše aplikovat i na jiné platformy sloužící pro vývoj vestavěných systémů a aplikací s mikroprocesorovou technikou.

4.1/ Infrastruktura

Infrastruktura VLAM je zobrazena na Obr. 16. Skládá se z následujících HW prvků:



Obr. 16: Infrastruktura VLAM

1. Fyzický virtualizační server: Námi použitý virtualizér KVM vyžaduje, aby procesor fyzického serveru podporoval rozšíření Intel VT nebo AMD-V (obecný název pro tato rozšíření je anglicky Hardware Virtual Machine). Tomuto požadavku vyhovují procesory většiny běžně prodávaných serverů pro obecné použití,

pouze výjimečně se lze setkat se serverem, který tuto technologii nepodporuje. Většinou se jedná o speciální servery pro úsporné aplikace s CPU Intel Atom.

2. Ethernetový přepínač (angl. *switch*) v učebně: Pro vzájemnou izolaci pracovišť laboratoře je potřeba, aby použitý přepínač podporoval standard 802.1q (tagované VLANy). Tomuto požadavku dnes vyhovují už i velmi levné přepínače. V našich prototypech jsme použili přepínače 3COM 4210 (26 portů), HP Procurve 2524 (J4813A) a Dlink DGS-1210-24.

3. Převodníky USB2Ethernet jsou potřeba pro připojení přístrojů a vývojových desek s USB rozhraním v laboratoři k virtuálním pracovním stanicím, které běží na fyzickém serveru v serverovně. Požadavky na převodníky byly následující:

   - Převodník musí korektně spolupracovat s vývojovými deskami Freescale, Xilinx a s měřicími přístroji Agilent.
   - K jednomu převodníku musí být možné připojit více USB zařízení, a to i z více laboratorních pracovišť – z hlediska pořizovacích nákladů je ideální stav takový, aby pro jednu učebnu stačil jeden převodník.
   - SW ovladač pro pracovní stanici, která využívá některé USB zařízení připojené k převodníku, nesmí umožňovat ovládání USB zařízení připojených k jiné pracovní stanici.

   V rámci projektu VLAM jsme vyzkoušeli všechny HW převodníky, které jsou na trhu (viz roční zprávy projektu VLAM za rok 2010 [8] a 2011 [9]), bohužel žádný z nich nevyhověl všem našim požadavkům. Jediné produkty, které splnily všechny požadavky, byly SW ovladače „USB over Ethernet“ firem Kernelpro a FabulaTech LLP. Na straně laboratoře jsme tyto drivery instalovali do levného úsporného miniaturního PC Alix 3D3, které má USB i Ethernet rozhraní a lze na něm provozovat jak standardní OS Windows XP, tak Linux x86. Cena jedné licence driveru Kernelpro pro 5 USB zařízení je 179 USD, spolu s cenou Alix 3D3 se tedy celková cena jednoho převodníku pohybuje kolem 6000 Kč. Vzhledem k tomu, že žádné jiné funkční řešení jsme zatím nenašli, je tato cena stále přijatelná.

4. Webové kamery: V prototypech VLAM laboratoře jsme používali běžné webové kamery s Ethernet rozhraním. Konkrétně se jedná o dnes již nevyráběný typ Axis 207, jeho současnou náhradu Axis M1011 a cenově zajímavý model TP-LINK TL-SC3171.
   Z hlediska obrazu Axis 207 vykazuje lepší kvalitu obrazu než novější Axis M1011, který má více šumu a jen fixní fokus. Kamera TP-LINK poskytuje překvapivě dobré výsledky, její výhodou pro nepřetržitý provoz v laboratoři je výkonný přísvit nočního vidění a elektromechanicky ovládaný IR filtr. Pro uživatele s OS Linux má tato kamera nevýhodu v tom, že její webová aplikace nepodporuje streamování MPEG4 v Linuxu, takže uživatelé tohoto OS jsou odkázáni pouze na sledování neplynulého sledu JPEG obrázků nebo musí použít nějakou externí aplikaci s podporou protokolu RTSP a MPEG4.

5. Ovládání napájení 230V po TCP/IP: Vzdálené ovládání napájení jednotlivých laboratorních pracovišť je řešeno pomocí zařízení AP7920, což je PDU (Power Distribution Unit) firmy APC s osmi ovladatelnými zásuvkami, určené do rackových rozvaděčů šířky 19“. Pro laboratoře s menším počtem pracovišť lze s výhodou použít zařízení NETIO se čtyřmi zásuvkami, které jsme také úspěšně nasadili.

6. Rackové rozvaděče a kabeláž: Z hlediska integrace infrastruktury VLAM do fyzické laboratoře představuje materiál kabelů a rozvaděčů velmi malou část pořizovacích nákladů. Např. cena kabelů a rozvaděče pro učebnu s dvanácti pracovišti nepřekročí 5000 Kč. Vytvoření projektu a jeho realizace je ovšem časově náročnější než integrace ostatních komponent. Příklady projektů kabeláže pro 2 prototypy VLAM laboratoří jsou uvedeny v ročních zprávách projektu VLAM za rok 2010 [8] a 2011 [9]).

4.2/ Podporované vývojové platformy

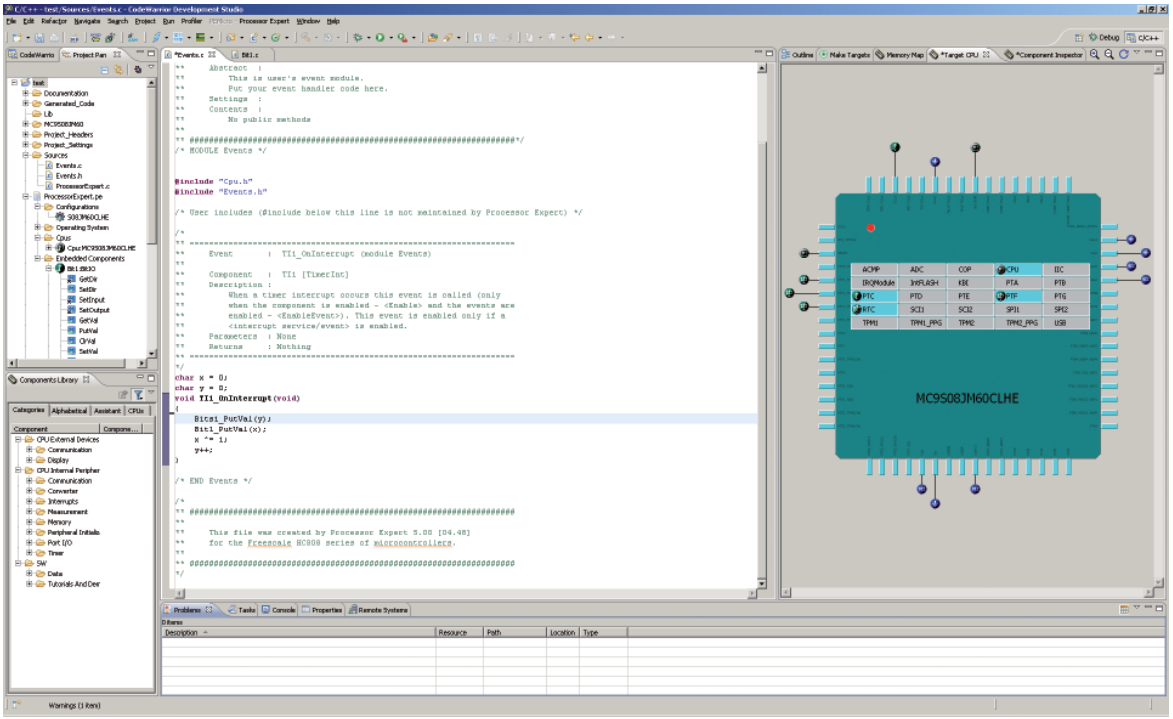
Fyzické pracoviště je vybaveno několika vývojovými přípravky tak, aby pokrylo výuku nejen v kurzech zaměřených na návrh vestavěných systémů, ale také návrh číslicových systémů a hardware-software codesign. Zatímco v případě vestavěných systémů je výuka zaměřena výhradně na práci s mikrokontroléry s důrazem na pochopení základních principů a technik, v případě číslicových systémů je kladen důraz na schopnost orientovat se v moderním návrhu číslicových systémů s využitím rekonfigurovatelných hradlových polí FPGA. Získané znalosti z implementace jednoduchých kombinačních a sekvenčních obvodů studenti využívají při tvorbě komplexního projektu obsahujícího konečný automat a řadu periferií (LCD displej, klávesnice). Nabyté znalosti z obou oblastí jsou zužitkovány v následujícím semestru, kdy se studenti věnují technice hardware-software codesign. Stejnojmenný kurz obsahuje projekt (typicky filtrace zvukového či audio signálu), jehož cílem je demonstrovat výhody akcelerace části úlohy uvnitř FPGA. Kromě vývojových platforem obsahuje každé pracoviště dvoukanálový analogový osciloskop a generátor analogového signálu. Oba přístroje vybaveny digitálním rozhraním.

4.2.1/ Vývojová platforma Flexis

Vývojová platforma Flexis (viz Obr. 16) byla navržena začátkem roku 2010 na Fakultě informačních technologií a v témže roce byla úspěšně nasazena do výuky. Během návrhu byly jednotlivé komponenty voleny především s ohledem na potřeby výuky vestavěných systémů a periferních zařízení. Snahou bylo renovovat doposud používaný a již poměrně zastaralý přípravek s cílem vytvořit a nabídnout studentům platformu reflektující soudobé moderní technologie. Výsledkem je flexibilní vývojový přípravek obsahující pestrou škálu běžně dostupných moderních periferií. Flexibilita přípravku spočívá v možnosti snadné a pohodlné výměny rozšiřujícího a procesorového modulu. V současné době jsou k dispozici dva typy procesorových modulů: (1) modul osazený osmibitovým mikrokontrolérem z produktové řady FLEXIS firmy Freescale a modul osazený šestnáctibitovým mikrokontrolérem stejné řady. Při dodržení rozmístění signálů v propojovacím konektoru je však možné použít modul obsahující mikrokontrolér jiné produktové řady či jiného výrobce. K dispozici jsou také dva typy rozšiřujících modulů – modul grafického LCD displeje s rezistivní dotykovou vrstvou (viz Obr. 16 vpravo dole) a modul obsahující nepájivé kontaktní pole používaný pro prototypování (viz Obr. 16 vpravo nahoře).

Obr. 17: Vývojová platforma Flexis (vlevo), procesorové (uprostřed) a rozšiřující (vpravo) moduly

Vývojové nástroje pro obvody od firmy Freescale jsou k dispozici jak v prostředí Windows, tak i Linux. K práci s kitem se používá integrované vývojové prostředí CodeWarrior (viz Obr. 18), které kromě vývoje a překladu aplikace dovoluje též ladění aplikace, což je nepostradatelný prvek zejména pro začínající vývojáře. CodeWarrior je v omezené verzi (omezení spočívá v maximální dovolené velikosti kódu) poskytován zdarma, pro výuku jsou zakoupeny plnohodnotné licence a využíván licenční server. Nedílnou součástí vývojového prostředí CodeWarrior je modul označovaný jako Processor Expert, který umožňuje pohodlnou konfiguraci periferií bez nutnosti detailního rozboru příslušného katalogového listu. Řada úloh je koncipována tak, aby využívala modul Processor Expert. Studenti tak mají možnost srovnat vývoj aplikace bez a s využitím tohoto nástroje.



Obr. 18: Vývojové prostředí CodeWarrior. Pravý a levý panel obsahuje akce spojené s modulem ProcessorExpert.

Vývojový kit je navržen tak, aby nebyly zapotřebí žádné další přípravky, a to nejen pro účely programování, ale i ladění cílové aplikace. Toho je dosaženo integrací volně dostupného programátoru OpenSourceBDM [24] určeného pro mikrokontroléry Freescale rodiny S08 (8-bit) a ColdFire V1 (32-bit). Programátor je vybaven standardním USB rozhraním a poskytuje následující funkcionalitu: programování a verifikace paměti Flash uchovávající firmware, ladění programu (spuštění, zastavení, krokování, body zastavení, přístup do paměti a k registrům) a možnost napájení přípravku přes BDM port. Samozřejmostí je podpora vývojového prostředí CodeWarrior. Další možností, kterou platforma nabízí, je využití externího BDM programátoru. Právě tento přístup je používán v prostředí laboratoří, kde je každé pracoviště vybaveno programátorem Cyclone PRO (viz Obr. 19) firmy P&E Microcomputer Systems [29]. Důvodem k tomuto kroku byla skutečnost, že tyto programátory byly používány již s předchozím vývojovým přípravkem. Mimo to programátor poskytuje dva typy rozhraní, které mohou být provozovány současně – standardní USB a ethernetové rozhraní. Právě ethernetové rozhraní je výhodou z pohledu realizace virtuální laboratoře, neboť odpadá nutnost používat nákladné převodníky z USB na Ethernet, a připojení přípravku se tím pádem redukuje na vytvoření virtuálního LAN okruhu a nastavení IP adresy.

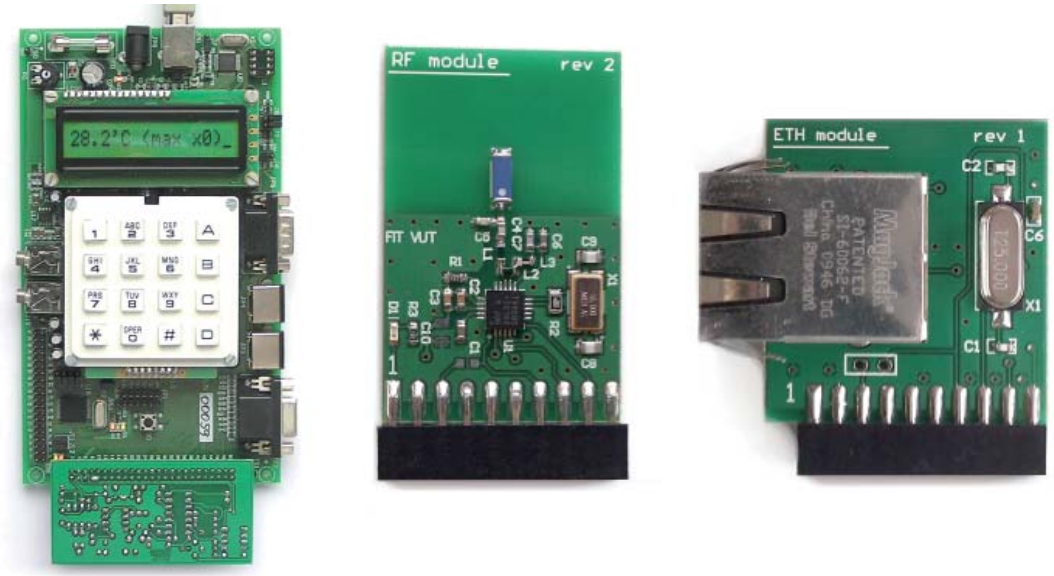
Základním stavebním prvkem platformy je procesorový modul, jehož porty jsou připojeny k dostupným periferiím. Nejjednodušší úlohou, která je řešena v rámci laboratoří, je práce se vstupy a výstupy. K tomuto účelu slouží čtyři mechanická tlačítka s LED podsvícením. Dále jsou zastoupeny úlohy seznamující studenty s principy datové komunikace a komunikačními protokoly, které využívají následujících periferií vývojové platformy: asynchronní sériové rozhraní RS232 (UART), rozhraní pro SD kartu (protokol SPI), dotykovou klá-

vesnici a posuvník (protokol I2C) a rozhraní USB pro připojení mikrokontroléru k PC jakožto zařízení USB slave. Pro demonstraci využití A/D modulu a PWM modulu se používá potenciometr a alfanumerický LCD displej s RGB podsvícením, jehož intenzitu a zabarvení lze řídit pomocí PWM.

Obr. 19: Programátor Cyclone PRO od firmy P&E Microcomputer Systems

4.2.2/ Vývojová platforma FITkit

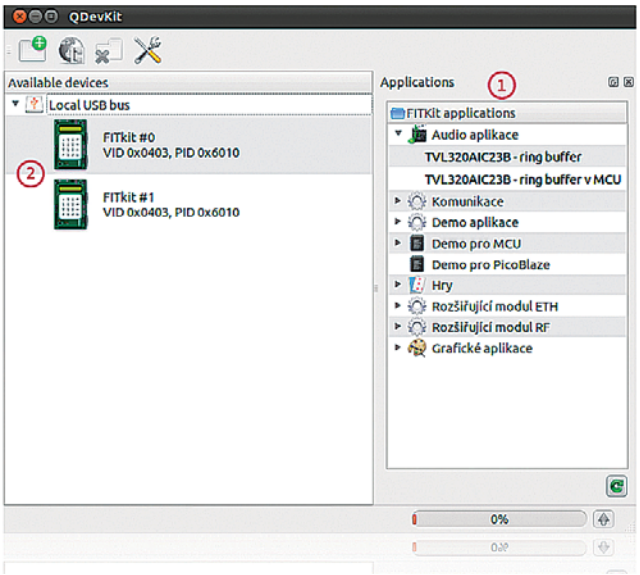
FITkit je svým konceptem jedinečná výuková platforma vyvinutá na Fakultě informačních technologií v roce 2006, která kombinuje programovatelné hradlové pole FPGA s nízkopříkonovým mikrokontrolérem rodiny MSP430 firmy Texas Instruments a řadou základních periferií (viz [41]). Napájení, stejně tak jako komunikaci a programování, zajišťuje jediné USB rozhraní. Vývojová platforma je vybavena pestrou škálou běžně dostupných periferií, které umožňují studentům pochopit principy komunikace a zpracování dat s využitím programovatelných hradlových polí. Kit (viz Obr. 20) je osazen následujícími periferiemi: dvouřádkový znakový LCD displej, maticová klávesnice 8x8, rozhraní RS232 pro asynchronní sériovou komunikaci, rozhraní PS/2 pro připojení myši a klávesnice, konfigurovatelný kodek pro zpracování zvukového signálu a rozhraní VGA umožňující generovat výstup na LCD monitor. FITkit obsahuje dva široké konektory, které slouží k připojení rozšiřujících modulů. V současné době je k dispozici FITkit ve verzi 2 a existují tři rozšiřující moduly – modul pro komunikaci po síti Ethernet, modul pro bezdrátovou komunikaci v bezlicenčním pásmu 2,4 GHz a PWM modul osazený LED diodami a audio konektorem, který slouží k prvnímu seznámení se s návrhem hardware.



Obr. 20: Vývojová platforma FITkit (verze 1) s rozšiřujícím modulem pro přesné měření teploty, další rozšiřující moduly pro FITkit (vpravo)

Programovatelné součástky byly voleny s ohledem na možnost pracovat s přípravkem výhradně pomocí volně dostupných nástrojů, a to nejen v prostředí OS Windows, ale i Linux. Pro generování konfiguračního řetězce (angl. *bitstream*) pro FPGA se používá volně dostupná verze produktu Xilinx ISE Webpack vyvíjená firmou Xilinx. Jedná se o profesionální vývojové prostředí, které nabízí studentům možnost si projít celým procesem vývoje hardware od návrhu až po vygenerování konfiguračního bitstreamu sloužícího k naprogramování FPGA. V současné době vývojové prostředí integruje plnohodnotný událostně řízený simulátor s grafickou nadstavbou, který umožňuje odladit chyby vzniklé v průběhu tvorby modelu hardware. S mikrokontrolérem je možné pracovat pomocí volně dostupného překladače na bázi GCC dostupného pod kódovým označením MSP430-GCC. Součástí toolchainu je také GDB server umožňující ladění aplikací. S ohledem na dostupnost použitých součástek a nenáročnost na programování lze FITkit využívat také jako prototypovací platformu. Pomocí přípravku má student možnost realizovat řešený problém ve formě prototypu a následně na základě získaných zkušeností navrhnout vlastní hardware, který obsahuje pouze nezbytné periferie.

Podobně jako předchozí kit je i FITkit navržen tak, aby nevyžadoval kromě USB připojení další externí prvky. Rozhraní USB zajišťuje tři úlohy: naprogramování mikrokontroléru a FPGA, komunikaci s naprogramovanou aplikací a napájení celého přípravku. Použitý dvoukanálový převodník umožňuje současně komunikovat s aplikací běžící uvnitř mikrokontroléru a hardware, který je implementován uvnitř FPGA. Zatímco pro komunikaci s mikrokontrolérem je používána asynchronní sériová linka, na straně FPGA lze volit z několika různě efektivních synchronních i asynchronních přenosových režimů.

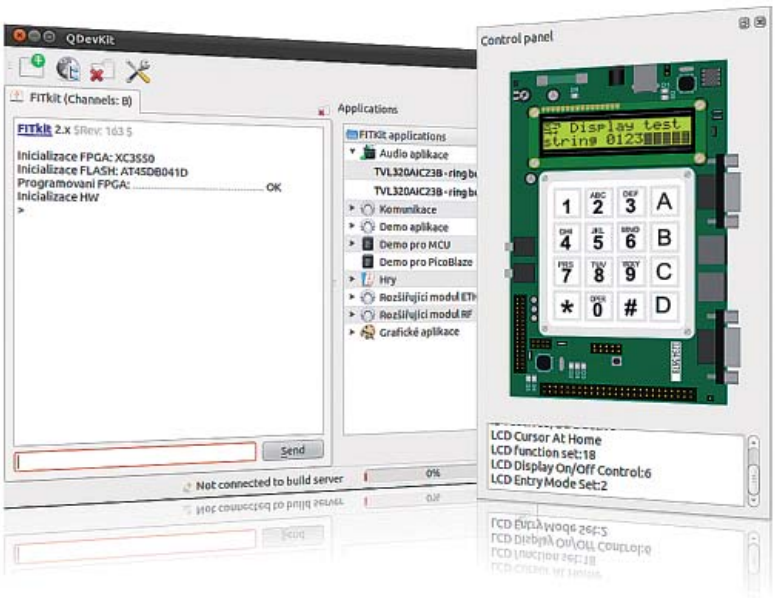


Obr. 21: Ukázka vzhledu aplikace QDevKit.
(1) Seznam úloh nacházejících se v repozitáři, (2) seznam připojených zařízení.

Ačkoliv je možné programování provádět pomocí libovolné terminálové aplikace (programování mikrokontroléru probíhá přes standardní sériovou linku, programování FPGA zajišťuje bootloader, který je nahrán do mikrokontroléru), byla pro práci s kitem vytvořena aplikace QDevkit (viz Obr. 21), jejímž cílem je zpříjemnit práci s přípravkem, konkrétně programování platformy, správu aplikací, umožnit snadný vývoj s využitím nástrojů Xilinx (simulace, syntéza, editace kódu bez nutnosti ručně tvořit projektové soubory) a především sjednotit prostředí mezi různými platformami (Linux, Windows). Aplikace navíc obsahuje prostředky pro rychlé programování mikrokontroléru, které rapidně snižují dobu potřebnou k jeho přeprogramování (aplikace msp430-bsl vyžaduje desetkrát více času). Pro komunikaci s FITkitem byla vytvořena multiplatformní knihovna libkitclient, která umožňuje komunikaci s čipy firmy FTDI. Navržená knihovna poskytuje uživateli jednotné rozhraní napříč operačními systémy i implementačními jazyky. Knihovna je dostupná pro jazyk C/C++ a jako modul i pro Python, kde umožňuje rychlé prototypování vlastních aplikací využívajících přípravek FITkit. Součástí aplikace QDevKit je správa zdrojových souborů a demonstračních aplikací, které jsou uloženy

ve veřejném SVN repozitáři. Uživatel je vždy informován o případné změně, která nastala ve zdrojových souborech, čímž byla odstraněna nutnost řešit problémy spojené s neaktuálností knihoven apod. V současné době je k dispozici více než 50 demonstračních aplikací, které seznamují s obsluhou jednotlivých periferií. Repozitář zdrojových souborů dnes obsahuje aplikace od jednoduchých ukázkových až po komplexní, jako je webový server nebo hardware pro rasterizaci trojúhelníků.

Jelikož přípravek FITkit nabízí z pohledu virtualizace větší možnosti v porovnání s předchozí platformou, byla v rámci projektu VLAM navrhnuta technika dovolující virtualizovat základní periferie platformy FITkit. Navržený přístup využívá přítomnosti programovatelného hradlového pole a skutečnosti, že veškeré periferie jsou připojeny k FPGA čipu. Koncept použitý při realizaci FITkitu dovoluje poměrně snadno realizovat virtualizační platformu využívající virtualizačního subsystému, který je přikompilován k vyvíjené aplikaci. Tímto přístupem sice ztrácíme možnost pracovat s prostředky, které jsou alokovány pro podporu virtualizace (např. bloková paměť, jeden komunikační kanál, logika uvnitř FPGA apod.), avšak získáváme flexibilní virtualizační platformu dovolující virtualizovat libovolnou periferii bez nutnosti fyzicky zasahovat do přípravku. Subsystém zajišťující virtualizaci je napojen na sledované periferie, tzn. na rozhraní LCD displeje, rozhraní klávesnice, LED diody apod., a neovlivňuje tedy nikterak činnost zbytku systému. Výjimku tvoří klávesnice, kdy je žádoucí, abychom měli možnost klávesnici ovládat vzdáleně. Abychom mohli periferie sledovat, je virtualizační komponenta napojena na volný kanál USB převodníku, pomocí něhož jsou periferie zpřístupněny v obslužné aplikaci. Vzhledem k tomu, že použitý převodník neumožňuje iniciovat přenos na straně FPGA a datové přenosy skrze USB jsou vždy iniciovány ze strany počítače, nevyhneme se neustálému dotazování na stav komponent. Abychom však redukovali množství dat, je virtualizační subsystém vybaven schopností detekovat a uchovávat pouze změny. Pro účely virtualizace je navržen modul pro aplikaci QDevKit (viz Obr. 22), který zobrazuje aktuální stav FITkitu (tzn. obsah LCD displeje a stav LED diod) a pomocí kterého je možné ovládat klávesnici, aniž bychom k ní museli mít fyzický přístup.



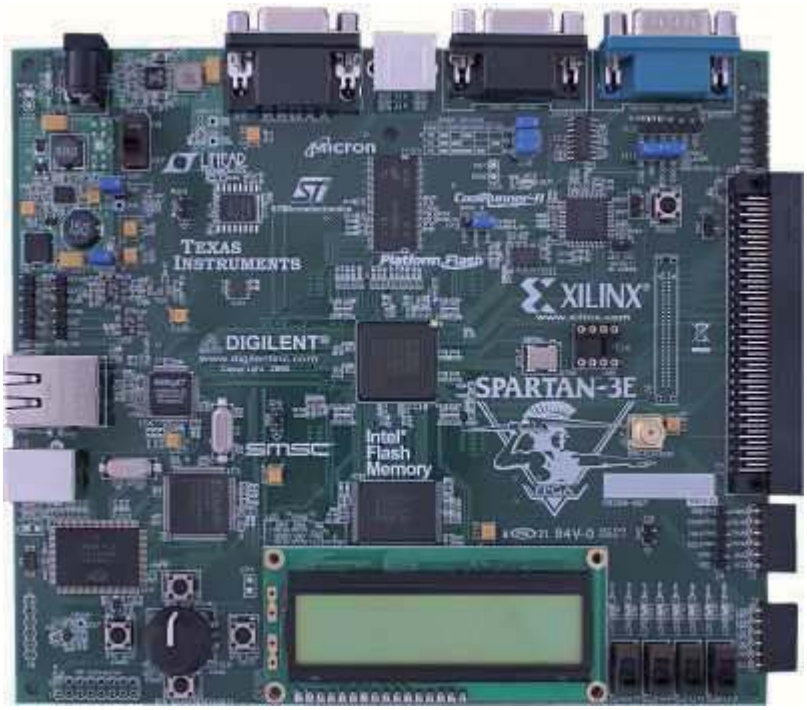
Obr. 22: Ukázka vzhledu modulu pro podporu virtualizace v aplikaci QDevKit. Otevřený terminál (vlevo), panel obsahující interaktivní FITkit (vpravo).

K virtuálnímu pracovišti je vývojový přípravek připojen skrze převodník USB2Ethernet převádějící rozhraní USB na rozhraní Ethernet a naopak, který je realizován s využitím miniPC platformy Alix. Na převodník je připojen USB konektor z vývojového kitu, dále USB konektor VGA grabberu, který zajišťuje přenos obrazu generovaného na VGA port a zvukový vstup a výstup.

4.2.3/ Digilent Spartan 3E-1600

Vývojová deska Digilent Spartan 3E-1600 (viz Obr. 23) je velmi podobná desce Digilent Spartan 3E Starter Board, liší se především větším osazeným FPGA typu XC3S1600E. Na obou těchto deskách je možné v FPGA realizovat CPU MicroBlaze a díky velkým externím pamětem (Flash 128Mbit, 256Mbit DDR SDRAM) dokonce provozovat Linux, čehož využívají některé laboratorní příklady, které byly v rámci projektu VLAM vyvinuty.

Pro interakci s uživatelem deska obsahuje LCD modul, tlačítka, přepínače, pro pokročilejší úlohy pak konektory VGA, PS/2, 2x RS-232, RJ-45 (Ethernet), SMA pro vstup vysokofrekvenčních hodin, 100-pin Hirose FX2 a 3x 6ti pinové konektory typu Pmod. USB konektor na desce slouží pouze pro JTAG programovací logiku.

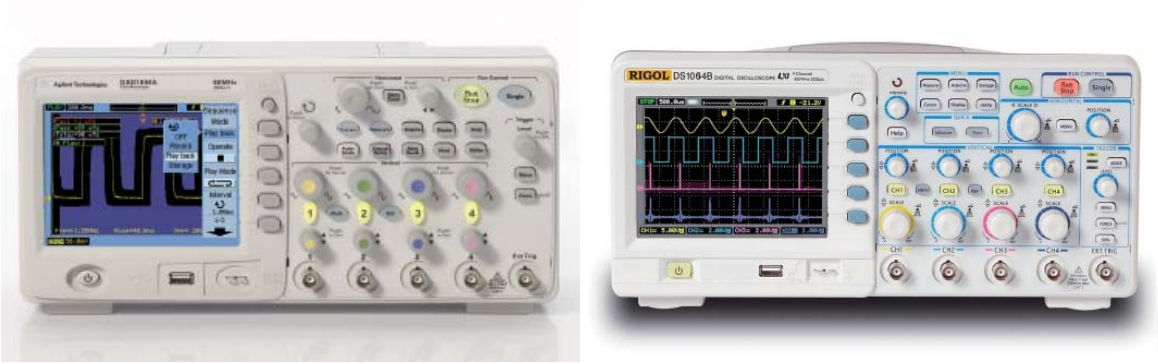


Obr. 23: Vývojová deska Digilent Spartan 3E-1600

4.3/ Měřicí vybavení

Všechna pracoviště jsou vybavena digitálními čtyřkanálovými osciloskopy se šířkou pásma 60 MHz a vzorkováním 2 Gsa/s (dvě miliardy vzorků za vteřinu) značky Agilent nebo Rigol, konkrétně se jedná o produkty Agilent DSO1004A a Rigol DS1064B (viz Obr. 24). Zatímco osciloskop značky Agilent umožňuje propojení s počítačem pouze pomocí rozhraní USB, osciloskop Rigol poskytuje kromě USB konektivity také možnost připojení do standardní sítě LAN, čehož se s výhodou využívá pro realizaci propojení s virtuálním pracovištěm. Pracoviště osazená osciloskopy Agilent jsou k příslušnému virtuálnímu pracovišti připojena skrze převodník USB2Ethernet.

Osciloskopy jsou osazeny 5,7“ barevným LCD displejem s rozlišením 320x240 bodů. Přístroje jsou vybaveny pamětí vzorků umožňující zaznamenávat sledovaný průběh a jsou schopny pracovat se standardními i pokročilými spouštěcími mechanismy (angl. *triggers*) sensitivními na hranu, hladinu, šířku pulzu a statický, či alternující vzorek. Mimo napětí jsou schopny měřit až dvacet dalších veličin, jako je frekvence, amplituda, napětí špička-špička apod. Rozhraní USB podporuje připojení externího paměťového prvku, na který je možné ukládat naměřená data.



Obr. 24: Osciloskop Agilent DSO1004A a Rigol DS1064B

Kromě osciloskopu jsou pracoviště vybavena také jednokanálovými funkčními generátory analogových průběhů tak, aby si studenti byli schopni snadno ověřit například korektní práci s D/A převodníky, časovači, komunikačními periferiemi apod. Podobně jako v předchozím případě jsou použity přístroje značek Agilent nebo Rigol, konkrétně se jedná o produkty Agilent DSO1004A a Rigol DG2041A (viz Obr. 25). Oba generátory umožňují propojení s počítačem pomocí rozhraní USB nebo standardního síťového rozhraní LAN.

Generátory jsou schopny generovat kromě několika běžných průběhů (stejnosměrné napětí, sinus, obdélník, trojúhelník, pila, šum atd.) také libovolný uživatelsky definovaný signál, a sice se vzorkovací frekvencí až 50 MSa/s, případně 100 MSa/s u modelu Rigol. Frekvence generovaného signálu se může pohybovat od 1 μHz až po 20 MHz, případně 40 MHz u modelu Rigol. Kromě generování průběhů generátory nabízí možnost využít řadu modulačních technik (např. amplitudová modulace, frekvenční modulace, fázová modulace, pulzně šířková modulace apod.). Amplitudu lze nastavit od 20mV až po 10V s rozlišením 14 bitů.

Obr. 25: Generátor Agilent DSO1004A a Generátor Rigol DG 2041A

Pro účely laboratoře VLAM byl vytvořen nástroj umožňující komunikovat s měřicími přístroji prostřednictvím rozhraní USB a LAN, který poskytuje jednotné ovládací rozhraní bez ohledu na konkrétní typ měřicího přístroje. S využitím tohoto nástroje je uživatel schopen vzdáleně (např. ve virtuálním pracovišti) nastavit parametry generovaných průběhů, provést měření a získané výsledky uložit do svého počítače v rastrové i vektorové podobě. V případě lokální práce s měřicími přístroji (tzn. práce v laboratoři) se využívá USB rozhraní, které je připojeno k fyzickému pracovišti, a dovoluje tak ovládat přístroje z počítače nacházejícího se v bezprostřední blízkosti.

5/ Softwarové vybavení virtuálních pracovišť

Tato kapitola navazuje na popis softwarové architektury virtuální laboratoře a popisuje softwarové prostředky, které bylo potřeba zajistit pro bezproblémový chod a používání virtuálních pracovišť uživateli. Po analýze požadavků následovala rešerše existujících nástrojů a jejich zhodnocení pro účely pracoviště s danou hardwarovou konfigurací. Nástroje, které bylo možné využít, byly na virtuální pracoviště nainstalovány a patřičně nakonfigurovány. Některé softwarové nástroje bylo třeba upravit, nebo dokonce implementovat vlastními silami.

Software obsažený na virtuálním pracovišti:

- operační systém Microsoft Windows XP (Service Pack 3) CZ,
- základní bezpečnostní software (Microsoft Essentials atd.),
- základní uživatelský software (Internet Explorer 8, Adobe Acrobat Reader 10, Java Runtime Environment, .NET atd.),
- ovladače hardware na pracovišti (Agilent, Rigol atd.),
- vývojářský software (Freescale CodeWarrior, Xilinx ISE Webpack, Eclipse s VLAM IDE, QDevKit, Vlaxicon),
- podpůrné uživatelské a vývojářské programy (PSPad, IrfanView, PDFCreator, WinRar, Putty, WinSCP).

V následujícím textu se detailněji seznámíme s volbou operačního systému pro virtuální pracoviště, tvorbou a vlastnostmi integrovaného vývojového prostředí VLAM a nakonec tvorbou a vlastnostmi nástroje Vlaxicon.

5.1/ Operační systém

Vzhledem k tomu, že většina vývojových nástrojů používaných při práci s HW prostředky laboratoře je vytvořena pro OS Windows, byla volba OS pro VPS omezena pouze na výběr vhodné verze systému MS Windows. Kritéria pro výběr byla následující:

- OS musí být hardwarově nenáročný,
- OS musí umožnit práci se všemi SW nástroji prostřednictvím vzdálené plochy/publikace aplikace,
- licence OS nesmí omezovat jeho použití pro velký, dynamicky se měnící, počet uživatelů (studentů).

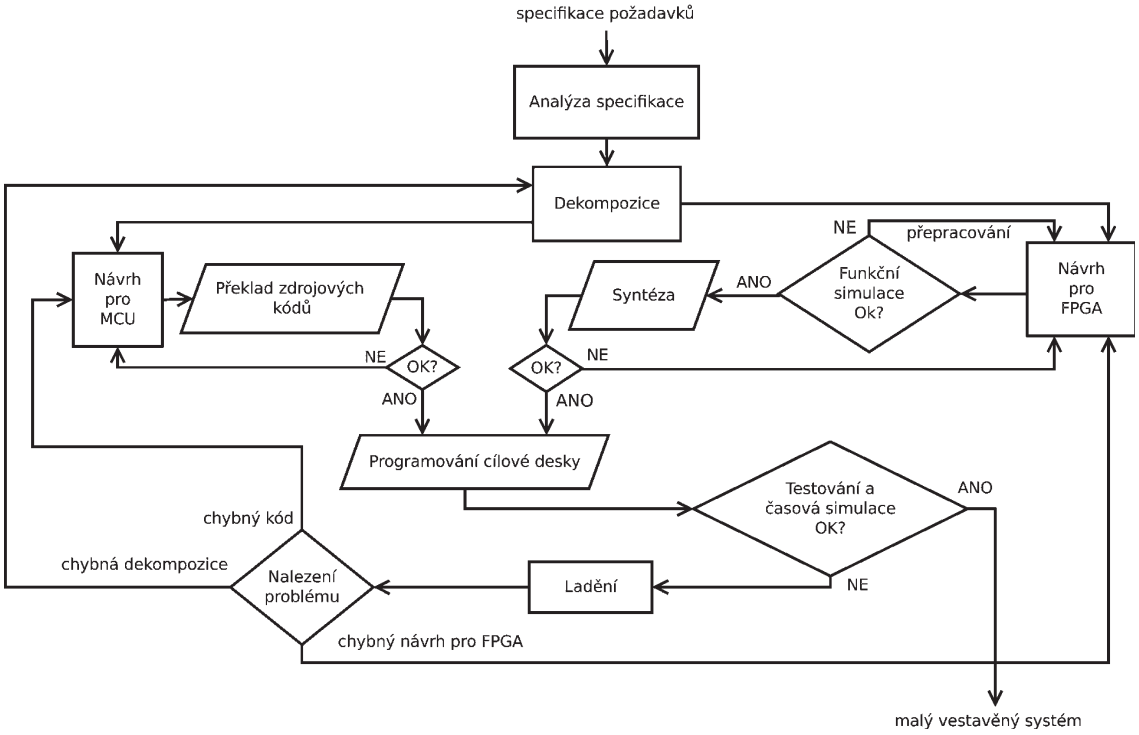
Na základě výše uvedených kriterií byl jako vhodný operační systém VPS zvolen systém MS Windows XP (Service Pack 3).

Z hlediska konfigurace OS se jedná o standardní instalaci s dílčími úpravami, mezi něž patří změna komunikačního portu protokolu RDP dle nastavení VPS v portálu VlamGateway, povolení portu na Firewallu

OS Windows a vhodného nastavení identifikace VPS v síti tak, aby nedocházelo ke kolizím s ostatními VPS. Vzhledem k použité virtualizační technologii (KVM) není nutná žádná dodatečná úprava OS nezbytná pro spolupráci s hostujícím hypervizorem. Jedinou podmínkou je podpora a korektní nastavení technologie ACPI zajišťující vypnutí virtuálního stroje na podnět hypervizoru. Podrobná specifikace všech instalačních kroků a nastavení je uvedena v dokumentu „Poznámky k instalaci virtuální pracovní stanice“ [5].

5.2/ Integrované vývojové prostředí VLAM

V současné době je vývoj vestavěných systémů (viz Obr. 26) prováděn v komplexních integrovaných vývojových prostředích (IDE), jako například CodeWarrior firmy Freescale nebo EDK od firmy Xilinx. Studenti začínající s těmito prostředími jsou většinou zahlceni detailním popisem obecných postupů v těchto prostředích pro vytvoření jednoduchých příkladů. Největší nevýhodou je, že není dostatek času vysvětlovat důvod většiny kroků, takže student často slepě postupuje podle návodu, aniž by chápal, proč některé kroky činí. Toto je hlavní motivací pro návrh a implementaci nového vývojového prostředí vhodného pro studenty začínající s vestavěnými zařízeními. Toto prostředí není tak obecné ani efektivní pro vývoj složitých vestavěných systémů, ale pohodlným způsobem by mělo provést studenta základními kroky vývoje vestavěného systému na jednoduchých školních příkladech.



Obr. 26: Postup návrhu s využitím souběžného návrhu hardware a software

Dalším přínosem našeho nového IDE označovaného jako *VLAM IDE* bude také integrace dalších vývojových nástrojů, které dosud musel student ovládat buď z příkazové řádky, nebo aplikací s nejednotným grafickým uživatelským rozhraním.

Přestože všechny zdrojové texty pro různé programovací jazyky využité při vývoji vestavěných systémů (jazyk C, jazyk symbolických adres, VHDL) lze editovat v libovolném editoru čistého textu (např. Poznámkový blok, PSPad či VIM), tak v dnešních moderních vývojových nástrojích je uživatel zvyklý na podporu i této části vývoje. Především se jedná o zvýrazňování syntaxe, upozorňování na lexikální i syntaktické (a často i séman-

tické) chyby nebo našeptávání celého jména identifikátoru, který je použitelný v daném kontextu. V případě programování pro platformu Freescale mají studenti k dispozici CodeWarrior IDE, které nabízí integrovaný editor zdrojových kódů v různých podporovaných jazycích. Návrh obvodů pro FPGA je poté nutné provádět ručně v jazyce pro popis hardware (většinou VHDL). Jako alternativa pro studenty mohou sloužit nástroje ISE Webpack firmy Xilinx, které jsou v omezených edicích stejně jako CodeWarrior k dispozici zdarma. Xilinx ISE nabízí komfortní editaci zdrojového kódu a integraci všech návrhových etap (syntéza, mapování, routování, simulace, generování modelu atd.). Bohužel jeho použití pro studentskou platformu FITkit je příliš těžkopádné. Na simulaci a verifikaci lze využít ModelSim (samostatná aplikace) nebo Xilinx ISIM (integrována do Xilinx ISE). V závislosti na použité architektuře pak studenti používají i překladače pro různé procesory (např. MSP430GCC, PicoBlaze C Compiler).

Abychom studentům zjednodušili kompilaci a syntézu vytvořeného návrhu, byl vyvinut nástroj *QDevKit* s jednoduchým grafickým uživatelským rozhraním a se zaměřením na vývoj pro platformu FITkit. Aplikace dále umožňuje spravovat katalog demonstračních úloh, které lze pohodlně zkompilovat a naprogramovat do platformy FITkit. Některé části a vlastnosti tohoto nástroje budou též integrovány do VLAM IDE.

5.2.1/ Návrh a architektura vývojového prostředí

Představme si požadavky kladené na moderní uživatelské rozhraní vývojových nástrojů, které mají sloužit pro návrh a implementaci malých vestavěných systémů s důrazem na souběžný návrh hardware a software. Základními požadavky jsou uživatelská přívětivost, intuitivnost a využívání efektivních nástrojů s dostatečnou flexibilitou jejich konfigurace. Pokročilejší požadavky lze identifikovat na základě etap vývoje, které musí vývojové prostředí podporovat ať už po stránce funkčnosti nebo odpovídajícího grafického uživatelského rozhraní:

1. Návrh hardware (obvodů) pro realizaci v čipech FPGA na dané vývojářské platformě.
2. Editace zdrojového kódu pro cílový procesor.
3. Programování vývojářské platformy.
4. Simulace, ladění a verifikace vytvořeného vestavěného systému.

Dalším cenným zdrojem inspirace jsou různé komplexní profesionální nástroje, jako například komerční Xilinx EDK (angl. *Embedded Development Kit*) určený pro zkušené vývojáře. VLAM IDE nemá ambice být takto obecným a přitom efektivním nástrojem, ale zaměřuje se na studenty s důrazem demonstrování hlavních etap a problémů vývoje.

Implementace a integrace jednotlivých zásuvných modulů (angl. *plug-in*) VLAM IDE pro splnění těchto požadavků jsou diskutovány dále.

Jako softwarová platforma pro vývoj navrženého vývojového prostředí byla zvolena platforma Eclipse [12], která nabízí vývoj v jazyce Java s mnoha rozšiřitelnými a flexibilními rámci, nástroji i běhovými prostředími pro vytváření, distribuci a spravování softwarových a také hardwarových produktů. Eclipse je navíc multiplatformní řešení podporované na obou hlavních typech operačních systémů (Microsoft Windows, Linux). Hlavní výhodou Eclipse je jeho rozšiřitelnost a flexibilita. Pomocí systému zásuvných modulů není problém do této platformy integrovat požadovanou funkčnost buď pomocí existujících modulů třetích stran (např. editory zdrojového kódu, správce zdrojů, pohledy na projekt podporující proces vývoje atd.), nebo modulů námi vytvořených na míru.

5.2.2/ Integrace nástrojů do vývojového prostředí

Po představení obecné architektury celého prostředí bude popsán způsob integrace jeho jednotlivých modulů a komponent.

Jelikož studenti inklinují k preferování softwarové části souběžného návrhu, tak jednou z motivací při vytváření modulů VLAM IDE bylo, aby byla posílena dekompozice systému do hardwarové složky. Za tímto účelem vznikl modul implementující grafický komponentní editor pro návrhy hardware, který nabízí uživatelsky přívětivé a intuitivní ovládání. Modul primárně slouží pro popis hardwarové části systému navrženého studentem s cílovou platformou v čipu FPGA (např. na deskách FITkit nebo Xilinx). Zadní část modulu pak využívá překladový systém z aplikace QDevKit, který pro syntézu používá zadní část aplikace Xilinx ISE Webpack. Datová vrstva modulu využívá rámec Eclipse Modeling Framework pro správu modelu návrhu. Prezentační vrstva potom stojí na rámci Graphical Editing Framework.

Jelikož u začínajícího vývojáře (studenta) nelze předpokládat detailní znalost cílové platformy, tak je třeba, aby měl tyto informace pohodlně k dispozici a aby si je každý nástroj byl schopen automaticky nebo poloautomaticky převzít z centrální báze znalostí, která zahrnuje popis hardwarových komponent i parametry jejich případného propojení či zobrazení v grafickém uživatelském rozhraní. Tyto informace pomohou validovat studentský návrh systému po syntaktické a částečně i sémantické stránce. Dalším mechanismem vyvinutým na podporu začínajícího vývojáře při návrhu propojení potřebných hardwarových komponent je poloautomatický algoritmus na odvození propojení komponent. Tento algoritmus na vstupu od uživatele očekává dvě komponenty, které se má pokusit nejlepším způsobem propojit a výsledná akceptovatelná propojení vrátit jako výsledek. Uživatel potom vybere pro něj nejvhodnější propojení, které případně ještě manuálně upraví do finálního stavu.

Báze znalostí, inferenční algoritmus a komponentní editor návrhů tvoří fundamentální základ celého prostředí VLAM IDE. Poté, co uživatel dokončí deklarativní návrh hardware, je třeba automaticky vygenerovat odpovídající zdrojový kód v jazyce pro popis hardware (VHDL). Podobně jako u navrhovaných propojení mezi komponentami, i zde je možné provést ruční úpravy nebo rozšíření za pomocí integrovaného editoru VHDL (zásuvný modul Eclipse Verilog Editor [14]). Jako další krok je z vygenerovaného a upraveného kódu ve VHDL syntetizován bitový obraz (angl. *bitstream*), který je určen k naprogramování cílové platformy. K tomu slouží integrovaná zadní část nástroje QDevKit využívající překladače a syntézu Xilinx ISE.

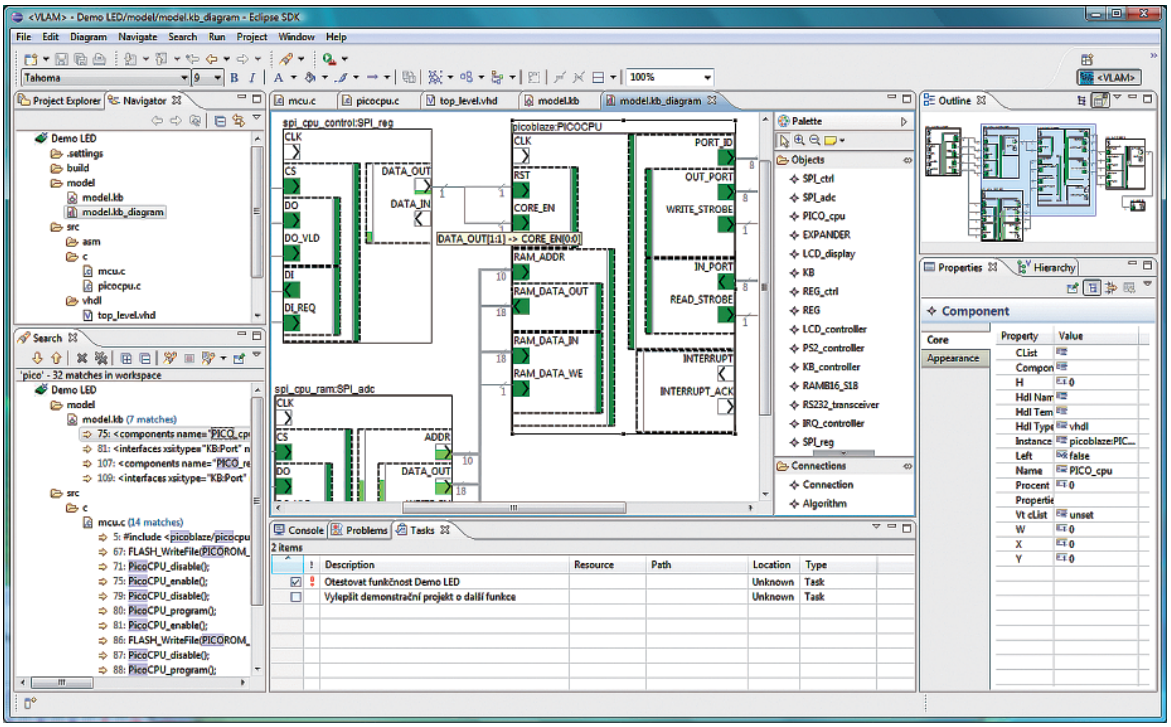
Spolu s tvorbou části systému implementovanou v hardware je nutné navrhnout a implementovat také softwarovou část systému, což uživatel provádí vytvořením řídícího programu pro procesor osazený na cílové platformě. Dle preferované míry abstrakce lze zdrojový kód pro tento procesor vytvářet buď v nízkoúrovňovém jazyce symbolických instrukcí, anebo vysokoúrovňovém jazyce C. Za tímto účelem je třeba integrovat do VLAM IDE překladače pro různé jazyky a pro různé cílové procesory. V rámci uživatelsky přívětivého ovládání nejsou opomenuty ani editory zdrojových kódů pro podporované jazyky.

Poslední krok návrhu zahrnuje kompilaci zdrojových kódů a syntézu hardware podle jeho popisu ve VHDL. Za tímto účelem se využívá rozhraní příkazové řádky nástroje QDevKit, které umožňuje generovat odpovídající soubory pro překladový systém, kdy je třeba správě konvertovat potřebné cesty k souborům se zdrojovými kódy a dalším zdrojům. Pak využijeme utilitu *GNU make*, která zkompiluje a vysyntetizuje všechny potřebné kódy v projektu.

Pro vytvoření vysoce uživatelsky přívětivého vývojového prostředí je třeba integrovat také mnoho dalších nástrojů příkazové řádky, například překladače, spojovací programy, assembler, simulátory, ladicí nástroje, syntézu a programovací nástroje. Využívaje informace z konfigurace projektu v Eclipse lze nastavit a automaticky spustit většinu nástrojů.

5.2.3/ Uživatelské rozhraní

Kromě samotné integrace nástrojů bylo třeba vyvinout i grafické uživatelské rozhraní pro VLAM IDE. Vzhledem k současným trendům je využito prostředí Eclipse, kde jsou v rámci nové perspektivy VLAM poskytovány různé nástroje a pohledy napomáhající při vývoji malých vestavěných systémů nejen studentům. Pohled je většinou reprezentován oknem, které je možné v perspektivě přesouvat, měnit jeho velikost, skrývat nebo maximalizovat dle uživatelových přání. Na Obr. 27 lze vidět základní uživatelské rozhraní obsahující titulek, menu, nástrojovou lištu, stavový řádek a několik speciálních panelů (angl. *Views*): (1) horní prostřední je komponentní editor skládající se z palety s komponentami a propojeními a návrhové plochy, (2) vpravo nahoře je celkový náhled na diagram, (3) vpravo dole je panel s vlastnostmi vybrané entity a (4) vlevo nahoře vidíme navigátor zobrazující zdroje uživatelského projektu, jako například modely diagramy, zdrojové kódy a generované binární soubory a logovací soubory.



Obr. 27: Ukázka vývojového prostředí VLAM IDE

V následujících sekcích této kapitoly budou detailně popsány nejdůležitější moduly vývojového prostředí VLAM IDE.

5.2.4/ Komponentní editor pro návrh hardware

Komponentní editor je grafický editor, který umožňuje na pracovní plochu (tzv. *návrhové plátno* či *plocha*) rozmístit různé typy hardwarových komponent (z báze znalostí) a ty následně propojit pomocí konektorů neboli propojení. Komponenty se vybírají z palety entit a v pomocném dialogovém okně jsou dle potřeby upřesněny jejich generické parametry (tzv. generiky). Na panelu nástrojů vybíráme také entitu konektoru nutnou pro propojení dvou vybraných komponent.

Komponentní editor (prostřední horní panel na Obr. 27) je zásuvný modul prostředí Eclipse, který zajišťuje dvě hlavní funkcionality: (1) grafické modelování hardwarových diagramů a (2) uživatelské rozhraní pro propojovací inferenční algoritmus (viz níže).

Implementace editoru je založena na rámci Eclipse Modeling Framework (EMF), který umožňuje pohodlnou definici modelů s různými reprezentacemi. Pro zajištění vizualizace modelu použijeme rámec Graphical Modeling Framework (GMF), který je nadstavbou Graphical Editing Framework (GEF, zajišťující podporu vytváření editorů zadaného modelu) a EMF.

Model, respektive *konfigurace*, může být chápán jako multigraf s komponentami jako vrcholy a propojeními jako hranami. Každá komponenta obsahuje několik portů, které obsahují podporty. Tím je v každé komponentě vytvořena hierarchická struktura portů.

Vývojové prostředí umožňuje dva druhy zobrazení modelu. Prvním je zobrazení diagramu reprezentovaného množinou grafických elementů. Druhou možností je hierarchická vizualizace modelu reprezentovaná jako stromová struktura. V tomto případě má uživatel možnost editovat veškerá data, nejen ta graficky zobrazená. V obou případech je možnost přidávat či mazat komponenty, vzájemně je propojovat a měnit jejich parametry.

Pro zachování intuitivního zobrazení zobrazujeme v editoru data v několika úrovních detailů. Využíváme různých typů zobrazení, jako jsou bublinové nápovědy (angl. *tooltip*), vlastní grafické komponenty a barvy.

Komponenty a jejich propojení pomocí konektorů obsahuje řadu informací a vlastností, které je třeba zprostředkovat uživateli. Dále si ukážeme několik přístupů zobrazení těchto informací.

Nejjednodušším způsobem zobrazení těchto dat je využití tabulky v okně vlastností. Toto okno je nativní součástí prostředí Eclipse a zobrazuje dostupné vlastnosti komponent a jejich hodnoty. Tento způsob zobrazení se zpřístupní při výběru konkrétní komponenty či konektoru. Většinou se jedná o výběr myší (kliknutí levým tlačítkem myši na danou entitu). Tzv. *Property viewer* se automaticky obnoví a zobrazí aktuální informace, které přísluší zvolenému objektu. Hodnoty lze měnit a sdružovat do různých skupin zobrazovaných v oddělených záložkách. Tento způsob zobrazení dat je vhodný, pokud se jedná o nestrukturovaná data.

Po aktivaci grafické entity (např. dvojklikem myši) se vykoná uživatelsky předdefinovaná akce přiřazená zvolené entitě. V případě portů se jedná o jejich rozbalení/zabalení. Pro komponenty je také možné otevřít specializované dialogové okno pro zadávání či editaci generických parametrů této komponenty.

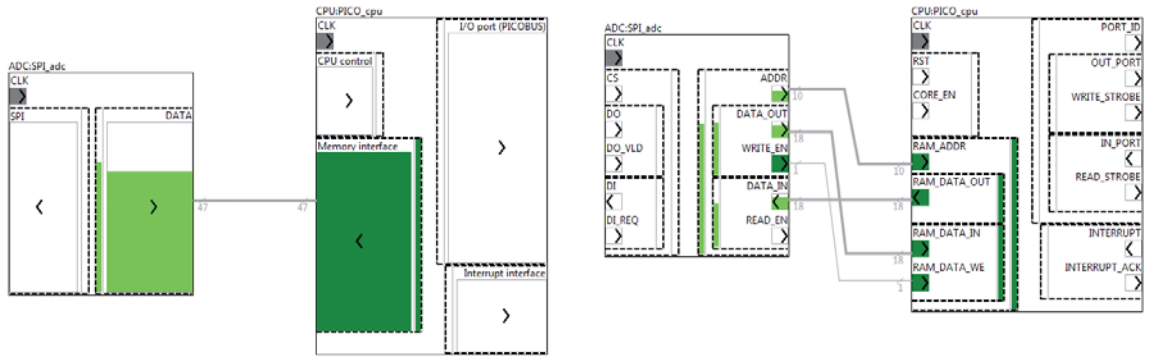
Třetí používaný postup projekce informací je zaměřen především na data textového charakteru. Některé informace je nutné zobrazit přímo na návrhové ploše editoru. Jedná se především o informace nutné pro rychlou a přesnou orientaci v návrhu. Jsou to například jméno komponenty, jejích portů a podportů. Tyto důležité informace jsou zobrazovány pomocí statických návěští přímo na návrhovém plátně, s patřičnou velikostí fontu. Další užitečná data se vizualizují pomocí bublinové nápovědy. Jedná se o malá dynamicky zobrazovaná okénka (tzv. *bubliny*) se statickým textem, která se zviditelní pouze při přejetí kurzoru myši nad příslušnou grafickou entitou. Bublinová nápověda slouží pro sdělení několika málo krátkých textových informací. Například v našich diagramech využíváme bubliny dvou typů. První typ zobrazuje informace o propojení a obsahuje informace o zdrojovém a cílovém portu, což ilustruje Obr. 27 (žlutá bublina s textem „DATA_OUT[1:1] -> CORE_EN[0:0]“). Druhý typ bublin zobrazovaný pro porty informuje o směru, šířce a procentuální obsazenosti portu včetně případného seznamu podportů, obsahuje-li port podporty a byl-li zabalen.

Pro zobrazení výčtových informací lze využít zakódování do různých vizuálních atributů (např. barva, výplň, tloušťka čar apod.). Například virtuální porty jsou ohraničeny přerušovanou čarou. Šířka propojení je graficky vizualizovaná šířkou zobrazované linky. Ukázka tohoto kódování je na Obr. 28. Nedostupný port je vybarven šedým pozadím. Nepropojený port má pozadí bílé. Podle zaplněnosti portu je pozadí v různých odstínech zelené (částečně zaplněné – světle zelená, plně obsazené – tmavě zelená). Pro snadnější zapamatování a dostatečnou rozlišitelnost barevných kódů je používáme v omezeném množství.

Oblast vyjadřující místo pro napojení konektoru je vizualizovaná jako obdélník. Do něj pak vyznačujeme směr toku signálů v tomto portu (tj. vstupní, výstupní či vstupně-výstupní port).

Pro lepší orientaci v grafické reprezentaci modelu umožňuje editor přibližování a oddalování. Abychom neztratili přehled o celém zapojení, obsahuje prostředí nástroj zobrazující miniaturu celého modelu se zvýrazněnou částí, jež je aktuálně viditelná. Tímto nástrojem můžeme zvýrazněnou částí posouvat a editor následně aktualizuje svůj výběr.

Pokud chceme sledovat model jako celek v původní velikosti, postupně (zejména v rozsáhlejších návrzích) ztrácíme přehled. Prostor nutný pro zobrazení komponenty byl tedy také minimalizován. Proto editor využívá možnosti rozbalování a zabalování portů, a tím současně proběhne aktualizace zobrazení jejich propojení.



Obr. 28: Zobrazení propojení komponent na návrhovém plátně s menší úrovní detailů (vlevo) a vyšší úrovní detailů (vpravo)

Jak již bylo zmíněno výše, návrhové plátno může být nahlíženo v různých úrovních detailů. Na nejvyšší úrovni (viz levá část Obr. 28) abstrahujeme konfiguraci na jednoduché propojení mezi jednotlivými komponentami pouze na globální úrovni portů (bez uvažování dělení na podporty). Přesněji řečeno, pro každé dvě komponenty, u nichž jsou propojené libovolné porty či podporty, bude znázorněno virtuální propojení mezi jejich globálními porty. V této úrovni jsou více detailní informace skryty. Pro lepší přehlednost jsou komponenty a viditelné porty opatřeny návěštími s jejich jmény.

Pro detailnější úroveň zobrazení je třeba aktivovat některý port nebo konektor. Když například uživatel aktivuje port označený BUS, port se rozbalí a zobrazí se jeho strukturní detaily. Aktivuje-li uživatel konektor, oba nelistové porty na jeho koncích budou rozbaleny. Pro zabalení pak stačí aktivovat odpovídající nadport. Chceme-li vyvolat zabalení celého konektoru, je možné vyvolat tuto akci z kontextového menu pro dané propojení.

Dynamickou úpravou množství zobrazovaných informací při zobrazování se snaží grafické uživatelské rozhraní poskytnout maximální použitelnost a uživatelskou přívětivost, kdy uživatel není přehlcen některými nedůležitými informacemi. Další detaily jsou dosažitelné v dodatečných dialozích a pohledech na vyžádání uživatele.

V tomto duchu si popišme základní interakci s uživatelem v rámci tohoto editoru. Pro přidání nové komponenty na plátno vybere uživatel vhodnou komponentu v paletě (angl. *Palette*) a umístí ji na plátno. Před propojením dvou komponent musí být nachystán (viditelný) zdrojový i cílový port (rozbalený, je-li nutno). Nyní vybereme z palety entitu propojení (konektor), dále pak zdrojový port a následně cílový port. Poté je vytvořeno maximální možné syntakticky validní propojení (viz inferenční algoritmus níže), které může uživatel dále upravovat. Dalšími základními operacemi je odstranění entity nebo změna jejích vlastností přístupných přes kontextové menu nebo patřičný pohled nad vybranou entitou.

Pro rychlý start vývoje je při vytvoření nového projektu ve vývojovém prostředí spuštěn průvodce, který se uživatele postupně dotazuje na základní vlastnosti celého systému (architektura, pracovní frekvence, základní funkce atd.), aby mohl uživateli nabídnout předpřipravené návrhové plátno s komponentami reprezentujícími připojení externích periferií k FPGA čipu. Detailněji viz kapitola 6.2.1.

5.2.4.1/ Inferenční algoritmus

Algoritmus pro odvození (inferenci) propojení hardwarových komponent je určen pro automatizaci takovéhoto propojení v editoru návrhů FPGA. Usnadňuje uživateli vytváření schémat zapojení. Uživatel vybere dvě komponenty k propojení, algoritmus poté navrhne několik nejlepších variant a uživatel z nich nakonec vybere tu nejvhodnější, kterou může ještě ručně upravit.

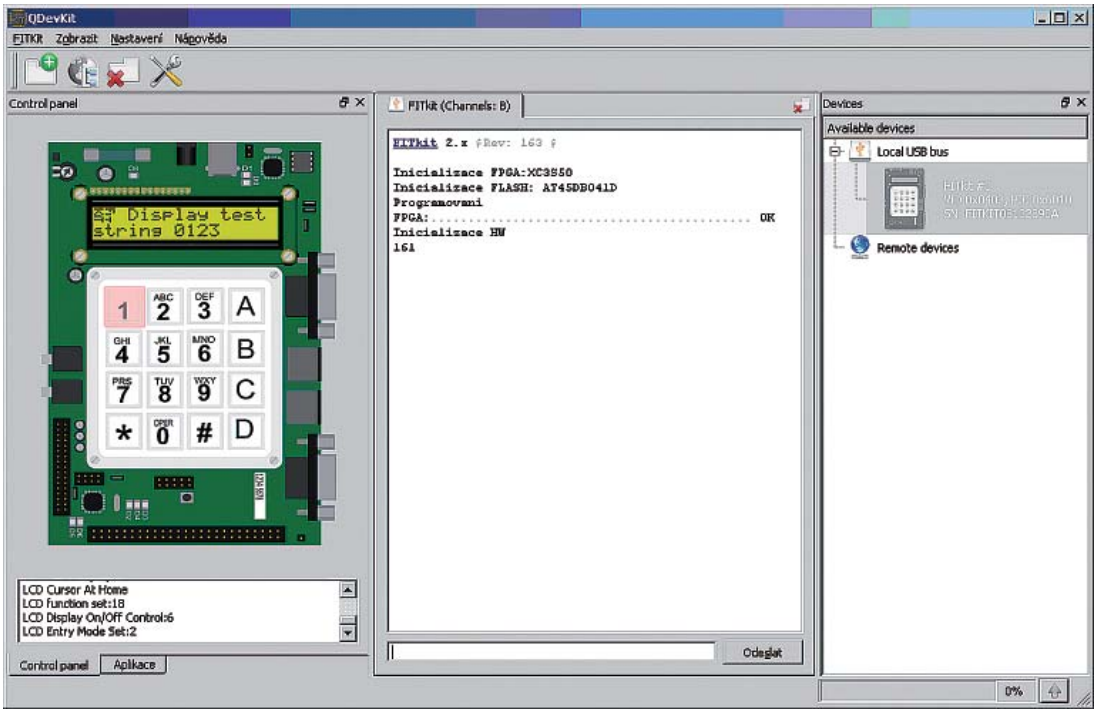
Algoritmus pracuje s popisem portů dvou zadaných komponent, který je uložen v připojené bázi znalostí. Je popsán směr portů, typ (datový, adresový, řídící) a hierarchická stromová struktura portů, kdy jeden port komponenty může obsahovat více podportů.

Problém odvození nejlepšího propojení úzce souvisí s *přiřazovacím problémem* (Assignment Problem) a *bipartitním párováním* (Bipartite Matching) spadajícími do oblasti *síťových algoritmů* [31]. Obě oblasti algoritmů byly za účelem návrhu algoritmu blíže zkoumány, kdy je ve výsledku problém nejlepšího párování ve stromové struktuře portů určen a omezen jako instance *lineárního přiřazovacího problému* (Linear Assignment Problem), pro nějž je na rozdíl od jiných zkoumaných druhů přiřazovacích problémů, znám efektivní algoritmus řešení pracující s polynomiální časovou složitostí [28].

Testování algoritmu proběhlo na šesti aplikacích z výuky a na zadaných dvojicích propojovaných komponent mělo 90% úspěšnost. Neúspěšných 10 % hledání souvisí s omezením z definice lineárního přiřazovacího problému.

5.2.5/ QDevKit

Aplikace QDevKit (viz Obr. 29) vznikla za účelem podpory práce se studenty oblíbenou vývojářskou platformou FITkit, která byla představena v kapitole 4.2.2. Některé její části je možné využít samostatně, a tak byly integrovány do VLAM IDE.



Obr. 29: QDevKit

QDevKit umožňuje spravovat repozitář demonstračních úloh a příkladů, které slouží pro výuku v předmětech se zaměřením na vestavěné systémy. Dále umožňuje automatickou konfiguraci a kompilaci včetně programování platformy FITkit, což je využito také ve VLAM IDE pro zajištění lepší podpory platformy FITkit. Navíc je též možnost využít příkazové řádky pro dávkové či automatizované zpracování projektu. Zadní část nástroje QDevKit totiž generuje všechny potřebné skripty pro překlad, ošetřuje závislosti a řeší správnost cest k souborům. Navíc umožňuje QDevKit také rozšiřování pomocí zásuvných modulů (angl. *plug-in*) napsaných v jazyce C++ nebo Python.

Příkladem takového rozšiřujícího modulu je modul pro virtualizaci FITkit (viz levá část Obr. 29 s označením *Control Panel*). Tento modul nabízí vzdálené řízení, správu a monitorování desky FITkit včetně některých periferií. Modul umožňuje sledovat stav LCD displeje nebo vzdáleně stisknout tlačítko na maticové klávesnici. Uprostřed Obr. 29 lze vidět okno terminálu popisující aktuálně prováděné akce. Přes toto okno je také možné zasílat příkazy vývojové desce FITkit. Pravý panel potom vypisuje všechna připojená podporovaná zařízení.

V budoucnu je plánováno, že prostředí VLAM IDE zcela nahradí QDevKit, aby tak skloubil jednoduchost návrhu vlastního projektu spolu s jeho implementací a programováním do cílové platformy.

5.2.6/ PicoBlaze C Compiler

V dnešní době jsou vývojáři zvyklí na tvorbu software ve vysokoúrovňových programovacích jazycích. Nezáleží dokonce, zda se jedná o jednoduchý skript nebo rozsáhlou aplikaci. Navíc v případě vývoje pro programovatelný hardware (např. FPGA) jsou často k dispozici tzv. *soft-core* procesory, které je možné před naprogramováním do FPGA dokonce i modifikovat nebo jich v návrhu využít libovolné množství závislé pouze na velikosti FPGA čipu. Mezi nejmenší soft-core procesory pro FPGA patří procesorové jádro PicoBlaze firmy Xilinx. Před několika lety existoval nedotažený projekt překladače jazyka C pro tento procesor (zkráceně pojmenovaný *PCComp*), který vedl Ital Francesco Poderico. V průběhu projektu byl však tento projekt ukončen, a dokonce stažen z oficiálních webových stránek. To byl impulz pro vývoj kvalitnější verze překladače *PicoBlaze C Compiler* (*PBCC*) vlastními silami. Tato sekce popisuje návrh, implementaci a použití tohoto nového nástroje.

5.2.6.1/ Existující nedokončený překladač PCComp

Pro soft-core procesor PicoBlaze již existuje jednoduchý překladač z jazyka C, nazvaný PCComp (Picoblaze C Compiler). Avšak tento překladač se dále nevyvíjí a jeho poslední verze je z roku 2005. Zároveň trpí zásadními neduhy, které jeho využitelnost značně degradují.

Při generování pracuje na principu zásobníkového kódu, což je vzhledem k absenci zásobníku uvnitř tohoto procesoru a vzhledem k velikosti datové paměti neefektivní. V některých případech je dokonce výsledný kód nefunkční. Zároveň překladač neprovádí žádné optimalizace, což je vzhledem k omezeným zdrojům cílového procesoru zásadní chyba. Dalším omezujícím faktorem je podpora maximálně 16bitového celočíselného datového typu a pouze jednorozměrných polí. Dále není podporována ukazatelová aritmetika, z čehož vyplývá, že parametrem funkce nemůže být ukazatel ani pole.

5.2.6.2/ Architektura a návrh překladače

Na nejvyšší úrovni abstrakce můžeme architekturu překladače rozdělit na dvě části:

1. Přední část (angl. *front-end*) – část závislá na vstupním jazyce.

2. Zadní část (angl. *back-end*) – část závislá na cílové platformě.

U této architektury neprobíhá překlad přímo, ale je rozdělen do dvou kroků. Nejprve je zdrojový text zpracován lexikální a syntaktickou analýzou, se kterou je úzce spjata i sémantická analýza (sémantické kontroly), která následně generuje vnitřní reprezentaci vstupního zdrojového textu (tzv. *mezikód*).

Druhý krok je ve znamení transformace vnitřní reprezentace do kódu pro cílovou platformu. V našem případě máme situaci ulehčenu tím, že není třeba generovat binární kód, ale pouze jazyk symbolických adres, který lze přeložit nástrojem firmy Xilinx (tzv. *assembler*) do binární podoby vhodné pro následující syntézu hardware.

Výhodou této architektury je rozdělení kompilátoru na dvě relativně nezávislé jazykové transformace, přičemž překlad ze zdrojového jazyka do mezikódu je společný pro více cílových platforem a až typem zadní části se určuje cílová platforma. Navíc záměnou přední části, při zachování struktury mezikódu, lze teoreticky docílit podpory pro jiný zdrojový jazyk [2].

Naším cílem bylo vytvořit zadní část překladače pro soft-core procesor Xilinx PicoBlaze-3 za využití již existující přední části překladače jazyka C.

Přední část překladače

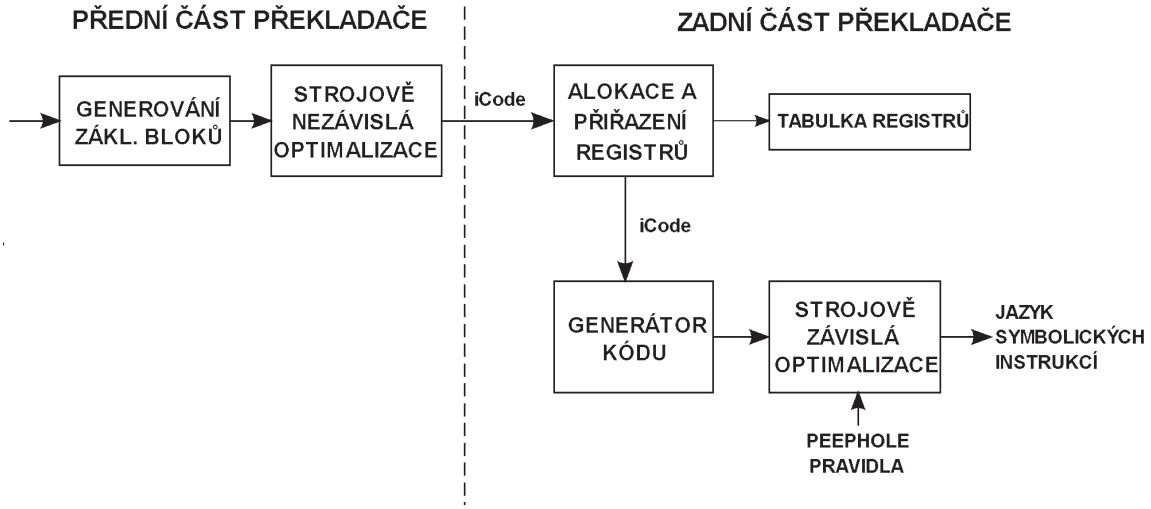
Jako nejvhodnější se jevila volba rámce *Small Device C Compiler* (*SDCC*). Tento rámec pro tvorbu překladače jazyka C byl vyvinut především pro 8bitové procesory a zároveň je distribuován pod licencí *GNU General Public License*. Nad tímto rámcem byla již realizována řada překladačů jazyka C, jako například pro procesory řady PIC, MCS51 nebo Z80. Překlad v SDCC probíhá ve dvou krocích. V prvním kroku je generován mezikód jazyka, který je v SDCC označován jako *iCode*. Struktury reprezentující tento mezikód obsahují všechny důležité údaje nezbytné pro generování cílového kódu (zejména typ operace a seznam operandů). Operand každé instrukce je reprezentován strukturou *operand*, kde jsou definovány příznaky (*global*, *volatile* apod.) a určen typ operandu buď jako symbolický operand (*symbol*), nebo literální hodnota (*value*). U symbolického operandu je navíc důležitá informace o životi odpovídající proměnné, což umožňuje efektivně spravovat registry.

Zadní část překladače

Obr. 30 znázorňuje návrh architektury s detailnějším zaměřením na návrh zadní části našeho překladače včetně navázání na přední část. Pro účely dokumentace naší části implementace dostačuje zmínit, že přední část generuje seznam základních bloků, na které jsou aplikovány různé strojově nezávislé optimalizace.

Prvním blokem zadní části je *Generátor kódu*, který na základě typu instrukce definované mezikódem provede vygenerování odpovídající posloupnosti instrukcí jazyka symbolických instrukcí.

Druhým blokem je *Alokace a přiřazení registrů*, jehož smyslem je udržovat informace o obsazenosti registrů a paměti dat konkrétními proměnnými a na základě těchto poznatků efektivně využívat registry v instrukcích generovaných prvním blokem. Případně při nedostatečném množství registrů se provede jejich odložení do paměti (angl. *register spilling*) tak, aby dopady na výslednou výkonnost byly minimální.



Obr. 30: Architektura překladače PBCC [20]

Posledním blokem zadní části překladače je *Strojově závislá optimalizace*. Tato optimalizace pro zefektivnění cílového kódu je většinou založena na tzv. *peephole optimalizaci* popsané například v knize [2] využívající op-

timalizační pravidla připomínající vyhledávání neoptimálních sekvencí několika instrukcí a jejich náhradou za efektivnější zápis v cílovém jazyce.

Takto popsaný návrh, kdy bloky pro generování kódu a práci s registry spolupracují v rámci jednoho průchodu instrukcemi mezikódu, mohl být zvolen vzhledem k jednoduchosti samotného procesoru PicoBlaze a jeho instrukční sady. Rámec SDCC je však navržen pro komplexnější ač 8bitové procesory, jako např. PIC16, HC08 nebo MCS51 a samotné generování je zde rozděleno do dvou fází. Prvním průchodem se pouze přiřadí registry jednotlivým instrukcím mezikódu (tzv. *pCode*), a až při druhém průchodu se provede vygenerování výstupních instrukcí. Tyto průchody se provádí pro každou funkci jazyka C zvlášť v pořadí shodném s jejich uvedením ve zdrojovém kódu. Při tomto způsobu zpracování ale postrádáme například informace o všech globálních proměnných a znemožňuje to efektivně rozvrhnout datovou paměť, které má procesor PicoBlaze k dispozici velmi omezené množství. Výsledné realizované řešení tedy sice nevyužívá pCode instrukce, ale používá jednodušší strukturu pro ukládání iCode instrukcí (obousměrný vázaný seznam seznamů, které odpovídají jednotlivým funkcím). Samotné generování na základě seznamu iCode instrukcí se provede až po zpracování celého zdrojového textu programu (konečná fáze rámce SDCC nazvaná angl. *glue*), takže při generování cílového kódu máme k dispozici již všechny potřebné informace. Tímto jsme se vyhnuli nutnosti modifikovat přední část překladače. Kvůli tomuto řešení však bylo třeba ještě modifikovat způsob práce s návěštími, která jsou generována během generování sekvencí iCode, ale jejich další zpracování opět probíhá až v konečné fázi SDCC.

Implementace PBCC je oproti dalším procesorům nad SDCC unikátním tím, že kvůli primitivnosti cílové platformy nepoužívá mapování adresových prostorů, což významně komplikuje vstupně/výstupní operace a je třeba navrhnout jejich specifickou implementaci. Procesor PicoBlaze komunikuje se svými periferiemi pomocí sady očíslovaných portů. Vždy je možné zapisovat (instrukce *OUTPUT*) nebo číst (instrukce *INPUT*) pouze jeden z portů, k čemuž slouží dva parametry: (1) číslo portu a (2) data pro zápis nebo cíl/registr pro čtení. Z možných realizací byla zvolena implementace vestavěného pole označeného pseudoproměnnou *__PORT*, která reprezentuje hodnoty všech portů procesoru a pomocí indexace k nim lze přistupovat jak pro čtení, tak pro zápis.

Vzhledem k tomu, že procesor PicoBlaze neobsahuje žádné dedikované instrukce pro násobení, dělení a operaci modulo, bylo nutné tyto operace implementovat pomocí instrukcí sčítání a posunu v případě násobení, a instrukcí odečítání a posunu v případě operací dělení a modulo. Princip těchto algoritmů je popsán například v [30]. Menší komplikací jsou operace se znaménkovými čísly, kdy je třeba před samotnou operací zkontrolovat, zda je operandem záporné číslo a případně jej převést na kladné s tím, že se tato změna poznamená. Po provedení samotné operace se poté přihlédne ke znaménkům vstupních operandů a na základě znaménkových pravidel u násobení a dělení se patřičně upraví znaménko výsledku operace.

Překladač má samozřejmě také nevýhody, jako například neimplementovanou plnou podporu ukazatelové aritmetiky nebo čísel s plovoucí řádovou čárkou, což je ovšem u tak malého kontroléru pochopitelné.

Detailně je celý překladač i jeho implementace popsána v rámci diplomové práce Jakuba Horníka [20] a jeho příspěvku na studentskou soutěž [19].

5.2.7/ Editor jazyka symbolických instrukcí pro procesor PicoBlaze

Editor jazyka symbolických instrukcí (angl. *assembly language*) slouží k podpoře vytváření a editaci zdrojových textů v jazyce symbolických instrukcí pro procesor PicoBlaze. Je navržen jako zásuvný modul do vývojového prostředí Eclipse. Podporovanými dialekty jazyka jsou syntaxe KCPSM3 a syntaxe vývojového prostředí pBlazIDE [42].

Pro vytvoření editoru byl využit open source nástroj Xtext [3], který umožňuje poměrně jednoduchým způsobem na základě gramatiky cílového jazyka vygenerovat jeho editor.

Vytvořený editor je vybaven sémantickým zvýrazňováním syntaxe jazyka symbolických instrukcí, validací zdrojového textu s výčtem chyb typickým pro prostředí Eclipse, kontextovou nápovědou při editaci dokumentu a zobrazením základní sémantické struktury dokumentu v přehledovém okně (angl. *Outline*) prostředí Eclipse.

Zároveň byla do editoru naimplementována i možnost vzájemného převodu obou výše zmíněných dialektů jazyka. Vstupem je aktuálně otevřený soubor daného dialektu, výstupem potom funkčně odpovídající kód převedený do druhého dialektu.

5.2.7.1/ Xtext

Nástroj Xtext [3] je určen k vytváření editorů především pro *doménově specifické jazyky* (*Domain-Specific Language*, DSL), ale také pro *jazyky pro obecné použití* (*General Purpose Language,* GPL). Vygenerovaný editor následně funguje jako zásuvný modul pro vývojové prostředí Eclipse. Xtext poskytuje pro daný jazyk mimo jiné syntaktický analyzátor, nástroj pro formátování kódu, *abstraktní syntaktický strom* (*AST*), serializér, validátor, kompletování kódu, generátor kódu nebo interpret. Všechny vytvářené komponenty jsou založené na *Eclipse Modeling Framework* (EMF), což dále umožňuje jejich integraci s dalšími rámci založenými na EMF, např. *Graphical Modeling Project* (GMF). Xtext využívá Google Guice (Dependency Injection Framework), takže je možné výchozí implementace jednotlivých součástí editoru přehledně nahrazovat vlastními.

5.2.7.2/ Implementace editoru

Základem pro vytvoření nového editoru pomocí nástroje Xtext je zadání *gramatiky* jazyka (obvyklou definicí *terminálů* a *neterminálů*), pro který je editor určen. Při zadávání gramatiky je možné využít dědičnosti, která umožňuje rozšířit již existující gramatiku o nové prvky. Nutností při zadávání gramatiky je definice tzv. *slotů*, které poté slouží ke konstrukci EMF modelu (AST) aktuálně editovaného dokumentu.

Následuje neúplná ukázka gramatiky pro editor jazyka symbolických instrukcí se syntaxí dialektu KCPSM3:

```
grammar pbpsmeditor.PBPsm with org.eclipse.xtext.common.Terminals
generate pBPsm „http://www.PBPsm.pbpsmeditor"

RootPsm:
      ((EOL)*(lines+=Line)((EOL)+ lines+=Line)*(EOL)*)?;
Line:
      (name=ID ':') | (line = (Instruction | Directive ));
Instruction:
      instruction = (... | CallJump |...
CallJump:
      ins = ('CALL' | ,JUMP') (cond = Condition del=',')?(named=[Label...]
      | addr=Const10bit);
Const10bit:
      val=HEXA3;
Label...:
      Line|...
terminal EOL:
      '\n';
terminal HEXA3:
      ('0'..'9'|ALPHAHEX)('0'..'9'|ALPHAHEX)('0'..'9'|ALPHAHEX);
terminal ALPHAHEX:
      ('a'..'f')|('A'..'F');
```

Tato ukázka obsahuje některé specifické konstrukce gramatiky. Např. bylo potřeba vyjádřit tzv. *křížový odkaz* (cross-reference, označuje se hranatou závorkou) mezi názvem *návěští* a jeho použitím (*Line*:*name* ↔ *Label*), kdy poté, co máme v kódu zadáno návěští, chceme se na něj dále odkazovat např. v instrukci JUMP a nechat

editor validovat a kompletovat zadávání kódu uživatelem v souladu s touto vazbou (obdobně i pro pojmenované konstanty a registry). Ukázka dále obsahuje definici *počátečního neterminálu* gramatiky (*RootPsm*), pěti neterminálů a tří terminálů. ID je terminálem dovezeným z knihovny `org.eclipse.xtext.common.Terminals`. Mimo syntaxi KCPSM3 je potřeba definovat i syntaxi pBlazIDE. Za tímto účelem jsme ve výsledku vytvořili editory dva – po řadě PBPsmEditor (aktivované pro příponu souborů *.psm) a PBPsmEditor (*.asm). Gramatika pro PBAsmEditor vznikla pomocí tzv. *mixovaných gramatik* (grammar mixins). Jedná se o schopnost Xtextu odvodit jednu gramatiku z jedné a více již existujících. Syntaxe pBlazIDE totiž zřejmě rozšiřuje sadu direktiv dialektu KCPSM3 a přejmenovává některé jeho instrukce (viz [42]). Výsledné editory (zásuvné moduly) jsou ve vztahu jednosměrné závislosti ve směru PBAsmEditor → PBPsmEditor a následně v tomto smyslu mohou sdílet i kód, který bude zmiňován dále.

Zadáním gramatik jsme v tuto chvíli po spuštění generování editorů dosáhli schopnosti validovat uživatelem zadávaný text, a také kontextově nabízet možné vstupy přes obvyklou zkratku *Ctrl+Mezera*. Pro rozšíření výchozího zvýrazňování syntaxe (v tomto stavu by byly od okolního textu odlišeny pouze klíčová slova a komentáře) je potřeba většího programátorského zásahu.

Pro rozšířené zvýrazňování syntaxe je potřeba definovat vlastní třídu implementující rozhraní `ISemanticHighlightingCalculator`. Ta pro předaný AST vypočítá rozsahy výskytů sledovaných neterminálu, a tak sděluje, které pozice v dokumentu mají být zvýrazněny konkrétním definovaným stylem. Styly vztahující se k jednotlivým skupinám neterminálů se definují implementací rozhraní `IHighlightingConfiguration`. Vytvořené třídy je potřeba následně ručně navázat na implementaci editoru přes Google Guice.

Dalším požadovaným funkčním prvkem editoru je přehledový pohled. Jedná se o sémantickou strukturu editovaného kódu zobrazovanou v prostředí Eclipse. Pro naše účely je vhodné a postačující zobrazit návěští vyskytující se ve zdrojovém kódu. Pro nastavení neterminálů, které se mají v přehledovém pohledu objevit (resp. jejich pojmenování definované nastavením slotu *name*), je potřeba ze zobrazení odfiltrovat nežádoucí neterminály. Vzhledem ke struktuře gramatiky dostačovalo ignorovat všechny potomky neterminálu *Line* a tento samotný nechat zobrazit, protože jeho nastaveným jménem, pokud je definováno (viz ukázka gramatiky), je právě návěští.

Xtext také umožňuje nastavit vlastní pravidla pro validaci (doplněním definice vygenerované třídy PB?smJavaValidator). Toho bylo využito pro kontrolu rozsahu číselných konstant.

5.2.7.3/ Popis editoru

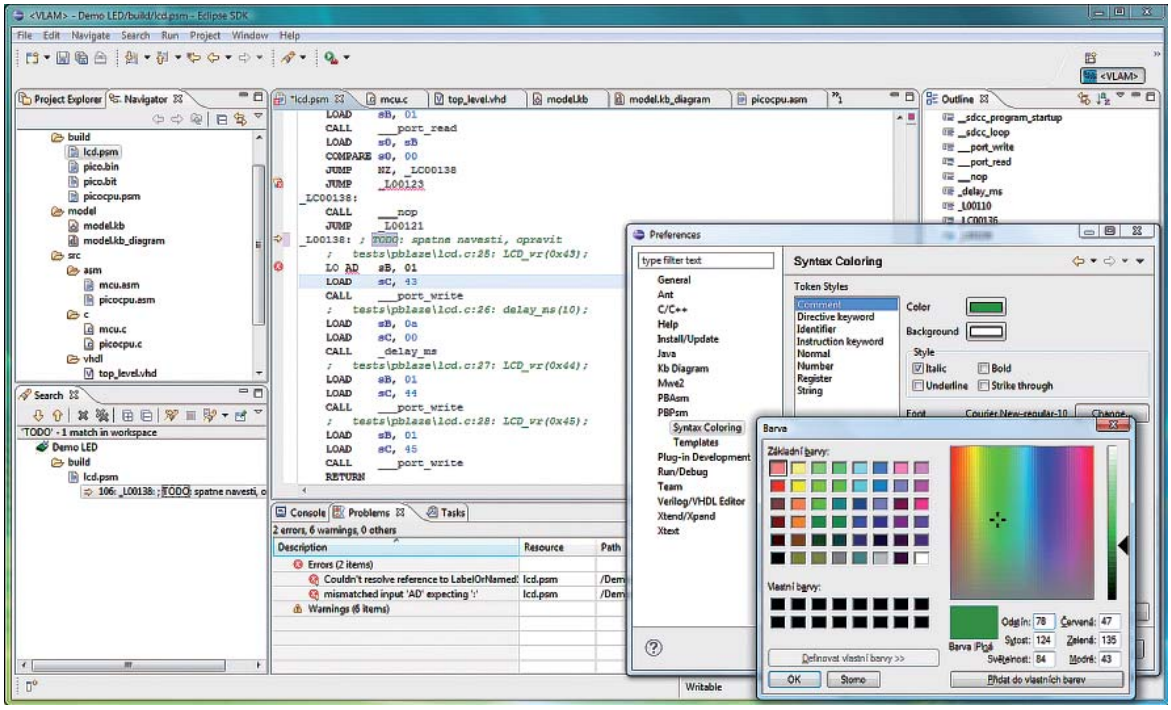
Na Obr. 31 vidíme ukázku editoru s otevřeným souborem v jazyce symbolických instrukcí pro procesor PicoBlaze. Vpravo nalezneme přehledový pohled (*Outline*), vlevo strukturu projektu (*Navigator*), dole je zobrazen seznam syntaktických a případně i některých sémantických chyb (*Problems*). Na dialozích v popředí vpravo dole je demonstrována možnost konfigurace zvýrazňování syntaxe (dialog *Preferences*).

5.2.7.4/ Použití editoru

Editory obou dialektů jazyka symbolických instrukcí pro PicoBlaze jsou distribuovány jako 6 závislých, Xtextem automaticky vygenerovaných zásuvných modulů (`pbasmeditor.jar`, `pbasmeditor.generator.jar`, `pbasmeditor.ui.jar`, `pbpsmeditor.jar`, `pbpsmeditor.generator.jar` a `pbpsmeditor.ui.jar`). Instalace modulů probíhá pro Eclipse obvyklým způsobem, je však potřeba mít nainstalován zásuvný modul Xtext [3].

5.2.7.5/ Shrnutí

Vytvořený editor jazyka symbolických instrukcí je nakonec plně funkční. Zásadní komplikací při implementaci byl ovšem neúplný a místy neaktualizovaný manuál [3], který navíc předpokládá obeznámenost s konceptem rámce EMF a Google Guice. Toto však vyvažuje online komunita vytvořená kolem vývoje zásuvného modulu Xtext a podpora dostupná prostřednictvím Xtext fóra.



Obr. 31: Editor jazyka symbolických instrukcí včetně ukázky konfigurace zvýrazňování syntaxe (na popředí)

5.2.8/ Ostatní zásuvné moduly

Kromě vytvářených zásuvných modulů jsou ve VLAM IDE využívány i moduly třetích stran, které prostředí obohacují o funkčnost požadovanou od moderního vývojového prostředí.

Pro editaci zdrojových kódů VHDL slouží zásuvný modul *Eclipse Verilog editor* [14], který dokáže zpracovávat také jazyk Verilog. Pro oba podporované jazyky nabízí editor kódu se zvýrazňováním syntaxe a přehledem struktury také asistenci při psaní kódu (doplňování identifikátorů či klíčových slov dle několika počátečních písmen). Editor si v prostředí Eclipse asociuje soubory s koncovkou VHD a ty se v něm pak otevírají automaticky.

Pro editaci zdrojových kódů jazyka C byl zvolen zásuvný modul *C Development Tools* [13], což je velmi sofistikovaná úprava celého prostředí Eclipse včetně bohaté podpory pro konfiguraci, překlad a ladění. Modul bylo potřeba doplnit o klíčová slova, která přidává rámec SDCC do standardního jazyka C.

Krom výše zmíněných modulů lze samozřejmě VLAM IDE uživatelsky dále rozšiřovat o další vhodné nekolizní zásuvné moduly pro Eclipse (např. modul pro podporu verzovacího systému pro správu historie i aktuální verze vyvíjeného projektu).

5.2.9/ Instalace

Prostředí VLAM IDE je distribuováno jako balíček obsahující sadu zásuvných modulů pro Eclipse (tzv. *Feature*), dále je nezbytné mít nainstalován software, jehož distribuci nemáme licencovánu, a tak si jeho instalaci musí zajistit sám uživatel (např. ISE WebPack, KCPSM3 Assembler, nástroj make, …). Po instalaci modulu bude vyžádán restart prostředí Eclipse. Při následném spuštění se aktivuje dialog s nejnutnější konfigurací nástrojů třetích stran (např. cesta k nástroji make, implicitní cesta k repozitáři FITkit atd.).

5.2.10/ Shrnutí

Na základě studia problémů studentů při návrhu malého vestavěného systému jsme provedli analýzu, navrhli a implementovali jednoduchý vývojový nástroj VLAM IDE. Předchozí část detailně popisovala návrh grafického uživatelského prostředí i některých dalších implementovaných algoritmů.

Mezi základní vlastnosti VLAM IDE patří vlastnosti s výhodou zděděné od prostředí Eclipse, na kterém staví. VLAM IDE je multiplatformní a snadno rozšiřitelný o zásuvné moduly třetích stran. V budoucnu bychom rádi rozšířili bázi znalostí pro popis pro více platforem a vytvořili část prostředí lépe podporující ladění a simulaci.

5.3/ Vlaxicon

Program Vlaxicon [39], [38] slouží pro ovládání měřicích přístrojů používajících pro komunikaci standardu příkazů SCPI a rozhraní pro připojení USB nebo TCP/IP. Program je distribuován pod licencí GPLv2 pro operační systémy MS Windows a Linux.

Důvodem pro vytvoření programu zahrnutého do projektu VLAM bylo umožnění vzdáleného přístupu k měřicímu přístroji. Výhodné je to především při plnění laboratorních úloh studenty, kdy mohou tyto úlohy provádět přímo z domova nebo jiné, vzdálené lokace.

5.3.1/ Stávající řešení

Program Vlaxicon je navržen jako alternativa k následujícím SW nástrojům:

5.3.1.1/ Agilent VEE Pro

Jedná se o grafický programovací SW firmy Agilent. Jde o vysokoúrovňový programovací jazyk využívající grafické zobrazení obdobné vývojovým diagramům. Prakticky to tedy znamená, že uživatel si pomocí grafických bloků a jejich propojení sestaví sekvenci, která může být vykonána. Nutná je při tom znalost SCPI příkazů, které je potřeba do některých bloků doplnit. Ačkoliv je výrobcem firma Agilent, daný software díky standardům podporuje zařízení téměř všech výrobců. Aplikace je tedy především díky možnosti grafického návrhu programu uživatelsky přívětivá, nicméně má i své nevýhody. Mezi ně patří, že aplikace je pouze pro platformu MS Windows a licence není bezplatná. Uživatel si může stáhnout třicetidenní zkušební verzi a poté, pokud chce program dále používat, si musí pořídit jednu z placených licencí. Pro běh programu je nutná aplikace Agilent IO Libraries Suite, která je prostředkem pro zpřístupnění měřicích přístrojů a jejich případné konfiguraci. Tato aplikace je zdarma ke stažení z internetu.

5.3.1.2/ LabView

Software společnosti National Instruments je taktéž grafickým programovacím prostředím, obdobně jako Agilent VEE Pro. Princip fungování je zde také podobný. Rozdíl je především v tom, že LabView nabízí i distribuci pro jiné operační systémy než Microsoft Windows. Hlavní nevýhodou zde je, že použití software je zpoplatněno poměrně vysokou částkou.

5.3.2/ Ovládání pomocí webového rozhraní

Pokud je měřicí přístroj připojen pomocí LAN, pak je možné se na něj připojit pomocí internetového prohlížeče. Podmínkou v tomto případě je, aby měl přístroj přidělenu IP adresu. Tu může získat třemi způsoby:

- je mu přidělena automaticky prostřednictvím DHCP serveru,
- je mu nastavena pomocí programu „Agilent Connection Expert“, který je součástí Agilent IO Libraries Suite,
- je nastavena přímo na přístroji.

Díky tomu, že ovládací prostředí je uloženo na přístroji, je toto řešení zdarma a je multiplatformní. Nevýhodou zde je, že takto ovládat lze jen zařízení připojená pomocí TCP/IP.

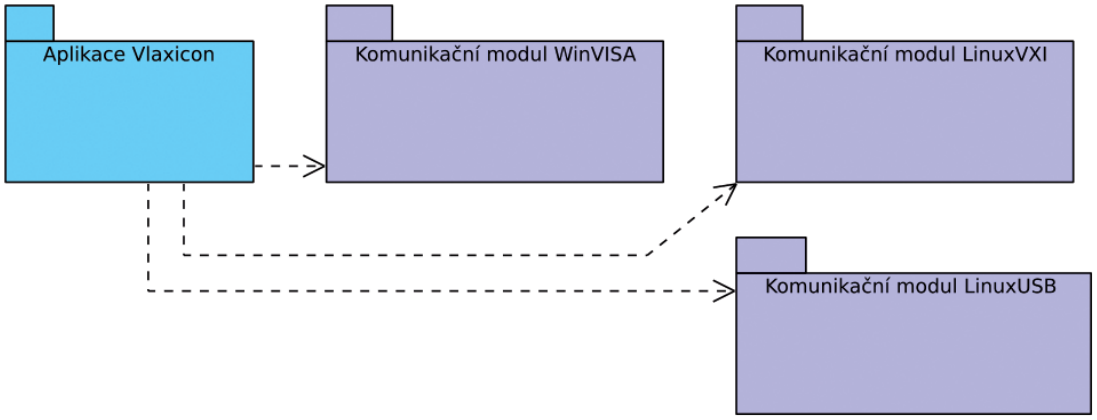
5.3.2.1/ Možnosti pod systémem Linux

Pod systémem Linux je jako grafická prostředí možné použít LabView nebo webového rozhraní. Dalšími dostupnými nástroji jsou jen jednoduché programy typicky dodávané spolu s ovladači pro dané zařízení. Jde o konzolové aplikace, jejichž použití je pro složitější práci obtížné.

5.3.3/ Struktura aplikace

Program Vlaxicon byl vytvořen pomocí programovacího jazyka C++ a multiplatformní knihovny wxWidgets [35]. Pro tvorbu bylo využito také rozšiřujících komponent, které nejsou přímou součástí standardní knihovny, a to komponent wxShapeFramework [7] a wxPlotCtrl [26].

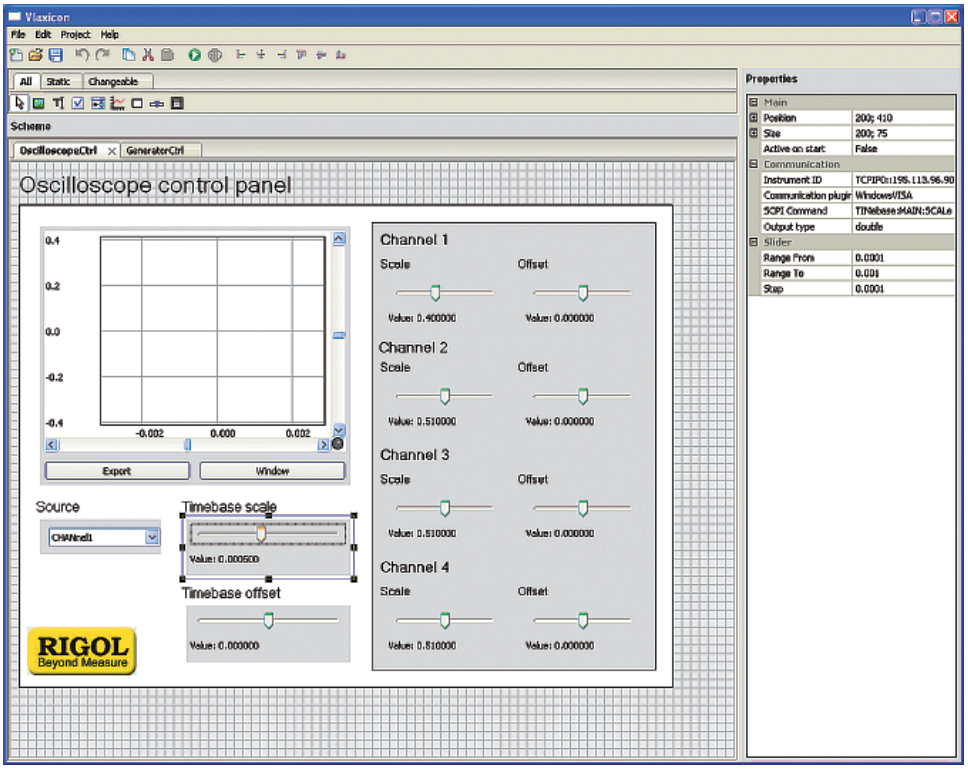
Jedná se modulární aplikaci, kde jednotlivé zásuvné moduly zajišťují komunikaci hlavní aplikace s jednotlivými měřicími přístroji pomocí podporovaných komunikačních rozhraní a protokolů (viz Obr. 32). Pro tvorbu těchto zásuvných modulů bylo pro systém MS Windows využito knihoven VISA, které jsou součástí instalace Agilent IO Libraries Suite a jsou schopny komunikovat jak přes USB, tak TCP/IP. Pro komunikaci pod systémem Linux byly použity ovladače USBTMC [25] (pro USB) a VXI11 [36] (pro TCP/IP).



Obr. 32: Komponenty aplikace Vlaxicon

Vlaxicon byl navržen jako MDI (Multiple Document Interface) aplikace, což znamená, že umožňuje práci s několika projekty najednou. Z hlediska terminologie aplikace Vlaxicon rozumíme projektem formulář s návrhem uživatelského rozhraní ovládaného měřicího přístroje. Každý ovládací prvek projektu může být asociován s libovolným fyzickým měřicím přístrojem a může generovat libovolný podporovaný SCPI příkaz odeslaný měřicímu přístroji prostřednictvím jednoho z dostupných komunikačních zásuvných modulů.

Hlavní okno aplikace na Obr. 33 sestává z následujících částí:



Obr. 33: Hlavní okno aplikace Vlaxicon v editačním módu (projekt pro ovládání osciloskopu)

V horní části okna je menu a nástrojová lišta pro ovládání základních funkcí programu, jako je uložení a načtení projektu, editace formuláře, popř. spouštění/zastavení ovládání přístroje. Níže jsou záložky s ovládacími prvky, které je možno vkládat do panelu projektu. Tento panel je přímo pod záložkami s ovládacími prvky. V pravé části okna je pak tabulka vlastností, která slouží pro nastavení ovládacích prvků.

V současné verzi (v1.2.0) jsou dostupné následující ovládací prvky:

První skupinou prvků jsou statické grafické prvky, jejichž účelem je umožnit definovat přehledné uživatelské rozhraní ovládaného měřicího přístroje. Společnými editovatelnými vlastnostmi těchto prvků jsou poloha a velikost. Patří mezi ně:

- Prvek ***Bitmap*** představuje statickou bitmapu (podporovaným formátem je pouze BMP). Nastavuje se u něj cesta k souboru obrázku, který tento prvek zobrazuje.

- Prvek ***Edit Text*** představuje statický editovatelný text. Jde o prvek, při jehož vložení je vyvolán dialog pro nastavení textu tohoto prvku. Nastavit zde lze vlastnosti textu jako typ písma, jeho barvu, velikost apod.

- Prvek ***Rectangle*** představuje statický obdélník, pomocí kterého lze vizuálně seskupit další prvky projektu. U tohoto prvku je možné nastavit také jeho barvu.

Druhou skupinou ovládacích prvků jsou GUI prvky, které lze interaktivně využívat pro ovládání funkcionality a sledování měřicího přístroje. Společnými editovatelnými vlastnostmi těchto prvků jsou poloha, velikost, unikátní adresa (ID) přístroje a použitý komunikační zásuvný modul. S výjimkou prvku *Plot* je nutné specifikovat příslušný SCPI příkaz. Do této skupiny patří:

- Prvek ***Check Box*** slouží pro nastavování dvoustavových hodnot. Je zde potřeba nastavit hodnotu parametru příkazu pro případ zaškrtnutí (*Parameter if TRUE*) a pro případ nezaškrtnutí (*Parameter if FALSE*) prvku.

- Prvek ***Choice*** slouží jako roletové menu, kde vybraná položka je parametrem SCPI příkazu odeslaného přístroji. U tohoto prvku je potřeba nastavit parametr *Choices*, který představuje jednotlivé položky výběru.

- Prvek ***Plot*** slouží pro zobrazení průběhů veličin získaných z měřicího přístroje (typicky osciloskopu). U tohoto prvku je možné zobrazení grafu v samostatném okně pomocí tlačítka *Window* a uložení zobrazeného grafu do souboru pomocí tlačítka *Export*.

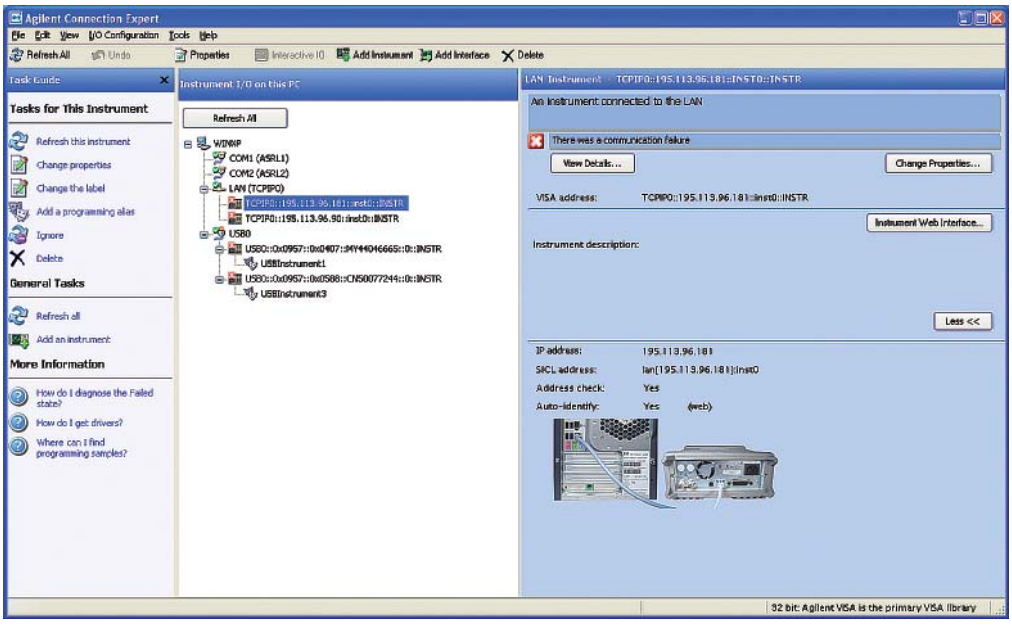
- Prvek ***Slider*** je posuvníkem, jehož hodnota je parametrem SPCI příkazu odeslaného přístroji. U tohoto prvku je třeba nastavit horní (*Range To*) a dolní (*Range From*) mez hodnot a krok (*Step*) posunu.

- Prvek ***Universal*** slouží pro univerzální účely; to znamená, že pomocí něj lze zadávat libovolné příkazy a dotazy ve formě SCPI příkazů. Jde-li o dotaz, bude odpověď přístroje zobrazena v tomto prvku.

Unikátní ID přístroje lze získat dvěma způsoby:

- U přístrojů připojených pomocí protokolu TCP/IP je identifikátorem přístroje jeho IP adresa. Tu lze zjistit/nastavit prostřednictvím ovládacího panelu fyzického přístroje. Tento způsob identifikace je použit například v komunikačním modulu „LinuxVXI“.

- U přístrojů připojených prostřednictvím USB nebo TCP/IP v prostředí OS Windows pomocí komunikačního balíku Agilent IO Libraries Suite je identifikátorem položka „VISA Address“ zobrazená ve vlastnostech připojeného přístroje v aplikaci „Agilent Communication Expert“, jak je vidět na Obr. 34. Tento způsob identifikace je použit v komunikačním modulu „WindowsVISA“.



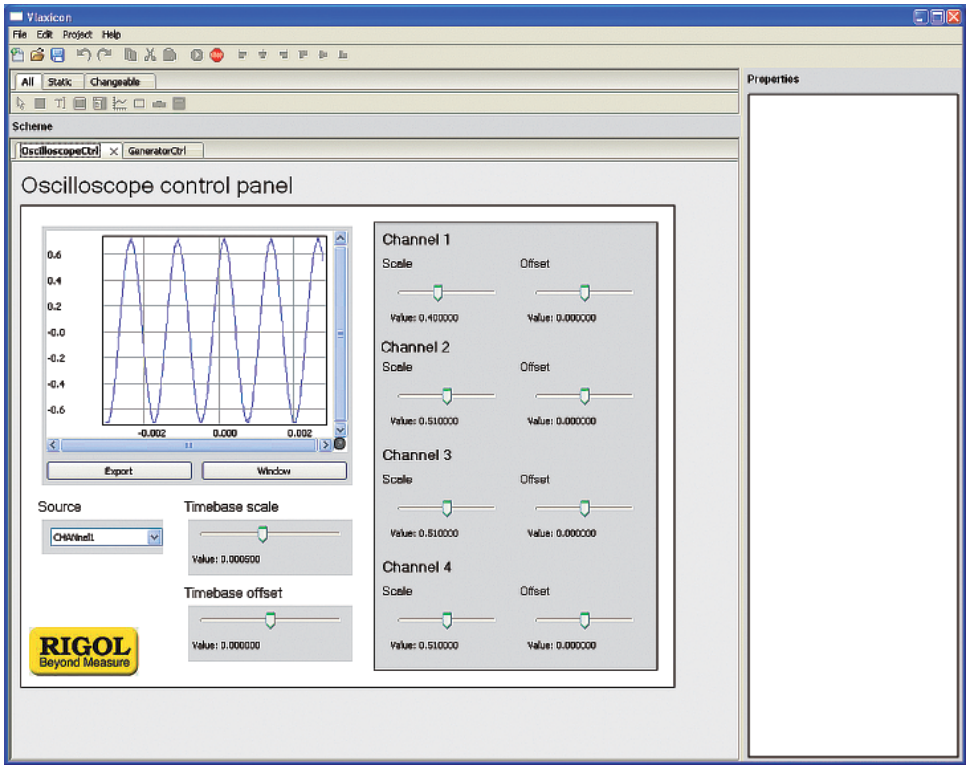
Obr. 34: Aplikace „Agilent Communication Expert“ balíku Agilent IO Libraries Suite

5.3.4/ Použití programu

Chceme-li pomocí aplikace Vlaxicon ovládat měřicí přístroj, musí být tento připojen buďto k PC, na němž je Vlaxicon spuštěn (v případě připojení přístroje pomocí USB rozhraní), nebo musí být dostupný v rámci segmentu lokální sítě, do níž je PC připojeno (v případě připojení pomocí protokolu TCP/IP, popř. technologie „USB over Ethernet“ [23]). Každý připojený měřicí přístroj musí mít přiděleno unikátní ID nebo IP adresu.

Po spuštění aplikace Vlaxicon a po vytvoření nového/otevření existujícího projektu je možné do tohoto projektu pomocí myši, popř. klávesových zkratek pro práci se schránkou, vkládat ovládací prvky popsané v předchozí kapitole. Po označení konkrétního prvku pomocí myši lze editovat jeho vlastnosti v tabulce vlastností zobrazené na pravé straně rámcového okna aplikace. Polohu a velikost ovládacích prvků lze rovněž měnit tažením pomocí myši.

Po nastavení všech ovládacích prvků projektu lze ovládání přístroje spustit tlačítkem *Start* na nástrojové liště hlavního rámcového okna, popřípadě pomocí příslušné položky menu *Project*. Po spuštění projektu jsou dočasně zakázány editace obsažených ovládacích prvků, ty jsou poté aktivovány, čímž je umožněno vzdáleně ovládat měřicí přístroje (viz Obr. 35). Ukončení ovládání přístroje lze provést stiskem tlačítka *Stop* na nástrojové liště, popřípadě analogicky z menu *Project*.



Obr. 35: Vlaxicon s běžícími projekty pro ovládání generátoru a osciloskopu

5.3.4.1/ Menu a nástrojová lišta

Jak menu, tak nástrojová lišta aplikace Vlaxicon obsahují sobě odpovídající volby, pomocí kterých lze přistupovat k základní funkčnosti programu.

Menu *File* slouží pro vytvoření nového projektu (položka *New*), načtení již existujícího projektu (*Open*), uložení stávajícího projektu (*Save*) a ukončení programu (*Exit*). Zároveň obsahuje podmenu se seznamem naposledy otevřených projektů (*Recent files*).

Menu *Edit* obsahuje funkce pro úpravy stávajícího projektu. Mezi těmito funkcemi je krok zpět (*Undo*), krok vpřed (*Redo*), vložení vybraného ovládacího prvku do schránky (*Copy*, *Cut*), vložení prvků ze schránky do aktivního projektu (*Paste*) a funkce pro zarovnání více vybraných prvků (*Align*).

Menu *Project* slouží pro spuštění (*Run*) a zastavení (*Stop*) stávajícího vytvořeného projektu a menu *Help* obsahuje informace o programu (*About*).

5.3.4.2/ Ovládací prvky a jejich nastavení

Ovládací prvek lze do projektu vložit tak, že bude vybrán v záložkách prvků v hlavním rámcovém okně pomocí myši a po kliknutí na plochu projektu bude vložen na příslušné místo. Pokud je zároveň při vložení držena klávesa CTRL, pak zůstane tento výběr aktivní a je možné do projektu vkládat další prvky téhož typu. V opačném případě je po vložení prvku aktivován mód editace.

Prvky vložené do projektu je možné po jejich označení (je povoleno i označení více prvků najednou) mazat stiskem klávesy DEL. S označenými prvky lze také standardně pracovat pomocí systémové schránky.

6/ Příklady použití (laboratorní úlohy)

Pro demonstraci funkčnosti virtuální laboratoře bude v této kapitole popsáno několik jednoduchých případů použití celého systému. Nejprve se zaměříme na využití vývojového prostředí CodeWarrior verze 10.1 a modulu Processor Expert pro práci s vývojovým kitem Flexis. Pak budeme demonstrovat práci s vývojovým kitem FITkit pomocí prostředí VLAM IDE. V poslední podkapitole bude demonstrován pokročilejší studie použití OS Linux ve větších čipech FPGA.

6.1/ Práce s vývojovým kitem Flexis s využitím modulu Processor Expert

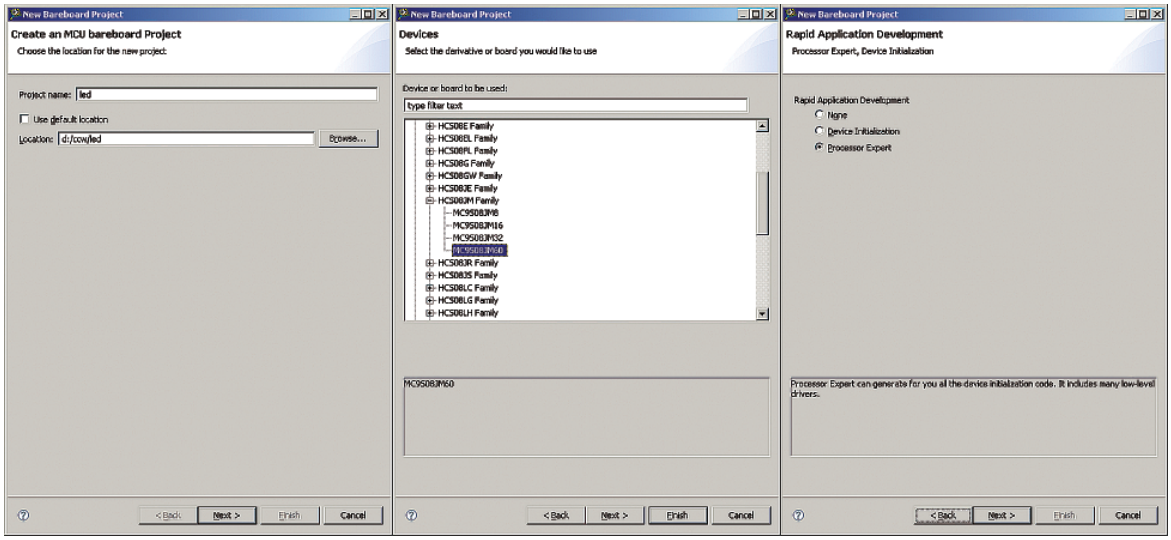
Dvě následující jednoduché úlohy demonstrují ovládání LED diod na vývojové desce Flexis. Celý vývoj bude podpořen použitím prostředí CodeWarrior s modulem Processor Expert, který velmi zjednodušuje použití periferií při programování vestavěných systémů.

6.1.1/ Úloha 1: Blikající diody LED

Cílem této jednoduché úlohy bude rozblikat několik LED diod osazených na kitu s periodou přibližně jedna sekunda. K řízení blikání využijeme vestavěnou periferii časovače, pomocí které budeme generovat přerušení s periodou přibližně půl sekundy. V obsluze přerušení pak stačí posílat na příslušný pin střídavě hodnotu nula nebo jedna. Nejjednodušší přístup je použít proměnnou, jejíž hodnotu zapíšeme na port a zvýšíme její hodnotu o jedna.

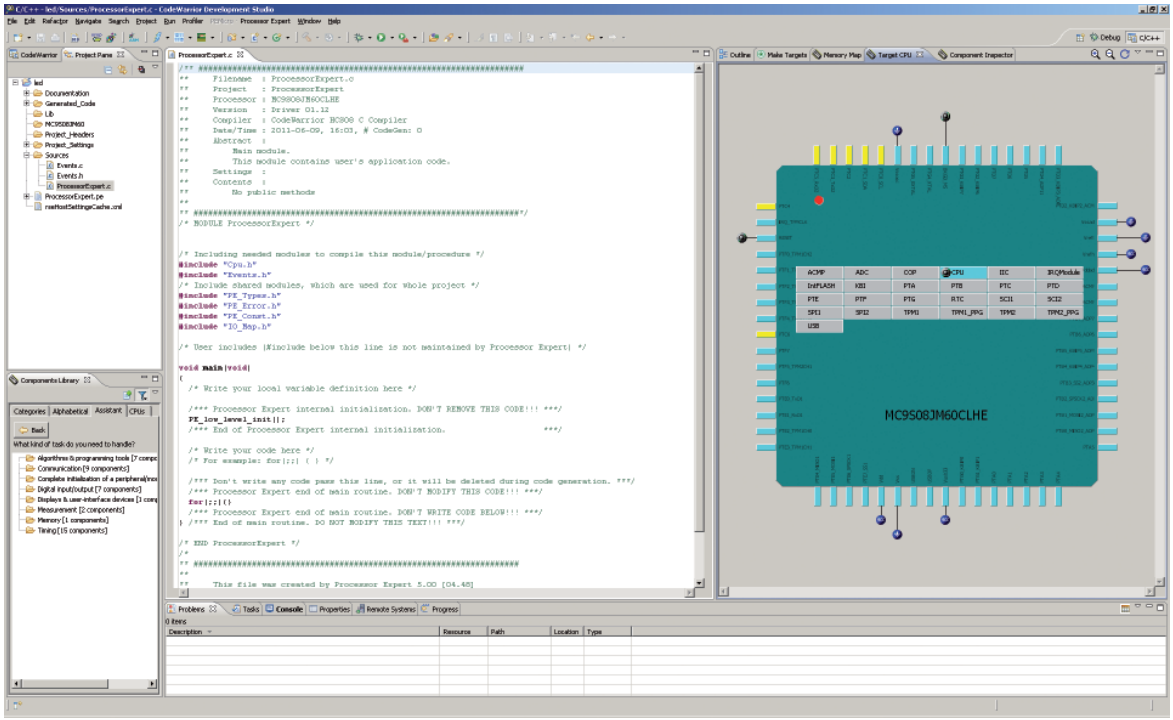
V prvním kroku je nutné spustit vývojové prostředí CodeWarrior a založit nový projekt. V této ukázce založíme projekt tak, abychom mohli využít možností nabízených modulem Processor Expert, který použijeme ke konfiguraci periferií. Jak si ukážeme dále, Processor Expert výrazně usnadní tvorbu kódu, neboť není nutné detailně studovat příslušnou technickou dokumentaci (angl. *datasheet*), abychom byli schopni nakonfigurovat potřebné periferie.

Založení nového projektu provedeme pomocí hlavního menu a nabídky *File*, kde nalezneme položku *New* obsahující *BareBoard Project*. Následně zadáme název projektu (v našem případě LED) a umístění projektu na disku (viz Obr. 36 vlevo). Nezbytným krokem je volba použitého mikrokontroléru, kde zvolíme model MC9S08JM60 rodiny HCS08JM (viz Obr. 36 uprostřed). V dalším kroku je zapotřebí zvolit programátor, který bude použit pro programování a ladění, v našem případě se jedná o variantu *P&E Cyclone PRO USB* nebo *P&E Cyclone PRO Ethernet*. První varianta se používá v případě lokální práce s vývojovým kitem, druhá varianta pro práci ze vzdáleného pracoviště. IP adresa programátoru je detekována automaticky. Na další záložce je možné přidat již existující soubory a dále zvolit implementační jazyk. V obou případech ponecháme výchozí stav. Další krok (záložka *Rapid Application Development*) slouží k usnadnění návrhu aplikace, zde máme možnost využít zásuvný modul Processor Expert (viz Obr. 36 vpravo). Následně jsme vyzváni, abychom zvolili konkrétní pouzdro použitého mikrokontroléru, v tomto kroku označíme variantu MC9S08JM60CLHE a potvrdíme. Nyní se vytvoří čistý projekt obsahující kostru programu, hlavičkové a konfigurační soubory využívající modul Processor Expert. Hlavní program se nachází v souboru pojmenovaném jako `ProcessorExpert.c`.



Obr. 36: Tvorba nového projektu s podporou modulu Processor Expert v prostředí CodeWarrior

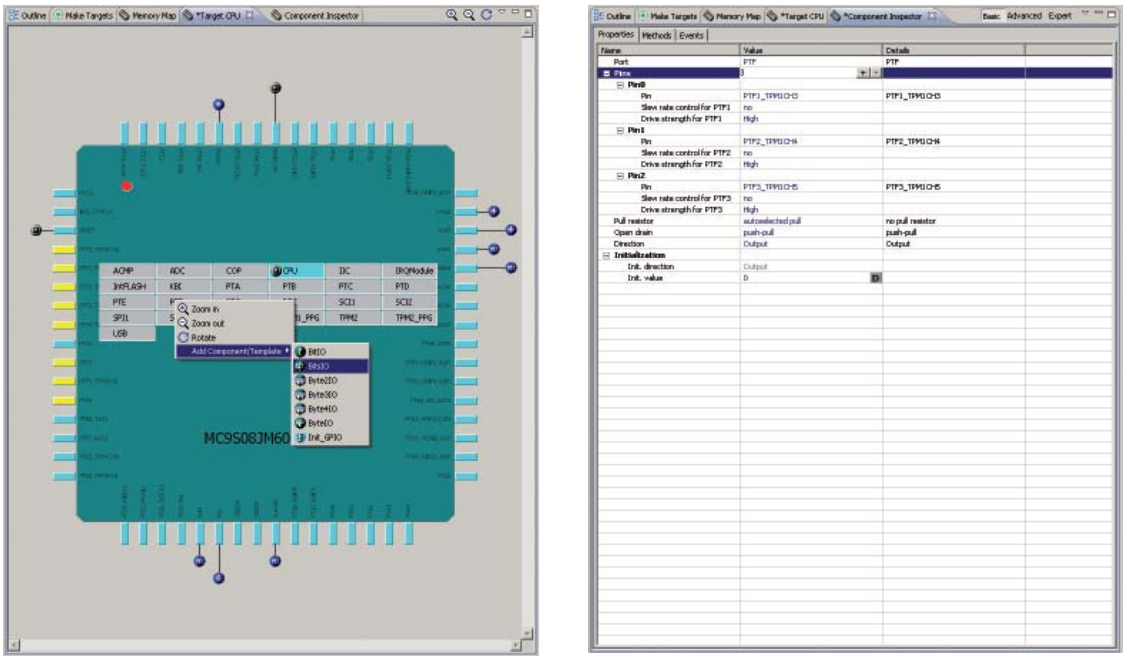
V záložce Processor Expert hlavního menu zvolit *Show views*, v pravém panelu se zobrazí okna přidružená k tomuto modulu. Zvolíme okno *Target CPU*, kde je schematicky znázorněn mikrokontrolér, dostupné periferie a dále alokace jednotlivých pinů pouzdra (viz Obr. 37).



Obr. 37: Vývojové prostředí CodeWarrior s otevřeným modulem Processor Expert (vpravo) zobrazující mikrokontrolér, dostupné periferie, alokaci pinů a periferií

V prvním kroku je zapotřebí alokovat piny, které jsou připojené ke třem LED diodám RGB podsvícení LCD displeje. Dle schématu kitu Flexis jsou diody připojeny následovně: červená – pin 1 portu F (signál PTF1), zelená – pin 2 portu F (signál PTF2), modrá – pin 3 portu F (signál PTF3).

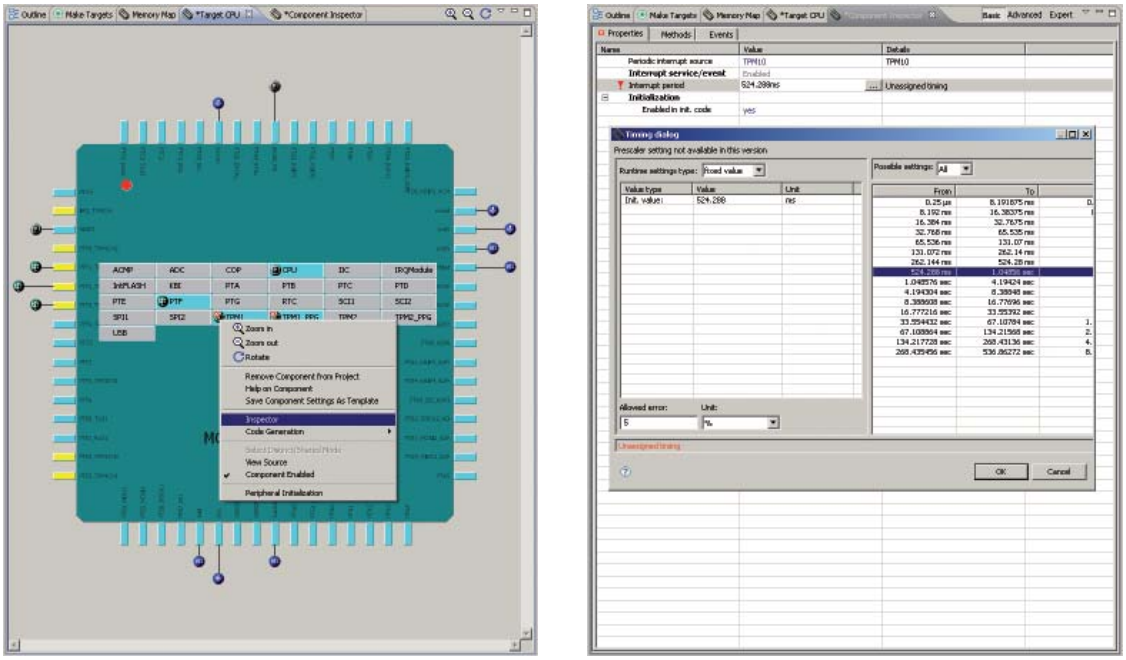
Konfiguraci pinů v modulu Processor Expert lze provést dvěma způsoby. První možností je využít schematického znázornění mikrokontroléru (okno *Target CPU*, viz Obr. 38 vlevo), kde klikneme pravým tlačítkem na modul pojmenovaný PTF a pomocí položky *Add Component/Template* vložíme komponentu s názvem *BitsIO*. Tato komponenta umožní alokovat a používat část portu PTF ve formě bitového pole. Druhý způsob je založen na přímém vložení komponenty z panelu *Component Library* nacházejícím se vlevo dole. Komponenta se nachází v následujícím umístění stromu komponent: *CPU Internal Peripherals > Port I/O*.



Obr. 38: Alokace pinů pro jednotlivé LED diody v modulu Processor Expert. Nejprve přidáme komponentu BitsIO k portu F, na kterém se nacházejí LED diody, které budeme řídit (vlevo). Následně komponentu nakonfigurujeme tak, aby jednotlivé bity byly mapovány na piny s LED diodami a byly ve správném režimu (vpravo).

Jakmile vložíme komponentu *BitsIO*, zobrazí se u portu PTF ikona indikující, že s tímto modulem je svázána určitá komponenta. V kontextovém menu této ikony potom nalezneme položku *Inspector*, která otevře záložku *Component Inspector* obsahující nastavení související s vloženou komponentou (viz Obr. 38 vpravo). V záložce musíme definovat, že si přejeme vytvořit tříbitový interface (bits: 3) a postupně u každého bitu zvolit mapování na PTF1, PTF2 a PTF3. Nastavení zakončíme definicí směru jednotlivých pinů, kde zvolíme možnost *Output* jak pro výchozí, tak pro provozní režim.

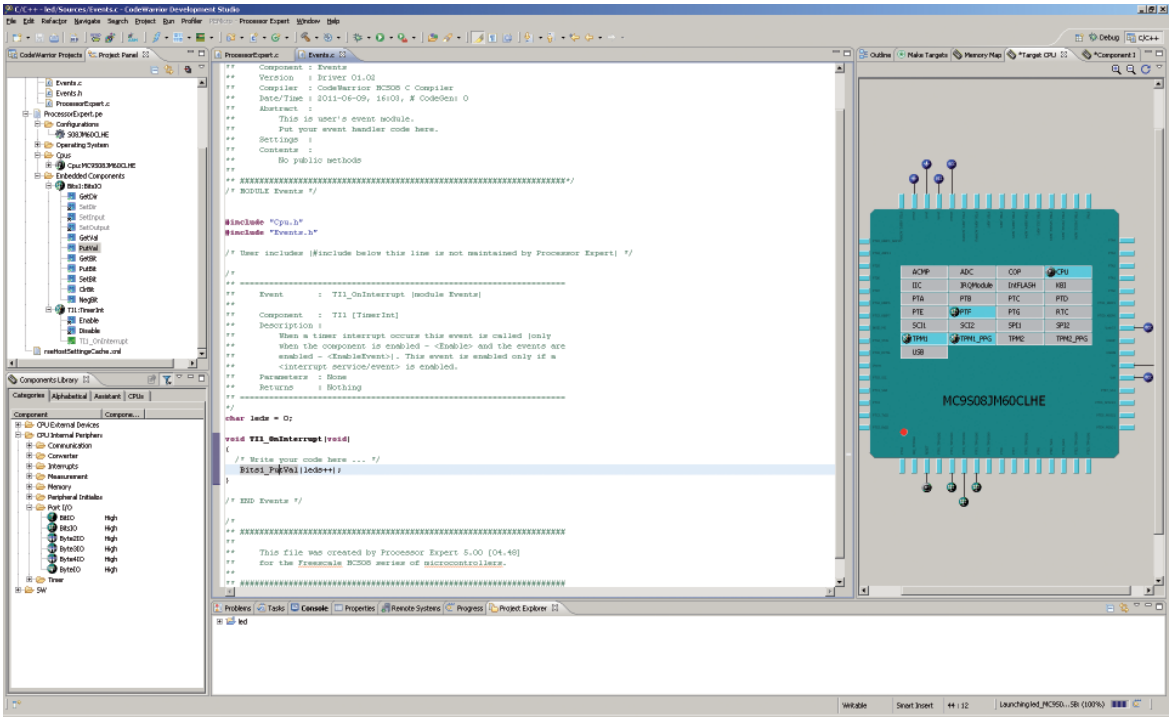
V dalším kroku využijeme možnosti vygenerovat nastavení pro vestavěný časovač. Podobně jako při vkládání komponenty umožňující pracovat s porty vložíme pomocí kontextového menu modulu pojmenovaného jako TPM1 (časovač 1) komponentu *TimerInt*, která umožní nakonfigurovat časovač generující v pravidelných intervalech přerušení. Opět podobně jako v předchozím případě pomocí položky *Inspector* vyvoláme dialog umožňující definovat parametry časovače (viz Obr. 39 vlevo). Klikneme na prázdné políčko *Interrupt period*, které vyvolá dialog, v rámci něhož si můžeme vybrat z několika přednastavených intervalů. Zvolíme např. hodnotu 524,288 ms (viz Obr. 39 vpravo).



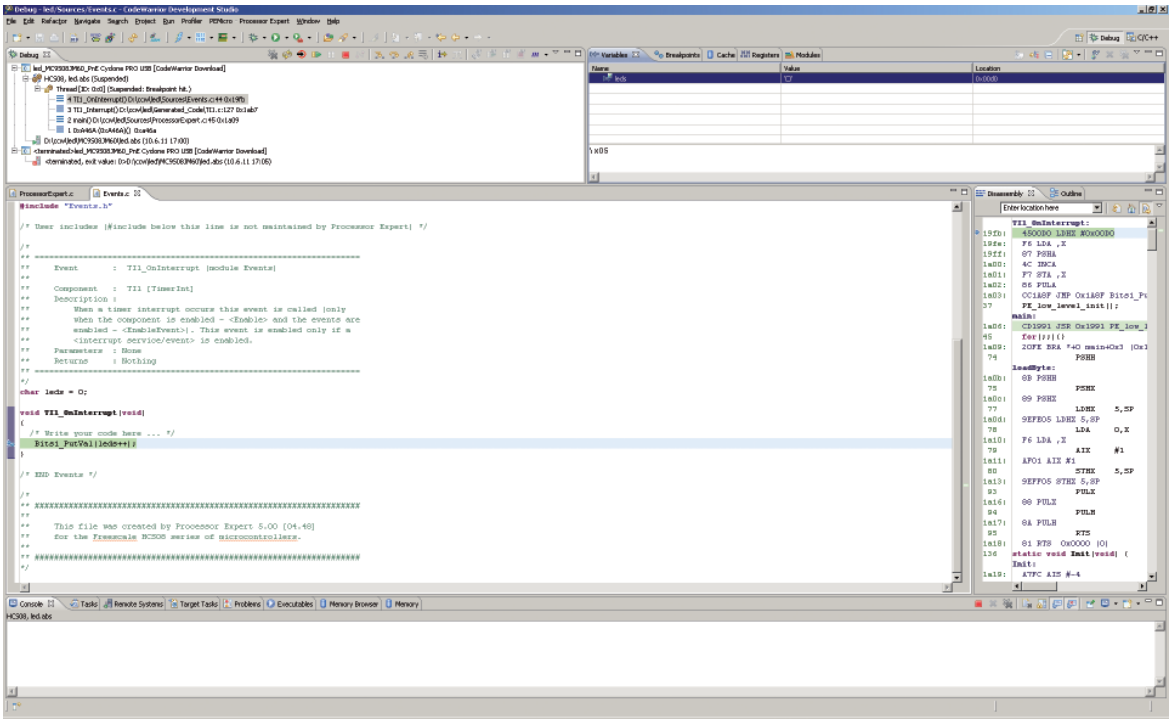
Obr. 39: Konfigurace časovače TPM1 tak, abychom měli k dispozici rutinu obsluhy přerušení, která se bude aktivovat každých cca 0,5 s. Přidání komponenty TimerInt svázané s periferií časovače TPM1 a vyvolání panelu s nastavením (vlevo). Panel s nastavením umožňující vygenerovat obsluhu přerušení a nastavit periodu, se kterou bude obslužná rutina aktivována (vpravo).

Nyní máme nakonfigurovány potřebné periferie a můžeme přistoupit k tvorbě kódu pro mikrokontrolér. Nejprve je však nutné promítnout změny provedené pomocí modulu Processor Expert do stávajících zdrojových kódů. Tento krok provedeme tak, že v menu *Project* zvolíme akci *Generate Processor Expert Code*. Zvolená akce provede vygenerování kódu pro mikrokontrolér dle zvolené konfigurace tak, aby konfigurace jednotlivých periferií odpovídala zvoleným parametrům. Přegenerování zdrojových kódů je možné vyvolat kdykoliv, kdy narazíme na nutnost modifikovat konfiguraci, aniž bychom přišli o uživatelsky definované rutiny.

Otevřeme soubor Events.c obsahující kostru obslužné rutiny časovače pojmenovanou TI1_OnInterrupt. Tato rutina se dle nastavení provedeného v předchozích krocích bude aktivovat každé půl sekundy. Aby se LED diody rozblikaly, je nutné vložit kód, který aktivuje a deaktivuje jednotlivé RGB kanály LED podsvícení. Pokud se již v prostředí orientujeme, můžeme kód napsat ihned, avšak abychom se seznámili s prostředím Processor Expert, zvolíme přístup jiný. V panelu *Project Panel* se nachází ve stromu položka *ProcessorExpert.pe* zastřešující komponenty vygenerované pomocí modulu Processor Expert. V rámci této položky nalezneme nejen konfiguraci mikrokontroléru ale také jednotlivé komponenty, které jsme pomocí modulu Processor Expert vložili. V našem případě se zde budou nacházet instance dvou komponent a sice Bits1 typu *BitsIO* a dále instance komponenty *TimerInt* pojmenovaná jako TI1. Každá komponenta dále obsahuje seznam souvisejících funkcí a seznam událostí (obslužných rutin). Dvojklikem na událost pojmenovanou TI1_OnInterrupt se otevře soubor Events.c s kurzorem umístěným na tělo obslužné rutiny přerušení od časovače. Nyní můžeme napsat vlastní kód nebo využít *Project Panel* a přetáhnout funkci pojmenovanou jako PutVal portu Bits1 do kódu (viz Obr. 40).



Obr. 40: Jednoduchý kód umožňující rozblikat jednotlivé kanály RGB podsvícení. Vlevo nahoře se nachází Project Panel obsahující seznam použitých komponent. Pro každou komponentu je zobrazen seznam dostupných metod a událostí, které je možno použít v projektu.



Obr. 41: Ukázka perspektivy Debug umožňující ladit zdrojový kód přímo uvnitř hardware. Uprostřed okna se nachází kód napsaný v jazyce C, napravo odpovídající přeložený kód v jazyce symbolických instrukcí. V horní části okna je uveden výpis zásobníku, dostupné proměnné, body zastavení atd.

Po kliknutí na tlačítko *Run* v nástrojové liště se provede překlad aplikace a naprogramování aplikace do vývojového kitu. Jakmile je kit úspěšně naprogramován, začne se podsvětlení LCD displeje skokově měnit od červené barvy až po bílou. Výsledek je možno vidět na snímku z kamery umístěné nad pracovištěm.

V případě problémů je též možné využít vestavěný ladicí modul (angl. *debugger*). Po kliknutí na tlačítko *Debug* v nástrojové liště se právě aktivní aplikace přeloží, naprogramuje do vývojového kitu a aktivuje se režim ladění, který způsobí okamžité zastavení aplikace v hlavní smyčce funkce main. Nyní můžeme kód krokovat nebo přidat body zastavení (angl. *breakpoint*) na místo, které nás zajímá, a mikrokontrolér aktivovat. Přepneme se do souboru Events.c a vložíme nový bod zastavení (kliknutím na hlavičku příslušného řádku) na řádek, který nás zajímá (viz Obr. 41). Poté klikneme v nástrojové liště na ikonu *Resume* (F8), která způsobí povolení činnosti mikrokontroléru. Jakmile dojde k přerušení, mikrokontrolér je pozastaven a kurzor se objeví na řádku, kde byl definován bod zastavení.

6.1.2/ Úloha 2: Plynulé řízení LED diod

V předchozí ukázce byla předvedena jedna ze základních laboratorních úloh věnující se konfiguraci, řízení a přístupu na port. Další typickou laboratorní úlohou je využití pulzně šířkové modulace (PWM) například k plynulému řízení svitu LED diody. Vzhledem k tomu, že modul PWM je jednou z periferií mikrokontroléru, můžeme při tvorbě opět s výhodou využít modul Processor Expert.

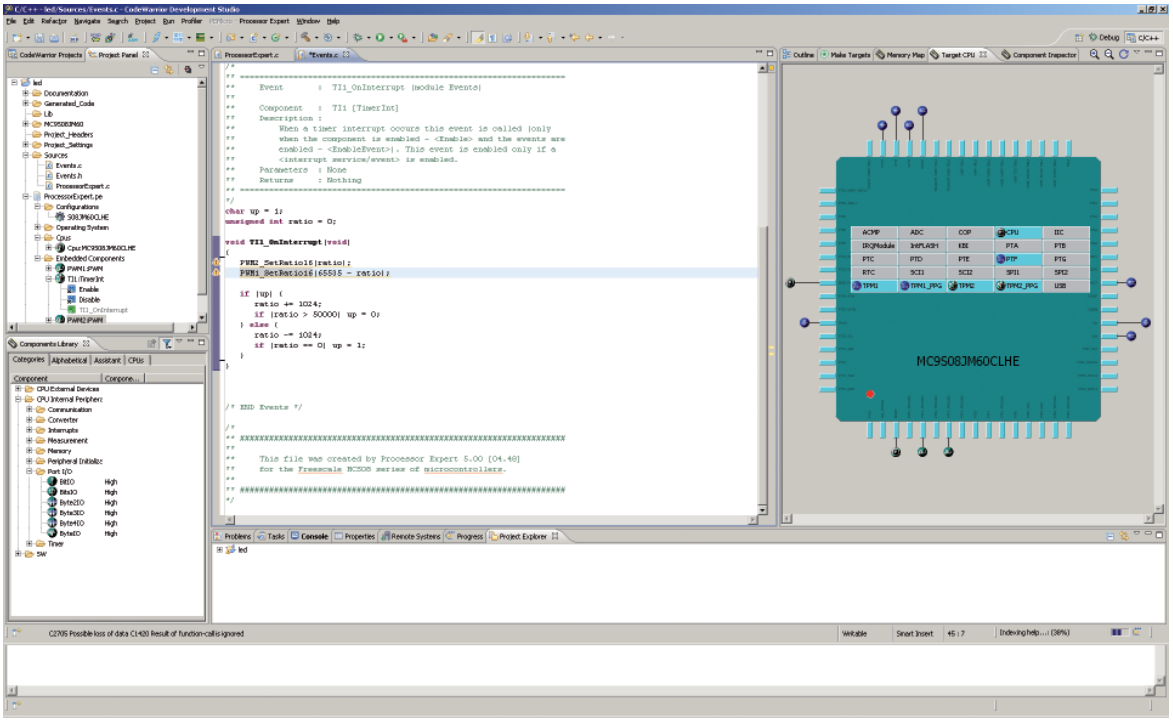
Cílem druhé demonstrační úlohy bude plynule měnit intenzitu svitu červené a modré LED diody, které jsou součástí RGB podsvícení LCD displeje. K tomuto využijeme předchozí projekt. Abychom mohli řídit dvě LED diody, musíme na každou LED diodu připojit výstup PWM modulu. Nahlédnutím do schématu zapojení výukového kitu lze zjistit, že všechny tři LED diody jsou připojeny na výstupy PWM modulu časovače TPM1.

V prvním kroku odstraníme pomocí Processor Expert v panelu *Target CPU* komponentu *BitsIO* a časovač z modulu TPM1. Nyní nakonfigurujeme časovač stejných parametrů, jako měl časovač TPM1 na modulu TPM2. Tento krok je nezbytný z toho důvodu, že TPM1 využijeme pro potřeby PWM. Periodu časovače nastavíme cca na 100 ms. V druhém kroku pomocí kontextového menu TPM1 vytvoříme dvě instance komponenty PWM. První instanci pojmenovanou jako PWM1 nakonfigurujeme tak, aby výstup PWM byl připojen na červenou LED diodu (vlastnost *Output pin* odpovídá hodnotě PTF1_TPM1CH3, tedy třetí kanál modulu TPM1), měl periodu i výchozí šířku pulzu cca 8 ms (vlastnost *Period* i *Starting pulse width* nastavena na hodnotu 8,192 ms). Parametry druhé instance pojmenované jako PWM2 nakonfigurujeme obdobně s tím rozdílem, že výstup bude připojen na modrou LED diodu (vlastnost *Output pin* se musí rovnat hodnotě PTF3_TPM1CH5, tedy pátý kanál modulu TPM1). Nyní je zapotřebí promítnout změny provedené pomocí modulu Processor Expert do stávajících zdrojových kódů. V menu *Project* zvolíme akci *Generate Processor Expert Code*.

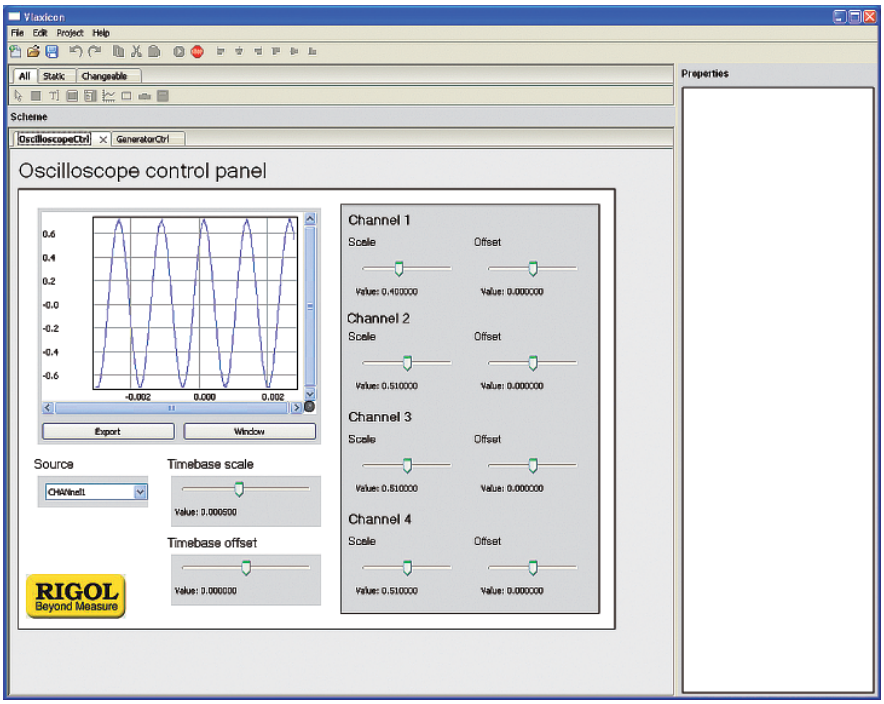
Předgenerovaný kód nabízí u každého PWM modulu funkci PWMx_SetRatio16, pomocí které můžeme plynule měnit intenzitu osvětlení, přičemž hodnota 65535 odpovídá stavu, kdy LED dioda nesvítí, a hodnota 0 stavu, kdy LED dioda svítí maximálním jasem. Jednoduchý kód, který postupně zvyšuje intenzitu svitu jedné LED a snižuje intenzitu svitu LED druhé, a tím pádem realizuje efekt přechodu z jedné barvy na druhou, je na Obr. 42.

Po kliknutí na tlačítko *Run* v nástrojové liště se aplikace přeloží a naprogramuje do vývojového kitu. Jakmile je kit úspěšně naprogramován, začne se podsvětlení LCD displeje plynule měnit z modré barvy na červenou a naopak.

Průběhy signálů generovaných PWM moduly lze sledovat také pomocí nástroje Vlaxicon, jak ilustruje Obr. 43. Po připojení jednoho měřicího kanálu osciloskopu na pin číslo 5 (PTF1) a druhého měřicího kanálu na pin číslo 7 (PTF3) konektoru P2 nacházejícího se v pravém dolním rohu vývojového kitu lze průběhy signálů střídavě sledovat na panelu vzorového projektu pro ovládání osciloskopu (OscilloscopeCtrl.vlap). Zdroj zobrazovaného měřeného signálu lze na panelu projektu přepínat pomocí rolovací nabídky *Source*.



Obr. 42: Ukázka obsluhy dvou PWM modulů, jejichž výstup je připojen na dvě LED diody a umožňuje tak řídit intenzitu svitu diod. V obslužné rutině přerušení od časovače je postupně zvyšována intenzita svitu první diody, a naopak u druhé snižována. Jakmile se narazí na maximální hodnotu, dojde k postupnému snižování intenzity svitu první diody a zvyšování druhé. Výsledkem je efekt plynule se měnící barvy podsvětlení od modré přes fialovou po červenou a zpět.



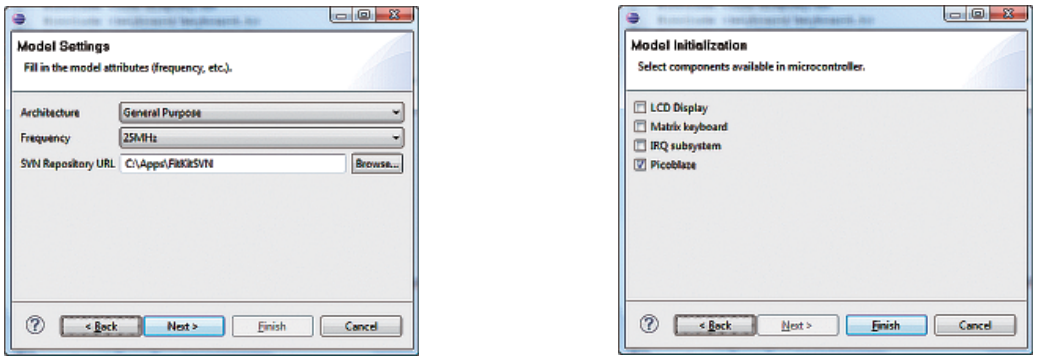
Obr. 43: Vzorový ovládací panel osciloskopu v programu Vlaxicon

6.2/ Využití vývojového prostředí VLAM IDE pro tvorbu aplikace pro PicoBlaze

Cílem této jednoduché demonstrační úlohy je vytvořit uvnitř FPGA čipu hardware využívající soft-core procesor PicoBlaze, který obsahuje kód blikající s LED diodou připojenou na jeho výstupní port. Úloha bude řešena s využitím vývojového prostředí VLAM IDE, které umožňuje snadno vytvořit kód pro FPGA s využitím grafického editoru, aniž by uživatel musel znát detailně VHDL či aspekty vývoje hardware. Kromě hardware je možné pomocí tohoto IDE vyvinout i firmware pro procesor PicoBlaze, a to nejen v jazyce symbolických instrukcí, ale též v dnes již v běžně používaném jazyku C. VLAM IDE integruje také obecné editory zdrojových textů v jazycích C a VHDL, pomocí kterých lze upravit kód vygenerovaný pro mikrokontrolér (Texas Instruments MSP430) či FPGA.

6.2.1/ Založení projektu

V prvním kroku je zapotřebí založit nový projekt. V hlavním menu *File* zvolíme položku *New* a kliknutím na *VLAM Project* vyvoláme průvodce založením nového projektu. V průvodci postupně vyplníme název a umístění projektu. V dalším dialogu (viz Obr. 44, vlevo) máme možnost zvolit architekturu (General Purpose), umístění SVN repozitáře s HDL popisy komponent a frekvenci systému (např. 25 MHz). V následujícím dialogu (viz Obr. 44, vpravo) pak vybereme komponenty, které budou v aplikaci připojeny k mikrokontroléru (v nabídce nalezneme například LCD displej, klávesnice, PS/2, PicoBlaze atd.). Generické parametry získají implicitní hodnoty. V tomto kroku zvolíme, že si přejeme použít pouze procesor PicoBlaze. Tato volba způsobí, že vygenerovaný kód pro mikrokontrolér bude obsahovat rutiny pro naprogramování PicoBlaze a kód pro FPGA komponentu PicoBlaze připojenou k rozhraní SPI spojující mikrokontrolér s FPGA. Po dokončení průvodce tlačítkem *Finish* se automaticky vytvoří základní adresářová struktura projektu včetně projektového souboru project.xml, jenž obsahuje odkazy na zdrojové soubory aplikace a knihovny zvolených periferií pro zajištění jejích připojení k mikrokontroléru. Dále se automaticky vygeneruje kostra aplikace pro mikrokontrolér (soubor mcu.c) a FPGA (soubor top_level.vhd). Po dokončení průvodce je zobrazen komponentní editor pro návrh hardwarové části.



Obr. 44: Záložky průvodce vytvořením nového projektu

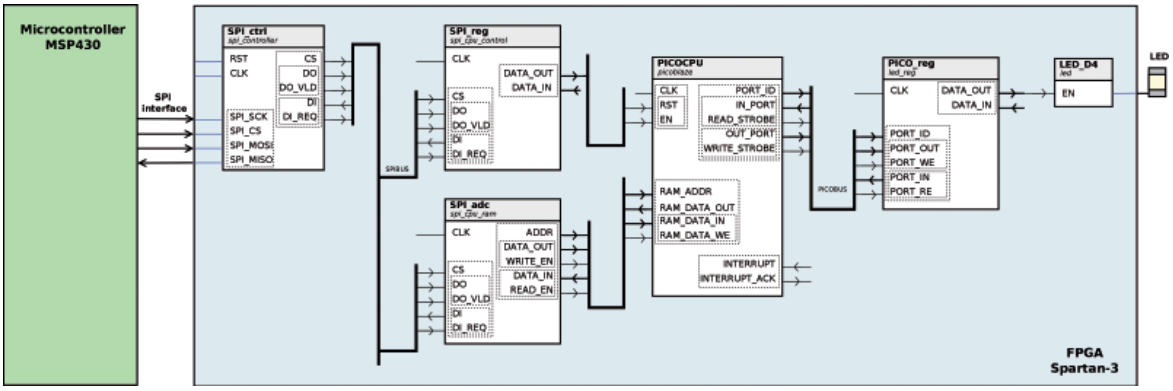
6.2.2/ Návrh hardwarové části pomocí grafického rozhraní

Centrální prvek systému tvoří procesor PicoBlaze, který je ve VLAM IDE k dispozici jako knihovní komponenta PICOCPU. Použitá komponenta má čtyři rozhraní – a) řídicí část obsahující hodinový, nulovací a povolovací signál, b) datové rozhraní blokové paměti RAM o kapacitě něco přes 2 kB, která slouží k uchování programů o velikosti až 1024 instrukcí, c) osmibitové datové rozhraní pro přístup k periferním zařízením a d) rozhraní pro práci s hardwarovým přerušením.

Abychom mohli řídit činnost procesoru, je řídicí rozhraní napojeno na osmibitový registr (prostá 8bitová paměť), jehož hodnotu je možné měnit skrze rozhraní SPI, kterým je FPGA propojeno s mikrokontrolérem. Jelikož prakticky každá aplikace vyžaduje určitou formu řízení ze strany mikrokontroléru, byl pro FITkit vytvořen sběrnicový systém na bázi SPI, který umožňuje připojit takřka libovolné množství dekodérů a registrů, jejichž datovou šířku je možné konfigurovat pomocí generických parametrů. Adresový dekodér (komponenta SPI_adc) převádí SPI rozhraní vedené po SPIBUS na jednoduché paralelní rozhraní o dané datové šířce a případně poskytuje adresový výstup. Adresový dekodér má čtyři generické parametry – datovou šířku, adresní šířku, bázovou adresu a počet adresních bitů použitých na výstupu z dekodéru. Pomocí těchto parametrů lze flexibilně specifikovat adresový prostor, do něhož má být připojená periferie mapována. Parametry však není možné volit libovolně, je nutné dodržet požadavek SPI, a sice počet adresních i datových bitů musí být násobek osmi. Při volbě parametrů je dále nutné brát ohled na možnost nežádoucího překrývání adresových prostorů v důsledku nesprávného nastavení. Pokud se však používá pro návrh aplikace VLAM IDE, uživatel se o tyto restrikce starat nemusí.

Rozhraní paměti programu je napojeno na sběrnici SPIBUS, v tomto případě pomocí adresového dekodéru (komponenta SPI_adc), jehož adresová šířka je 16 bitů. Adresový dekodér sleduje sběrnici SPIBUS, a jakmile se objeví zápis/čtení do/z adresového prostoru dekodéru, generuje zápisové a čtecí signály.

Podobně jako v případě SPI potažmo SPIBUS, byl i pro usnadnění návrhu aplikací vyžívajících PicoBlaze vytvořen jednoduchý sběrnicový systém PICOBUS, na který je možné připojit adresové dekodéry a registry s konfigurovatelnou datovou šířko



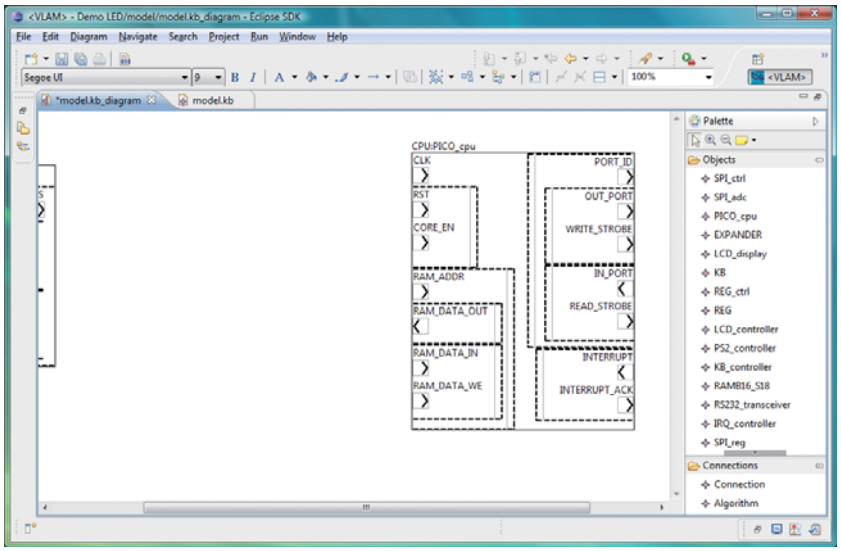
Obr. 45: Schéma návrhu hardware pro FPGA

Postup návrhu v grafickém editoru pro vytvoření hardware na Obr. 45:

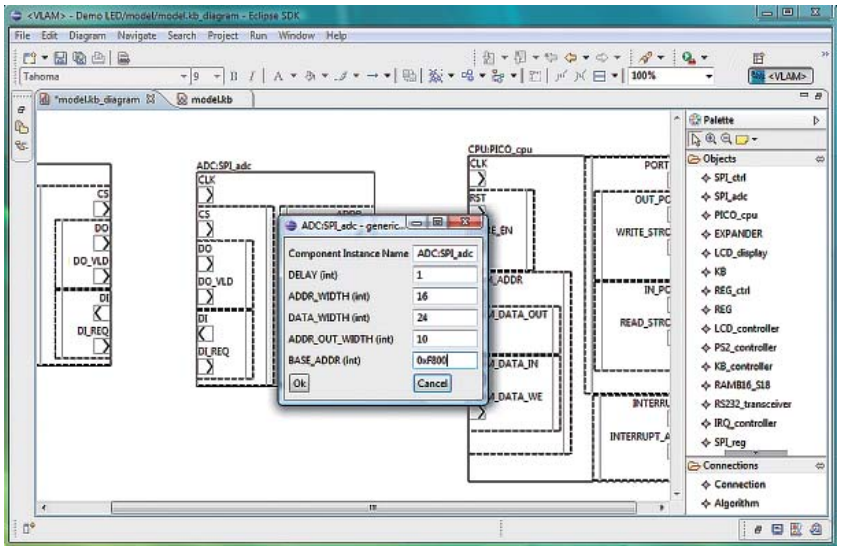
1. Vložit *picoblaze* (komponenta v seznamu komponent je označena PICOCPU, viz Obr. 46).

2. Vložit komponentu SPI_adc pojmenovanou jako *spi_cpu_ram*, nastavit generické parametry ADDR_WIDTH=16, DATA_WIDTH=24, ADDR_OUT_WIDTH=10, BASE_ADDR=0xF800, tzn. komponenta bude mapována do adresového prostoru 0xF800 - 0xFBFF; zápis na jednu z těchto adres z mikrokontroléru způsobí zápis do určité buňky paměti, DELAY=1 značí, že připojená komponenta generuje výstup až v následujícím taktu, což odpovídá chování zabudované synchronní blokové paměti, 24 bitů dat je zapotřebí nastavit, protože PicoBlaze má 18bitové instrukce a nejbližší vyšší násobek 8 je 24 (viz Obr. 47). Propojit komponentu s PICOCPU.

3. Vložit a propojit SPI_reg, nastavit parametry ADDR_WIDTH=8, DATA_WIDTH=8, BASE_ADDR=0xC0; na pin DATA_OUT(0) připojit signál EN, na pin DATA_OUT(1) připojit signál RST.



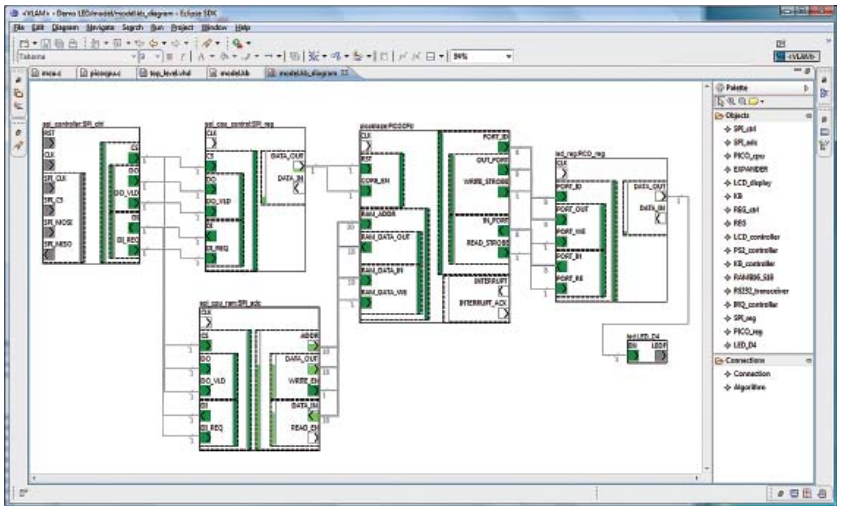
Obr. 46: Ukázka vkládání komponent na návrhové plátno



Obr. 47: Ukázka zadání generických parametrů pro SPI_adc

4. Vložit a propojit PICO_reg, nastavit parametry BASE_ADDR=0xC0, DATA_WIDTH=8. Jelikož máme nezapojený vstupní port a budeme-li chtít při čtení z adresy 0xC0 vrátit naposledy zapsanou hodnotu, je zapotřebí nastavit generickou proměnnou BYPASS na TRUE. To způsobí, že se vnitřně propojí výstup se vstupem.

5. Připojit na pin DATA_OUT(0) komponenty PICO_reg LED diodu D4 (LED dioda stejně tak jako hodinový signál CLK a signály SPI rozhraní jsou připojené přímo k modelované entitě – jsou tedy zabudované/předdefinované). Kompletní návrh je zobrazen na Obr. 48.

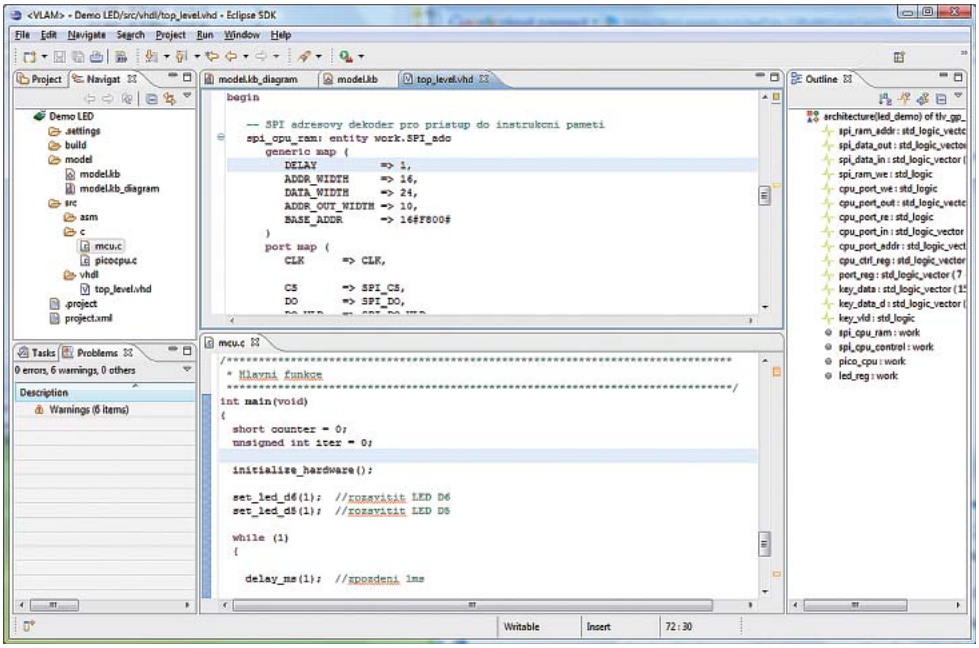


Obr. 48: Celý návrh pomocí komponentního editoru

6. Skončil-li uživatel práci s grafickým editováním návrhu hardware, může vygenerovat odpovídající VHDL kód (`top_level.vhd`, viz Obr. 49) a aktuální zdrojový soubor pro mikrokontrolér (`mcu.c`). Při přegenerování již existujícího návrhu bude uživatel dotázán na případný přepis původní verze.

6.2.3/ Vývoj firmware pro PicoBlaze

Jelikož pro jednoduchou platformu není třeba modifikovat předgenerovaný HDL kód, ani řídicí program pro mikrokontrolér (`mcu.c`), tak můžeme pokračovat editací programu pro soft-core procesor PicoBlaze, který je malou řídicí jednotkou našeho návrhu. Program je uložen v souboru dle jména procesoru, `picocpu.c` a obsahuje předgenerované vložení knihoven dle vybraných komponent v průvodci při vytváření nového projektu a kostru funkce `main`.



Obr. 49: Vygenerované HDL (horní prostřední panel) a kód pro mikrokontrolér (dolní prostřední panel)

Překladač pro PicoBlaze podporuje speciální globální pole `char PORT[256]`, které slouží pro vyvolání komunikace s porty procesoru (indexace pro čtení a přiřazení pro zápis). Hlavní program bude obsahovat nekonečnou smyčku, která bude postupně zapínat a vypínat LED diodu přibližně s vteřinovou periodou. Následuje úplný výpis jednoduchého kódu v jazyce C.

```
#define set_led(val) PORT[0x80] = val

#define led_on() set_led(0)
#define led_off() set_led(1)

// blikani s LED, perioda 1 sekunda
void main() {
  while (1) {
     len_on();
     delay1s();

     led_off();
     delay1s();
  }
}

//cekaci smycka ~ 1s
void delay1s() {
  for (char m = 0; m < 100; m++)
      for (char i = 0; i < 255; i++)
          for (volatile char j = 0; j < 255; j++);
  }
```

Když dokončíme editaci zdrojového kódu pro PicoBlaze, tak můžeme provést vygenerování dalších, nezbytných částí projektu (menu *Project* položka *Build*). Při akci *Build* bude proveden překlad pro mikrokontrolér, syntéza konfiguračního řetězce pro FPGA a překlad programu do procesoru PicoBlaze. Průběh a mezivýsledky překladu jsou zobrazovány v konzoli (viz Obr. 50). Po akci *Build* je možné zobrazit program pro PicoBlaze v jazyce symbolických instrukcí.

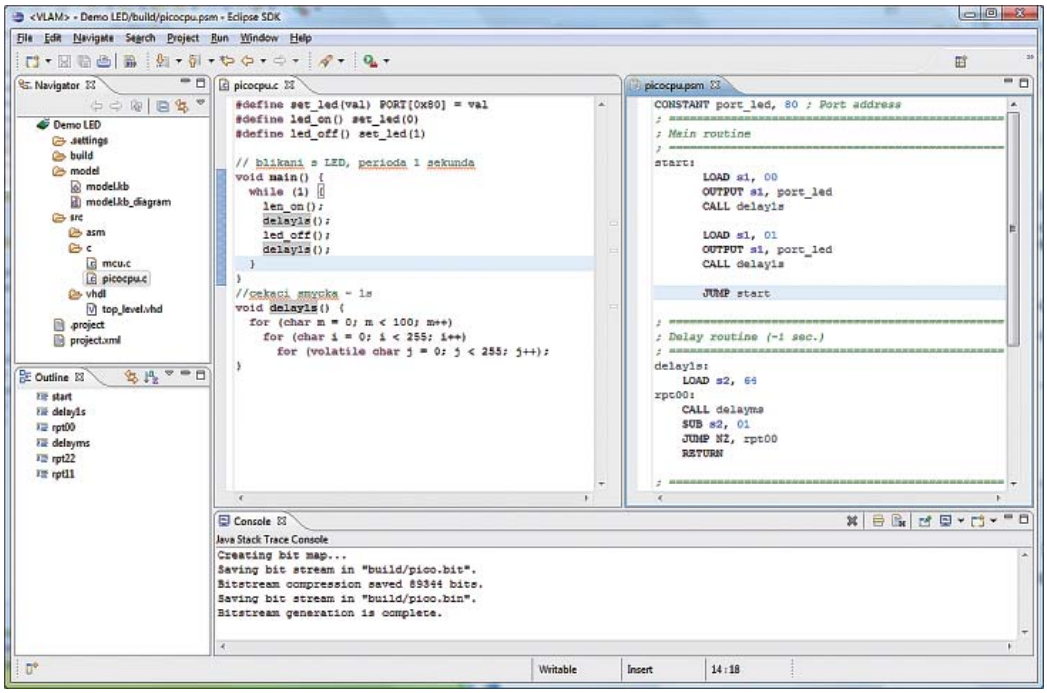
Po úspěšném dokončení překladu je pro spuštění celé aplikace nutno nakonfigurovat cílovou platformu (tj. FITkit), což provedeme akcí *Run* nebo *Debug* z nástrojové lišty. Akce se sestává ze dvou etap (viz ukázkové výpisy z konzole).

```
...
Creating bit map...
Saving bit stream in „build/pico.bit".
Bitstream compression saved 89344 bits.
Saving bit stream in „build/pico.bin".
Bitstream generation is complete.
```

První etapa naprogramuje FITkit do perzistentní FLASH paměti, aby bylo možné danou aplikaci spouštět bez nutnosti kompletního přeprogramování z počítače i při novém zapnutí napájení vývojové desky.

```
************************************************************************
* FIKTKIT PROGRAMMING                                                  *
************************************************************************
FITkit FLASH utility version: 1.7  (C) 2009 Zdenek Vasicek
Find device  OK
Device Info: VID: 0403  PID: 6010  SN: B DESCR: Dual RS232 B
Invoking BSL
CPU: F2x family; Device: F26F; BSLHEX rev: 20090326
Elapsed:2
Programming (BIN file: build/pico.bin HEX file: build/pico_f2xx.hex)
Navazování spojení
Čtení informací
Programování (MCU)
Programování (FPGA)
```

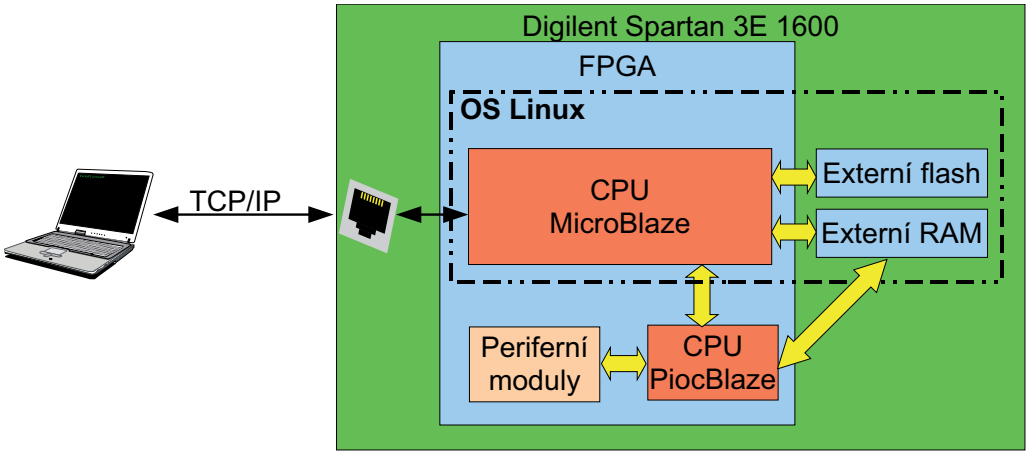
Druhá etapa spouští terminál, který automaticky nahraje do paměti FPGA program pro procesor PicoBlaze, čímž celý systém začne pracovat a dioda v sekundových intervalech blikat.



Obr. 50: Ukázka editorů jazyka C (prostřední panel) nebo jazyka symbolických instrukcí (panel vpravo) včetně zobrazené konzole pro výpisy průběhu generování a překladu

6.3/ Pokročilé příklady – Linux v FPGA

V rámci projektu VLAM vznikly příklady pro pokročilé studenty a vývojáře, které demonstrují možnosti použití OS Linux běžícího na soft-core procesoru MicroBlaze implementovaném v FPGA. Příklady jsou určeny pro vývojové desky Digilent Spartan 3E a jsou založeny na společné architektuře, která je znázorněna na Obr. 51:



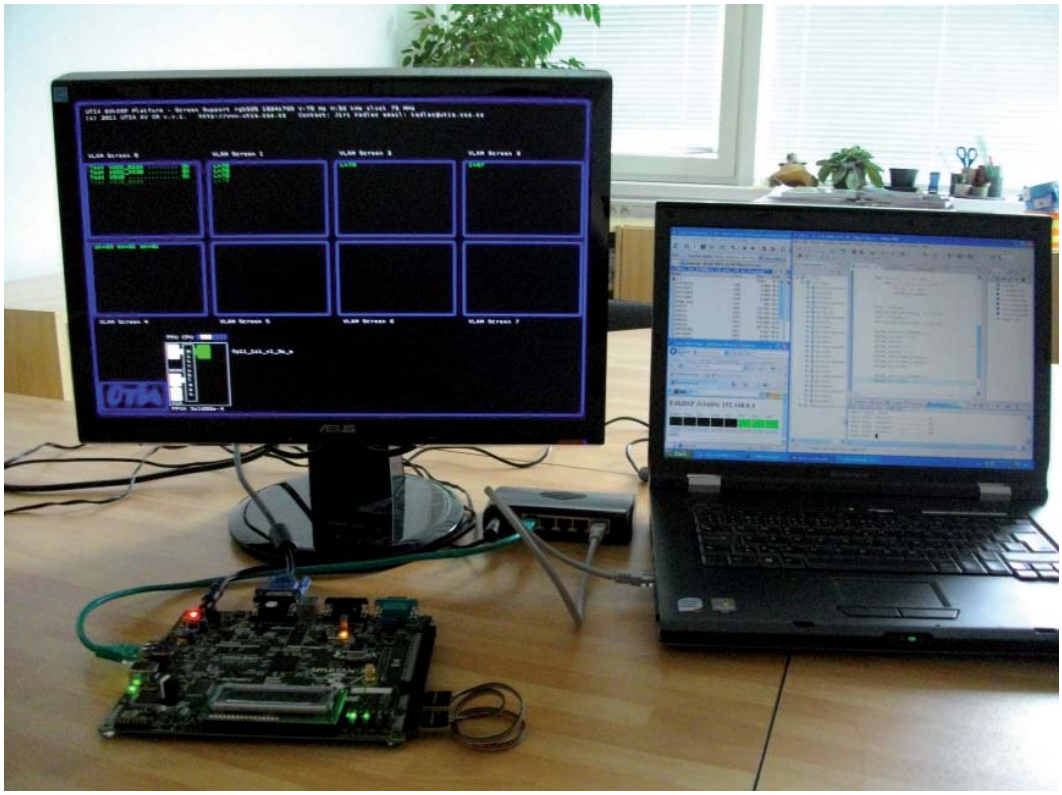
Obr. 51: Architektura pokročilých příkladů s OS Linux v FPGA

Studenti mají v rámci příkladů k dispozici funkční konfiguraci pro vývojové desky Digilent Spartan 3E 1600, která obsahuje konfigurační bitstream pro FPGA a binární image OS Linux. Po spuštění této konfigurace je OS Linux na desce dostupný na IP adrese 192.168.8.XYZ a lze se na něj připojit přes službu telnet, FTP popř. HTTP.

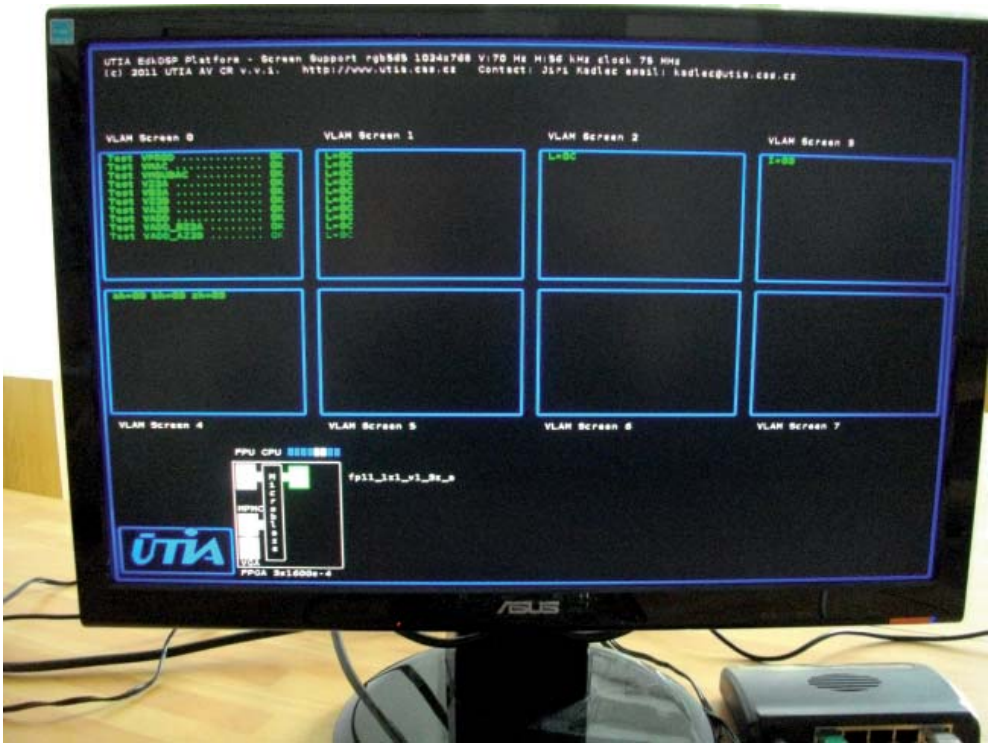
Kromě procesoru MicroBlaze, na kterém běží OS Linux, obsahuje FPGA také soft-core procesor PicoBlaze, jenž plní funkci koprocesoru pro obsluhu periferií. Studenti mohou pomocí protokolu FTP nahrát na desku binární kód ve formě ASCII souboru. Prostřednictvím aplikačního programu spustitelného přes telnet v Linuxu běžícím na procesoru MicroBlaze mohou nahrát tento binární kód do programové paměti koprocesoru PicoBlaze a následně jej spustit.

Vzdálené ovládání tlačítek a přepínačů desky je řešeno pomocí jednoduchého WWW rozhraní generovaného procesorem MicroBlaze tak, že VLAM periferní modul BASIC_IO má na vstupu sloučeny přes OR bitovou funkci jak stavy tlačítek a přepínačů, tak data z paralelního externího portu a data generovaná procesorem MicroBlaze podle stavu WWW rozhraní. Díky tomu může student ve svém WWW prohlížeči vzdáleně ovládat stavy tlačítek a přepínačů stejně, jako přímo na desce v laboratoři, pro koprocesor PicoBlaze je ale vzdálené ovládání jeho vstupů naprosto transparentní. WWW rozhraní je řešeno pomocí serveru běžícího trvale na procesoru MicroBlaze generujícího dynamicky jednoduchou stránku, která representuje tlačítka a přepínače tak, jak je znázorněno na Obr. 52.

Při běhu aplikačního programu na procesoru MicroBlaze má aplikace běžící na koprocesoru PicoBlaze k dispozici sadu komunikačních funkcí dovolujících ASCII znakový výstup na 8 samostatných terminálových oken, zobrazovaných na LCD obrazovce s rozlišením 1024x768 bodů. To dovoluje zobrazení výsledků programů, hodnot na vstupech přepínačů a tlačítek, proměnných a registrů programu tak, jak je znázorněno na Obr. 52 a Obr. 53.



Obr. 52: Vzdálené programování, ovládání a uchovávání výsledků testovaných úloh



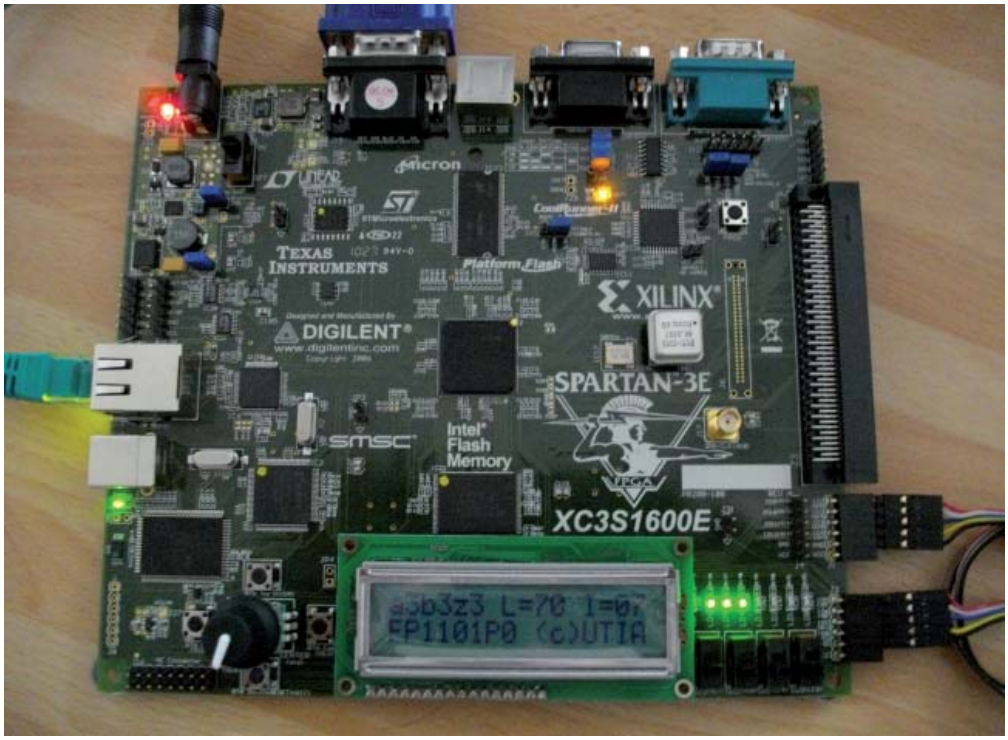
Obr. 53: Zobrazení výsledků programů, LED a stavu procesoru na obrazovce

Data, která byla předaná z testovacího programu na PicoBlaze, jsou současně ukládána do souboru pro dokumentaci běhu programu a následnou analýzu výsledků.

Součástí grafického výstupu na LCD obrazovce je také jednoduchá grafická representace řádky 8 LED diod na desce. Během komunikace mezi PicoBlaze a MicroBlaze je současně přenášen a zobrazován na LCD obrazovce aktuální stav LED diod na desce. Obdobně je zobrazováno, zda koprocesor PicoBlaze právě vykonává nahraný program, nebo čeká na nový program.

Jednotlivé řadiče periferií jsou tvořeny opět koprocesory PicoBlaze s pevným aplikačním programem. Tyto řadiče běží autonomně. Například VLAM periferní modul BASIC_IO má na vstupu tlačítka, přepínače a zpracovává data z elektronického potenciometru. Jeho výstupem je ovládání 8 LED na desce podle algoritmu daného pevným programem v PicoBlaze řadiči. Takto probíhá ovládání LED pomocí elektronického potenciometru zcela samostatně bez nutnosti spuštění programu studenta v PicoBlaze procesoru nebo koordinace s hlavním procesorem MicroBlaze.

Pokud je program studenta v koprocesoru PicoBlaze spuštěn, má tento aplikační program možnost číst stav tlačítek a přepínačů desky a současně možnost komunikace s procesorem MicroBlaze a tím i možnost vizualizace na LCD obrazovce a zápisu dat do souboru, viz Obr. 54.



Obr. 54: Výpis na dvouřádkový zobrazovač z programovatelného PicoBlaze

V rámci projektu VLAM jsme realizovali následující jednoduché příklady autonomních modulů založených na procesoru PicoBlaze použitelných bez vzdáleného připojení:

- **VLAM periferní modul BASIC_IO**: Tento HW modul byl vytvořen pro snazší použití základních periferií vývojové desky Spartan 3E Starter Kit. Modul může obsluhovat až osm tlačítek, osm přepínačů, jeden elektronický potenciometr a osm LED.

- **VLAM periferní modul LCD**: Modul LCD slouží k ovládání znakového dvouřádkového LCD displeje s řadičem HD44780. Uživatelské rozhraní představuje paměť RAM. Data zapsaná do této paměti se zobrazí na LCD displeji.

- **VLAM periferní modul ADC:** Modul ADC je řadič dvoukanálového A/D převodníku a programovatelného zesilovače s rozhraním SPI. Uživatelský interface představují čtyři řídící signály a jedna výstupní paměť FIFO, do které jsou zapisována data přečtená z A/D převodníku.

- **VLAM periferní modul DAC**: Modul DAC představuje univerzální řadič D/A převodníku s rozhraním SPI. Uživatelský interface tvoří tři řídící signály a vstupní paměť FIFO, do které se zapisují data pro D/A převodník.

- **VLAM periferní modul FREQ_CNT**: Modul FREQ_CNT představuje HW modul univerzálního čítače frekvence. Modul umožňuje přesné měření frekvencí v rozsahu 10Hz až 100MHz s možností výpočtu průměrné hodnoty frekvence. Vstupem je signál měřeného kmitočtu freq_for_meas, vstupní registr pro počet průměrovaných vzorků a výstupní registr obsahující naměřenou frekvenci.

- **VLAM periferní modul FREQ_GEN**: Modul FREQ_GEN představuje univerzální generátor frekvence. Modul generátoru umožňuje dělení vstupní uživatelské frekvence danou hodnotou.

- **VLAM periferní modul NOR_FLASH**: Modul řadiče paměti 128Mbit (16MB) Intel StrataFlash parallel NOR Flash PROM. Modul umožňuje řízení paměti pomocí sady generických příkazů. Modul podporuje zřetězení příkazů pomocí vstupního bufferu a blokové přenosy dat.

- **VLAM periferní modul PWM**: Modul PWM je řadič osmi-kanálové PWM. Uživatelský interface je tvořen 8 registry, z nichž každý obsahuje hodnotu pro jeden kanál PWM a jeden registr pro povolení zápisu do těchto 8 registrů.

- **VLAM periferní modul SPI_FLASH**: Modul řadiče paměti M25P16 SPI Serial Flash. Modul umožňuje řízení paměti pomoci sady generických příkazů. Syntaxe příkazů a způsob řízení je shodný s modulem NOR_FLASH.

Přehled realizovaných demo příkladů sestavených integrací autonomních modulů založených na PicoBlaze řadičích spolu s programovatelným PicoBlaze procesorem dovolujícím vzdálené připojení, programování, vizualizaci a záznam výsledků do souboru:

- Vzdálené ovládání tlačítek a přepínačů, 8 LCD obrazovek, zápis výsledků do souboru. Sestava v FPGA: 1x programovatelný PicoBlaze, 1x MicroBlaze.

- Elektronický potenciometr ovládající LED spolu se vzdáleným ovládáním tlačítek a přepínačů, 8 LCD obrazovek, zápis výsledků do souboru. Sestava v FPGA: VLAM periferní modul BASIC_IO, 1x programovatelný PicoBlaze, 1x MicroBlaze.

- Dvouřádkový LCD display spolu se vzdáleným ovládáním tlačítek a přepínačů, 8 LCD obrazovek, zápis výsledků do souboru. Sestava v FPGA: VLAM periferní modul LCD, 1x programovatelný PicoBlaze, 1x MicroBlaze.

- Dvouřádkový LCD display a elektronický potenciometr ovládající LED spolu se vzdáleným ovládáním tlačítek a přepínačů, 8 LCD obrazovek, zápis výsledků do souboru. Sestava v FPGA: VLAM periferní modul LCD, VLAM periferní modul BASIC_IO, 1x programovatelný PicoBlaze, 1x MicroBlaze.

7/ Závěr

Přestože projekt VLAM již skončil, jeho výsledky jsou a budou dále rozvíjeny. Například v oblasti vzdáleného přístupu a virtualizace díky celosvětové synergii zájmů vývojářů, správců a uživatelů probíhá překotný vývoj nových verzí virtualizačních produktů, které činí provozování a správu virtualizační infrastruktury stále jednodušší. Nelze předpokládat, zda v budoucnosti převáží řešení založená na technologiích VMware, Red Hat, Microsoft nebo Citrix a z hlediska projektů typu VLAM to není ani důležité, protože infrastruktura typu VLAM je na použitém virtualizéru z velké míry nezávislá, což jsme prakticky vyzkoušeli při přechodu z VMware ESX(i) na Red Hat KVM.

Jak oba prototypy laboratoří, tak naše zkušenosti získané s různými virtualizačními produkty, vzbudily zájem dalších subjektů. Nasazení virtualizovaných laboratoří a počítačových učeben nyní zvažují další fakulty UTB ve Zlíně, zaznamenali jsme také zájem o virtualizaci laboratorních přípravků projektu iSES.

Jako u každého softwarového nástroje, i software vzniklý na projektu VLAM bude dále vylepšován a případně i rozšiřován o novou funkčnost například prostřednictvím bakalářských a diplomových prací. Ve VLAM IDE se zaměříme na zlepšování uživatelské přívětivosti a podporu dalších výukových platforem. Pro překladač jazyka C pro PicoBlaze plánujeme implementaci novější verze cílového jazyka pro procesor PicoBlaze verze 6, který byl představen na konci minulého roku a přinesl nové možnosti umožňující rozšířit implementovanou podmnožinu standardu jazyka C. V posledních týdnech byla také započata spolupráce s vývojáři staršího zásuvného modulu SDCC pro Eclipse, na němž bychom také rádi spolupracovali pro zvýšení podpory editace zdrojových textů jazyka C rozšířeného o specifika vývoje pro vestavěné systémy. V rámci dalšího vývoje nástroje Vlaxicon bude kladen důraz na zlepšení odezvy jeho GUI při déletrvajících asynchronních operacích.

8/ Reference

[1] 2X Software Ltd. *2X ApplicationServer - Application & Desktop Publishing for Windows Terminal Services*. Dostupné z WWW: *http://www.2x.com/applicationserver/*. [cit. 2011-07-11].

[2] Aho, A.V. et al. *Compilers: Principles, Techniques, and Tools*. Prentice Hall, ISBN: 0321486811.

[3] Behrens, H. *Xtext User Guide*. Dostupné z WWW: *http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.html*. [cit. 2011-07-22].

[4] Bližňák, M. *Poznámky k instalaci fyzického serveru systému VLAM*. Příloha k Výroční zprávě VLAM 2010 č. A8-12_bliznak_001_VLAMServer, Vysoké Učení Technické v Brně, Fakulta Informačních Technologií, 2011.

[5] Bližňák, M. *Poznámky k instalaci virtuální pracovní stanice*. Příloha k Výroční zprávě VLAM 2010 č. A8-12_ bliznak_003_VPS, Vysoké Učení Technické v Brně, Fakulta Informačních Technologií, 2011.

[6] Bližňák, M. *Uživatelská příručka webové aplikace VlamGateway*. Příloha k Výroční zprávě VLAM 2010 č. A8-12_ bliznak_004_VlamGateway_UserGuide, Vysoké Učení Technické v Brně, Fakulta Informačních Technologií, 2011.

[7] Bližňák, M. *wxShapeFramework*. Dostupné z WWW: *http://sourceforge.net/projects/wxsf/*. [cit. 2011-07-11].

[8] Bližňák, M. et al. *Výroční zpráva 2010 - průběžná zpráva projektu Virtuální laboratoř aplikace mikroprocesorové techniky MŠMT 2C06008*. FIT VUT Brno, Unis a.s., UTIA, 2010. Dostupné z WWW: *http://www.msmt-vyzkum.cz/YYYYY/ePROJEKTY/*. [cit. 2010-07-27].

[9] Bližňák, M. et al. *Výroční zpráva 2011 - průběžná zpráva projektu Virtuální laboratoř aplikace mikroprocesorové techniky MŠMT 2C06008*. FIT VUT Brno, Unis a.s., UTIA, 2011. Dostupné z WWW: *http://www.msmt-vyzkum.cz/YYYYY/ePROJEKTY/*.

[10] Budíková, V. *Virtuální laboratoř mikropočítačů se vzdáleným přístupem*. Diplomová práce, Univerzita Tomáše Bati ve Zlíně, Fakulta Aplikované Informatiky, 2011. Dostupné z WWW: *https://portal.utb.cz/stag?urlid=prohlizeni-prace-detail&praceIdno=20779*. [cit. 2011-07-11].

[11] Drutarovský, M. *FPGA Virtual Lab*. Dostupné z WWW: *http://www.kemt.fei.tuke.sk/fpga/kega/index.html*. [cit. 2011-07-22].

[12] Eclipse Foundation. *Eclipse - The Eclipse Foundation open source community website*. Dostupné z WWW: *http://www.eclipse.org/*. [cit. 2011-07-22].

[13] Eclipse Foundation. *Eclipse CDT*. Dostupné z WWW: *http://www.eclipse.org/cdt/*. [cit. 2011-07-22].

[14] Eclipse Foundation. *Eclipse Verilog editor*. Dostupné z WWW: *http://sourceforge.net/projects/veditor/*. [cit. 2011-07-22].

[15] Elusiva. *Open Source - Elusiva Java RDP Client*. Dostupné z WWW: *http://www.elusiva.com/opensource/*. [cit. 2011-07-11].

[16] Elusiva. *Terminal Server Pro*. Dostupné z WWW: *http://www.elusiva.com/products/TerminalServerPro.aspx*. [cit. 2011-07-21].

[17] Gilibert, M. et al. *80C537 Microcontroller Remote Lab for E-Learning Teaching*. Dostupné z WWW: *http://online-journals.org/index.php/i-joe/article/view/364*. [cit. 2011-07-25].

[18] Golod, J. *Enabling Multiple Remote Desktop Sessions in Windows XP Professional and Media Center Edition 2005*. Dostupné z WWW: *http://www.golod.com/2005/10/enabling-multiple-remote-desktop-sessions-in-windows-xp-professional-and-media-center-edition-2005/*. [cit. 2011-07-21].

[19] Horník, J. *Language C Compiler Back-end for* Soft-core *Microcontroller*. V *Proceedings of the 17th Conference and Competition STUDENT EEICT 2011*, FIT VUT, Brno, 2011, ISBN: 978-80-214-4273-3.

[20] Horník, J. *Zadní část překladače podmnožiny jazyka C pro 8-bitový procesor*. Diplomová práce, FIT VUT v Brně, 2011.

[21] Hruška, F. *Laboratoře integrované automatizace na UTB ve Zlíně*. Dostupné z WWW: *http://www.odbornecasopisy.cz/index.php?id_document=34259*. [cit. 2011-07-22].

[22] Jeff Walters. *PHP Booking Calendar*. Dostupné z WWW: *http://sourceforge.net/projects/bookingcalendar/*. [cit. 2011-07-11].

[23] KernelPro Software. *KernelPro - usb over network*. Dostupné z WWW: *http://www.kernelpro.com/*. [cit. 2011-07-11].

[24] Kondr, A. a Brejl, M. *OpenSourceBDM – jednoduchý programátor pro mikrokontrolery Freescale S08 a ColdFire V1*. Dostupné z WWW: *http://hw.cz/teorieapraxe/konstrukce/art2558-opensourcebdm-jednoduchy-programator-pro-mikrokontrolery-freescale-s*. [cit. 2011-07-22].

[25] Kopp, S. *USBTMC Kernel Driver Documentation*. Dostupné z WWW: *http://www.home.agilent.com/upload/cmc_upload/All/usbtmc.html?&cc=CZ&lc=eng*. [cit. 2011-07-11].

[26] Labenski, J. *wxCode - wxPlotCtrl*. Dostupné z WWW: *http://wxcode.sourceforge.net/showcomp.php?name=wxPlotCtrl*. [cit. 2011-07-11].

[27] Lustig, F. et al. *iSES - Internet School Experimental System*. Dostupné z WWW: *http://www.ises.info/index.php/en/ises*. [cit. 2011-07-22].

[28] Mehlhorn, K. a Näher, S. *Leda: a platform for combinatorial and geometric computing*. Cambridge University Press, ISBN: 9780521563291.

[29] P&E Microcomputer. *P&E Microcomputer Systems*. Dostupné z WWW: *http://www.pemicro.com/*. [cit. 2011-07-22].

[30] Patterson, D.A. a Hennessy, J.L. *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface*. Morgan Kaufmann, ISBN: 0123744938.

[31] Pentico, D.W. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*. 176, 2 (led. 2007), s. 774-793, DOI: doi: 10.1016/j.ejor.2005.09.014.

[32] Red Hat, Inc. *Applications using libvirt*. Dostupné z WWW: *http://libvirt.org/apps.html*. [cit. 2011-07-21].

[33] Red Hat, Inc. *KVM Main Page*. Dostupné z WWW: *http://www.linux-kvm.org/page/Main_Page*. [cit. 2011-07-11].

[34] Red Hat, Inc. *KVM Management Tools*. Dostupné z WWW: *http://www.linux-kvm.org/page/Management_Tools*. [cit. 2011-07-11].

[35] Roebling, R. et al. *wxWidgets*. Dostupné z WWW: *http://www.wxwidgets.org/*. [cit. 2011-07-11].

[36] Sharples, S.D. *VXI11 Ethernet Protocol for Linux*. Dostupné z WWW: *http://optics.eee.nottingham.ac.uk/vxi11/*. [cit. 2011-07-11].

[37] Thinstuff s.r.o. *XP/VS Server*. Dostupné z WWW: *http://www.thinstuff.com/products/xpvs-server/*. [cit. 2011-07-21].

[38] Trbušek, J. *Vlaxicon*. Dostupné z WWW: *http://sourceforge.net/projects/vlaxicon/*. [cit. 2011-07-11].

[39] Trbušek, J. *Vzdálené řízení a monitoring měřicích přístrojů*. Diplomová práce, Univerzita Tomáše Bati ve Zlíně, Fakulta Aplikované Informatiky, 2011. Dostupné z WWW: *https://portal.utb.cz/stag?urlid=prohlizeni-prace-detail&praceIdno=20871*. [cit. 2011-07-11].

[40] VMware, Inc. *VMware vCenter Server Virtualization Management (formerly VMware Virtual Center)*. Dostupné z WWW: *http://www.vmware.com/products/vcenter-server/overview.html*. [cit. 2011-07-11].

[41] Vašíček, Z. *FITkit, Úvod*. Dostupné z WWW: *http://merlin.fit.vutbr.cz/FITkit/docs/uvod/uvod.html*. [cit. 2011-07-22].

[42] Xilinx. *PicoBlaze 8-bit Embedded Microcontroller User Guide*. Dostupné z WWW: *http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf*. [cit. 2011-07-22].

[43] *NetLab - The Online Remote Laboratory*. Dostupné z WWW: *http://netlab.unisa.edu.au/index.xhtml*. [cit. 2011-07-22].

Seznam obrázků:

Seznam tabulek:

Seznam zkratek:

Zkratka	Význam, popis
ACPI	Advanced Configuration and Power Interface
CPU	Central Processing Unit (Procesor)
FPGA	Field-programmable gate array (Programovatelné hradlové pole)
GPIB	General Purpose Interface Bus
GUI	Grafické uživatelské rozhraní
IPC	Industrial PC
iSES	internet School Experimental System
IR	Infrared (Infračervené)
HW	Hardware
LABI	Laboratoř integrované automatizace
LAN	Local Area Network
MCU	Mikrokontrolér
MS	Microsoft
OS	Operační systém
PDU	Power Distribution Unit (zařízení s několika zásuvkami 230V, které mohou být ovládány přes TCP/IP)
PLC	Programmable Logic Controller
PLD	Programmable Logic Device (programovatelné logické zařízení)
RDP	Remote Desktop Protocol (Protokol vzdálené plochy)
RTSP	Real Time Streaming Protocol
SCPI	Standard Commands for Programmable Instruments
SW	Software
TS	Microsoft Windows Terminal Services
UPS	Uninterruptible Power Supply (nepřerušitelný zdroj napájení)
VISA	Virtual Instrument Software Architecture
VLAN	Virtual Local Area Network
VPS	Virtuální pracovní stanice

Autorský kolektiv

Michal Bližňák / Věra Budíková / Tomáš Dulík

Ota Jirák / Jiří Kadlec / Zbyněk Křivka

Nela Olšarová / Josef Trbušek / Zdeněk Vašíček

Virtuální laboratoř

pro vývoj aplikací s mikroprocesory a FPGA

Vydalo AKADEMICKÉ NAKLADATELSTVÍ CERM®, s.r.o. Brno v roce 2011
Vytiskl FINAL TISK s.r.o. Olomučany
Recenzent doc. RNDr. Petr Šaloun, Ph.D.
Vydání první

Tato publikace neprošla redakční ani jazykovou úpravou

ISBN 978-80-7204-754-3