Uživatelská příručka VLAM IDE

Zbyněk Křivka, Ota Jirák, Nela Olšarová

krivka@fit.vutbr.cz, ijirak@fit.vutbr.cz, xolsar00@stud.fit.vutbr.cz

zpráva projektu
Virtuální laboratoř aplikace mikroprocesorové techniky
MŠMT 2C06008

http://www.vlam.cz
UNIS spol. s r.o, R&D; ÚTIA AV ČR; VUT Brno, FIT, ÚIFS

Obsah

Revize

Číslo	Datum	Autor	Popis změny
1	9. 8. 2010	O. Jirák	Vytvoření dokumentu
2	1. 8. 2011	Z. Křivka	Revize dle současného stavu
		N. Olšarová	Ovládání PB[Asm|Psm]Editoru
3	10. 8. 2012	Z. Křivka, N. Olšarová	Popis dialogu pro automatické propojování komponent (nástroj *Algorithm*)

Abstrakt dokumentu

Dokument obsahuje uživatelskou příručku nástroje VLAM IDE. Zabývá se uživatelským ovládáním VLAM IDE včetně komponentního editoru a dalších zásuvných modulů.

Klíčová slova

VLAM IDE, uživatelská příručka, Eclipse

Vztah k dílčímu cíli

Aktivita A8-18 je součástí Dílčího cíle 8.

Vztah k aktivitě a ostatním aktivitám

Vztahuje se k aktivitám A8-13 (tvorba editoru), A8-14 (tvorba IDE) a A8-18 (aktualizace dokumentace dle aktuálního stavu).

Úvod

V následujícím textu se uživatel postupně seznámí s vytvořením projektu, vytvořením diagramu konfigurace, jeho úpravami a generováním kódu.

Instalace VLAM IDE

Při instalaci vlastního vývojového prostředí můžeme využít již existující instalaci Eclipse, do které bude standardním způsobem doinstalován nový software z adresy uvedené na www.vlam.cz ([3]). Během instalace bude vaše prostředí dle nutnosti dovybaveno nezbytnými zásuvnými moduly třetích stran (např. běhová část projektu GMF a EMF). Prostředí VLAM IDE verze 1.0 bylo vyvíjeno a testováno v Eclipse verze 3.6.2 (Helios).

Následující popis provází postupnou instalací krok po kroku včetně instalace nového prostředí Eclipse. Návod je pro platformu MS Windows, ale analogicky se instalace provádí i na linuxovém operačním systému.

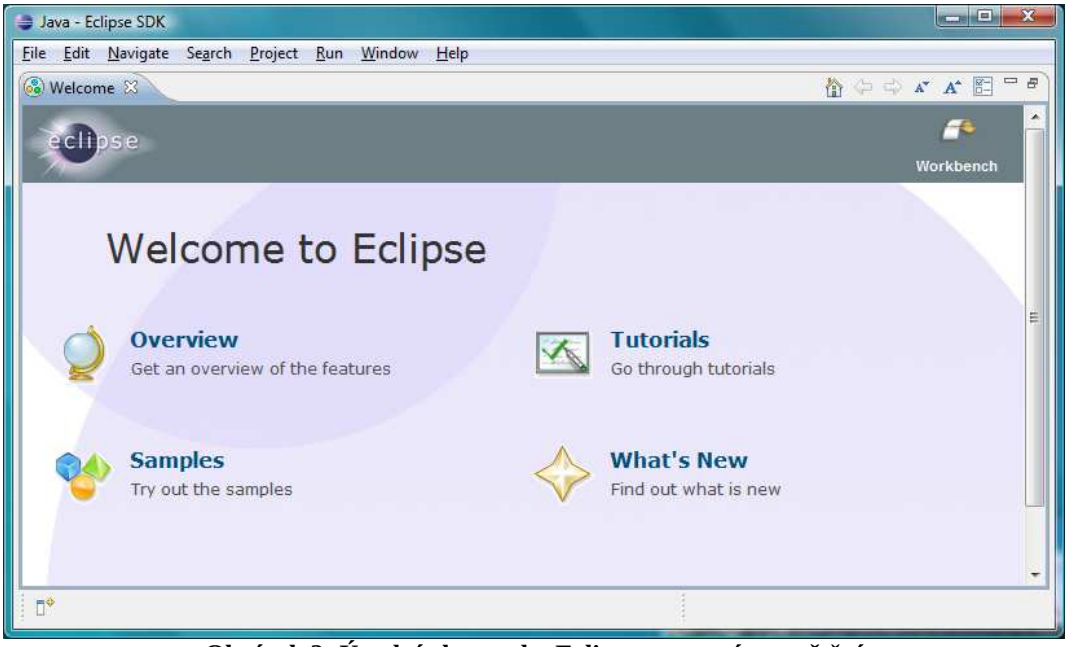
1. Na adrese http://www.eclipse.org/downloads/packages/release/helios/sr2 si stáhněte balíček *Eclipse Classic* 3.6.2 (je možné začít i s jiným balíčkem, např. *Eclipse IDE for Java Developers* nebo *Eclipse IDE for C/C++ Developers*, postup bude totožný). Stažený balíček rozbalte do adresáře, kde budete chtít mít Eclipse (je nutno ručně vytvořit, Eclipse nemá vlastní instalátor, protože nepotřebuje žádné zásahy do operačního systému ani prostředí; všechny jeho konfigurace jsou v jeho vlastních konfiguračních souborech nebo v souborech projektů vytvořených v Eclipse). Např. D:\eclipseclassic
2. Vytvořte adresář, kam budete chtít ukládat vytvořené projekty (každý projekt vytvoří svůj podadresář; adresář s projekty lze měnit, pokud pracujete s větším počtem projektů).

Obrázek 1: Ukázka mnoha pohledu

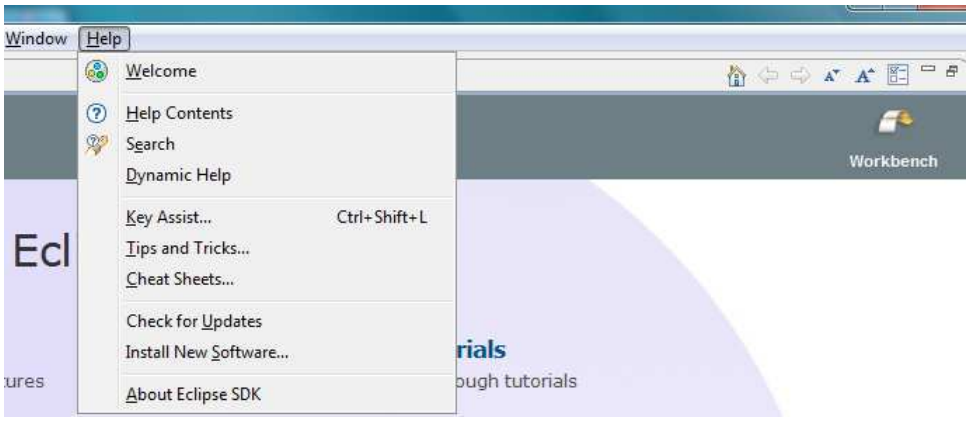
3. Spusťte Eclipse.exe. Budete dotázání na *Workspace* adresář (viz Obrázek 1), což je onen vytvořený adresář pro Vaše projekty. Např. D:\eclipseclassic\workspace. Pokud

nechcete být při každém spouštění Eclipse dotazováni na cestu k Workspace, tak zatrhněte tlačítko „Use this as default and do not ask again“. Stiskněte OK.

4. Po spuštění Vás přivítá úvodní obrazovka Eclipse (viz Obrázek 2). Do vývojového prostředí se přepneme kliknutím na ikonu *Workbench* v horním pravém rohu, ale zatím to není nutné. Nyní vyberte menu *Help* a položku *Install New Software…* (viz Obrázek 3)

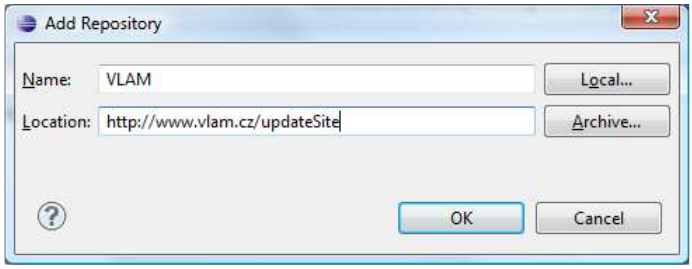


Obrázek 2: Úvodní obrazovka Eclipse po prvním spuštění



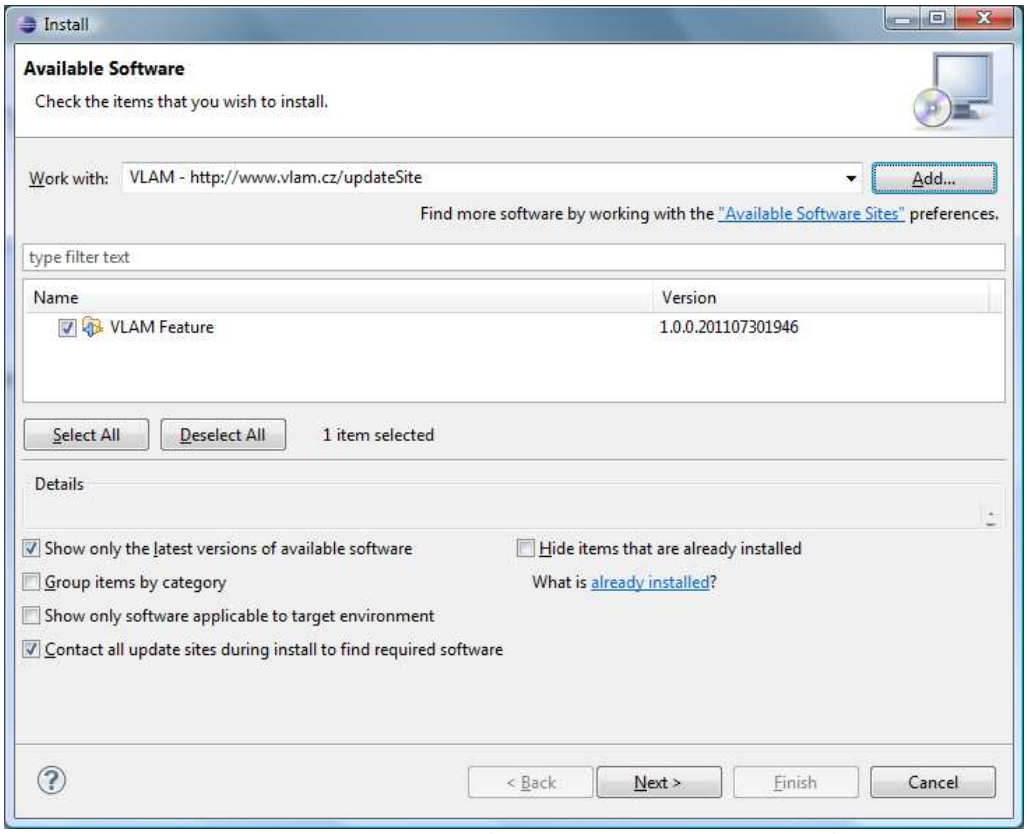
Obrázek 3: Spuštění instalace nových zásuvných modulů do Eclipse

5. V zobrazeném dialogu *Install* klikněte na tlačíko *Add…* pro přidání nového úložiště (tzv. repozitář), ze kterého lze stahovat zásuvné moduly do Eclipse. Do dialogu *Add Repository* (viz Obrázek 4) vyplňte *Name*: „VLAM“ (je na Vás, jak si repozitář pojmenujete) a *Location*: zadejte „http://www.vlam.cz/updateSite“. Je možné, že se tato adresa změní, o čemž bude případně informace na oficiální domovské stránce projektu VLAM [3]. Tomuto repozitáři se též někdy říká tzv. *Update site* a Eclipse je schopen na jejím základě provádět instalace či automatické aktualizace.



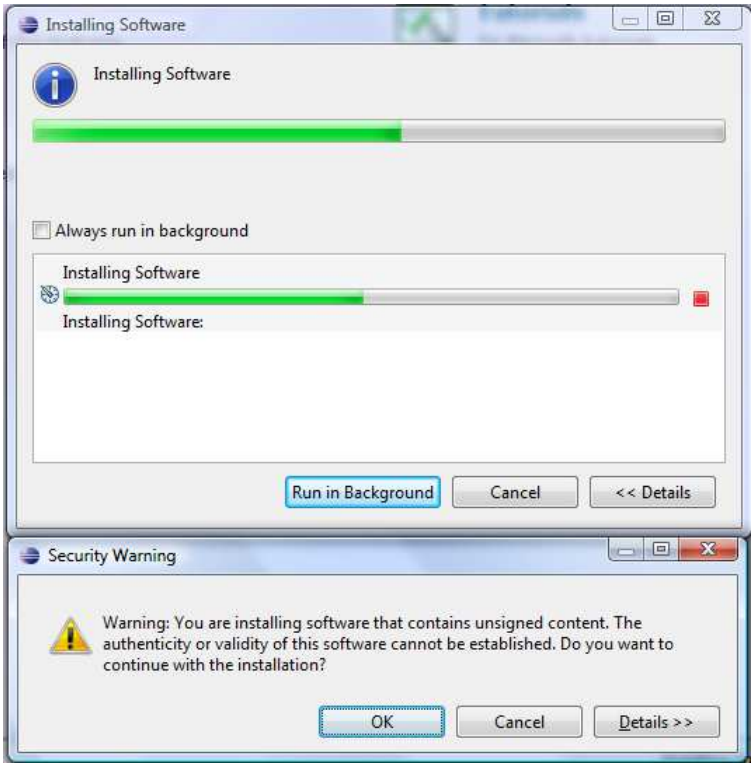
Obrázek 4: Dialog pro přidání nového repozitáře software pro Eclipse

6. Po načtení zásuvných modulů k instalaci vymažte případné zatržení položky „*Group items by category*“, která může způsobit nezobrazení položky *VLAM feature*, kterou vyberte (viz Obrázek 5). Zašktněte volbu „*Contact all update sites during install to find required software*“ (Poznámka: Pro doinstalování závislostí je nutné mít vytvořen oficiální repozitář Helios - http://download.eclipse.org/releases/helios, což je většinou v nové instalaci automatické.). Klikněte na *Next >*.



Obrázek 5: Dialog pro výběr balíčků k instalaci a nastavení detailů instalace

7. V dalším okně odsouhlaste různá licenční ustanovení a stiskněte opět tlačítko *Next >*. Pak dojde ke stahování a instalaci všech potřebných zásuvných modulů (viz Obrázek 6), přičemž můžete být varováni, že instalovaný software není elektronicky podepsán, což je třeba odsouhlasit (tlačítko OK na Obrázek 6), aby instalace pokračovala.



Obrázek 6: Průběh instalace včetně varování o nepodepsaném obsahu

8. Po dokončení instalace budete vyzváni k restartu prostředí Eclipse (viz Obrázek 7).

Obrázek 7: Dotaz na dokončení instalace spojené s doporučeným preventivním restartem prostředí Eclipse

9. Po restartu je VLAM IDE nainstalováno, což lze ověřit v menu *Help*, položka *About Eclipse SDK*. Otevřený dialog musí obsahovat modrozelenou ikonu detailu hardwarové desky (viz Obrázek 8). Kliknutím na tuto ikonu máte možnost revidovat všechny nainstalované zásuvné moduly a další závislosti VLAM Feature.

Obrázek 8: Ověření nainstalovaného VLAM Feature v dialogu About Eclipse SDK

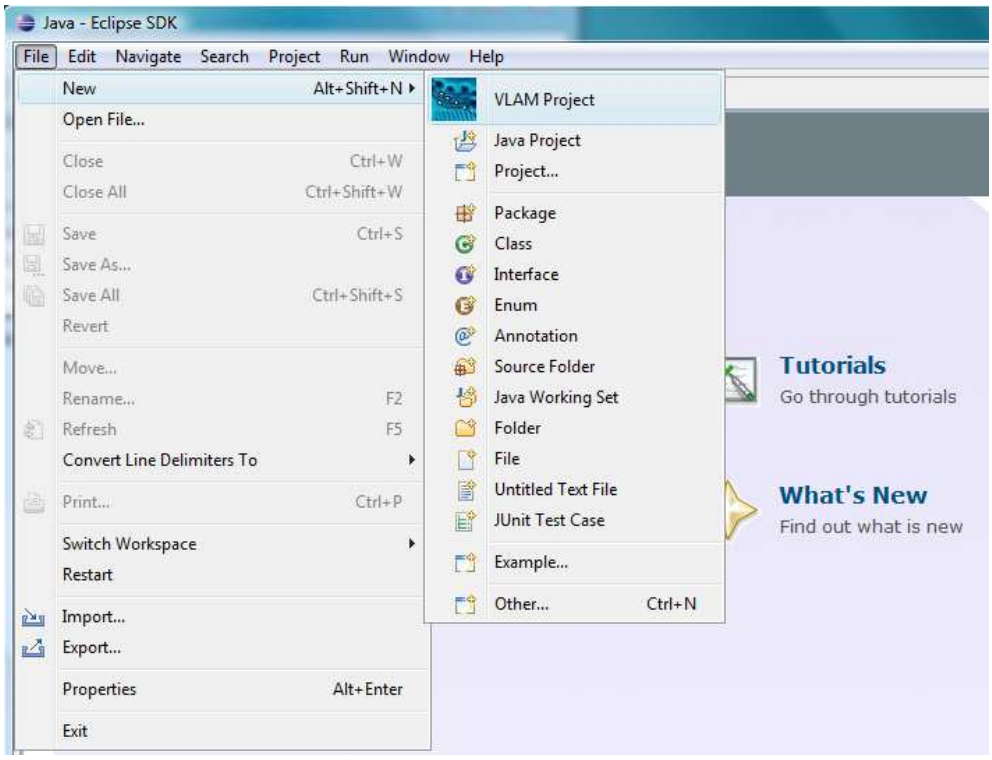
10. Nyní je VLAM IDE řádně nainstalováno a můžete začít pracovat.

Vytvoření projektu

V Eclipse se vaše práce člení na projekty. Každý projekt lze nezávisle překládat, má vyhrazen speciální adresář ve Workspace a má své vlastní specifické nastavení např. dle typu projektu. Podle nainstalovaných zásuvných modulů umí Eclipse pracovat s různými typy projektů (například knihovna v jazyce Java, program v jazyce C/C++, webová aplikace). VLAM IDE obsahuje podporu pro jeho vlastní typ projektu, *VLAM Project*. Tento projekt slouží pro souběžný návrh hardware a software a má tedy specifické vlastnosti (např. generování HDL kódu nebo překlad), které jsou navíc specializovány pro malé vestavěné systémy.

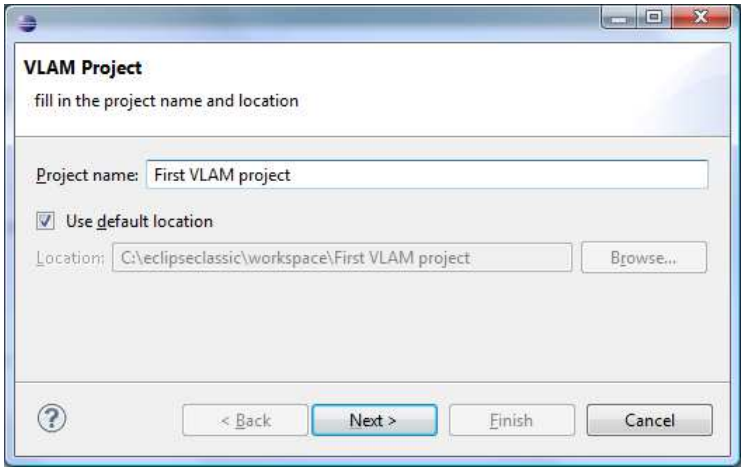
Nejjednodušším způsobem, jak vytvořit nový projekt, je využít nabízeného průvodce vytvořením nového projektu, jak ukazuje následující návod.

1. V menu *File* zvolte položku *New* a následně položku *VLAM Project*. (viz Obrázek 9)

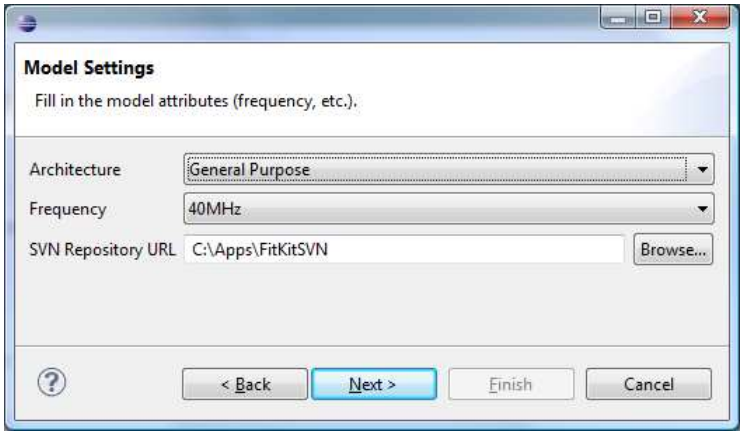


Obrázek 9: Spuštění průvodce vytvoření nového VLAM projektu

2. Nyní postupně projdeme průvodcem a vyplníme a nakonfigurujeme požadované vlastnosti projektu. Nejprve zadáme unikátní jméno a umístění projektu (viz Obrázek 10). Umístění se provádí implicitně do podadresáře dle jména projektu v aktivním *Workspace*. Pokračujeme stiskem tlačítka *Next >*.
3. Další dialog průvodce požaduje provést výběr architektury (General Purpose), požadované frekvence výsledného návrhu a adresář, kde je umístěn repozitář komponent (resp. jejich HDL reprezentace potřebná pro budoucí syntézu), jak demonstruje Obrázek 11.

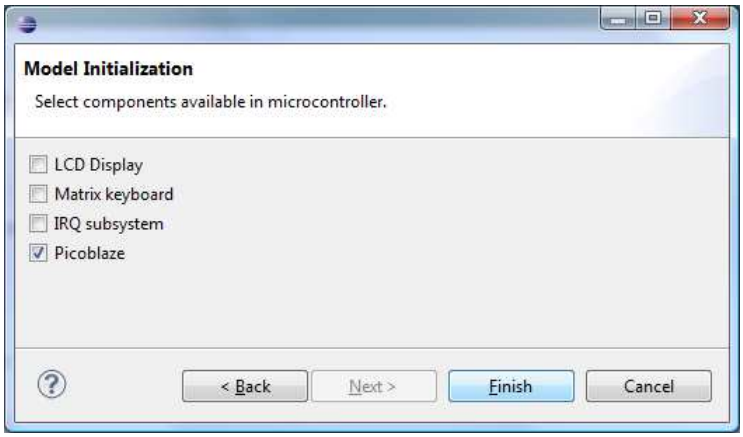


Obrázek 10: Zadání jména a výběr uložení



Obrázek 11: Výběr architektury, frekvence návrhu a umístění repozitáře implementací komponent

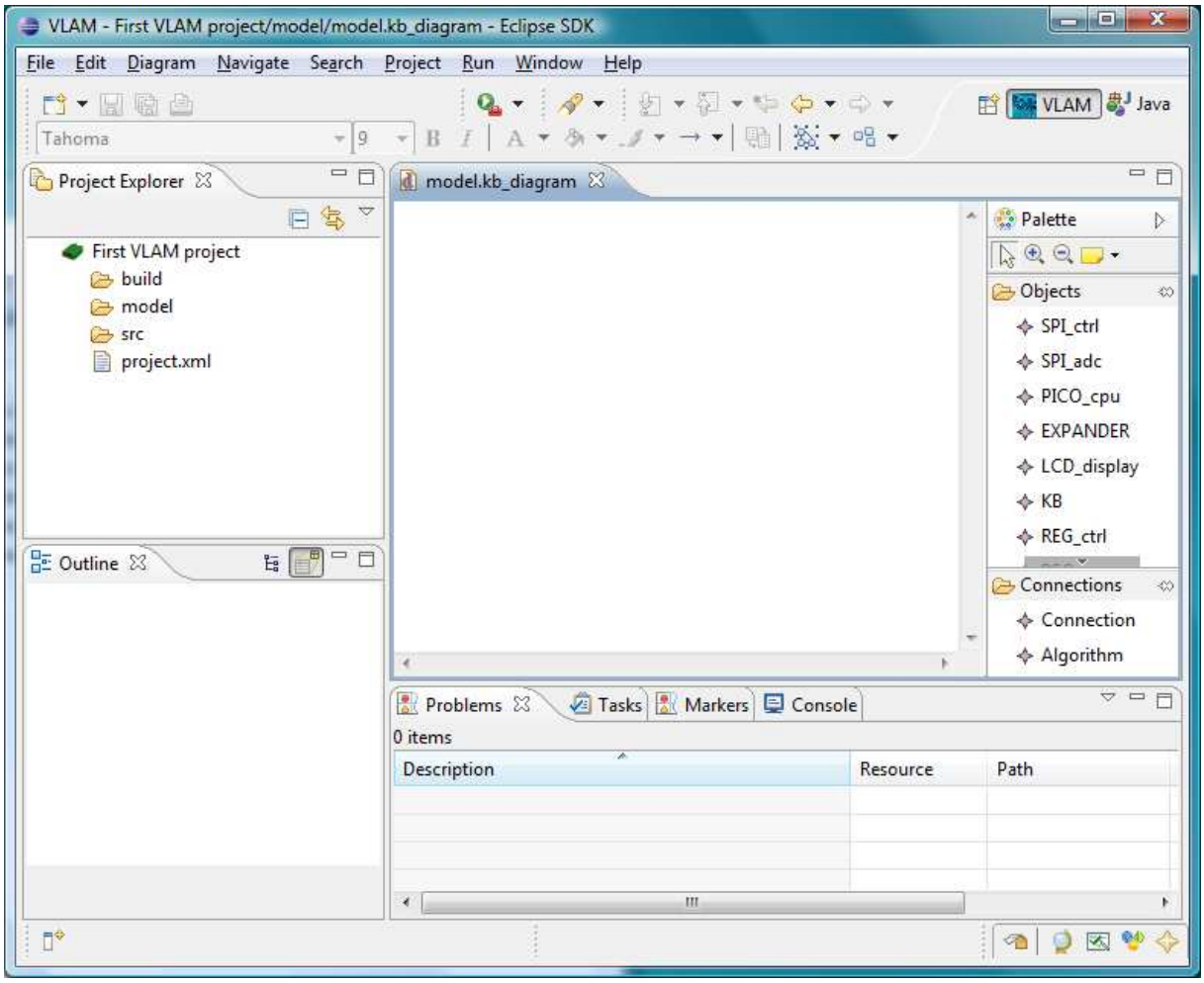
4. V poslední části průvodce (viz Obrázek 12) vybíráme komponenty, pro než vyžadujeme podporu při generování softwarové části kódu (např. podpora soft-core procesoru PicoBlaze).



Obrázek 12: Výběr komponent pro lepší podporu generování kódu v projektu

5. Průvodce dokončíme stisknutím tlačítka *Finish*. Vytvořit se projekt, jeho vnitřní struktura, vygenerují překladové skripty a případně i zdrojové kódy pro softwarovou část. Nový projekt je otevřen a prostředí Eclipse (resp. VLAM IDE) je přepnuto do perspektivy VLAM, která nabízí mnoho nástrojů pro podporu souběžného návrhu

hardware a software (angl. HW-SW codesign). Více o perspektivě VLAM v následující kapitole.



Obrázek 13: Nově vytvořený projekt v perspektivě VLAM

Chceme-li začít editovat hardwarový diagram, jak demonstruje Obrázek 13, musíme v pohledu *Project Explorer* rozbalit položku *model* a dvakrát kliknout na zdroj `model.kb_diagram`, což otevře požadovaný komponentní editor. Analogicky lze iniciovat editaci i dalších zdrojů projektu. **Poznámka**: Pro editaci zdrojových textů v jazyce C je zapotřebí doinstalovat *C/C++ Development Tooling*.

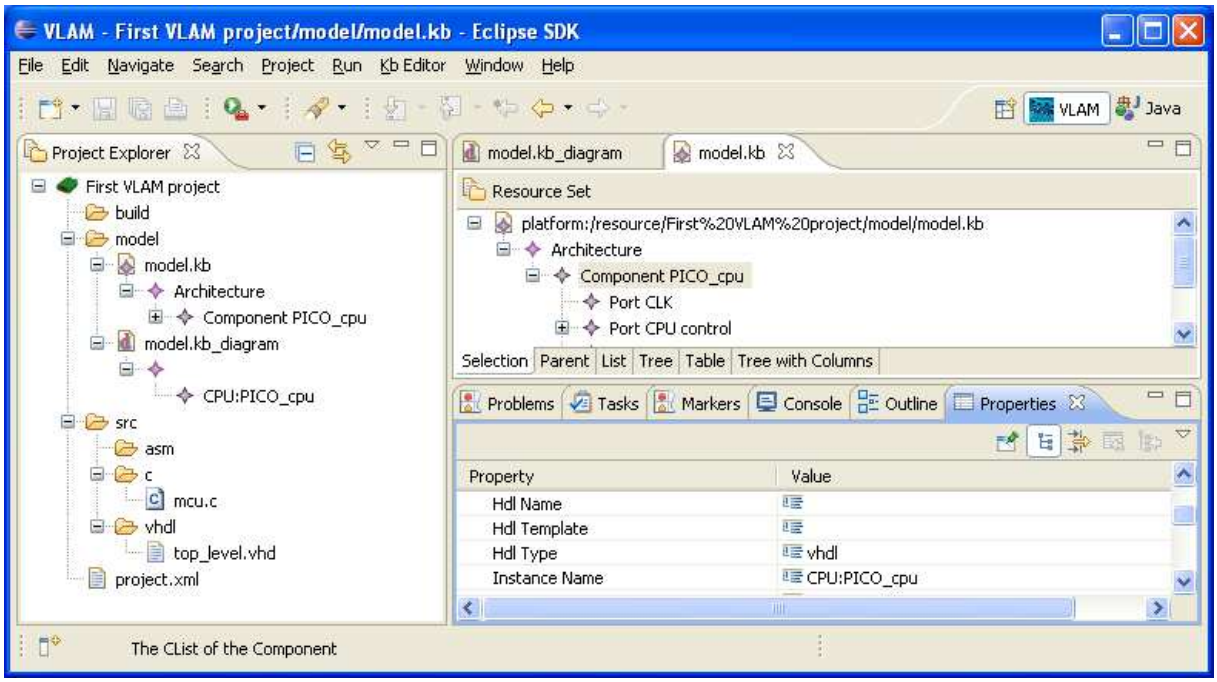
Struktura projektu

Po vytvoření obsahuje jinak prázdný projekt již nějaké soubory a vytvořenou adresářovou strukturu. Strukturu projektu můžeme sledovat v pohledu Project Explorer (viz Obrázek 14). Ve skutečnosti tento pohled zobrazuje celý pracovní prostor (Workspace), takže při vytvoření dalších projektů ve stejném prostoru budou kořenovými uzly právě tyto jednotlivé projekty (zelená ikona a v návěští zadaný název projektu). Tento pohled však skrývá některé soubory, které nejsou primárně určeny pro editaci uživatelem bez využití speciálních dialogů či editorů (např. vlastnosti projektu).

Nyní si popišme adresářovou strukturu detailněji:

- adresář **build** – vygenerovaná binární data po překladu a syntéze. Slouží k naprogramování hardware, logovací soubory

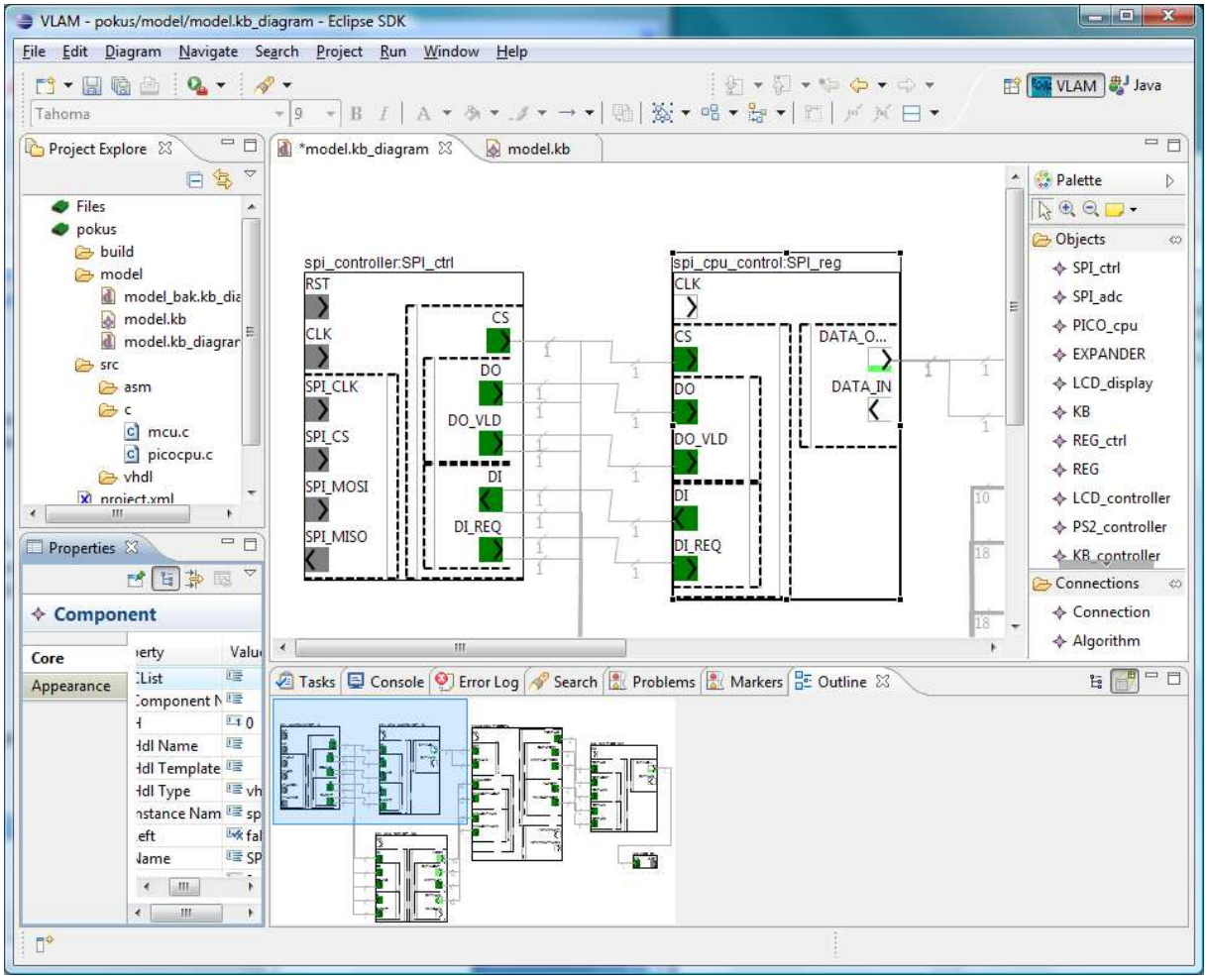
- adresář **model** – soubory reprezentující konfigurace (např. `model.kb`), diagramy (např. `model.kb_diagram`); všimněme si, že model (resp. konfigurace) lze dále prozkoumávat formou stromového zobrazení.
- adresář **src** – přidan vlastní zdrojové soubory, vygenerované zdrojové texty (HDL, ASM)
  - **asm** – kód v jazyce symbolických instrukcí
  - **c** – kód jazyka C
  - **vhdl** – VHDL kód
- *.project* – základní nastavení projektu pro Eclipse
- *project.xml* – detailní nastavení projektu využívané pouze projektem
- *Makefile* – vygenerovaný překladový skript



Obrázek 14: Vlevo pohled Project Explorer s rozbalenou strukturou nově vytvořeného projektu s jednou komponentou, vpravo nahoře strukturovaný výpis modelu projektu (model.kb) a vpravo dole pohled Properties sloužící pro editaci vlastností aktuálně vybrané entity.

Perspektiva

Pro lepší rozložení požadovaných oken, pohledů a editorů je ve VLAM IDE vytvořena speciální perspektiva VLAM. Jejím výběrem se automaticky nastaví základní rozložení oken a otevření nejčastěji používaných pohledů při vývoji.



Obrázek 15: Ukázka mnoha pohledu otevřených v perspektivě VLAM

Nástroje

Nástroje perspektivy VLAM lze rozdělit do několika kategorií:

- pohledy na modely a stav projektu
- editory zdrojových textů, konfigurací a diagramů
- překladače

Editory

Editor je zásuvný modul Eclipse, který se zabývá editací jistého typu zdroje. Většinou bývá editor asociován na nějakou příponu či jiný identifikační aspekt zdroje projektu (např. *.vhd pro otevření editoru VHDL). Nejčastěji se editují textové soubory (zdrojové texty) a grafické reprezentace (diagramy).

Komponentní editor

Toto je hlavní editor hardwarové části návrhu.

Editor jazyka symbolických instrukcí procesoru PicoBlaze

Editor jazyka symbolických instrukcí (angl. *assembly language*) slouží k podpoře vytváření a editaci zdrojových textů v jazyce symbolických instrukcí pro procesor PicoBlaze (verze 3). Podporovanými dialekty jazyka jsou syntaxe KCPSM3 a syntaxe vývojového prostředí pBlazIDE. Editor je vybaven sémantickým zvýrazňováním syntaxe jazyka symbolických instrukcí, validací zdrojového textu s výčtem chyb typickým pro prostředí Eclipse (pohled *Problems* či *Markers*), kontextovou nápovědou při editaci dokumentu a zobrazením základní sémantické struktury dokumentu v přehledovém okně (angl. *Outline*) prostředí Eclipse.

Editor jazyka VHDL

Eclipse Verilog Editor (vEditor, viz [2]) je zásuvný modul pro Eclipse, který dodává možnost editovat soubory s příponou vhd v editoru, který podporuje zvýrazňování syntaxe i doplňování kódu. Je tedy vhodné jej využít jak v případě úpravy, tak i pouhého studia vygenerovaných kódů pro popis hardware v jazyce VHDL. Jak název tohoto modulu napovídá, krom VHDL dokáže pracovat i s jazykem Verilog(IEEE-1364).

Editor jazyka C/C++

Součástí balíku C/C++ Development Tooling (CDT, [1]) je i pokročilý editor jazyka C, potažmo C++. Některé speciální aspekty jazyka C specifické pro vývoj pro vestavěné zařízení (a především procesor PicoBlaze-3) zatím nejsou tímto editorem zohledňovány, takže ne vždy lze spoléhat tvrzení o chybách tohoto jinak velmi komfortního editoru. Na vylepšení této situace usilovně pracujeme, ale modifikace takto složitého projektu vyžaduje mnoho studia a experimentů.

Pohledy

Pohled slouží pro grafickou reprezentaci zdrojů projektu (souborů, dat, textů, obrázků, diagramů atd.). Pohled je vlastně záložka vývojového prostředí, která v grafickém uživatelském prostředí Eclipse umožňuje měnit svoji velikost umístění a viditelnost. Není to však klasické modální dialogové okno.

Eclipse obsahuje řadu obecných pohledů, které mohou být přizpůsobovány každému typu projektu na míru. Pokud některý pohled není v aktivní perspektivě zobrazen, lze využít menu *Window* a položku *Show View >*, kde lze ze seznamu vybrat libovolný další pohled ke zobrazení. Pokud není hledaný pohled v základním výpisu, můžete zvolit položku *Other…*, která zobrazí dialog s kompletním výpisem všech nainstalovaných pohledů.

Nyní krátce popišme nejzajímavější pohledy při práci na projektu typu VLAM.

Project Explorer

S tímto pohledem jsme se již setkali. Zobrazuje logickou strukturu projektu. Kromě samotných zdrojů projektu často zobrazuje i závislosti na jiných projektech či externích zdrojích. Zobrazení je formou stromu, kdy uzly mohou mít ikonu a popisek. Podstromy lze zabalovat a rozbalovat pro efektivní prozkoumávání celé struktury, která může být poměrně rozsáhlá.

Navigator

Tento pohled zobrazuje na rozdíl od Project Explorer fyzickou strukturu projektu, tedy soubory a adresáře v aktuálním pracovním prostoru. V tomto pohledu lze vidět i zdroje, které jsou uživateli v pohledu Project Explorer skryté, např. konfigurační soubory projektu.

Outline

Náhledový nebo též přehledový pohled umožňuje abstraktnější zobrazení aktivního zdroje. Například u diagramu je to grafická zmenšenina, ve které lze vyvolat přesun hlavního okna editoru diagramu. U zdrojových textů jsou to seznamy obsažených třídy, potažmo metod či jiných významných syntaktických prvků. Například u editoru jazyka symbolických instrukcí je to seznam návěští programu.

Properties

Zobrazení tabulky s vlastnostmi aktivní entity. Vlastnosti vybrané entity lze případně v tomto okně také editovat, a tím měnit model celého návrhu. Je tak možno upravovat i vlastnosti, které nemají vizuální charakter nebo nejsou dostupné jiným způsobem (například vnitřní jména komponent, šířky portů apod.). Skupiny vlastností mohou být sdružovány do nezávislých záložek podle typu vlastnosti.

Palette

Tento pohled je ve skutečnosti část komponentního editoru obsahující seznam komponent a propojení, které je možné umísťovat na návrhové plátno (do diagramu). Pokud Vám nevyhovuje jeho umístění přímo v editoru, je možné jej osamostatnit a přesunou kamkoli jinam.

Console

Konzole slouží jako textové rozhraní s externími programy příkazové řádky. Především většina integrovaných překladačů může komunikovat propojením standardního vstupu a především standardního výstupu (případně ještě standardního chybového výstupu) na tuto konzoli. V jednom okamžiku lze mít otevřeno více konzolí a tedy komunikovat s více externími aplikacemi příkazové řádky současně.

Problems, Tasks, Markers, Error Log, Search

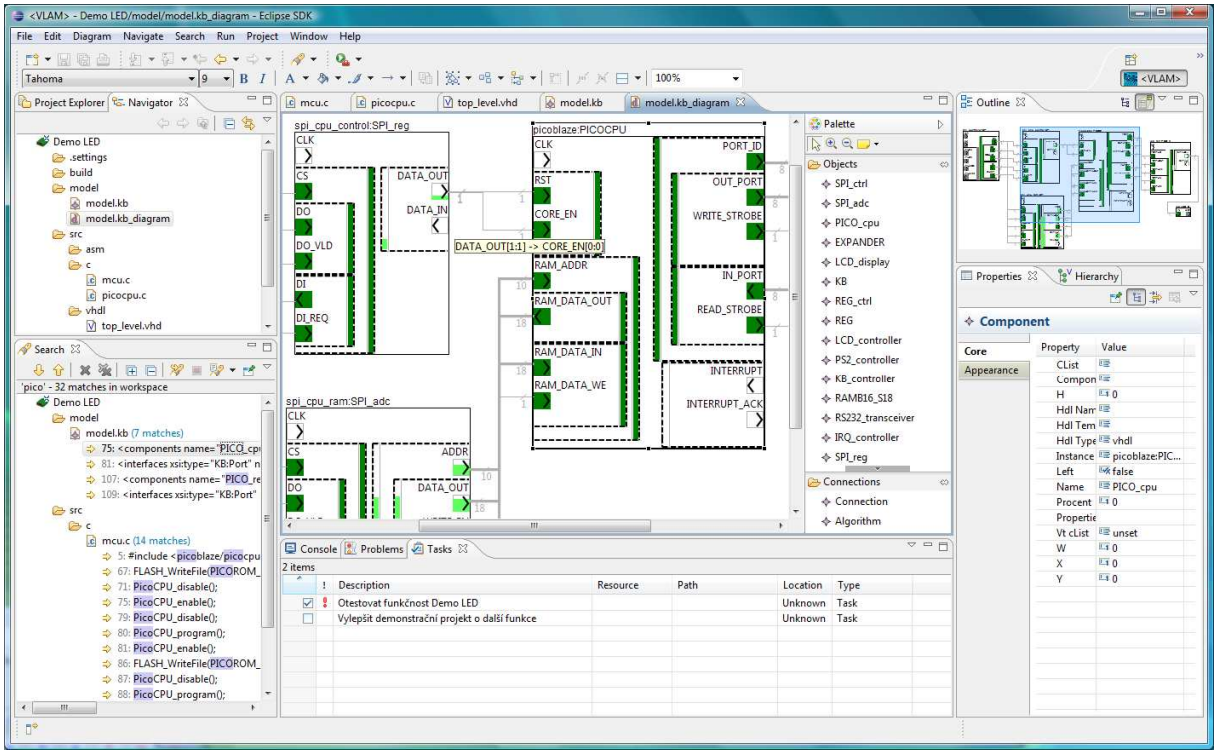
Tyto pohledy jsou tradiční pro všemožná vývojová prostředí. Všechny se zaměřují na pohodlnější práci a vývoj a různým způsobem informují o aktuálním stavu projektu. Chyby a varování jsou zaneseny do pohledu *Problems*. Úkoly a jejich plnění lze udržovat v pomocníkovi *Tasks*. Různá upozornění a doporučení se mohou zobrazovat s pohledu *Markers*. Chybové výpisy se často pro později zpracování uchovávají v souborech s příponou log. Tyto soubory lze dodatečně analyzovat v pohledu *Error Log* (kde je možno také importovat externí logovací soubor). Poslední zmíněný pohled, *Search*, slouží pro přehledné stromové zobrazení výsledku hledání v celém projektu, či jeho části.

Překladače

VLAM Feature implementuje a s tímto typem projektu váže jeden překladač (angl. *Builder*) pro prostředí Eclipse, který zajišťuje spuštění celého procesu překladu, syntézy a případně i programování cílové platformy.

Komponentní editor

Studenti mají často problém zorientovat se ve velmi komplexních a zároveň univerzálních vývojových prostředích používaných v praxi jako je CodeWarrior nebo Xilinx ISE či EDK. Proto jsme představili komponentně založený editor, který je zaměřen na tvorbu malých řešení jednoduchých úloh, pomocí kterých se studenti seznamují se základními principy souběžného návrhu hardware a software. Po osvojení těchto principů by pak měli být schopni lépe chápat výše zmíněná pokročilá prostředí a dále zvyšovat svou kvalifikaci.



Obrázek 16: Ukázka komponentního editoru s rozpracovanou konfigurací

Grafický editor hardwarového návrhu (tzv. konfigurace) má dvě části. První je paleta nástrojů a druhou samotná návrhová/kreslící plocha/plátno editoru (viz Obrázek 16). Paleta (angl. *Palette*) obsahuje dvě základní skupiny entit či nástrojů: (1) *Objects* (tzv. Komponenty) a (2) *Connections* (tzv. propojení nebo konektory). Komponenta se pro uživatele tváří jako částečně černá skříňka s několika málo konfigurovatelnými parametry (jméno, typ, generické parametry atd.). Tímto zvyšujeme abstrakci nízkoúrovňových detailů. Rozhraní komponenty je tvořeno *porty*, které tvoří hierarchii rozhraní. Porty lze zabalovat a rozbalovat (tzv. *virtuální porty*) a tak ještě více zvyšovat abstrakci při návrhu propojení mezi komponentami umístěnými na návrhové plátno.

Ovládání

Pro ovládání editoru je nejvhodnější kombinace myši a klávesnice. U všech akcí je možné využít volby *Undo*, příp. *Redo* (v menu *Edit*), které odpovídají českým ekvivalentům „*Zpět*“ a „*Znova*“.

Práce s komponentou

Základní pojmy (Některé jsou demonstrovány na Obrázek 17 a Obrázek 19.):

- *Entita* – libovolný grafický prvek v paletě nebo na plátně

- *Komponenta* resp. *instance komponenty* – grafický prvek obdélníkového tvaru; komponenta obsahuje množinu globálních portů (tj. porty na nejvyšší úrovni v rámci dané komponenty)
- *Port* – hierarchické rozhraní komponenty, porty na nejvyšší úrovni nazýváme globální. Porty mohou obsahovat libovolné množství podportů. Základní vlastnost portu je jeho šířka (množství signálů či pinů).
- *Virtuální port* – port, který obsahuje alespoň jeden podport. V rozbaleném stavu jsou reprezentovány pouze čárkovaným obdélníkem a ukazatelem obsazenosti (zelený pruh na vnitřním okraji). V zabaleném stavu se zobrazuje i jejich jméno (viz Obrázek 19)
- *Listový port* – na nejnižší úrovni jsou listové porty, které neobsahují žádné podporty a jsou reprezentovány malým čtvercem s popiskem. V případě zabalení jejich nadportu (je-li nějaký) dojde k jejich skrytí. Ukazatel zaplněnosti portu (kolik pinů je využitých) je v celém čtverci (nikoli pouze na vnitřním okraji jako u virtuálních portů).

Vlastnosti:

- Jméno typu komponenty – jméno, které je uvedeno v paletě
- Jméno instance komponenty – jméno, které je v rámci diagramu unikátní
- Barva – velmi důležitá vlastnost každého listového portu, demonstruje obsazenost jeho pinů či signálů. U virtuálního portu je tímto způsobem zabarven pouze vnitřní pruh.

Vložení nové komponenty na návrhové plátno

Vložení nové instance komponenty lze provést několika způsobu:

1) Dvojklik na požadovanou komponentu v paletě. Umístění na návrhové ploše bude vybráno automaticky.
2) Nejprve jedním kliknutím vybereme požadovanou komponentu v paletě a poté dalším klikem na požadované umístění na návrhové ploše komponentu vložíme.

Komponenta bude inicializována odvozeným jménem se zaručením unikátnosti pomocí číselné přípony a budou implicitně nastaveny její generické parametry.

Výběr instance komponenty

Na instanci komponenty na návrhové ploše jednou klikneme do oblasti pro uchopení komponenty (viz Obrázek 17).

Smazání (pozor na výběr)

Vybereme instanci komponenty na smazání a vyvoláme kontextové menu levým tlačítkem stisknutým nad oblastí pro uchopení komponenty (viz Obrázek 17). Vybereme příkaz *Delete from Model*.

Další akce nad komponentou

Otevřeme kontextové menu levým tlačítkem stisknutým nad oblastí pro uchopení komponenty (viz Obrázek 17) a vybereme požadovanou akci k provedení nad touto instancí komponenty (např. *Show Property View*).

Přesun komponenty

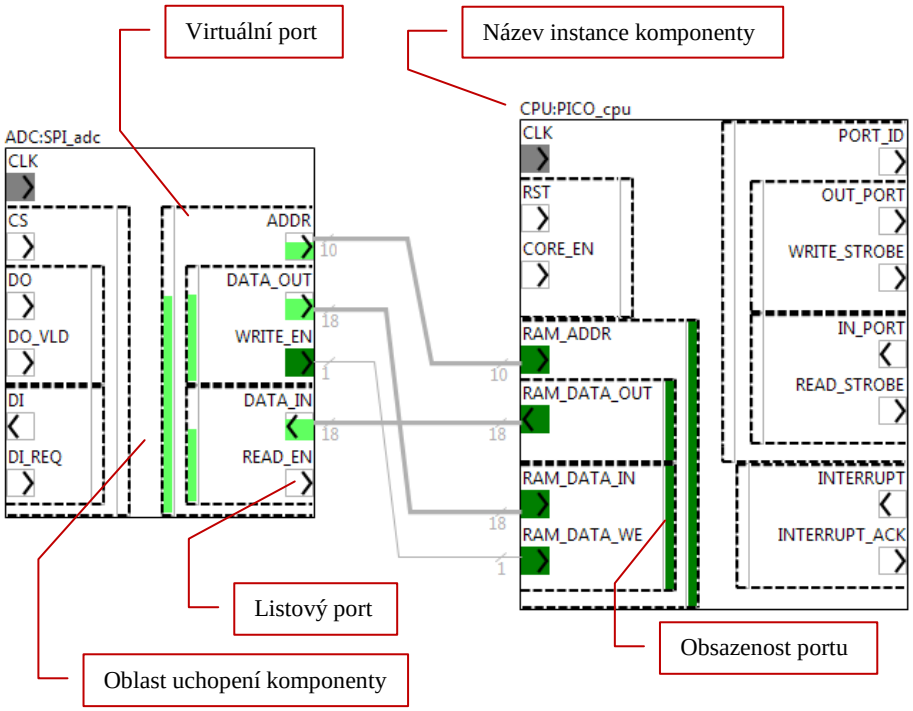
Instanci chytnu za oblast pro uchopení komponenty (viz Obrázek 17) a táhnu na novou pozici v návrhovém plátně.

Rozbalování a zabalování portů

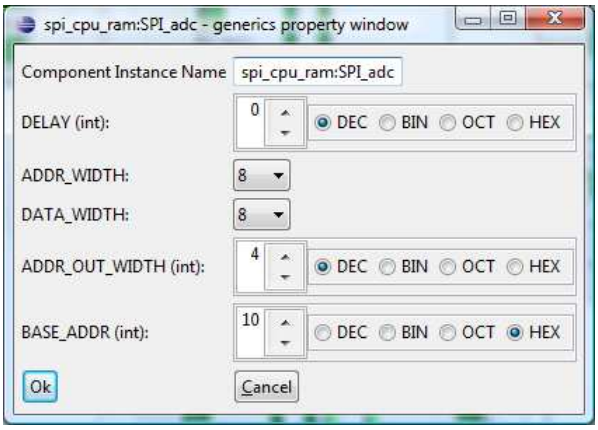
Pro rozbalení i zabalení virtuální portu klikni na vnitřní pruh zobrazující zaplněnost portu (viz Obrázek 19).

Editace generických parametrů nebo přejmenování instance komponenty

Proveďte dvojklik nad názvem instance komponenty a zobrazí se dialog podobný tomu na Obrázek 18. V případě, že odpovídající typ komponenty nenabízí žádné generické parametry, lze editovat pouze název instance.



Obrázek 17: Detail propojení dvou komponent včetně demonstrace (červeně) základních pojmů a značení



Obrázek 18: Dialog pro modifikaci generických parametrů komponenty

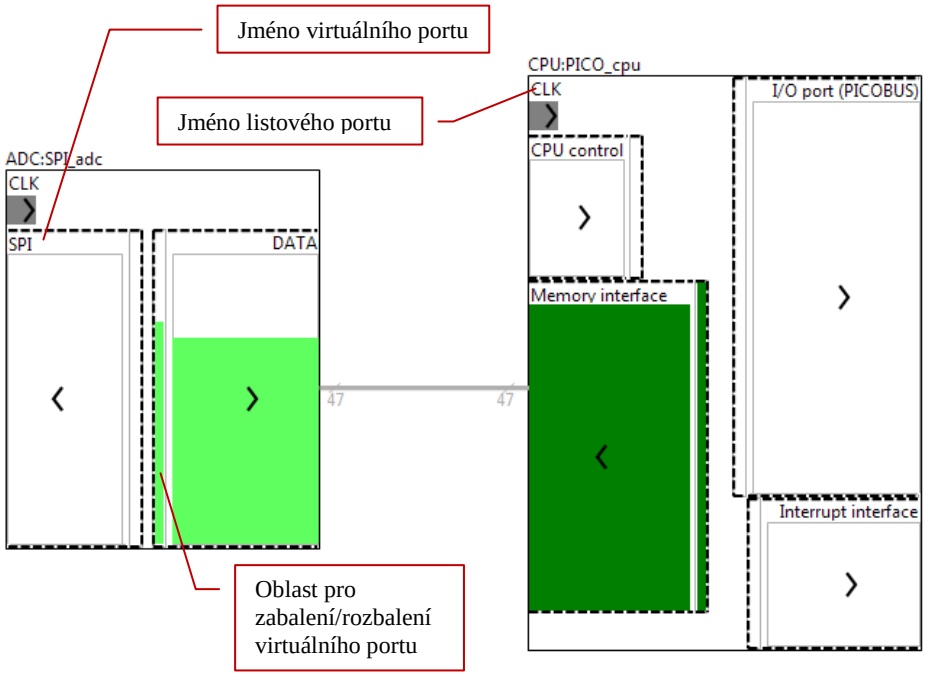
Propojení komponent (konektor)

Základní pojmy:

- Port – viz výše.
- Pin – minimální komunikační jednotka rozhraní (pro přenos signálu), nelze dále dělit, má jednotkovou šířku.
- Propojení (či konektor) – relace mezi dvěma porty různých instancí komponent v tomtéž diagramu. V diagramu je propojení zobrazeno různě tlustou linkou podle jeho šířky. Na okrajích má číselnou informaci o své šířce. Propojení také zobrazuje bublinovou nápovědu s pojmenováním zdrojového a cílového portu daného propojení.
- Typy propojení – Běžné (*Connection*), kdy uživatel vybírá zdrojový i cílový port ukazatelem myši nebo odvozované (*Algorithm*), které nabídne uživateli nejvhodnější propojení mezi vybranými komponentami nebo porty.

Vlastnosti:

- šířka propojení – té potom odpovídá tloušťka zobrazené linky
- zdrojový port, cílový port – těmto portům potom také odpovídá bublinová nápověda
- počáteční pin zdrojového portu a počáteční pin cílového portu



Obrázek 19: Detail propojení dvou komponent s vyšší úrovní abstrakce (zabalené virtuální porty)

Vytvoření propojení

V paletě editoru kliknu na typ připojení (tzv. nástroj *Connection*). Kliknutím vyberu zdrojový port a dalším kliknutím vyberu cílový port. Před druhý kliknutím mohu pozorovat průběžný náhled vznikajícího propojení. Není-li propojení validní, je u kurzoru myši zobrazena malá černobílá značka zákazu stání.

Manipulace s propojením

Propojení má význačné body, kde se linka láme. Tyto body jsou ve formě malých plných čtverečků. S tímto čtverečkem lze měnit vedení linky a případně kliknutím na linku přidat nový řídicí bod. Přetažení koncového bodu propojení na jiný port propojení změníme na nové.

Změna šířky propojení a dalších parametrů propojení

Dvojklikem na čísla informující o šířce propojení zobrazíme dialog (viz Obrázek 20), kde máme možnost změnit šířku propojení (kolik pinů bude obsazeno) a stanovit párování pinů zdrojového a cílového portu.

Obrázek 20: Konfigurace parametrů propojení (šířka, párování pinů portu)

Automatické odvození propojení

V paletě editoru je možné vedle nástroje *Connection* pro vytvoření propojení použít i nástroj *Algorithm*. Nástroj *Algorithm* slouží ke spuštění inferenčního algoritmu propojení komponent, který uživateli prostředí VLAM IDE napovídá propojení komponent označených v editoru návrhu schémat.

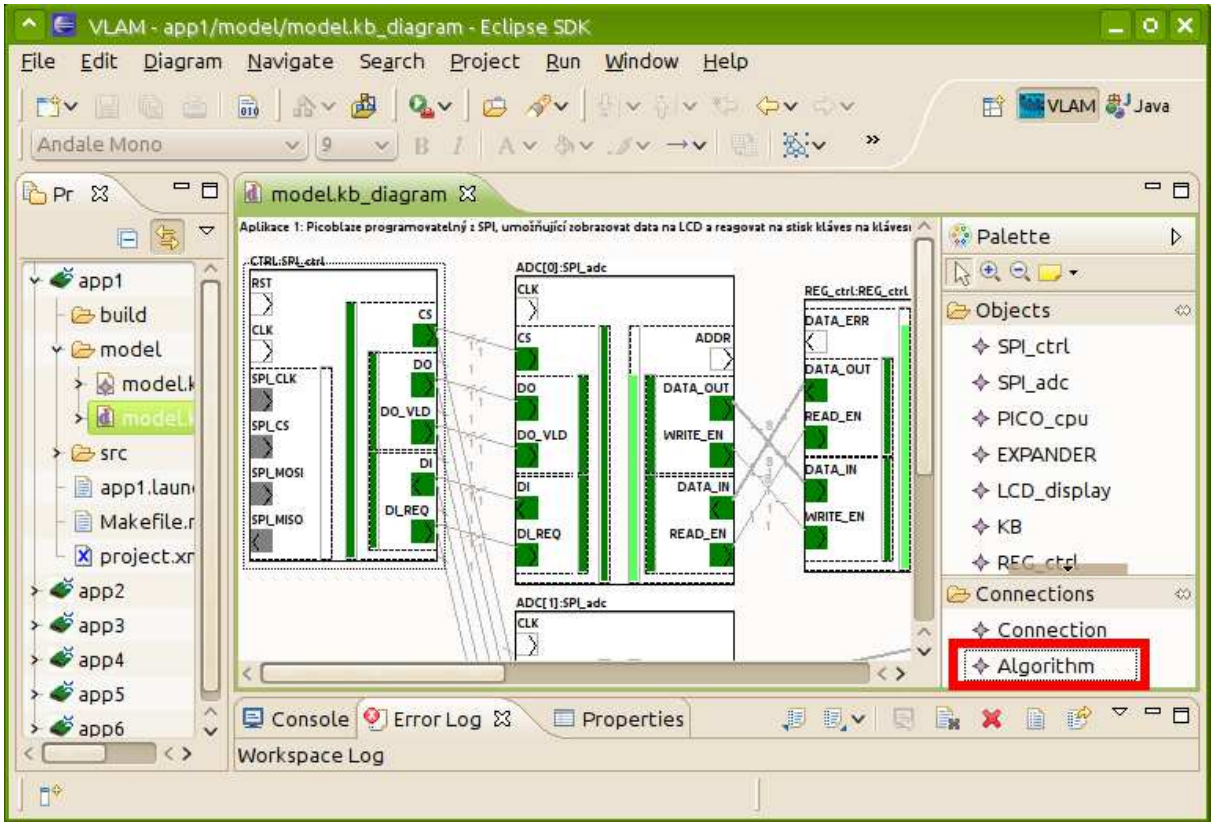
1. Spuštění algoritmu

Nástroj Algorithm se nachází ve VLAM IDE v paletě nástrojů skupiny Connections (viz Obrázek 21).

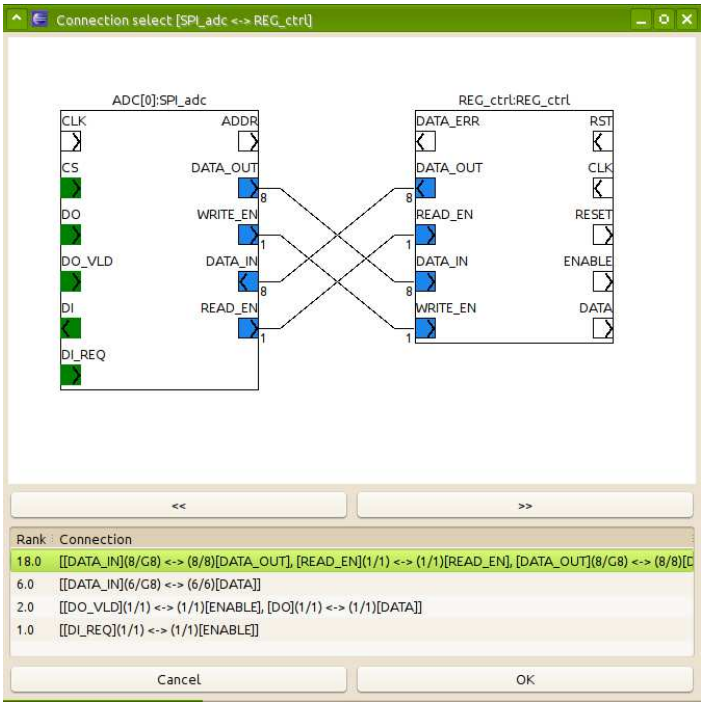
Po jeho aktivaci se kliknutím myši označí ve schématu zdrojová komponenta a tažením a uvolněním myši na druhé komponentě se spustí algoritmus pro tyto dvě vybrané komponenty. Je možné také specifikovat přímo některý port komponent. Aktuální výběr se uživateli znázorňuje jako rámeček kolem vybírané komponenty/portu. Kurzor myši se po aktivaci nástroje změní na kurzor indikující, zda lze komponentu či port, nad kterým se kurzor nachází, vybírat pro algoritmus.

2. Výběr a aplikace výsledku do schématu

Po spuštění algoritmu proběhne výpočet nad vybranými komponentami/porty a výsledek, pokud byl nalezen, se zobrazí uživateli ve formě grafického dialogu (viz Obrázek 23). Pokud nedošlo k nalezení žádného propojení, je uživatel informován informačním dialogem.



Obrázek 21: Nástroj *Algorithm* v prostředí VLAM IDE (červený rámeček)



Obrázek 22: Grafický dialog pro výběr odvozeného propojení

Okno grafického dialogu je rozděleno do dvou částí. V dolní části jsou vypsána všechna nalezená propojení, mezi kterými uživatel přepíná výběrem řádků tabulky, či použitím navigačních tlačítek. Aktuální výběr se zobrazuje v horní části dialogu, kde jsou schematicky znázorněny vybrané komponenty a propojení mezi nimi dané výběrem. Uživatel potvrdí

výběr tlačítkem *OK* nebo odmítne výsledky tlačítkem *Cancel*. Dialog je také ovladatelný z klávesnice, je možné použít navigační šipky pro přechod mezi výsledky, klávesu *Enter* pro potvrzení výběru, nebo klávesu *Escape* pro odmítnutí odvozených výsledků. Po potvrzení se vybrané propojení přidá do vytvářeného schématu.

3. Úprava výsledku v editoru schémat

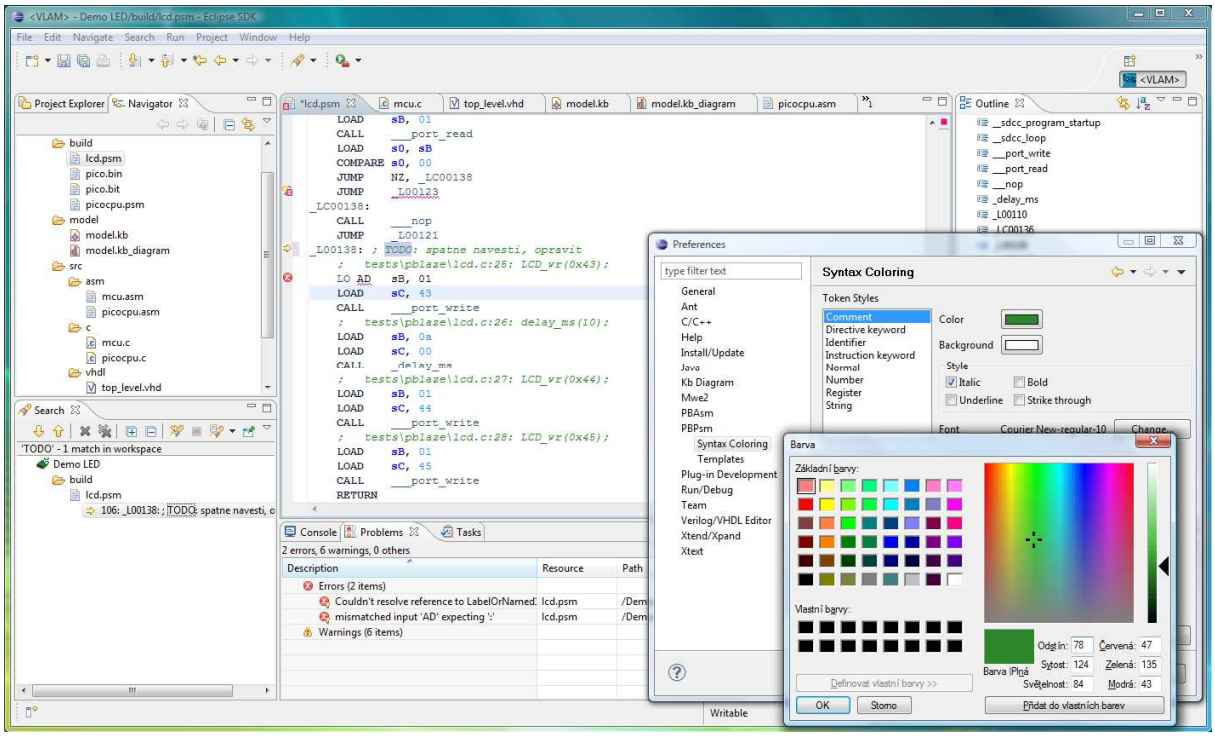
Uživatel následně může aplikované výsledky ručně upravit běžnými akcemi pro editaci propojení v editoru.

Editor jazyka symbolických instrukcí pro PicoBlaze-3

Editor jazyka symbolických instrukcí se aktivuje po otevření souboru s příponou asm (editor PBAsm pro syntaxi KCPSM3) nebo psm (editor PBPsm pro syntaxi prostředí PBlazIDE). Na Obrázek 23 vidíme editor s aktuálně otevřeným souborem lcd.psm. Na popředí je právě spuštěno nastavení editoru, konkrétně probíhá výběr barvy pro zvýraznění komentářů.

Editor rozšiřuje prostředí Eclipse o tyto funkce pro práci s asm a psm soubory:

- Zvýrazňování syntaxe – včetně nastavení fontů přes menu nastavení editorů v Eclipse.
- Přehledový pohled (*Outline*) – zobrazuje seznam návěští v editovaném souboru, a umožňuje tak lepší navigaci v kódu, kdy je možné se kliknutím v pohledu přemístit na místo výskytu v editovaném dokumentu.
- Kontrola syntaktické a částečně sémantické správnosti kódu – chyby jsou zobrazeny v chybovém pohledu (*Problems*), na Obrázek 23 se jedná o chybějící definici návěští a neplatný název instrukce.
- Kontextová nabídka s nápovědou při editaci – aktivuje se obvyklou zkratkou *Ctrl+Mezera*, nabízí doplňování instrukcí, registrů, definovaných návěští apod.

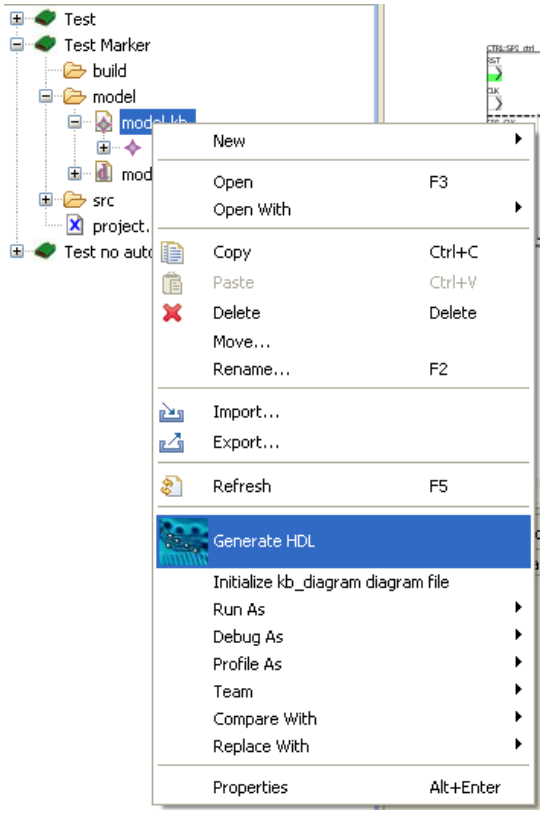


Obrázek 23: Ukázka editoru PBPsm včetně ukázky konfigurace zvýrazňování syntaxe (dialog *Preferences* a *Barva*)

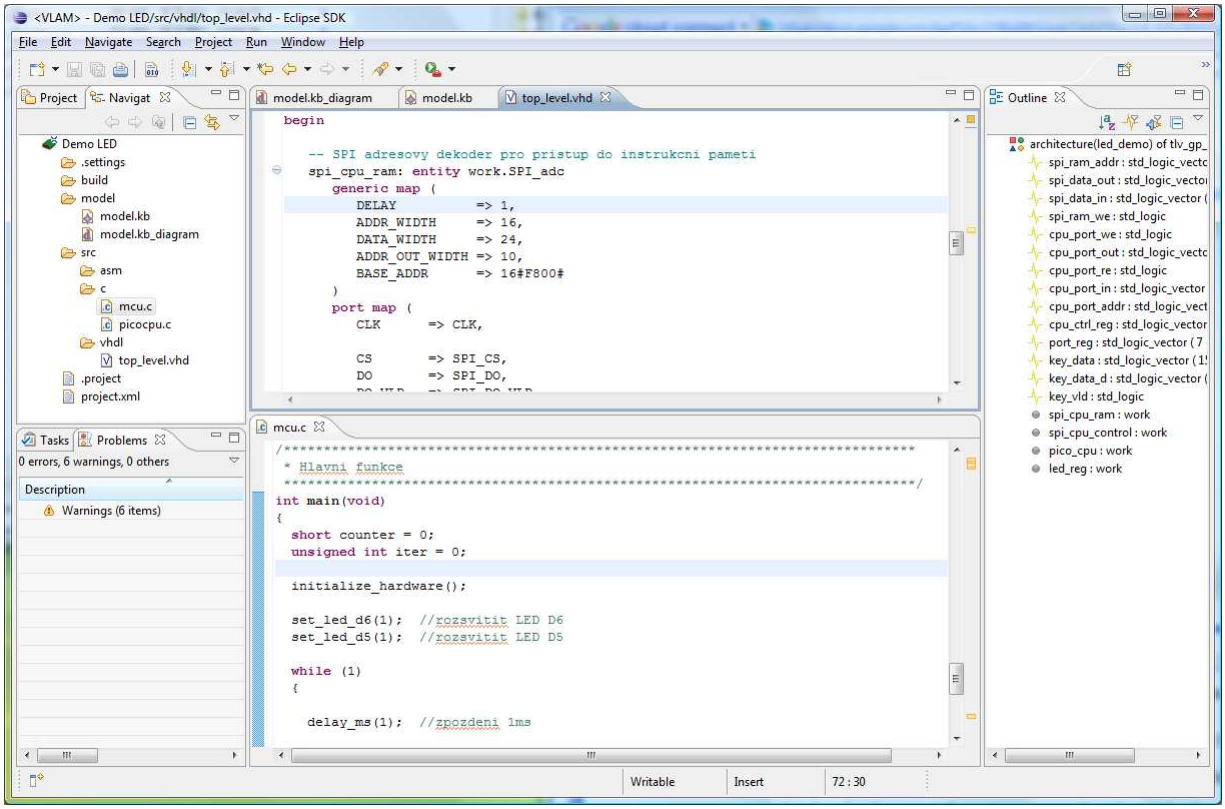
Generování kódu

Po dokončení editace výběrem z kontextového menu nad souborem modelu (přípona kb) spustíme akci *Generate HDL* (viz Obrázek 24). Dialogové okno nás informuje o výsledku. Jako výsledek se v projektu zobrazí soubor `top_level.vhd`, který obsahuje HDL kód popisující toto zapojení.

Pro jeho další editaci je součástí IDE editor podporující syntaxi VHDL (viz Obrázek 25).



Obrázek 24: Vygenerování HDL z hardwarového návrhu v kontextovém menu nad modelem propojení (model.kb)



Obrázek 25: Ukázka práce s Eclipse Verilog editorem (horní prostřední panel, *Outline* vpravo) na vygenerovaném VHDL (top_level.vhd)

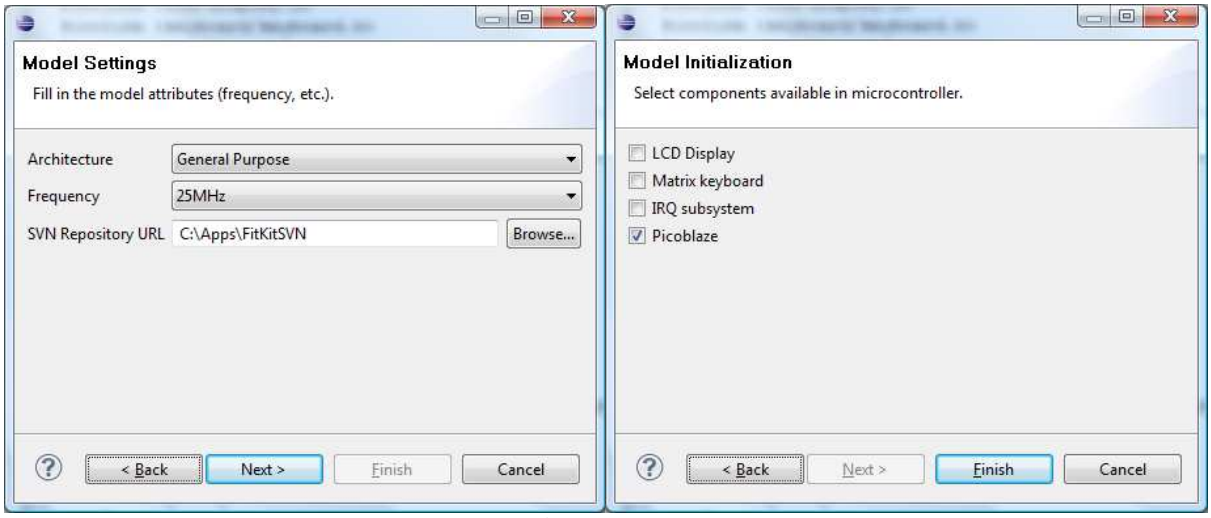
Příklad použití

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.

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 Obrázek 26, vlevo) máme možnost zvolit architekturu (General Purpose), umístění SVN repozitáře s HDL popisy komponent a frekvenci systému (např. 25 MHz). V následujícím dialogu (viz Obrázek 26, 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.



Obrázek 26: Záložky průvodce vytvořením nového projektu

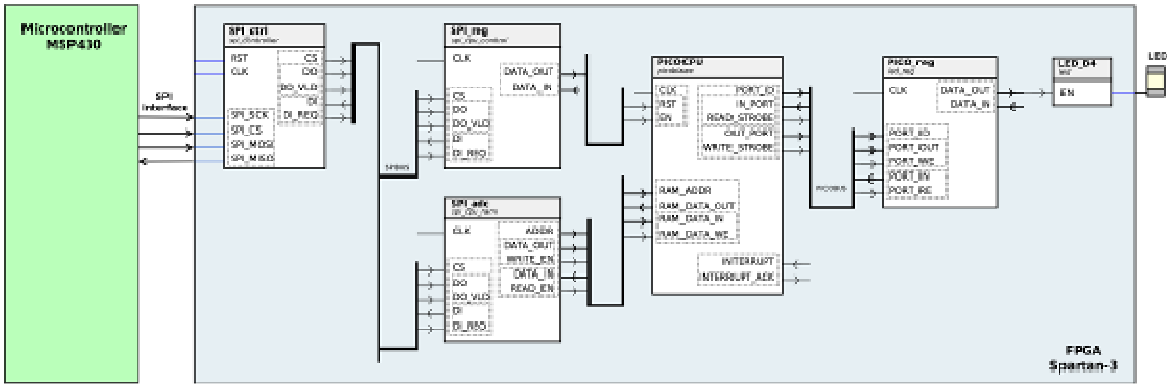
Návrh hardwarové části pomocí grafického rozhraní

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.

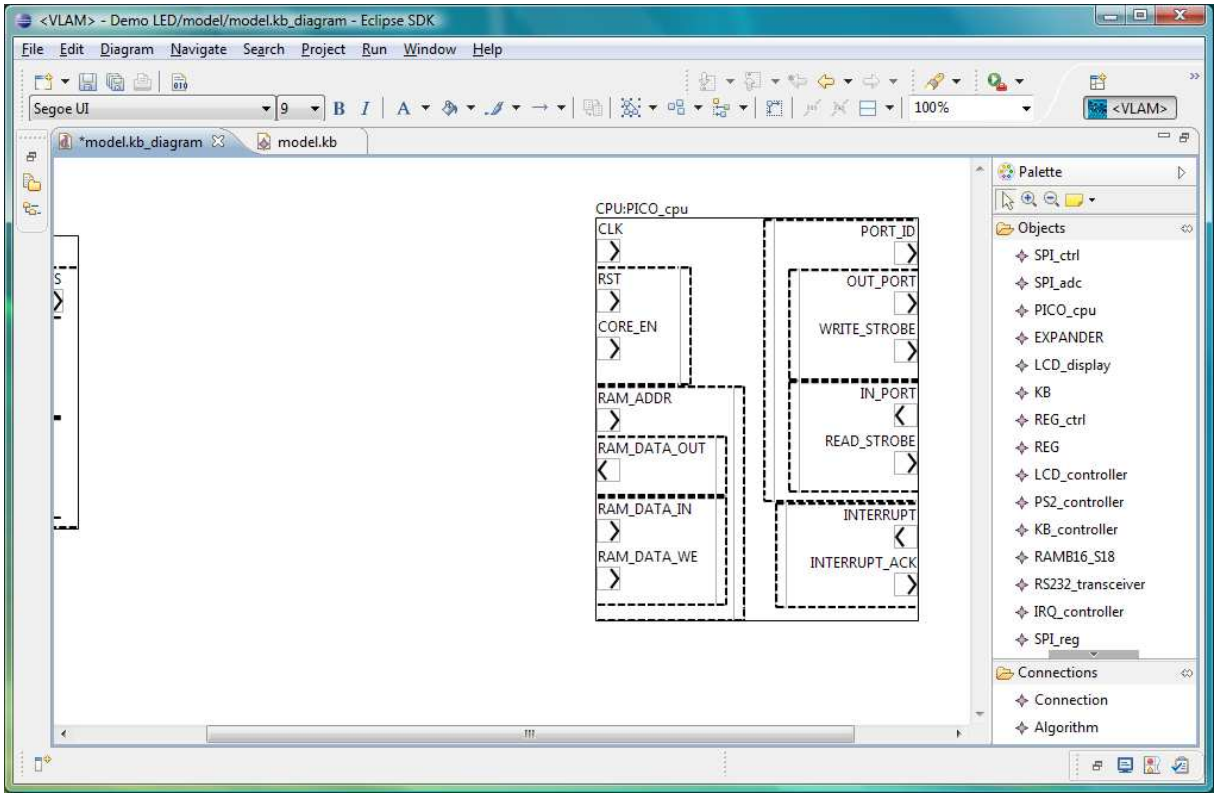
Podobně jako v případě SPI potažmo SPIBUS, byl i pro usnadnění návrhu aplikací vyžívajících PicoBlaze vytvořen jednoduchý sběrnicový systém PICOBUS, na který je možné připojit adresové dekodéry a registry s konfigurovatelnou datovou šířkou (opět vyžadují násobek osmi).



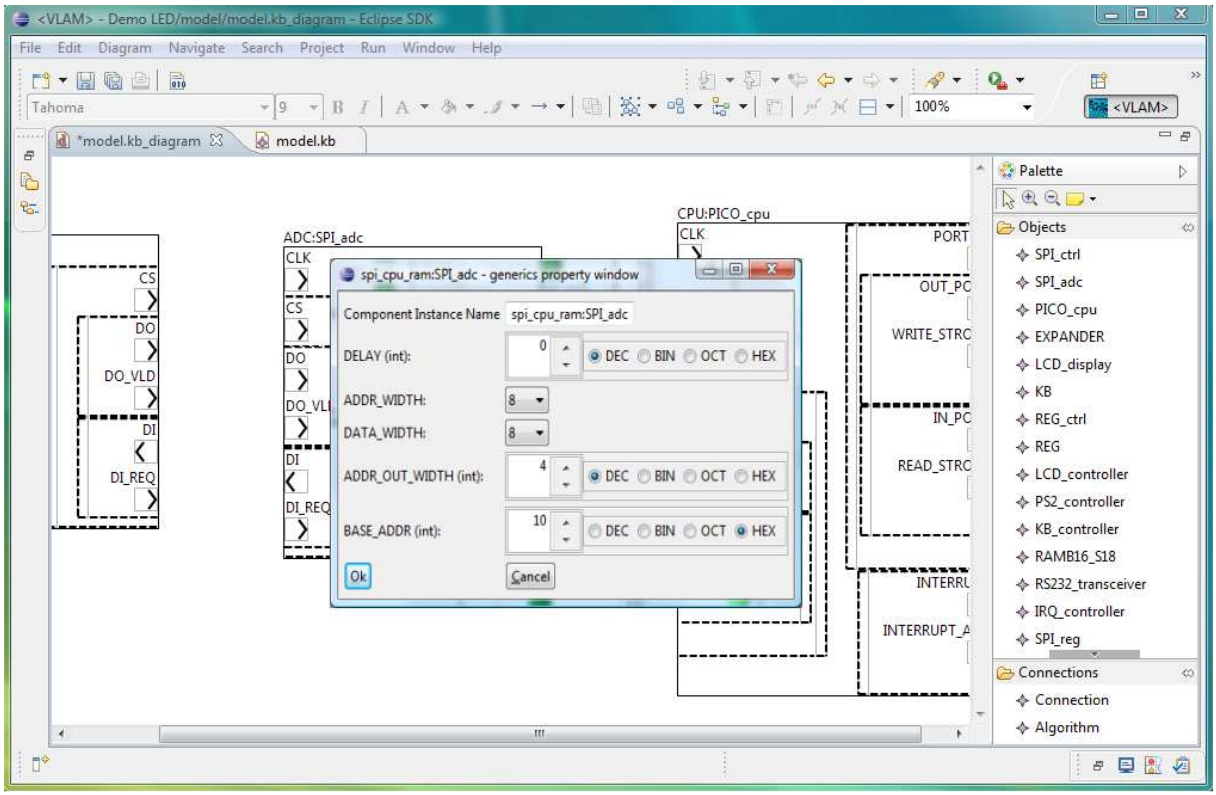
Obrázek 27: Schéma návrhu hardware pro FPGA

Postup návrhu v grafickém editoru pro vytvoření hardware na Obrázek 27:

1. Vložit *picoblaze* (komponenta v seznamu komponent je označena PICOCPU, viz Obrázek 28),
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 Obrázek 29). Propojit komponentu s PICOCPU.

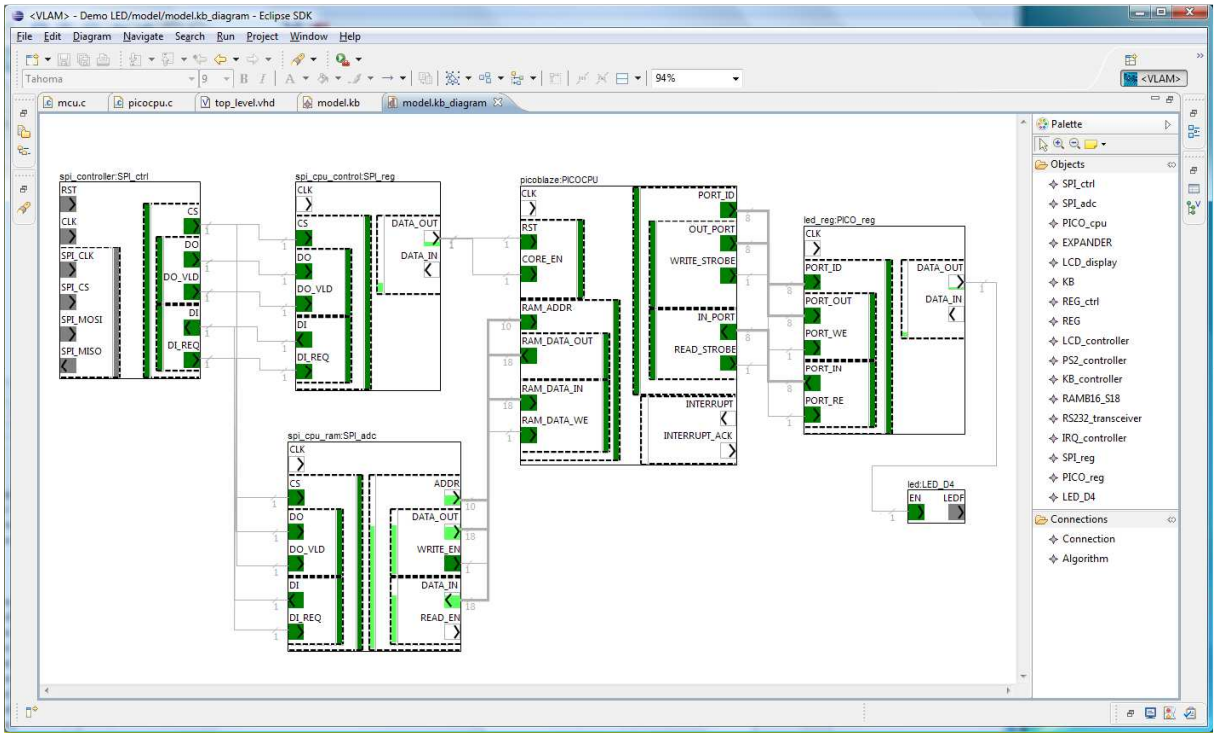


Obrázek 28: Ukázka vkládání komponent na návrhové plátno



Obrázek 29: Ukázka zadání generických parametrů pro SPI_adc

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.
4. Vložit a propojit PICO_reg, nastavit parametry BASE_ADDR=0xC0, DATA_WIDTH=8. Protože máme nezapojený vstupní port, tak pokud budeme 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 Obrázek 30.

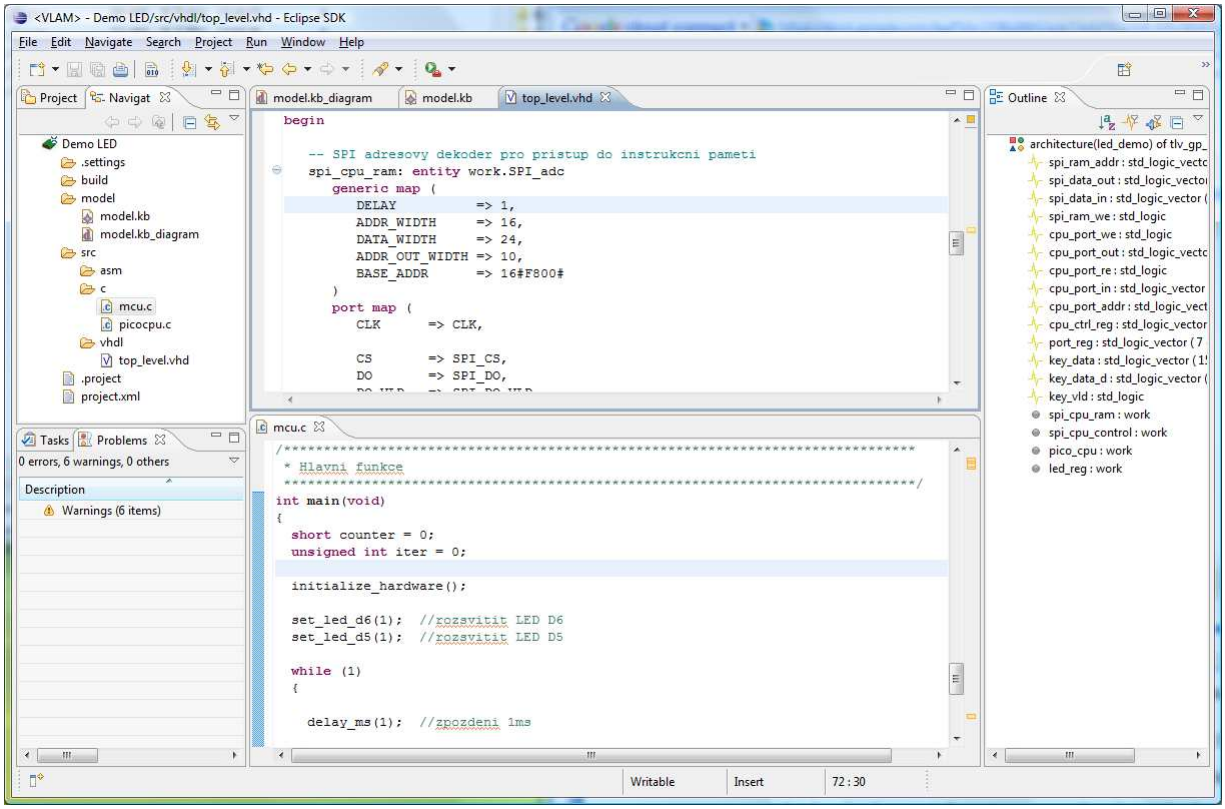


Obrázek 30: Celý návrh pomocí komponentního editoru

6. Skončil-li uživatel práci s grafickým editováním návrhu hardware, může vygenerovat odpovídající VHDL kód (`top_level.vhd`, viz Obrázek 31) a aktuální zdrojový soubor pro mikrokontrolér (`mcu.c`). Generování HDL se vyvolává položkou *Generate HDL* z kontextového menu zdroje `model.kb` v pohledu *Project Explorer*. Při přegenerování již existujícího návrhu bude uživatel dotázán na případný přepis původní verze.

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 softwarový 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`.



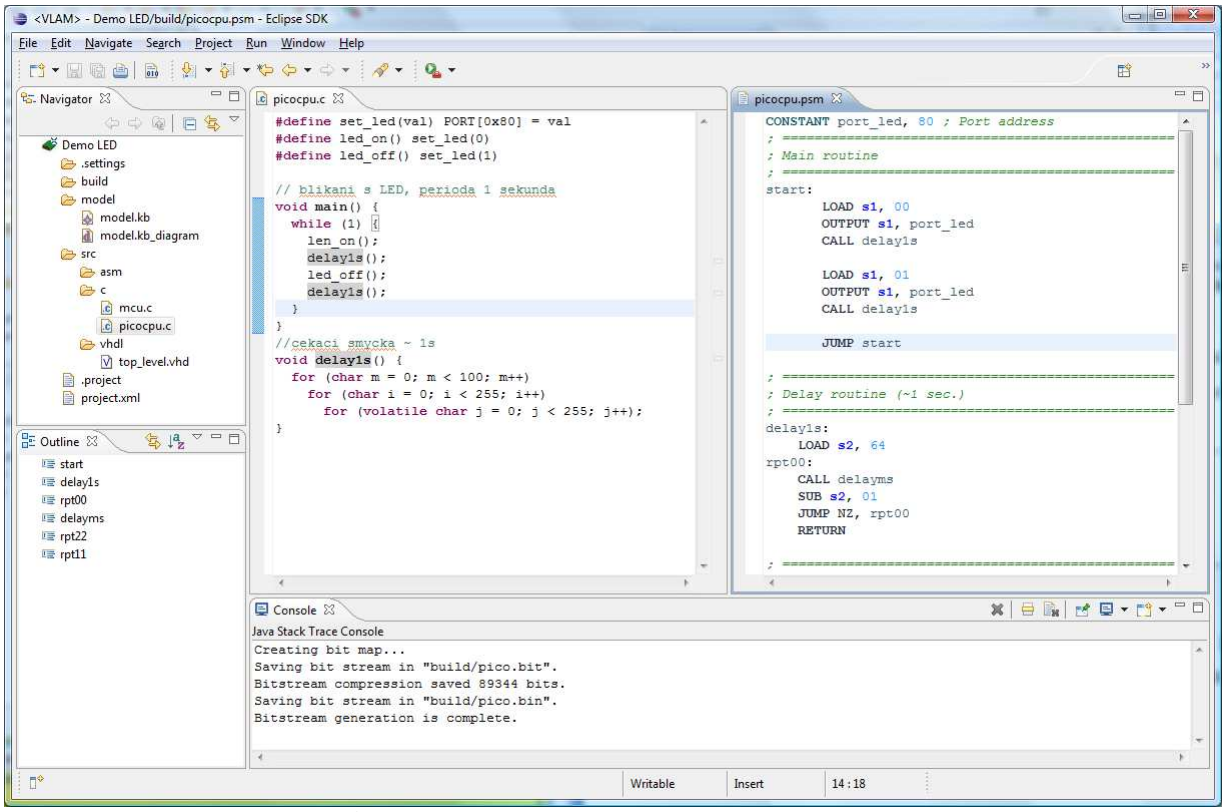
Obrázek 31: Vygenerované HDL (horní prostřední panel) a kód pro mikrokontrolér (dolní prostřední panel)

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 Obrázek 32). Po akci *Build* je možné zobrazit program pro PicoBlaze v jazyce symbolických instrukcí.



Obrázek 32: Ukázka editorů jazyka C (prostřední panel) nebo jazyka symbolických instrukcí (panel vpravo) včetně zobrazené konzole pro výpisy průběhu generování a překladu

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.

Závěr

Tato příručka popisuje ovládání integrovaného vývojového prostředí pro studenty postavené na platformě Eclipse.

V případě dotazů využijte kontakt na webových stránkách projektu VLAM a VLAM IDE. Vždy si také ověřte, zda pracujete s nejnovější verzí VLAM IDE i její nápovědy. Do budoucna bychom rádi zapracovali také integrovanou nápovědu do samotného prostředí a rozšířili množství tutoriálů.

Reference

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

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

[3] UNIS, s. r. o. et al. *VLAM*. Dostupné z WWW: *http://www.vlam.cz/*. [cit. 2011-08-07].