Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vývojové prostředí CycloneStudio

Bc. Lukáš Karlík

Diplomová práce
2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Lukáš Karlík**
Osobní číslo:	**I19280**
Studijní program:	**N0613A140007 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Vývojové prostředí CycloneStudio**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Cílem diplomové práce je návrh aplikace pro vytváření jednoduchých logických obvodů pro Intel Cyclone FPGA. Aplikace bude umožňovat nejen návrh vlastních schémat logických obvodů, ale také i vytváření uživatelských bloků pro opakované použití. Navržené schéma logického obvodu bude možné prakticky otestovat na zvoleném vývojovém kitu.
Aplikace pro svou činnost bude využívat vývojové prostředí Intel Quartus včetně předdefinovaných Verilog modulů.V teoretické části student provede rešerši podobných aplikací, dále bude podrobně popsán návrh vlastní aplikace. V praktické části student naprogramuje danou aplikaci a vytvoří ukázkové úlohy pro zvolený vývojový kit.

Rozsah pracovní zprávy: **50-60 normostran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

*JIVAN S. PARAB, RAJENDRA S. GAD a G. M. NAIK. Hands-on Experience with Altera FPGA Development Boards. Springer Verlag, 2017. ISBN 9788132237679.
*KOLOUCH, Jaromír. Jazyk Verilog a jeho užití při modelování a syntéze číslicových systémů: příručka. Brno: VUTIUM, 2012. ISBN 9788021445161.
*ŠŤASTNÝ, Jakub. FPGA prakticky: realizace číslicových systémů pro programovatelná hradlová pole. Praha: BEN – technická literatura, 2010. ISBN 9788073002619.

Vedoucí diplomové práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **6. listopadu 2020**
Termín odevzdání diplomové práce: **15. května 2021**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

prof. Ing. Antonín Kavička, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 30. listopadu 2020

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 18. 5. 2021

Bc. Lukáš Karlík

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat svému vedoucímu práce doc. Ing. Michaelu Bažantovi, Ph.D. a konzultantovi práce Ing. Miroslavu Dvořákovi, Dipl.tech. za cenné rady, pomoc a vstřícný přístup při vedení této diplomové práce. Děkuji také svým rodičům a kamarádům za trpělivost a podporu během studia.

ANOTACE

V diplomové práci je řešena problematika vlastního vývojového prostředí pro tvorbu logických obvodů pro vývojové kity. Výsledná aplikace bude sloužit jako grafický nástroj, tzv. diagram designer, pro tvorbu logických obvodů. Konkrétně se bude jednat o vývojové kity z rodiny Intel Cyclone FPGA. Aplikace bude pro svou činnost používat prostředí Intel Quartus a předefinované Verilog moduly. Dále bude umožňovat vytváření vlastních uživatelských bloků, projektů a praktické odzkoušení na vybraném vývojovém kitu.

KLÍČOVÁ SLOVA

Vývojové prostředí, logický obvod, Intel Quartus, vývojový kit, Verilog, C#, FPGA, Cyclone.

TITLE

CycloneStudio development environment.

ANNOTATION

In this diploma thesis addresses the issue of own development environment for the creation of logic circuits for development kits. The resulting application will serve as a graphical tool, the so-called diagram designer, for the creation of logic circuits. Specifically, these will be development kits from the Intel Cyclone FPGA family. The application will use Intel Quartus and predefined Verilog modules for its operations. It will also allow the creation of custom user blocks, projects, and practical testing on a selected development kit.

KEYWORDS

Development environment, logic circuit, Intel Quartus, development kit, Verilog, C#, FPGA, Cyclone.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

ARM	Advanced RISC Machine
CLB	Configurable Logic Blocks
CLI	Common Language Infrastructure
DC	Direct Current
DIP	Dual In-line Package
EEPROM	Electrically Erasable Programmable Read-Only Memory
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IDE	Integrated Development Environment
IOB	Input-Output Blocks
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
LUT	Look Up Table
PCB	Printed Circuit Board
SQL	Structured Query Language
SRAM	Static Random Access Memory
SoC	System-on-a-Chip
UML	Unified Modeling Language
VGA	Video Graphics Array
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

ÚVOD

Hlavním cílem diplomové práce je vytvoření funkční a uživatelsky příjemné aplikace pro tvorbu logických obvodů pro vývojové kity. Tato aplikace, která je nazvaná CycloneStudio, by měla sloužit uživatelům jako pomůcka pro seznámení s logickými operátory a sekvenčními obvody.

Pro tuto aplikaci byly zvoleny dva vývojové kity s FPGA z řady Intel Cyclone IV, a to konkrétně DE0-Nano a StormIV-E6. Ke kompilaci navrženého schématu na binární soubory určené k uploadu do FPGA bude použit kompilátor z vývojového prostředí Intel Quartus. Dále bude použit i programátor z daného vývojového prostředí. Obě tyto činnosti kompilace a uploud budou zautomatizovány prostřednictvím samotné aplikace.

Cílovou skupinou jsou pak především začátečníci, kteří nemají zkušenosti s logickými obvody. Těmto uživatelům pak bude umožněno vzdělávání v oblasti programování FPGA bez nutnosti se zároveň učit některý z jazyků HDL, a to buďto Verilog nebo VHDL. Tímto dojde k odstranění počáteční bariéry neznalosti základní syntaxe těchto jazyků.

Aplikace slouží jako grafický nástroj ve stylu diagram designeru, kde si uživatel může libovolně přidávat a vzájemně propojovat předdefinované moduly a vlastní vytvořené funkční bloky. Tyto moduly jsou graficky znázorněny na pracovní ploše aplikace. Uživatel si následně může propojovat vstupní a výstupní piny jednotlivých modulů. Základní pravidla pro spojování jednotlivých modulů pak obstará samotná aplikace. Jedná se především o propojení výstupního a vstupního pinu, nemožnost propojit piny na stejném modulu a znemožnění zapojení více výstupů do jednoho vstupu.

Informace o jednotlivých modulech jako jejich název, počet vstupních a výstupních pinů jsou načítány z hlaviček v předdefinovaných modulech. Aplikace obsahuje základní sadu modulů s možností jednoduchého přidání dalších. Samozřejmostí pak je vytváření projektů a vlastních bloků pro opakované použití.

Teoretická část práce se věnuje seznámení se s FPGA a také s prostředím Intel Quartus. Dále jsou v práci popsány použité vývojové kity a následuje část k základnímu seznámení s jazykem Verilog, který je používán k programování těchto kitů. Nakonec je zde provedena rešerše podobných aplikací a popsány požadavky na takovouto aplikaci.

V praktické části je detailně popsán návrh a také prostředí výsledné aplikace. Následuje popis některých Verilog modulů, a nakonec instalační a uživatelské příručky spolu s různými návody a ukázkovými úlohami.

1 PROGRAMOVATELNÁ HRADLOVÁ POLE

Tato část práce je zaměřena na vysvětlení pojmu programovatelná hradlová pole, jejich základní popis a vlastnosti. Dále je popsána volatilní a nevolatilní konfigurace FPGA. A také jsou vysvětleny soft procesory a hardwarové procesory ARM. Také budou zmíněny jazyky pro syntézu hradlových polí a výrobci hradlových polí.

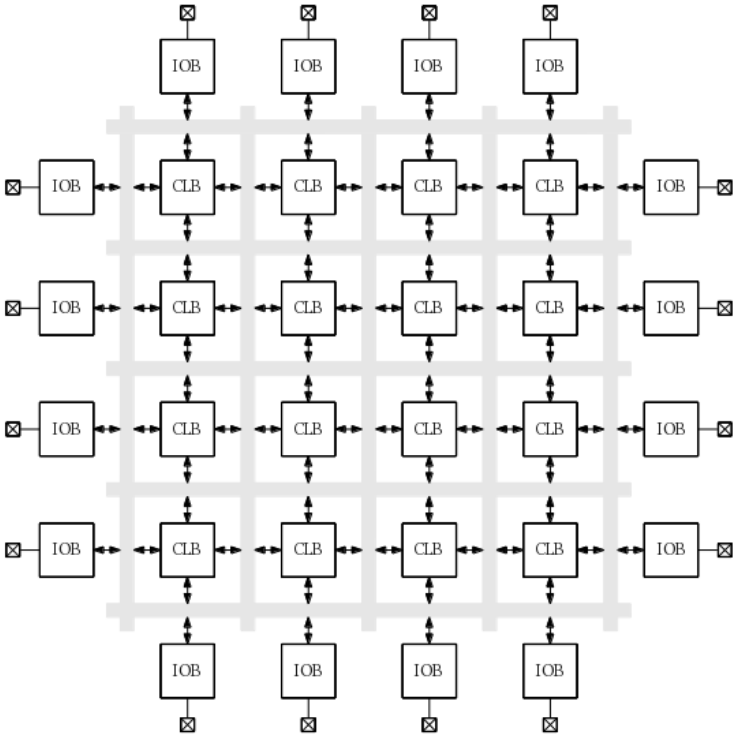
1.1 Popis FPGA

Programovatelná hradlová pole, nebo také FPGA, jsou speciální integrované logické obvody. Tyto obvody obsahují různě složité programovatelné bloky, které jsou propojené pomocí konfigurovatelné matice spojů. Právě programovatelnost je pak to, čím se tyto obvody odlišují od zákaznických integrovaných obvodů. Obvod totiž není nakonfigurován při výrobě, ale je nakonfigurován až u zákazníka (popř. zákazníkem), přímo pro řešení jeho specifického požadavku. [1]

FPGA jsou obvody s pravidelnou strukturou logických buněk, které jsou schopné realizovat jednoduché logické funkce. Propojováním těchto buněk pak lze získat rozsáhlejší a komplexnější funkce, k jejichž realizaci by jinak bylo zapotřebí použití mnoha různých obvodů. FPGA nejsou oblíbené pouze díky své programovatelnosti, dalšími důvody k jejich oblibě jsou například nízké výrobní náklady, možnost opakovaného použití a možnost za běhu měnit konfiguraci.

Díky své programovatelnosti mají FPGA obvody širokou škálu využití. Využívají se například v menší sérii navrhovaných zařízení, nebo třeba v oblasti prototypování složitějších návrhů zákaznických integrovaných obvodů, kdy FPGA umožňuje testovat navrhovaný systém už během samotného návrhu. [1]

Struktura typického FPGA obvodu se skládá ze základních logických buněk CLB, ty jsou propojeny pomocí konfigurovatelných přepínačů, také obsahuje vlastní spoje a programovatelnou matici, která je obklopena konfigurovatelnými vstupně výstupními buňkami IOB. Na obrázku č. 1 je možné vidět zjednodušenou strukturu charakteristickou pro FPGA obvody. Buňka obsahuje podporu pro implementaci kombinační logické funkce, která je realizována pomocí bloku LUT4. LUT je tabulka, která určuje, jakou úroveň bude nabývat výstup pro danou kombinaci vstupů. Jedná se tedy v podstatě o pravdivostní tabulku. Označení LUT4 udává 4 vstupovou tabulku. Moderní architektury pak často využívají i LUT s vícero vstupy. [2]



Obrázek 1 – Zjednodušená struktura obvodu FPGA [3]

Technologie, která je použita pro uložení konfigurace hradlového pole, je nejvýznamnější faktor pro výběr součástky pro výslednou aplikaci. Podle uložení konfigurace se rozlišují dva základní typy FPGA. První jsou FPGA s volatilní konfigurací a druhé FPGA s nevolatilní konfigurací. [1]

Programovatelná hradlová pole s volatilní konfigurací ukládají konfigurační informace do paměťových buněk, které jsou typu SRAM. Je třeba podotknout, že volatilní paměť má uloženou informaci pouze po dobu připojení ke zdroji energie. Tento typ FPGA má výhodu ve snadné konfiguraci a v rekonfigurovatelnosti i za běhu programu. Nevýhodou pak může být například to, že programovatelný obvod musí být po startu systému nakonfigurován, k čemuž je většinou potřeba externí paměť, což znamená, že je zapotřebí větší prostor na desce s plošnými spoji.

Programovatelná hradlová pole s nevolatilní konfigurací pak ukládají konfigurační bity v nevolatilních paměťových buňkách, typická je například flash paměť, EEPROM a antipojistky. Tyto paměti si uložené informace pamatují i bez napájení. Výhodou tohoto typu FPGA je vyšší odolnost proti radiaci a nižší spotřeba energie výsledným zařízením. Nevýhodou pak je složitější změna konfigurace.

Existují i taková FPGA, která jsou mezi oběma protipóly, což znamená obvody s SRAM konfigurací a flash pamětí integrovanou přímo v pouzdře FPGA obvodu (například Xilinx

Spartan 3AN nebo MachXO3LF od firmy Lattice). Výhoda kombinovaného přístupu je především ve zmenšení plochy a složitosti desky plošných spojů zařízení. [1]

Do FPGA se často implementují tzv. soft procesory, které z větší části nahrazují klasické mikrořadiče a umožňují tak větší flexibilitu návrhu. Architektura tohoto procesoru je plně implementována v programovatelných logických a paměťových blocích FPGA. Soft procesor je typ procesoru, který lze naprogramovat pouze pomocí logické syntézy. Tedy pouze za pomocí logických hradel. Nejjednodušší hradla, pomocí kterých lze napsat takový procesor jsou logické funkce AND, OR, NOT a XOR, kde jako paměť můžeme využít hradlo typu D, které je schopné v sobě uchovávat hodnotu. Není tedy problém si napsat vlastní soft procesor pomocí jazyka HDL. [4]

V poslední době se výrobci FPGA zaměřují na oblast SoC, např. Xilinx s řadou obvodů typu ZYNQ. Tyto obvody v sobě kombinují mikroprocesorové jádro, paměť, standardní periferie a programovatelnou logiku. Typicky se používá mikroprocesor z řady ARM. SoC používají při práci méně energie a zabírají méně místa a zjednodušují návrh zařízení. SoC jsou stále populárnější s růstem internetu věcí a mobilních počítačů. [5]

1.2 Jazyk pro syntézu hradlových polí

Logické obvody se v dnešní době skládají z propojených tranzistorů. Tyto obvody se navrhují pomocí hierarchických struktur. Tato hierarchická struktura nám umožňuje efektivně reprezentovat digitální obvody pomocí diagramů. Tento grafický přístup je intuitivní a vhodný pro začátečníky, ale je naprosto nevhodný pro složitější zapojení. Proto je vhodné logické obvody popisovat pomocí textového jazyka, který je k tomu konkrétně určen k jasnému a výstižnému popisu vlastností logického obvodu. [6]

Takové jazyky se nazývají hardware description language (HDL). HDL jazyky umožňují popsat logický obvod pomocí slov a symbolů a vývojový software pak následně může tento textový popis převést na konfigurační data. Ty jsou načtená do FPGA za účelem implementace požadované funkce. [6]

Hlavními jazyky, které se využívají pro programování obsahu hradlových polí, jsou VHDL a Verilog. Dříve se využíval i proprietární jazyk ABEL, který se dnes už nepoužívá. V praktické části práce je využit jazyk Verilog, z toho důvodu je mu v práci věnována samostatná kapitola a v této části práce je tedy popsán pouze stručně.

Verilog je hardwarový popisový jazyk, který se využívá k popisu digitálních systémů, jako jsou například mikroprocesory nebo síťové přepínače. Byl vyvinut za účelem zjednodušení

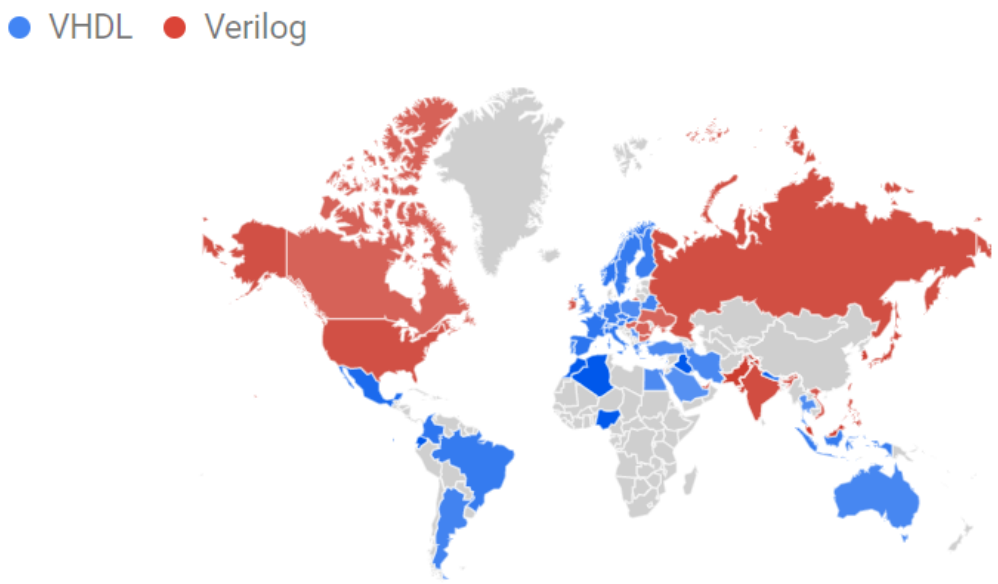
procesu a zvýšení odolnosti a flexibility HDL. Vytváří úroveň abstrakce, která pomáhá skrýt podrobnosti o jeho implementaci a technologii. [7]

VHDL patří mezi HDL jazyky, které jsou určené pro návrhy obvodů. Využívá se pro návrh a simulaci digitálních integrovaných obvodů, dá se využít i pro návrh analogových obvodů. Také se dá použít jako univerzální paralelní programovací jazyk.

Obrázek 2 – Porovnání Verilog a VHDL [7]

Oba tyto jazyky jsou velice populární. Porovnání zde není namístě, oba to jsou podobné jazyky a každý si může vybrat podle svých preferencí. Obecně platí, že VHDL je populárnější v Evropě a Verilog naopak v USA. VHDL je trošku víc podobný jazyku Pascal a Verilog zase připomíná C. VHDL je pak na rozdíl od Verilogu má přísnější pravidla syntaxe a jeho zápisy jsou podstatně složitější, viz obrázek č. 2. [5]

ABEL se stejně jako ostatní HDL jazyky používá k programování logických polí. Podporuje řadu vstupních forem chování včetně rovnic, stavových diagramů a tabulek pravdivosti. Jedná se o poměrně jednoduchý jazyk a ve svých vývojových prostředích ho podporují společnosti Xilinx a Lattice. Jak již bylo zmíněno výše, tento jazyk se využíval spíše dříve. Dnes převládají právě jazyky VHDL a Verilog.



Obrázek 3 – Graf vyhledávání VHDL a Verilogu na Googlu [16]

1.3 Výrobci hradlových polí

Mezi největší výrobce hradlových polí patří firmy XILINX, Altera/Intel a Lattice Semiconductor. V současné době Altera byla zakoupena firmou Intel a XILINX firmou AMD. Obě tyto akvizice jsou z důvodu rozšíření jednotlivých portfolií například v oblasti zpracování dat, strojového učení, datacenter a internetu věcí.

První bude popsána největší společnost, což je Intel. Tato firma v roce 2015 provedla akvizici společnosti Altera. Akvizice spojí produkty a výrobní procesy společnosti Intel s technologií Altera programovatelných hradlových polích. Toto spojení pak přineslo nové třídy produktů do portfolia FPGA Altery, především pak do segmentu datových center a internetu věcí. Současné nabízené řady FPGA jsou:

- Intel Max 10,
- Intel Cyclone 10,
- Intel Arria 10,
- Intel Stratix 10,
- Intel Agilex.

V současnosti Intel nabízí low-end FPGA z řad Intel Max a Intel Cyclone. Dále pak výkonnější FPGA s ARM procesorem z řad Intel Arria zaměřené na komunikaci a Intel Stratix se zaměřením na výpočetní výkon. Největší novinkou je řada Intel Agilex, která patří mezi nejvýkonnější FPGA nabízené Intelem určené do datacenter, kde je dbán důraz na aplikace náročné na propustnost a výpočetní výkon. [8]

Společnost Intel také nabízí své vlastní vývojové prostředí. Nazývá se Intel Quartus Prime a je dostupný ve třech edicích. Jsou to edice Pro, Standard a Lite. Kromě verze Lite jsou obě zbývající placené. Placené verze přinášejí rozšířené možnosti návrhového systému, například o prostředky ladění IP jader a také rozšířenou podporu. Pro běžné uživatele pak plně dostačuje Lite edice tohoto vývojového prostředí. Intel nabízí také svůj soft procesor. Nejnovější se nazývá Nios II a jde o 32bitového nástupce svého předchůdce se stejnou RISC architekturou. Tento procesor může být implementován na všech nových FPGA a SoC čipech od Intelu. Jedná se o nejrozšířenější soft procesor na světě. [8]

Společnost Xilinx, která je od října 2020 vlastněná společností AMD, je druhý největší výrobce FPGA čipů. I toto spojení proběhlo z důvodu vzájemného rozšíření portfolií, především pak o čipy Xilinx používané v zařízeních pro bezdrátové přenosové technologie. Nabízené řady společnosti Xilinx jsou:

- Xilinx Spartan 7,
- Xilinx Artix 7,
- Xilinx Kintex 7,
- Xilinx Kintex.

Řada Spartan je optimalizována jako I/O zařízení s ohledem na co nejnižší cenu. Artix je zaměřený na co největší propustnost při digitálním zpracování signálů. Kintex je zaměřen na co nejlepší poměr cena/výkon. A jako poslední pak řada Virtex určená pro nejvyšší výkon a kapacitu systému. [9]

Jako vývojové prostředí Xilinx nabízí Xilinx Vivado HL. Toto vývojové prostředí je nabízeno podobně jako u konkurence v placené verzi, tak i v neplacené. Neplacená verze se jmenuje WebPACK Edition a jedná se o velice omezenou verzi (obsahuje jenom některé FPGA). Placené verze jsou Design Edition a System Edition. Obě tyto verze jsou podobné a liší se především v několika drobnostech, například System Edition obsahuje nástroje pro PCB. Soft procesor nabízený společností Xilinx se nazývá MicroBlaze. Jedná se o 32 nebo 64bitový mikroprocesor s RISC architekturou. [10]

Poslední zde uvedenou společností je Lattice Semiconductor. Tato společnost má největší portfolio FPGA čipů ze všech uvedených. Vzhledem k velkému množství budou uvedeny pouze některé řady:

- Lattice Certus,
- Lattice CrossLink,
- Lattice iCE40,

- Lattice Mach.

Základní řadu FPGA tvoří obvody Certus-NX nebo EPC5. První uvedená řada obvodů je určena k obecnému použití a druhá je zaměřené na kompaktní aplikace, malé náklady a nízkou spotřebu energie. Dále tu jsou řady specializované na práci s videem nazvané CrossLink. I zde je možné volit tyto obvody dle výkonu a potřebných funkcí. Pro aplikace, kde je nutná nízká spotřeba pak slouží řada FPGA s označením iCE40, taktéž v různých výkonových verzích. Poslední řada FPGA s názvem Mach je pak zaměřená na řízení a bezpečnost. [11]

Vývojové prostředí dodávané společností Lattice je Lattice Diamond Software. Tento software dostupný jak v placené, tak v bezplatné verzi. Obě tyto verze se jmenují stejně a v zásadě se liší pouze dostupnými prostředky a podporovanými čipy. Nabízeným soft procesorem je LatticeMico32. Jedná se podobně jako u konkurence o 32bitový RISC procesor. Tento procesor je ovšem dodáván zdarma a pod open IP core licencí. [11]

2 POUŽITÉ VÝVOJOVÉ KITY

Tato kapitola bude věnována popisu použitých vývojových kitů, které byly použity k testování výsledné aplikace. Jedná se DE0-Nano a StormIV-E6. Oba tyto kity obsahují čip z řady Intel Cyclone IV E. První jmenovaný je zástupce firmy Terasic, která tyto kity vyrábí a jedná se o starší vývojový kit, stále však velmi oblíbený mezi uživateli. To je také důvod, proč se jeho cena drží na vyšší úrovni. Naproti tomu druhý kit je podstatně levnější. Jedná se o kit, který je navržen a vyroben v Číně. Jeho podstatnou nevýhodou je nedostatečná dokumentace a také to, že byl vyráběn jen omezenou dobu, ale nabízí stejné možnosti.

Čip řady Intel Cyclone IV E nepatří mezi nejnovější ani nejvýkonnější FPGA. Pro začátečníky jde však o dostačující a cenově dostupné řešení. Parametry této řady jsou ohledně výkonu 6272 až 114480 výpočetních logických prvků, 30 až 438 paměťových bloků M9K, 270 až 3888 Kbits vestavěné paměti a 179 až 528 I/O spojení. Všechny podrobné informace jsou v tabulce č. 1 níže. Informace ohledně velikostí samotných čipů jsou v tabulce č. 2. [12]

Zařízení	**EP4CE6**	**EP4CE10**	**EP4CE15**	**EP4CE22**	**EP4CE30**	**EP4CE40**	**EP4CE55**	**EP4CE75**	**EP4CE115**
Logické elementy	6 272	10 320	15 408	22 320	28 848	39 600	55 856	75 408	114 480
M9K paměťové bloky	30	46	56	66	66	126	260	305	432
SRAM (Kbits)	270	414	504	594	594	1 134	2 340	2 745	3 888
18 x 18 násobička	15	23	56	66	66	116	154	200	266
Phase-locked loops	2	2	4	4	4	4	4	4	4
I/O	179	179	343	153	532	532	374	426	528
Max. rozdílných kanálů	66	66	137	52	224	224	160	178	230

Tabulka 1 – Přehled rodiny Cyclone IV E FPGA [12]

Pouzdra	**E144**	**M164**	**M256**	**U256**	**F256**	**F324**	**U484**	**F484**	**F780**
Rozměry (mm)	22 x 22	8 x 8	9 x 9	14 x 14	17 x 17	19 x 19	19 x 19	23 x 23	29 x 29
Výška (mm)	0.5	0.5	0.5	0.8	1.0	1.0	0.8	1.0	1.0

Tabulka 2 – Velikosti pouzder pro Cyclone IV E [12]

Zajímavou a někdy i nutnou informací je pak také jaký je maximální počet I/O pinů, které jsou dostupné pro dané pouzdro, viz tabulka č. 3. Ne všechny piny musí být nutně na konkrétní desce použité. Obecně platí, čím více vývodů tím může být dané zařízení lépe vybaveno. S rostoucím počtem použitelných vývodů však vzrůstá cena čipů i cena montáže těchto čipů na PCB. Proto počet vývodů, respektive volba použitého pouzdra je vždy určitým kompromisem.

	E144	M164	M256	U256	F256	F324	U484	F484	F780
EP4CE6	91	—	—	179	179	—	—	—	—
EP4CE10	91	—	—	179	179	—	—	—	—
EP4CE15	81	89	165	165	165	—	—	343	—
EP4CE22	79	—	—	153	153	—	—	—	—
EP4CE30	—	—	—	—	—	193	—	328	532
EP4CE40	—	—	—	—	—	193	328	328	532
EP4CE55	—	—	—	—	—	—	324	324	374
EP4CE75	—	—	—	—	—	—	292	292	426
EP4CE115	—	—	—	—	—	—	—	280	528

Tabulka 3 – Maximální počet I/O pinů dle pouzdra [12]

2.1 DE0-Nano

Tento vývojový kit je vyroben společností Terasic. Jedná se o firmu s dlouholetou historií pocházející z Tchaj-wanu. Ačkoliv je tento kit již poměrně dlouho na trhu, stále je o něj zájem a stále je možné si ho zakoupit, a to například přímo na stránkách výrobce za cenu kolem 80 dolarů nebo za 61 dolarů při uplatnění akademické slevy. [13]

Tento kit je osazen čipem Cyclone IV EP4CE22F17C6N. Z tabulky č. 1 je jasné, že se jedná o středně vybavený kit. Výrobce udává následující konfiguraci:

- 22320 logických elementů,
- 594 Kbits SRAM,
- 66x 18x18 multiplexer,
- 4x univerzální PLL,
- 153 I/O pinů.

Obrázek 4 – FPGA vývojový kit DE0-Nano

Pro konfiguraci a nastavení kitu je kit přímo vybaven programátorem označovaným jako USB-Blaster pro programování FPGA a konfigurační paměti, což je flash paměť se sériovým rozhraním, do které se ukládají konfigurační data, která se načítají do FPGA po zapnutí napájení nebo při rekonfiguraci. [13]

K rozšíření možností kitu slouží dvojice 40 pinových rozhraní viz obrázek č. 4, rozhraní jsou umístěny na horní a dolní části (JP1, JP2). Tyto piny lze rozdělit na 72 pinů typu vstup/výstup využívající 3.3 V logiku, dále pak na napájecí piny obsahující napětí 3.3 V a 5 V. Poslední skupinou pinů jsou zemnící piny. Dále je kit vybaven 26 pinovým rozhraním na pravé straně kitu a poskytuje 16 digitálních 3.3 V I/O pinů a 8 analogových vstupů pro připojení senzorů. [13]

Co se týče pamětí pro uložení dočasných nebo trvalých dat jsou součástí kitu paměti 32 MB SDRAM a 2 Kb I2C EEPROM. Z uživatelských komponent tu je osm zelených LED diod, dvě tlačítka a čtyřpólový DIP přepínač. Kit dále poskytuje tříosý akcelerometr, 50MHz hodinový oscilátor a A/D převodník.

Pro napájení vývojového kitu slouží USB type mini-AB port, dva DC pětivoltové piny nebo speciální dvou pinový konektor pro připojení zdroje energie od 3.6 do 5.7 V. [13]

2.2 StormIV-E6

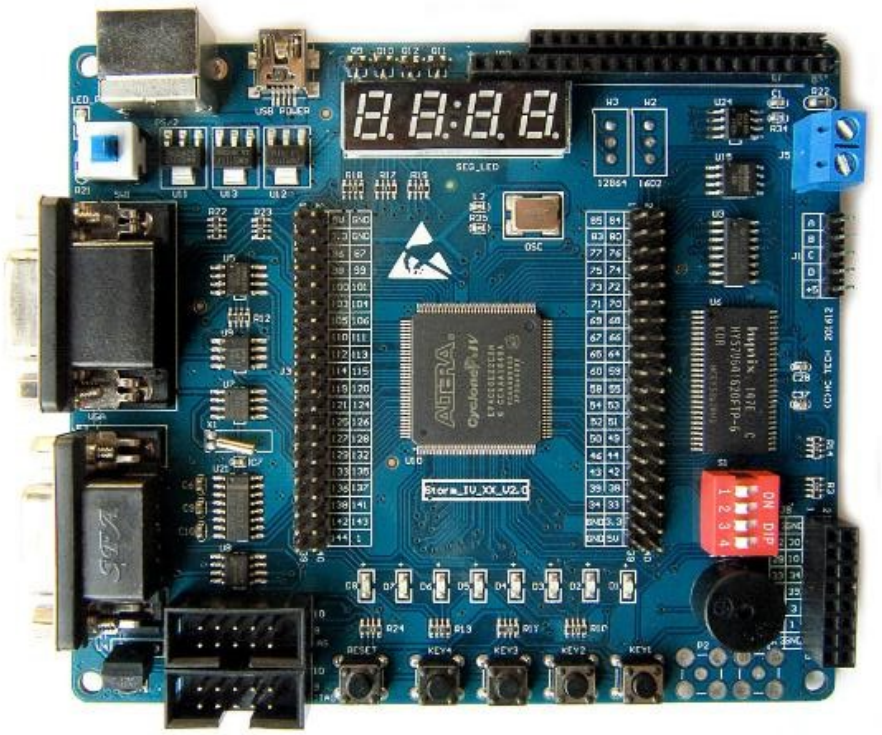
Tento kit pochází ze známého čínského e-shopu a obsahuje FPGA z řady Cyclone IV E, konkrétně EP4CE6E22C8, tedy nejméně výkonný z této řady. Oproti předešlému kitu se už tento konkrétní neprodává, ale stále se dají sehnat téměř identické kity s cenovkou okolo 28 dolarů. Konfigurace je následující:

- 6272 logických elementů,
- 270 Kbits SRAM,
- 15x 18x18 multiplexer,
- 2x univerzální PLL,
- 179 I/O pinů.

Ke konfiguraci a nastavení FPGA je dodán USB-Blaster a na vývojovém kitu se nachází sériová konfigurační paměť o velikosti 4Mbit. Co se rozšiřitelnosti funkcí týče, je vývojový kit vybaven dvěma 40 pinovými rozhraními umístěnými uprostřed kitu a jedním 16 pinovým rozhraním s female piny. Přesné rozvržení pinů je u tohoto vývojového kitu dostupné pouze ve schématu. [14]

K ukládání dat tu slouží 64Mbit SDRAM, 16Mbit serial flash a 16Kbit I2C EEPROM. Uživatelských komponent je tu oproti kitu DE0-Nano více. Na tomto kitu je osm červených

LED diod, čtyři tlačítka, jedno reset tlačítko a také čtyřpólový DIP přepínač a 50MHz hodinový oscilátor. Také je tu čtyř místný sedmisegmentový displej pro zobrazování naměřených hodnot. A také zde najdeme buzzer pro vytváření zvukových signálů. Z dalších hardwarových komponent tu je infračervený senzor, rozhraní pro ovládání DC a krokového motoru, PS/2 konektor, VGA konektor, senzor teploty, interface pro možnost připojení LCD displeje a připravené místo pro připájení slotu pro SD karty. O napájení vývojového kitu se stará USB micro-B port. Vzhledem k velkému množství různých hardwarových komponent jsou zde omezené možnosti využití jednotlivých pinů. [14]



Obrázek 5 – FPGA vývojový kit StormIV-E6

3 VERILOG

Jak již bylo uvedeno dříve Verilog je jazyk typu HDL. Jedná se o jeden z nejčastěji používaných jazyků pro programovatelná hradlová pole. Vzhledem k tomu, že v této práci byl použit právě tento jazyk z důvodu jeho větší přehlednosti, je pak níže podrobněji popsán. Dále je popsána adresářová struktura a samotné použité Verilog moduly použité v aplikaci. Poslední část této kapitoly je věnována popisu a vysvětlení účelu vlastních atributů v modulu, které neodpovídají žádné oficiální syntaxi toho jazyka.

3.1 Úvod do Verilogu

Základními stavebními bloky jazyka Verilog jsou moduly a vodiče, anglicky modules a wires. Modul se zvenčí tváří pouze jako blackbox s definovanými vstupy a výstupy. Modul představuje konstrukční jednotku, která implementuje způsob chování a během logické syntézy se převede na logický obvod. Na libovolnou kombinaci vstupů vždy modul poskytuje konkrétní definovanou hodnotu výstupu. To umožňuje využívat stejný modul opakovaně k vytváření větších a složitějších celků, které jsou pak implementovány do FPGA. Vodič pak slouží k vzájemnému propojení modulů a k přenášení dat. Pro vícebitové informace se používá sběrnice, anglicky bus.

Syntaxe pro blok modulu je velmi jednoduchá. Každý blok začíná klíčovým slovem **module**, za kterým následuje název daného modulu. V kulatých závorkách za názvem modulu jsou vyjmenované porty modulu. Tyto závorky jsou zakončeny středníkem. V případě, že modul nemá žádné porty, budou závorky prázdné. Typy portů můžou být **wire** (vodič, slouží pouze k přenášení informace) nebo **reg** (registr, slouží k uchování aktuálního stavu). Oba dva tyto typy pak ještě mají svůj prefix. Může jít o **input**, **output** nebo **inout**. Význam je stejný, jak už samotný název prefixu napovídá, tedy jedná se o vstupní, výstupní a vstupně výstupní port. Dále následuje samotná logika bloku a vše je ukončeno klíčovým slovem **endmodule**. Na obrázku č. 6 je vidět kód pro jednoduchý AND modul s dvěma vstupy a jedním výstupem. [15]

```
1  module cAND2(input wire A, input wire B, output wire Y);
2  assign Y = A && B;
3  endmodule
```

Obrázek 6 – Jednoduchý Verilog modul

Klíčové slovo **assign** slouží k nepřetržitému přiřazování hodnot. Nejčastěji se používá u typu **wire** nebo u podobných datových typů. Vodiče a další datové typy mohou být deklarovány s různou šířkou bitů (často označované jako sběrnice), například u vodiče jde o počet samostatných bitů, které se přes daný vodič posílají. I v samotném modulu je možné deklarovat

další vodiče nebo registry. Syntaxe je stejná jako v jiných programovacích jazycích, tedy určení typu, název a případně přiřazení hodnoty pomocí znaménka rovná se. Dalšími datovými typy pak jsou **integer**, **real**, **time** a **realtime**. [15]

```
1  wire w1;          //1-bit vodič
2  wire [7:0] w2;    // 8-bit vodič - vektor
3
4  reg one_reg;          //1-bit hodnota proměnné
5  reg [7:0] count_reg; //8-bit hodnota proměnné
```

Obrázek 7 - Ukázka deklarace

Velice často bude k práci potřeba použití různých operátorů. Bitové operátory se nejčastěji používají k provádění bitové operace se dvěma vstupy, výsledkem operace je pouze jedna hodnota, tj. jeden výstup. Logické operátory jsou základem kódu ve Verilogu a nejčastěji se nacházejí v podmínkových výrazech a v přiřazení (assign). Operátor posunu ve Verilogu se používá k posunu dat v proměnné. Levá strana operátoru obsahuje proměnnou k posunu, pravá strana operátoru obsahuje počet bitů, o které je hodnota posunuta.

Operátor	**Popis**
&	AND
~&	NAND
|	OR
~|	NOR
^	XRO
~^	XNOR
~	bitová negace
&&	logický AND
||	logický OR
!	logický NOT
<<	posun vlevo, logické (výplň nulami)
>>	posun vpravo, logické (výplň nulami)
<<<	posun vlevo, aritmetické (zachovává znaménko)
>>>	posun vpravo, aritmetické (zachovává znaménko)

Tabulka 4 – Operátory ve Verilogu

Další důležitou částí Verilog kódu jsou bloky **always**. Tento blok obsahuje seznam citlivostí, což je soupis proměnných, který slouží k aktivaci bloku při jakékoliv změně stavu proměnné z tohoto seznamu. Je zapsán za znaménkem @ v kulatých závorkách. Tento seznam může obsahovat jeden nebo skupinu signálů, jejichž změna hodnoty zapříčiní provedení obsahu **always** bloku. [15]

```
1  // provede se pokaždé když se "a" nebo "b" změní
2  always @(a or b)begin
3      //kod k provedeni
4  end
```

Obrázek 8 – Ukázka always bloku

Do každého modulu se pak dají vkládat i další moduly. A to jak vlastní moduly napsané uživatelem, tak moduly předpřipravené ve Verilogu, jako například moduly **nand**, **and**, **not** a podobně. Pro volání modulu je syntaxe jednoduchá. Stačí napsat název modulu následovaný jedinečným identifikátorem daného bloku v kódu. V ukázce na obrázku č. 9 je název modulu cAND2 a identifikátory jsou b1 a b2. Do kulatých závorek se následně zapíší porty pro daný modul. Toto se dá provést dvěma různými způsoby. První z nich je v ukázce níže na řádku číslo tři. Tento zápis je delší, ale přehlednější pro čtení a následné revize kódu. Jeho principem je, že pomocí tečky a názvu portu se přímo přiřazuje, se kterým vodičem bude port propojený. Zde není nutné dodržovat pořadí. Druhý způsob je ukázán na řádku čtyři. Tento zápis je jednodušší, ale je třeba znát přesné pořadí portů v použitém modulu a může tedy dojít k prohození v zapojení. [15]

```
1  wire w1,w2,w3;
2
3  cAND2 b1(.A(w1),.B(w2),.Y(w3));
4  cAND2 b2(w1,w3,w3);
```

Obrázek 9 – Ukázka volání modulů

3.2 Popis použitých Verilog modulů

Jak již bylo zmíněno dříve, tak samotná aplikace používá ke své práci Verilog moduly, které jsou v adresářové struktuře aplikace ve složce components. V této složce jsou pak moduly dále rozděleny do podadresářů. Každý podadresář má své logické opodstatnění a moduly jsou tak rozděleny do samostatných celků. Jedná se o adresáře bit, block, board, comb, io, logic a sec. Rozdělení má také svůj praktický důvod a to ten, že menu v aplikaci je dynamicky načítané, a tedy i rozdělené do stejných kategorií jako jsou rozděleny moduly. Toto rozdělení pak pomáhá v zjednodušení při načítání modulů do menu a do aplikace. Všechny moduly mají v názvu prefix písmena c. Toto písmeno značí, že se jedná o komponentu, a navíc zabraňuje kolizím s klíčovými slovy jazyka Verilog. Dále budou popsány jednotlivé adresáře spolu s moduly, které jsou v nich obsaženy.

3.2.1 Adresář bit

Tento adresář je nejmenší, co se obsahu použitých modulů týče. V sobě má obsaženy pouze moduly, které slouží jako vstup konstantních hodnot neboli signálů. Vzhledem k tomu, že se

jedná o vstup bitů, které mohou být pouze o hodnotách logická jedna nebo logická nula, jsou tu pouze dva moduly. Jsou jimi modul cOne a cZero. Jak již názvy napovídají modul cOne poskytuje konstantní bit o hodnotě jedna tzv. logická jedna a cZero pak dává logickou nulu.

3.2.2 Adresář block

Adresář block je zpočátku prázdný a není zde žádný předefinovaný modul. Toto není chyba, ale je tomu tak z důvodu jeho významu. V tomto adresáři jsou ukládány moduly, které si vytvořil samotný uživatel. Jedná se tedy o vlastní moduly s různými funkcemi, jaké jim byly uživateli nakresleny v aplikaci. Tyto moduly mohou být používány opakovaně v projektech, a tedy jejich myšlenka je v zapouzdření často opakovaných funkcionalit.

3.2.3 Adresář board

Asi nejzajímavější a nejsložitější adresář board obsahuje komponenty, které se vztahují přímo ke konkrétním vývojovým kitům. V tomto adresáři se nacházejí další podadresáře pojmenované podle názvu konkrétního kitu. V základním stavu aplikace jsou to podadresáře DE0-Nano a StormIV-E6. V těchto adresářích jsou moduly, které reprezentují nějakou z hardwarových funkcionalit daného kitu a obsažené moduly se liší dle vybavenosti kitu.

V adresáři DE0-Nano se nachází celkem 32 modulů. Takto velké množství pinů je dáno především možností tohoto kitu poskytovat velké množství vstupních a výstupních pinů. Popis modulů bude abecedně dle jejich názvů. Prvním je modul cCLK_50MHz, který poskytuje na svém výstupu 50MHz hodinový signál. Následují piny s názvy cIN_PIN11 až cIN_PIN20. Jedná se o moduly reprezentující konkrétní piny na vývojovém kitu a tyto moduly pak poskytují hodnoty vstupních signálů z těchto pinů. Odůvodnění, proč jsou piny očíslované od jedenáctky, a ne od jedničky je popsáno níže v kapitole popisující adresář io.

Po vstupních pinech následují moduly cKEY0 a cKEY1. Tyto moduly reprezentují dvě tlačítka na kitu. V obvodu pak na svém výstupu posílají signál logické nuly při zmáčknutí příslušného tlačítka. Dalšími moduly jsou cLED0 až cLED7 pro ovládání LED diod na kitu. Tyto moduly mají pouze vstup, do kterého je připojen signál z navrženého obvodu. Předposledními moduly jsou výstupní piny cOUT_PIN11 až cOUT_PIN20, které plní stejnou funkci jako vstupní piny, ale v tomto případě posílají signál po pinech ven z kitu. Posledním modulem v tomto adresáři je cSWITCH. Jak již název napovídá, jedná se o čtyřpólový DIP přepínač. Funguje podobně jako tlačítko, pouze s tím rozdílem, že se zde páčky přepínají mezi logickou jedničkou a nulou a nemusí se tedy držet zmáčknuté.

Druhý vývojový kit StormIV nemá možnost dodatečných vstupních a výstupních pinů, a tak má pouze sedmnáct IN/OUT modulů. Dalším modulem je cBeeper, s jehož pomocí je možné z kitu generovat různé frekvence zvuků. Podobně jako první kit i zde je modul CLK_MHz pro hodinový signál. Na rozdíl od prvního kitu má tento vývojový kit sedmisegmentový displej, který se ovládá přes modul cDISPLAY. Ovládáním je samozřejmě myšleno pouze možnost posílání signálů na jeho vstupy, čímž budou rozsvěceny jednotlivé segmenty displeje. I tento kit obsahuje tlačítka. V tomto případě jsou tu čtyři normální tlačítka cKEY1 až cKEY4 a jedno cKEY_RESET, které se ovšem chová podle naprogramování v obvodu a není to žádné speciální tlačítko na restart kitu. Dále tu jsou moduly pro ovládání osmi LED diod cLED1 až cLED8 a jako poslední modul tu je cSWITCH, který prezentuje čtyřpólový DIP přepínač.

3.2.4 Adresář comb

V tomto adresáři se nachází moduly pro kombinační obvody. Prvním je modul cBCD_DECODER, který slouží k dekódování čtyř signálů tvořících binární číslo od jedničky až po devítku a převádí je sedm výstupů určených pro sedmisegmentový displej. Zbylé moduly pak plní funkci pro složení a rozložení signálů vodičů z nebo do sběrnic. Moduly se jmenují cMERGE a cSPLIT a slouží k práci se sběrnicemi. Sběrnice jsou vodiče, které jsou sloučené do jednoho pomyslného spojení k zjednodušení návrhu obvodu. Číslo za názvem modulu uvádí, na kolikabitovou sběrnici budou samostatné vodiče sloučeny, respektive na kolik samostatných vodičů bude sběrnice rozdělena. Zde tedy sloučení či rozdělení ze dvou, čtyř nebo osmi vodičů.

3.2.5 Adresář io

V tomto adresáři jsou moduly sloužící k reprezentaci pinů, a to buď fyzických na vývojovém kitu nebo piny na vlastních modulech typu block. Jak již bylo zmíněno dříve, tento adresář obsahuje vstupní a výstupní piny, které odkazují na konkrétní piny na vývojovém kitu. Jmenují se cIN_PIN1 až cIN_PIN10 a cOUT_PIN1 až cOUT_PIN10. Dále je tu předpoklad, že každý vývojový kit má aspoň dvacet volných pinů pro tyto účely a pokud jich má více budou uvedeny jako moduly v samotné složce s kitem v adresáři board.

Poslední dva moduly jsou cIN_PIN a cOUT_PIN, které slouží jako šablona pro vytváření vlastních vstupních nebo výstupních pinů v uživatelských modulech typu block. Tyto moduly neodkazují na žádné fyzické piny a jedná se pouze o virtuální piny (vstup a výstup signálů) na modulech typu block.

3.2.6 Adresář logic

Základní stavební kameny logických obvodů se nacházejí v adresáři logic, pojmenované podle logických členů, které jsou zde uloženy. Je zde celkem 15 modulů a jedná se převážně o různé variace na logické obvody AND, NAND, NOR a OR s různými počty vstupu do modulu. Jednou tu jsou zastoupeny i logické obvody NOT, XNOR a XOR, které nemají různé variace na počet vstupů. Všechny prvně jmenované moduly jsou ve variacích s dvěma, třemi nebo čtyřmi vstupy do modulu. Pro popis fungování logických obvodů pak slouží tabulka níže.

Vstup		Výstup					
A	B	AND	NAND	OR	NOR	XNOR	XOR
0	0	0	1	0	1	1	0
0	1	0	1	1	0	0	1
1	0	0	1	1	0	0	1
1	1	1	0	1	0	1	0

Tabulka 5 – Pravdivostní tabulka logických členů

3.2.7 Adresář sec

Poslední adresář obsahuje vybrané sekvenční logické obvody. Konkrétně jde o tři klopné obvody, dva moduly pro dělení hodinového signálu a jeden modul s funkcí počítadla. Klopné obvody jsou v modulech s názvy cRS, cJK a cD. Klopné obvody jsou základním obvodem, který je schopen setrvat ve stavu logická nula nebo logická jedna bez zásahu vnějších logických úrovní. Jedná se tedy o jednoduché paměťové buňky.

Obvod typu R-S je jedním z nejzákladnějších obvodů. Tento modul má tři vstupy a dva výstupy. Výstupy jsou označeny Q a Qn. Výstup Q poskytuje aktuální uloženou hodnotu v obvodu a výstup Qn je jeho negované hodnota. Vstupy jsou označené jako S (set), R (reset) a CLK (clock). Jak již názvy napovídají, vstup S nastavuje hodnotu. Při přivedení hodnoty logická jedna se tak v obvodu uloží tato hodnota a je následně dostupná na výstupu Q. Vstup R slouží k vynulování uložené hodnoty na úroveň logická nula. V tomto případě s jedná o synchronní variantu obvodu R-S, což znamená, že obvod reaguje na signály ze vstupů pouze po dobu, kdy dostává hodnotu logická jedna z hodinového signálu vstupu CLK.

Druhý klopný obvod JK je funkčně podobný obvodu R-S. Má stejný počet vstupů i výstupů. V tomto případě se liší pouze názvy dvou vstupů, kde u toho obvodu jsou nazvány J a K. Vstup J nastavuje uloženou hodnotu na úroveň logická jedna a vstup K nuluje uloženou hodnotu na logickou nulu. Rozdíl spočívá především v tom, co se stane při přivedení hodnot logická jedna na oba vstupy najednou. Na rozdíl od obvodu R-S se nejedná o zakázaný stav, ale neguje se

hodnota uložená v klopném obvodu. I v tomto případě se veškeré změny provádí pouze po dobu pulzu hodinového signálu.

Posledním klopným obvodem je obvod D. Ten má pouze dva vstupy s názvy D (data) a CLK a dva výstupy Q a Qn, které jsou stejné jako v předešlých případech. Funkčností se odlišuje od předešlých obvodů. Klopný obvod D si ukládá logickou hodnotu ze vstupu D pouze s příchodem náběžné hrany hodinového signálu. Pouze v tomto okamžiku si obvod zapamatuje hodnotu, která zůstane na výstupu Q až do chvíle, než přijde další náběžná hrana hodinového signálu a uloženou hodnotu opět změní na aktuální hodnotu vstupu D.

Dále tu jsou dva moduly určené k dělení hodinového signálu, a to k dělení dvěma nebo deseti. Moduly se jmenují cDIVIDE_BY2a cDIVIDE_BY10. Posledním modulem je cCOUNTER sloužící jako počítadlo hodinového signálu.

3.3 Vlastní atributy

Velká většina předefinovaných modulů obsahuje speciální řádky kódu, které nepatří mezi oficiální syntaxi jazyka Verilog. Interně jsou tyto řádky nazývány atributy. Tyto řádky jsou zapsány jako jednořádkové komentáře, a tak nikterak nenarušují syntaxi jazyka. Jedná se o atributy hidden a position. V kódu modulů jsou zapsány jako //hidden: <parametry> a //position: <parametry>.

Moduly a jejich vstupní a výstupní piny jsou v aplikaci generovány a vykreslovány dynamicky vyčítáním informací z hlavičky modulu. Porty, které se uživateli nemají zobrazit, jsou uvedeny za atributem hidden. Schování pinů je zde z důvodu, že u některých modulů sice jsou vyjmenovány vstupní i výstupní piny, ale uživateli se budou zobrazovat pouze některé z nich. Nejlépe se toto vysvětlí na příkladu. Modul cIN_PIN1 má v hlavičce vyjmenovaný jeden vstupní a jeden výstupní vodič. Protože se jedná o vstupní pin, uživateli by se měla v aplikaci zobrazit pouze výstupní hodnota tohoto pinu, a tak je vstupní pin zapsán do atributu hidden a tím je před uživatelem v grafickém prostředí schován. Tento vstup do modulu je pak použit interně při napojení modulu na skutečný hardwarový pin vývojového kitu.

Druhý atribut position je uveden u všech modulů, které reprezentují nějakou hardwarovou komponentu na vývojovém kitu. V tomto atributu jsou vyjmenovány pozice x a y spolu s názvem příslušného kitu. Z těchto informací je aplikace schopná zobrazit obrázek kitu dle zadaného jména a pomocí uvedených souřadnic zobrazit pomocí šipky onu komponentu, kterou daný modul reprezentuje. Hlavním smyslem je tedy pomoci začínajícím uživatelům s hledáním a poznáváním komponent na vývojovém kitu.

4 REŠERŠE PODOBNÝCH APLIKACÍ

Další částí práce je popsání podobných aplikací. Podobností je myšlena bezplatná dostupnost dané aplikace. Vybrány byly dvě: Icestudio a Intel Quartus Lite Edition, které v době psaní práce bylo možné bezplatně stáhnout. Aplikace Icestudio byla vybrána z důvodu, že se jedná o nejznámější komunitou vyvíjenou aplikaci pro tvorbu obvodů.

Intel Quartus byl vybrán z důvodu, že se jedná o profesionální prostředí, které nabízí pokročilé funkcionality k složitějšímu navrhování obvodů. Dalším důvodem, proč bylo vybráno k popisu právě toto prostředí, je také jeho využití v rámci vyvíjené aplikace pro sestavení a nahrání obvodů do kitu. Obě tyto aplikace jsou zde popsány a zhodnoceny.

4.1 Icestudio

Toto vývojové prostředí nepatří pod žádnou ze společností vyvíjející nebo vyrábějící obvody FPGA. Tato aplikace je vyvíjena komunitou pod GPL-2.0 licencí a je k dostání kompletně zdarma. Tvůrci této aplikace jsou Jesús Arroyo Torrns, Carlos Venegas Arrabé a Juan Gonzáles Gómez, kteří vystupují pod značkou FPGAwars pod kterou vytváří další projekty určené pro FPGA. [17]

Autoři na tomto projektu spolupracují s komunitou, která pomáhá ve vývoji a udržování aplikace. Aktuální verze označovaná jako 0.5.0 je oficiálně z října 2019, nicméně na aplikaci se stále pracuje. Momentálně vývojáři pracují na opravování chyb a aplikace tedy nedostává nové funkce. Datum je tedy v tomto případě zavádějící a nejedná se tedy o starou a nepodporovanou aplikaci. Díky komunitě je také dostupná česká lokalizace aplikace, které je prozatím pouze částečně přeložena.

Aplikace je dostupná na oficiálních stránkách projektu nebo také na stránkách GitHubu, kde jsou dostupné i zdrojové kódy. Na těchto stránkách je k dispozici instalační a uživatelská příručka. Aplikace je zaměřena na středně pokročilé uživatele a částečně i na začátečníky. [17]

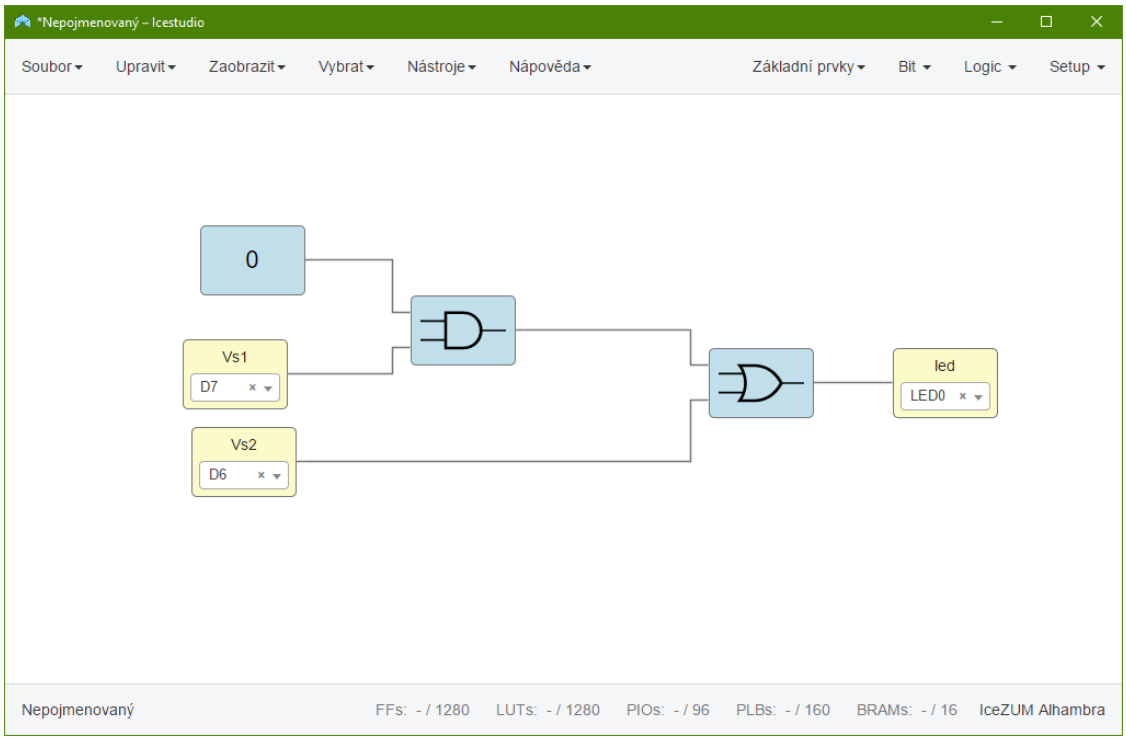
4.1.1 Instalace a popis

Instalační soubor je na dnešní dobu malý, jde o velikost do 150 MB. Po instalaci zabírá na disku okolo 230 MB, což je velice příznivá hodnota, protože nezabírá mnoho místa. Samotná instalace je pak velice rychlá a bez jakýchkoli zádrhelů. Je třeba také podotknou, že při instalaci není možné ovlivnit jazyk lokalizace aplikace, který se pravděpodobně nastavuje podle jazyka systému, ale v samotné aplikaci je možné si dodatečně jazyk zvolit.

Po nainstalování je ještě nutné si nainstalovat nástroj (anglicky Toolchain) Apio starající se o syntetizaci, simulování a nahrávání do podporovaného kitu. Nakonec je nutné připojit

vývojový kit a nastavit ovladače. Po těchto krocích je vše připravené k používání. Icestudio podporuje pouze několik vybraných open source FPGA. Je však možné přidání dalších vývojových kitů.

Jedná se o robustní a kvalitní nástroj. Výhodou je dostupnost pro všechny hlavní operační systémy na trhu, tedy Windows, Linux a MacOS. Standardně je aplikace v bílém provedení, ale je možné si ji stáhnou i v takzvaném night buildu, který je laděn do tmavých odstínů. Bohužel zde není možné rychlé přepínání mezi těmito barevnými schématy aplikace.



Obrázek 10 – Icestudio

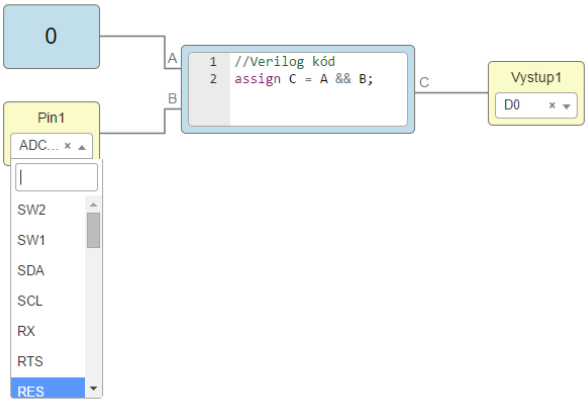
Prostředí Icestudia je jednoduché a vcelku přehledné. Rozložení je podobné jako v navrhované aplikaci s hlavním menu na horní straně okna a kreslící plochou pod ním. Samotné menu je přehledně uspořádáno do kategorií, což usnadňuje práci a učení s touto aplikací. Moduly se přidávají také pomocí položek v menu z nabídek umístěných na pravé straně tak, aby byly vizuálně odděleny od položek na levé straně sloužící k informačním anebo editačním účelům.

Aplikace umožňuje veškeré funkce očekávané od tohoto typu programu. Mezi několik hlavních funkcí patří možnost nahlížení do Verilog kódu všech modulů a je také možné si vytvářet bloky s vlastním kódem. Dále také možnost zkontrolování nakresleného obvodu a také jeho následné sestavení a nahrání do vývojového kitu. V neposlední řadě zobrazení rozložení a popisu vývojového kitu a možnost vytvoření projektů a vlastních bloků.

4.1.2 Hodnocení

Největší klady této aplikace jsou snadná dostupnost a jednoduchost v používání. Každý začátečník také ocení ukázkové příklady a dokumentaci. Aplikace je přehledná a dostupná pro více operačních systémů. Pro pokročilejší uživatele přijde vhod možnost psaní přímo do bloků pomocí Verilog kódu.

Najde se zde však i několik záporů. Přidání vlastního vývojového kitu je sice možné, bohužel není v dokumentaci nikde popsáno, což může způsobit problémy začátečníkům. Ovládání není nikde popsané, a tak první seznámení může být složité. Velice nešťastně řešen je pak způsob přidávání modulů pro jakoukoli hardwarovou komponentu na kitu, protože není možné si ji vybrat přímo v menu. Je nutné si vybrat modul pro vstup nebo výstup v něm nastavit pomocí dropbox menu název daného pinu nebo komponenty. A právě tyto názvy jsou dosti matoucí a je nutné je dohledávat v obrázku rozložení kitu, pokud je ovšem tento obrázek k vybranému kitu dostupný.



Obrázek 11 – Ukázka bloku s kódem a volba pinu

4.2 Intel Quartus Prime Lite Edition

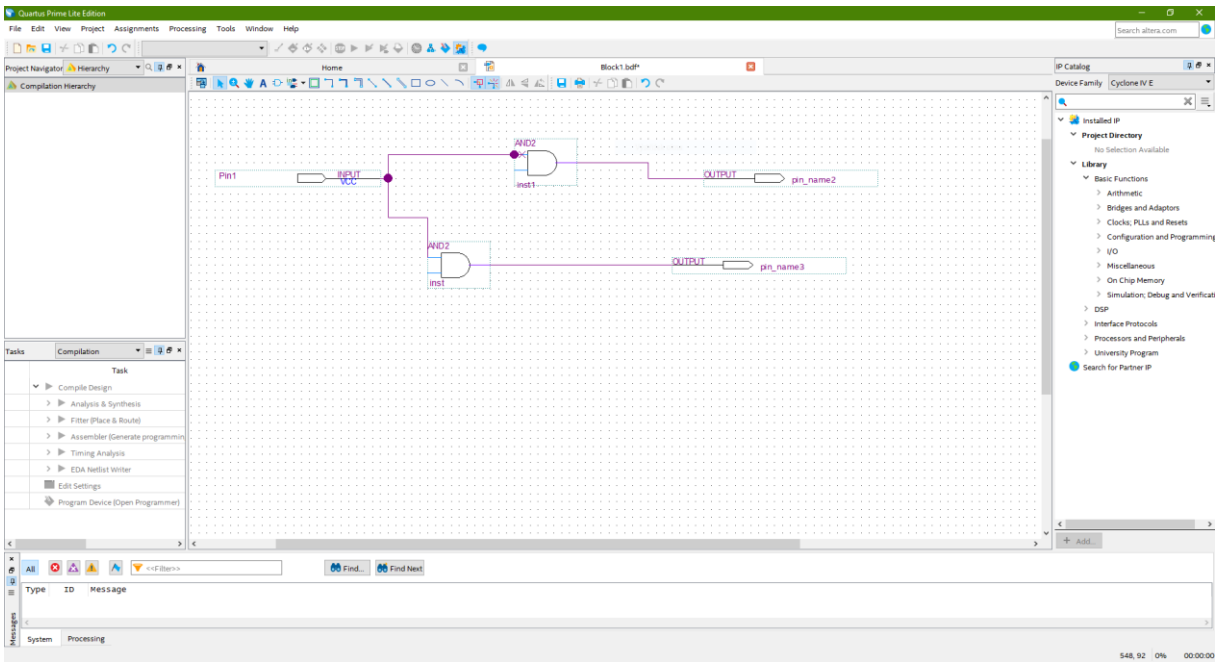
Jedná se o vývojové prostředí od společnosti Intel, dříve patřilo společnosti Altera a i díku tomu je možné v tomto programu stále najít různé odkazy nebo nápisy obsahující staré jméno společnosti. Tato konkrétní edice je k dostání zdarma, ovšem pro stažení je nutné mít založený uživatelský učet na stránkách Intelu. Aplikaci je možné si stáhnou přímo na stránkách Intelu. Jak bylo zmíněno již dříve, jsou k dostání tři různé edice, tak je nutné si stáhnou tu správnou. Aktuální verze má označení 20.1.1 a je z listopadu 2020. Na rozdíl od Icestudia se v tomto případě jedná o profesionální vývojářský nástroj pro FPGA a pro začátečníky je naprosto nevhodný. Největší omezení Lite verze je nemožnost vyvíjet pro FPGA řady Stratix a Arria, což však běžného uživatele nijak neomezí. [18]

4.2.1 Instalace a popis

Quartus Prime je k dostání na operační systémy Windows a také na hlavní distribuce Linuxu jako je Red Hat, CentOS nebo Ubuntu. Před stažením je nutné přihlášení pod účtem Intel. Registrace je vcelku rychlá, ovšem je nutné vyplnit podrobné osobní informace. Poté je už možné stažení instalačního balíčku.

U verze pro Windows je k dostání několik různých verzí lišících se v nainstalovaných knihovnách pro podporu různých řad FPGA. Tento kompletní instalační balík má šest gigabajtů. Mnohem výhodnější je si stáhnout samotné vývojové prostředí s knihovnou pro zvolenou řadu FPGA, čímž se velikost stahovaných souborů zmenší o více než polovinu. Samotná instalace je zdlouhavější než v případě Icestudio, ale to je dáno odlišnou složitostí programů. Po instalaci už není třeba nic dělat a je možné začít vyvíjet.

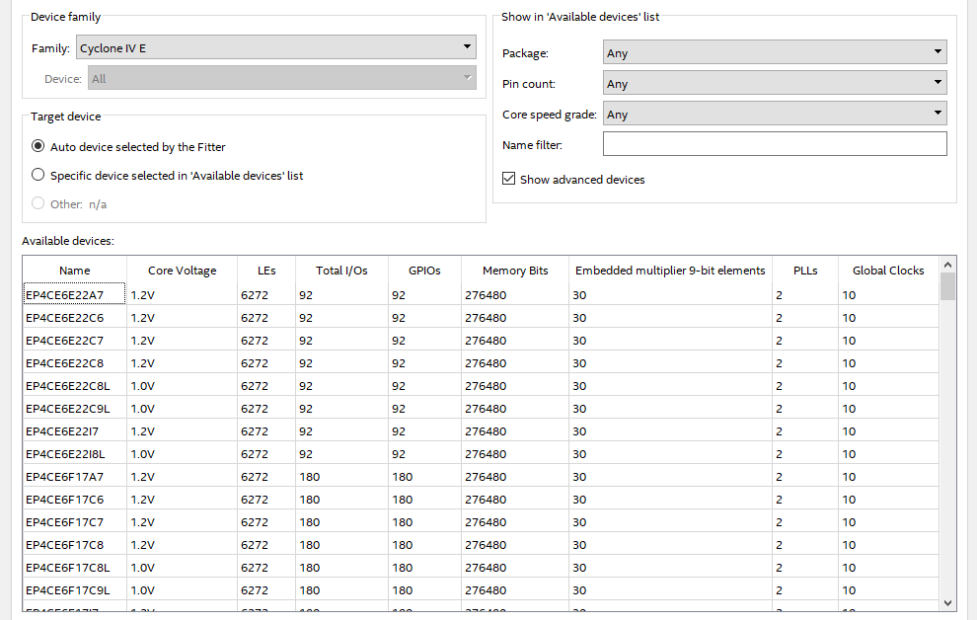
Vzhledem k tomu, že se jedná o profesionální nástroj tak tomu také odpovídá složitost samotného prostředí aplikace. V horní části je hlavní menu, vlevo jsou části pro práci s projektem, napravo jsou dostupné různé funkce z knihoven, dole je informační oblast a uprostřed pracovní okno. Vzhledem k povaze programu se zde dá veškerá práce provádět čistě za pomocí HDL jazyků nebo pomocí kreslení v grafickém prostředí. V případě použití grafického prostředí je práce složitá a je nutné mít znalosti z programování a prací s FPGA.



Obrázek 12 – Intel Quartus Lite Edition

Mezi velké výhody patří schopnost sestavení a nahrávání na FPGA. Toto vývojové prostředí má schopnost pracovat s jakýmkoliv vyráběným čipem ze zvolené řady FPGA, pro který bude potřeba daný systém vyvíjet. V tomto případě je to řada Cyclone IV E. Při vytváření projektu

je nutné vždy zadat konkrétní označení čipu na vývojovém kitu. Také zde nenajdeme omezení v HDL jazycích. Quartus Primo podporuje VHDL, Verilog nebo i další jazyky. Je nutné podotknout, že i v grafickém prostředí se velká část práce provádí nastavováním a psaním kódu.



Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elements	PLLs	Global Clocks
EP4CE6E22A7	1.2V	6272	92	92	276480	30	2	10
EP4CE6E22C6	1.2V	6272	92	92	276480	30	2	10
EP4CE6E22C7	1.2V	6272	92	92	276480	30	2	10
EP4CE6E22C8	1.2V	6272	92	92	276480	30	2	10
EP4CE6E22C8L	1.0V	6272	92	92	276480	30	2	10
EP4CE6E22C9L	1.0V	6272	92	92	276480	30	2	10
EP4CE6E22I7	1.2V	6272	92	92	276480	30	2	10
EP4CE6E22I8L	1.0V	6272	92	92	276480	30	2	10
EP4CE6F17A7	1.2V	6272	180	180	276480	30	2	10
EP4CE6F17C6	1.2V	6272	180	180	276480	30	2	10
EP4CE6F17C7	1.2V	6272	180	180	276480	30	2	10
EP4CE6F17C8	1.2V	6272	180	180	276480	30	2	10
EP4CE6F17C8L	1.0V	6272	180	180	276480	30	2	10
EP4CE6F17C9L	1.0V	6272	180	180	276480	30	2	10

Obrázek 13 – Výběr čipu při tvorbě projektu

4.2.2 Hodnocení

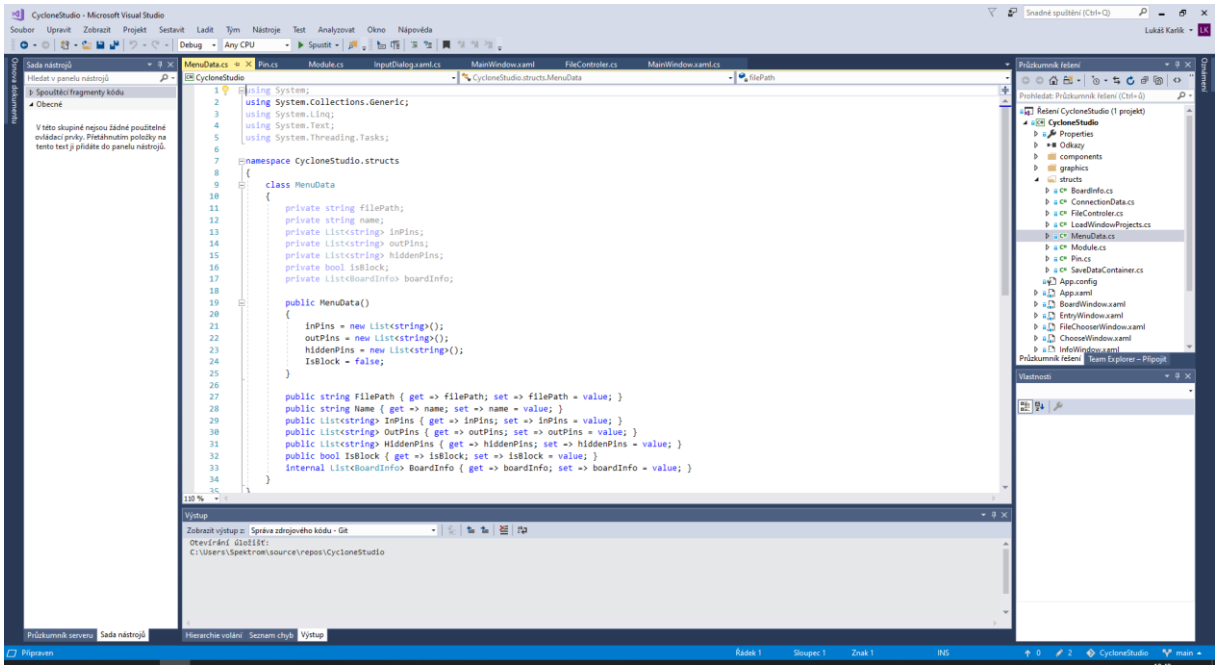
Jedná se o velice pokročilé a propracované vývojové prostředí pro vyvíjení pro FPGA. Výhodou je jeho všestrannost a dostupnost pokročilých funkcí. Toto vývojové prostředí bude také použito ve vyvíjené aplikaci, kde bude na pozadí bez přímého zásahu uživatele provádět sestavování a nahrávání binárních souborů do vývojových kitů. Za zápory se dá považovat snaha Intelu k donucení uživatelů k nákupu placené licence, pomocí různých omezení v dostupných řadách FPGA. Dále tu jsou občasné nefunkční odkazy jako například odkaz na uživatelskou příručku.

5 POUŽITÉ TECHNOLOGIE

Tato část práce je věnována technologiím, které byly použity při vytváření aplikace. Bude zde popsáno vývojové prostředí Visual Studio, programovací jazyk C# a framework pro tvorbu formulářových aplikací WPF.

5.1 Vývojové prostředí Visual Studio

Visual Studio je integrované vývojové prostředí, které bylo vyvinuto společností Microsoft za účelem vývoje grafického uživatelského rozhraní, konzolových a webových aplikací, mobilních aplikací, cloudu a webových služeb. Právě s pomocí tohoto IDE je možné vytvořit spravovaný i nativní kód. Nejedná se o jazykově specifické IDE, je možné ho využít k psaní kódu v C#, C++, Visual Basic, Python, JavaScript a mnoha dalších jazycích. Celkově poskytuje podporu pro 36 různých programovacích jazyků. Je k dispozici pro Windows, Linux i pro macOS. První verze Visual Studia byla vydána v roce 1997 a byla pojmenována jako Visual Studio 97 s verzí číslo 5.0. Nejnovější verze sady Visual Studio je 2019 verze 16.9. Tato verze má pak k dispozici tři různé edice. [19]



Obrázek 14 – Visual Studio

První z nich je Community. Tato edice je jediná bezplatná. Funkcemi se celkem podobá edici Professional. Commnutiy edice má však určitá omezení. Například v podnikové organizaci, pokud má organizace více než 250 počítačů a má roční tržby vyšší než 1 milion USD, tak nemá společnost oprávnění používat toto vydání. V nepodnikové organizaci může toto vydání používat až pět uživatelů. [19]

Druhou možností je již zmíněná edice Professional. Jedná se o komerční vydání sady Visual Studio. Ta poskytuje podporu pro úpravy XML a XSLT a zahrnuje nástroje jako je Průzkumník serveru a integrace s Microsoft SQL Serverem. Společnost Microsoft poskytuje bezplatnou zkušební verzi tohoto vydání. Hlavním účelem je poskytovat flexibilitu a s tím spojené profesionální vývojářské nástroje pro vytváření jakéhokoli typu aplikace, produktivitu, což zahrnuje výkonné funkce jako CodeLens, které zvyšují produktivitu a dále poskytuje spolupráci pomocí agilních nástrojů pro plánování projektů. [19]

Třetí možností je edice Enterprise. Jedná se o integrované komplexní řešení pro týmy jakékoli velikosti s náročnými požadavky na kvalitu a rozsah. Společnost Microsoft poskytuje 90denní bezplatnou zkušební verzi tohoto vydání. Hlavní výhodou tohoto vydání je, že je vysoce škálovatelné a poskytuje vysoce kvalitní software. [19]

Mezi hlavní výhody Visual Studia patří rychlé ladění, důkladné testování, správa verzí díky předem připravené integraci Gitu a GitHubu, spolupráce s využitím Live Share, která umožňuje celému týmu úpravy a ladění v reálném čase bez ohledu na jazyk nebo platformu. [20]

5.2 Jazyk C#

C# je jednoduchý, moderní, objektově orientovaný jazyk, který je odvozený z C++ a Javy. Jeho cílem je spojit vysokou produktivitu jazyka Visual Basic a hrubý výkon jazyka C++. Tento programovací jazyk byl vyvinut Andersem Hejlsbergem a jeho týmem během vývoje .Net Framework. Je navrženo pro CLI, což je složeno ze spustitelného kódu a běhového prostředí, které umožňuje použití různých jazyků vysoké úrovně na různých počítačových platformách a architekturách. Jde o strukturovaný jazyk, který je orientován na komponenty. [21]

Mezi hlavní výhody jazyka patří jeho jednoduchost. Na rozdíl od C++ nejsou v C# ukazatele, tj. není povolena přímá manipulace s pamětí. Jedná se o moderní jazyk, protože byl založen podle aktuálního trendu a je velmi výkonný a jednoduchý pro vytváření interoperabilních, škálovatelných a robustních aplikací. Za výhodu by se dalo považovat i to, že je objektově orientovaný a podporuje tedy zapouzdření dat, dědičnost, polymorfismus a rozhraní. Další výhodou je i to, že je typově bezpečný. [22]

5.3 Windows Presentation Foundation

WPF je zkratka pro Windows Presentation Foundation. Jedná se o výkonný framework pro vytváření aplikací Windows. WPF bylo nejprve představeno ve verzi .NET Framework 3.0 a poté byly do dalších verzí .NET Framework přidány další funkce. [23]

Framework nabízí velké množství hotových komponent, ze kterých je možné formulář jednoduše poskládat. Jedná se především o různá tlačítka, pole, popisky, posuvníky a další komponenty. Pokud by tyto komponenty nedostačovaly, tak je také samozřejmě možná tvorba vlastních ovládacích prvků. Ve WPF jsou prvky uživatelského rozhraní navrženy v XAML, zatímco chování je implementováno v procedurálních jazycích, jako například C#. Je tedy velmi snadné oddělit chování od kódu návrháře. S XAML mohou programátoři pracovat paralelně s designéry. [23][24]

```
<Window x:Class="HelloWorld.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:HelloWorld"
        mc:Ignorable="d"
        Title="HelloWorld" Height="450" Width="800" ResizeMode="NoResize">
    <Grid >
        <Label x:Name="helloLabel" Content="Text to change" Margin="279,29,0,0" FontSize="14"
               HorizontalAlignment="Left" VerticalAlignment="Top" />
        <Button Content="Change text" Click="Event_ChangeTxt" Margin="279,75,0,0"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="75" />
    </Grid>

</Window>
```

Obrázek 15 – Ukázka XAML pro HelloWorld aplikaci

S veškerými grafickými prvky se pracuje jako se samostatnými objekty. To platí nejen pro prvky používané ve formulářích, ale také pro grafické elementy jako jsou čtverce, kruhy atd. Každému grafickému objektu je tak možné přiřadit způsob chování, například při interakci s kurzorem myši, bez nutnosti složitých výpočtů, o jaký konkrétní objekt se jedná.

Kromě WPF je v .NET frameworku přítomný ještě starší formulářový framework Windows Forms. V současné době se paralelně využívají oba frameworky, ale WPF je technologicky mnohem pokročilejší. I když mnoho existujících aplikací stále používá Windows Forms, nové aplikace je již lepší vyvíjet právě pomocí WPF. [24]

Při porovnání WPF a Windows Form má WPF výhodu v tom, že Windows Forms nejsou uživatelské rozhraní založené na vektorech. Zatímco uživatelské rozhraní WPF je založené na vektorové grafice. Díky čemuž, umožňuje plynule škálovat komponenty uživatelského rozhraní, aniž by došlo k problémům se zkreslením velikosti. Další výhodou je, že ve Windows Form je obtížné přizpůsobit ovládací prvky, zatímco ve WPF je lze přizpůsobovat snadno. Také vykreslování ve WPF je rychlejší než ve Windows Form a WPF také poskytuje větší konzistenci mezi aplikacemi. [25]

6 NÁVRH APLIKACE

V této kapitole je popsána základní analýza návrhu aplikace. Nejdříve jsou zde popsány požadavky na vzhled, funkce a chování, které jsou kladené na finální aplikaci. Dále je jednoduše popsán UML diagram tříd. Nakonec jsou popsány vybrané programové části spolu s ukázkami kódu a také popis všech adresářů, které se nacházejí v projektu aplikace.

6.1 Požadavky na aplikaci

Nyní jsou vyjmenovány a popsány hlavní požadavky na výslednou aplikaci. Na aplikaci je kladeno celkem velké množství požadavků, všechny z nich jsou v této části práce zmíněny a stručně popsány. Mezi ty nejdůležitější z nich patří především:

- Funkční požadavky
    - Načtení a uložení projetu, nebo bloku
    - Dynamicky vytvářené menu
    - Vykreslování modulů podle hlavičky ve Verilog kódu
    - Umožnění propojování pinů modulů
    - Sestavení a nahrání obvodu do zvoleného kitu
    - Zamezení nevalidního propojení pinů
    - Zobrazení umístění komponent na vývojovém kitu
- Nefunkční požadavky
    - Pro systémy Windows
    - Pro FPGA Cyclone IV
    - Aplikace napsaná v C# WPF

Jak již bylo popsáno v úvodu bude výsledná aplikace sloužit jako grafický nástroj ve stylu diagram designéru. Ve stručnosti jde především o to, aby v aplikaci bylo možné vytvářet různé grafické bloky (moduly) a následně je bylo možné spojovat.

Při přidávání modulu na pracovní plochu aplikace je nutné moduly dynamicky graficky vykreslovat v závislosti na obsahu hlavičky každého modulu. Dle zjištěných informací následně vykreslit správný počet vstupních a výstupních pinů daného modulu. Někdy je také třeba některé piny nevykreslovat, viz popis vlastního atributu hidden v kapitole 3.3. Každý modul pak také musí obsahovat různé textové informace jako jeho název, názvy pinů a jeho vygenerovaný identifikátor. Samozřejmostí je možnost libovolného posouvání modulů a stejně tak jejich mazání z pracovní plochy.

Dále je nutné umožnit spojovat jednotlivé piny na modulech. Spojení může být dvojího typu, a to pomocí samostatného vodiče nebo pomocí sběrnice, která sdružuje více samostatných vodičů. Tyto propojení musí dodržovat určitá pravidla popsaná dále v práci a také musí obsahovat svůj identifikátor.

Menu, pomocí kterého se celá aplikace ovládá, obsahuje dynamicky vytvářené položky reprezentující celý obsah adresáře components. Toto má hlavní důvod v možnosti přidávání vlastních Verilog modulů. Aplikace pak není závislá na konkrétním obsahu adresáře. Horní menu se také bude svými povolenými možnostmi přizpůsobovat režimu projekt nebo blok a bude znemožňovat znovu použití některých z již použitých modulů.

Jak bylo zmíněno výše aplikace podporuje dva režimy tvoření obvodů, a to projekt a blok. Projekt je hlavní režim, kde v tomto režimu je umožněno obvod sestavit a nahrát do vývojového kitu. Také je zde možné používat již vytvořené bloky jako jakékoliv jiné moduly. I zde pak platí další podrobná pravidla, popsaná dále v práci. V režimu blok není umožněno provádět sestavení ani upload, protože je tento režim zamýšlen pro tvorbu uživatelských znovupoužitelných modulů. Z toho důvodu je zde možné vytvářet vstupní a výstupní piny z bloku, které se budou zobrazovat jako vstupní piny při použití v projektu. Až na tyto rozdíly je pak práce v projektu a bloku stejná. V obou těchto režimech bude umožněno ukládání a znovu načítání.

Dále jsou důležité požadavky na omezení používání více vývojových kitů a zamezení znovu použití již používané komponenty z kitu v jednom projektu, bloku a také v blocích použitých v projektu. Do vyvíjené aplikace je možné přidávat vlastní vývojový kit a návod, jak tak učinit je popsáno v kapitole 8.3.

O sestavení a nahrání obvodů do vývojového kitu se stará vývojové prostředí Intel Quartus. Jak již bylo napsáno jedná se o profesionální vývojové prostředí, ale díky své univerzálnosti je možné ho využívat na pozadí za pomocí dávkových souborů právě pouze pro sestavení a nahrání obvodu do vývojového kitu. Z důvodu využití prostředí Intel Quartus je také nutné určit adresářovou cestu ke kořenové složce tohoto programu. O tuto skutečnost se tedy vyvíjená aplikace musí postarat a zajistit vybrání této složky uživatelem.

Dalším požadavkem je možnost exportovat obsah pracovní plochy jako obrázek, aby bylo možné si rychle a efektivně vytvářet obrázky zapojení bez nutnosti používání jiných programů. Posledním funkčním požadavkem je umožnění zobrazení přesného umístění vybraných komponent na vývojovém kitu. K tomuto slouží atribut position, který je popsán v kapitole 3.3.

Nejzásadnějším nefunkčním požadavkem je jazyk, v jakém bude aplikace napsána. Zvoleným jazykem je C# konkrétně s grafickou nadstavbou WPF, protože se jedná o moderní programovací jazyk a nadstavba WPF patří k jedné z nejlepších pro tvorbu grafických aplikací,

což je stěžejní část výsledné aplikace. Se zvoleným programovacím jazykem úzce souvisí i požadavek na operační systém, kterým je Microsoft Windows. Nakonec je důležité určit pro jaké FPGA je vyvíjená aplikace určená, což je v tomto případě řada čipů Intel Cyclone IV.

6.2 Popis tříd

Tato část práce se věnuje popisu použitých tříd výsledné aplikace. K náhledu do tříd poslouží UML diagram tříd, který poskytuje zajímavý náhled na datovou strukturu aplikace, operace a souvislosti mezi různými objekty. Z důvodu velkého počtu tříd je diagram rozdělen na dva samostatné diagramy. První z nich znázorňuje třídy sloužící jako datové struktury a pomocnou třídou pro práci se soubory. Druhý diagram pak zobrazuje třídy, které slouží k zobrazení grafické části aplikace.

6.2.1 Popis datových a pomocných tříd

Tento diagram, který je přiložen jako příloha A na konci práce, popisuje strukturu datových tříd a jedné pomocné třídy sloužící k práci se soubory. Celkem je v aplikaci použito sedm datových tříd a jedna třída pro výčet. První třídou je LoadWindowProjects, která slouží k uložení a zobrazení informací a aktuálně dostupných projektech nebo blocích. Tato třída je použitá v grafických třídách EntryWindow a ChooseWindow.

Za hlavní datovou strukturu v aplikaci se dá považovat třída MenuData, která obsahuje informace potřebné k vytváření modulů. Jedná se o data načtená ze zdrojových Verilog modulů a uživatelských bloků. Tato třída uchovává data o struktuře modulu, tedy data jako název, cesta k souboru, vstupní a výstupní piny a skryté piny. V případě, že se jedná o komponentu z kitu je použita třída BoardInfo k uložení názvu kitu a souřadnic pro možnost zobrazení dané komponenty. Třída MenuData pak obsahuje seznam (List) tříd BoardInfo, který se následně předává i do třídy Module, protože tato třída je generována za pomoci informací z třídy MenuData.

Třída Module obsahuje informace o konkrétních modulech. O každém modulu je nutné mít informace (atributy) jako id, název, cesta ke zdrojovému Verilog kódu, vstupní a výstupní piny a také pár dalších informací. Tyto informace se také různí v závislosti, jestli jde o klasický Verilog modul nebo o uživatelský blok. V případě bloku se používají atributy s informací o použitých modulech v daném bloku a také obsahují cesty ke zdrojovému Verilog kódu.

Data o vstupních a výstupních pinech jsou ukládána ve třídě Pin, která je použita ve třídě Module. K určení, jestli se jedná o vstupní nebo výstupní pin slouží jednoduchá výčtová třída PinType. Třída Pin má atributy potřebné k uložení informací o konkrétním pinu na modulu.

Jedná se především o data jako o jaký pin se jedná, jestli o vodič nebo sběrnici, dále jsou důležitá data jako název vodiče, zda se jedná o skrytý pin nebo jestli je pin zapojený a popřípadě seznam aktivních spojení (třída ConnectionData) k pinu.

Aktivní spojení jsou ukládána ve vlastní třídě ConnectionData, kde jsou data o každém propojení, které bylo vytvořeno na pracovní ploše aplikace. Nejdůležitější atributy jsou název spojení, moduleInId a moduleOutId, které jednoznačně identifikují pomocí id počáteční a koncový modul a atributy pinInName a pinOutName sloužící k identifikaci konkrétních pinů na modulu pomocí názvu pinů.

Poslední datovou třídou je SaveDataContainer, která je používána, jak název napovídá, jako kontejner k uložení modulů, pinů a jejich spojení. Tato třída slouží k ukládání a načítání uložených projektů nebo bloků. Kromě seznamu modulů tu jsou také atributy k uložení posledního použitého id pro moduly a vodiče a také název použitého vývojového kitu.

Třída FileController slouží jako pomocná třída k veškeré práci se soubory v aplikaci, ať jde o načítání nebo jejich ukládání a je také nejobsáhlejší ze všech dříve zmíněných tříd. Atributy této třídy jsou hlavně konstanty s různými cestami ke zdrojovým souborům, jako jsou Verilog moduly, projekty, skripty pro sestavení a nahrání do kitu a také cesta k souboru s uloženou cestou ke vývojovému prostředí Intel Quartus. Tato třída také obsahuje větší množství metod a jejich celý výpis je možné vidět na obrázku v příloze A. Některé z nich jsou například GenerateMenuItems, která vytváří položky v dynamickém menu, CreateQuartusPathFile, pro vytvoření textového souboru s cestou k tomuto programu nebo OpenSaveFile, která načte uložený projekt nebo uživatelský blok.

6.2.2 Popis grafických tříd

Grafických tříd obsahuje výsledná aplikace celkem devět. Jedná se o třídy reprezentující různá grafická okna použitá v aplikaci. Vzhledem k velikosti obrázku je tento diagram přiložen jako příloha B na konci práce. Veškerá grafická okna jsou spouštěna z hlavní grafické třídy s názvem MainWindow.

Jedná se o hlavní okno aplikace s menu a pracovní plochou pro vytváření obvodů. Z této třídy je také volána většina metod z třídy FileController a také jsou zde použity datové třídy pro ukládání informací o aktuálním obsahu pracovní plochy. Tato třída je zodpovědná za veškerou práci s grafickými prvky, co se tvořených schémat týče. Je zde velké množství atributů potřebných ke správnému chodu aplikace jako například čítače pro identifikátory modulů a vodičů, seznam aktuálně vykreslených objektů Verilog modulů nebo konstanty pro velikosti

vykreslených modulů a jejich pinů. Toto je pouze malý výčet atributů a všechny je možné si prohlédnou v přiloženém diagramu.

Převážná většina metod slouží ke tvorbě grafiky modulů, vodičů nebo k jejich manipulaci či mazání. To jsou metody jako CreateModule nebo CreateConnection. Také tu je spousta různých eventů, které se přiřazují grafickým objektům (např. čtverec reprezentující modul) čímž se definuje jejich chování při různých událostech. Takovým eventem je například metoda EventMouseOver, která se vykonává při najetí kurzoru myši na daný objekt. Vzhled k tomu, že vyvíjená aplikace je z velké části založena na práci s grafickým prostředím, je v této hlavní grafické třídě velké množství metod a různých eventů pro obsluhu událostí.

Kromě hlavního okna aplikace tu jsou také pomocná grafická okna, tzv. dialogová okna. První třídou je EntryWindow, která je použita jako uvítací okno aplikace. Toto okno slouží jako základní rozcestník pro uživatele, který si může vybrat, zda chce otevřít projekt nebo blok, popřípadě jestli chce začít tvořit nový projekt nebo blok. Podobnou třídou, co se funkce týče, je ChooseWindow, které slouží pouze pro výběr projektu k otevření v aplikaci, potažmo bloku v závislosti na konkrétním použití této třídy.

Dále tu jsou třídy, které jsou vyloženě pouze dialogovými okny, které bylo nutné do aplikace dodat. První třídou je InputDialog, která slouží k zadávání textového vstupu od uživatele do aplikace, kde použití je například při pojmenování ukládaného projektu. Dále je tu třída FileChooserWindow, která slouží k výběru souboru a získání cesty vybraného souboru. V aplikaci je toto využito pouze pro vybrání cesty k prostředí Intel Quartus. V některých případech je také nutné vybrat vývojový kit, který chce uživatel použít a k tomuto slouží třída BoardWindow. Tato třída zobrazí okno, které má pouze rozbalovací seznam kitů a následně vrací vybraný kit.

Poslední tři třídy jsou třídami pro popis oken, slouží pouze k zobrazování informací a nemají žádný uživatelský vstup. První třídou z této kategorie je LoadingWindow. Jedná se o animované okno, které slouží jako načítací obrazovka v době sestavování nebo nahrávání obvodu do kitu. Text se mění pro sestavovaní a nahrávání na základě zadaných parametrů při vytváření této třídy. Po dokončení akce v LoadingWindow se zobrazí okno třídy InfoWindow, které zobrazuje textovou informaci o výsledku akce. Nakonec je tu třída PreviewWindow, kde se zobrazuje obrázek vybraného vývojového kitu spolu s ukazatelem v podobě žluté šipky, který ukazuje na komponentu na kitu. Tato třída slouží jako pomůcka pro identifikování umístění komponent na kitu.

6.3 Popis programových částí

Tato část práce obsahuje popis některých částí programu aplikace. Tyto části jsou vybrané funkční celky, jako například proces vytvoření nového modulu na pracovní ploše nebo proces sestavování obvodu v projektu. Každá vybraná část obsahuje popis, vysvětlení a také zde jsou ukázky částí samotného kódu.

6.3.1 Vytvoření grafického modulu

Celý proces vytvoření grafického znázornění modulu začíná spuštěním metody obsluhující událost při kliknutí na položku v menu odkazující na Verilog modul. Všechny metody, které jsou zde vyjmenovány jsou ve třídě MainWindow. K obsloužení této události slouží metoda MenuItemGenerateModule, která je počátečním bodem pro vytvoření jakéhokoliv modulu. V této třídě jsou nejprve získány informace o vytvářeném modulu z tzv. Tagu, ve kterém má položka menu uložen objekt typu MenuData s potřebnými informacemi o vytvářeném modulu.

```
private void MenuItemGenerateModule(object sender, RoutedEventArgs e)
{
    MenuItem el = sender as MenuItem;

    DeactivateMenuItem(el);

    MenuData data = el.Tag as MenuData;
    HashSet<string> usedModulesNames = new HashSet<string>();
    HashSet<string> usedModulesPaths = new HashSet<string>();

    if (data.IsBlock)
    {
        bool compatible = CheckBlockCompatibility(data, usedModulesNames, usedModulesPaths);
```

Obrázek 16 – Ukázka získání dat z položky menu

V případě, že se jedná o modul reprezentující komponentu kitu je tato skutečnost zjištěna a položka v menu bude deaktivována pro zamezení vytvoření nežádoucích duplicitních modulů. Také je zde kontrolováno, zda při vytváření modulu z uživatelského bloku není porušena kompatibilita (blok používá jiný kit nebo obsahuje již použité komponenty kitu na pracovní ploše). Po všech těchto kontrolách pak dojde k samotnému procesu vytváření grafiky modulu.

Vytváření samotného grafického znázornění modulu probíhá v metodě CreateModule, která se volá po předešlých kontrolních krocích. V této metodě je vytvořen kontejner typu grid, do kterého budou vkládány všechny grafické prvky, které bude výsledný modulu obsahovat. Díky umístění všech grafických prvků modulu do gridu, je pak možné při posouvání modulu po pracovní ploše přepočítávat souřadnice pouze jednoho objektu, čímž se zjednodušuje a optimalizuje proces manipulace s modulem. Šířka modulu je dána fixní hodnotou, ale výška modulu je vypočítávána podle počtu zjištěných pinů.

Prvním prvkem je objekt čtverce, který bude reprezentovat tělo modulu na pracovní ploše. Do tohoto čtverce je také umístěn objekt datové třídy Module, který obsahuje veškeré potřebné

informace o modulu, který bude reprezentován. Čtverec a kontejner grid jsou pak pouze vykresleny na pracovní plochu aplikace a jsou jim přiřazeny metody pro obsluhy událostí. Ukázka vytvoření čtverce reprezentující tělo modulu a také vytvoření gridu je na obrázku č. 17.

Po úspěšném vytvoření těla modulu jsou vytvářena grafická znázornění jeho pinů. K tomuto slouží metoda CreatePinsFromList, ve které jsou ze seznamu pinů vytvářeny jejich grafické reprezentace. Tato metoda se volá samostatně pro vytvoření vstupních a výstupních pinů, protože obě tyto skupiny pinů mají jiné barevné označení i umístění na těle modulu. Také zde dochází k různým kontrolám pro každý jednotlivý pin. Takovou kontrolou je zjištění, jestli se nejedná o skrytý pin, u kterého by nedošlo k vytvoření grafické podoby nebo k zjištění typu připojení daného pinu (vodič nebo sběrnice).

```
Rectangle mainRectangle = new Rectangle
{
    Margin = new Thickness(0, 0, 0, 0),
    Width = MODULEWIDHT,
    Height = height,
    RadiusX = 5,
    RadiusY = 5,
    Fill = GetLinearGradientFill(),
    Stroke = Brushes.Red,
    StrokeThickness = 0,
    Tag = module,
    Effect = GetShadowEffect()

};

hlavni = new Grid
{
    Margin = new Thickness(20, 20, 0, 0),
    Width = MODULEWIDHT,
    Height = height,
    Background = Brushes.Transparent,
    Tag = mainRectangle,
    Effect = GetShadowEffect()
};
```

```
hlavni.MouseEnter += EventMouseOverGrid;
hlavni.MouseLeave += EventMouseLeaveGrid;
hlavni.MouseLeftButtonDown += Module_MouseLeftButtonDown;
hlavni.MouseLeftButtonUp += Module_MouseLeftButtonUp;
hlavni.MouseMove += Module_MouseMove;
```

```
private static Label CreateTextBlock(int marginLeft,
    int marginTop, string text, HorizontalAlignment alignment) {
    return new Label
    {
        Content = text,
        Foreground = Brushes.Black,
        Width = 60,
        Height = 15,
        Padding = new Thickness(0, 0, 0, 0),
        HorizontalAlignment = HorizontalAlignment.Left,
        VerticalAlignment = VerticalAlignment.Top,
        HorizontalContentAlignment = alignment,
        FontSize = 9,
        Margin = new Thickness(marginLeft, marginTop, 0, 0)
    };
}
```

Obrázek 17 – Vytváření grafických objektů a obsluha události

Po těchto kontrolách je vytvářen objekt čtverce a textové položky s názvem pinu. Tyto grafické objekty jsou následně vykresleny přidáním do kontejneru grid a jsou přidány metody pro obsluhu událostí. Všechny vytvořené čtverce obsahují objekt typu Pin, pro zjednodušení práce při jejich propojování. Jedním z posledních kroků je vytvoření textových položek s názvem a identifikátorem modulu, které budou také umístěny do gridu modulu. K vytváření textových položek slouží metoda CreateTextBlock, která je na obrázku č. 17 vpravo dole a je použita také k vytváření názvu u pinů modulu.

```
CreateModule(data, out Module module, out Grid hlavni, 10 + pinsCount * 30, false);

CreatePinsFromList(data.InPins, data.HiddenPins, module, hlavni, 10, 15, PinType.IN);
CreatePinsFromList(data.OutPins, data.HiddenPins, module, hlavni, 130, 45, PinType.OUT);

hlavni.Children.Add(CreateTextBlock(30, 0, module.Name, HorizontalAlignment.Center));
hlavni.Children.Add(CreateTextBlock(30, (int)hlavni.Height - 15, module.Id, HorizontalAlignment.Left));
```

Obrázek 18 – Ukázka volání metod při vytváření modulu

6.3.2 Volání dávkových souborů

V aplikaci jsou volány dávkové soubory pro akce sestavení nebo nahrání obvodu na kitu, tyto akce jsou prováděny ve třídě FileController. Výchozí soubory pro tyto akce jsou ve složce scripts jak je podrobněji popsáno v kapitole 8.3. Proces volání dávkových souborů začíná po úspěšném vytvoření Verilog kódu pro nakreslený obvod a po zkopírování všech použitých modulů do složky projektu v adresáři workspace. Následuje příprava na kopírování souborů ze složky skriptů v metodě CopyBuildScriptFiles, kde dochází k výběru správné složky podle vybraného kitu a následnému kopírovaní souborů. Ke kopírování souborů se používá metoda CopyFile, která je používána i v jiných metodách.

```
string pureName = mainModuleName.Remove(mainModuleName.Length - 2);

text = text.Replace("TOP_LEVEL_ENTITY top", "TOP_LEVEL_ENTITY " + pureName);
text = text.Replace("VERILOG_FILE top.v", "VERILOG_FILE " + mainModuleName);

foreach (string modulePath in usedModules)
{
    string module = System.IO.Path.GetFileName(modulePath);
    string addModul = "\nset_global_assignment -name VERILOG_FILE " + module;
    text += addModul;
}
```

Obrázek 19 – Úprava konfiguračního souboru

Po vytvoření kopií všech potřebných modulů a skriptů je nyní nutné upravit dávkové a konfigurační soubory pro aktuální projekt. Tyto činnosti se dějí v metodě ChangeScriptsSetting, kde je nejprve upraven jeden konfigurační soubor (ostatní není potřeba upravovat) s příponou qsf. Tento soubor je nutné upravit, aby odkazoval na Verilog modul se stejným názvem jako má projekt. Jedná se totiž o hlavní modul, který obsahuje informace o propojení všech ostatních použitých modulů. Nakonec je v tomto konfiguračním souboru nutné vyjmenovat veškeré použité moduly.

Po těchto úpravách už jsou pouze upraveny dávkové soubory, aby obsahovaly správnou adresářovou cestu k prostředí Quartus. Nyní jsou veškeré přípravy hotové a je možné spustit samotný dávkový soubor pro sestavení a poté pro nahrání do kitu. K spuštění dávkového souboru je používán příkazový řádek systému Windows, který je spuštěn na pozadí, aby nedošlo k zobrazení příkazového řádku uživatelovi aplikace.

Po dobu vykonávání dávkového souboru je v aplikaci zobrazována animovaná načítací obrazovka. Aby bylo možné zobrazit animované okno je nutné vytvořit nové vykonávací vlákno, které bude zobrazovat načítací obrazovku zatím co bude hlavní vlákno aplikace čekat na dokončení dávkového souboru. K vytvoření nového vlákna dochází v metodě

CreateNewThread a vlákno je ukončováno hlavním vláknem pomocí metody CloseWindowSafe, která odešle signál k bezpečnému zavření načítací obrazovky, popřípadě k nucenému ukončení.

```
private void CreateNewThread()
{
    Thread newWindowThread = new Thread(new ThreadStart(ThreadStartingPoint));
    newWindowThread.SetApartmentState(ApartmentState.STA);
    newWindowThread.IsBackground = true;
    newWindowThread.Start();
}
private void CloseWindowSafe()
{
    if (loadingWindow.Dispatcher.CheckAccess())
    {
        loadingWindow.Close();
    }
    else
    {
        loadingWindow.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,
            new ThreadStart(loadingWindow.Close));
    }
```

Obrázek 20 – Ukázka vytvoření a ukončení načítacího okna

6.3.3 Čtení Verilog modulů

Při vytváření položek menu pro veškeré moduly v adresáři components je nutné nejen získat seznam modulů, ale také z nich získat veškeré potřebné informace. Získávání informací z kódu modulu se děje v metodě ReadAndProcessFile ve třídě FileController. Získávání informací je prováděno pomocí regulárních příkazů, které zajišťují rychlé a efektivní zpracovávání textů kódů.

Prvním krokem je načtení samotného kódu modulu a pomocí prvního regulárního příkazu rozdělení textu kódu na textové pole, kde jsou potřebné informace rozděleny do předem známých pozic v poli. V tomto případě dojde na rozdělní jména, seznamu pinů modulu a zbytku kódu modulu, který je ještě dále zpracováván. U jména modulu je nutné pouze odstranit prefix z názvu modulu. Seznam pinů je nutné očistit o bílé znaky a po rozdělení textu, podle znaku čárky, je už snadné identifikovat typ a název pinu. Jak název modulu, tak seznam pinů je uložen do objektu datové třídy MenuData, která ukládá tyto informace.

Dále je zpracován zbytek kódu, ve kterém se hledají vlastní atributy hidden a position spolu s příslušnými hodnotami. Pokud je regulární výraz úspěšný a je nalezen atribut hidden je kód rozdělen do skupin v novém poli. Pozice je zde opět předem známá a seznam atributů hidden je nachází v poli na druhém místě. Podobně jako u seznamu pinů, i zde je nutné odstranit bílé znaky a rozdělit seznam skrytých pinů podle znaku čárky. Tento rozdělený seznam je už pouze zpracován a uložen do objektu MenuData.

U atributu position se mohou vyskytovat informace pro více kitů najednou. V případě, že je uvedeno v atributu více kitů, jsou tyto informace rozděleny pomocí středníku. Je tedy nutné

nejprve rozdělit seznam kitů pomocí středníku a následně zpracovat informace pro každý kit zvlášť pomocí regulárního výrazu. Ten rozdělí text na tři skupiny, a to samostatné souřadnice x, souřadnice y a na jméno kitu. Tyto údaje jsou uloženy do datové třídy BoardInfo, která je také uložena do objektu MenuData, které bylo dříve vytvořeno.

Název modulu a seznam pinů

```
string text = File.ReadAllText(path);
string[] textSplited = Regex.Split(text, "module ([\\w\\d]+[ ]?)\\(\\s*([\\w\\d,_\\]\\[:\\n\\s(]*)\\);");
```

Očištění a rozdělení pinů

```
string[] pins = Regex.Replace(textSplited[2], @"\s+", "").Split(',');
```

Rozdělení podle hidden a position

```
string[] textSplitedTwo = Regex.Split(textSplited[3], "\\/\\/hidden:\\s([\\w\\d, ]+)[\\n|\\r\\n]" +
    "(\\/\\/position:\\s([;]?\\d{1,3},\\d{1,3},\\w*)*)?");
```

Rozdělení souřadnic a jména u atributu position

```
string[] coordinatesAndName = Regex.Split(oneBoard, "((\\d{1,3}),(\\d{1,3}),(\\w*))");
```

Obrázek 21 – Ukázky regulárních výrazů

6.4 Popis jednotlivých adresářů

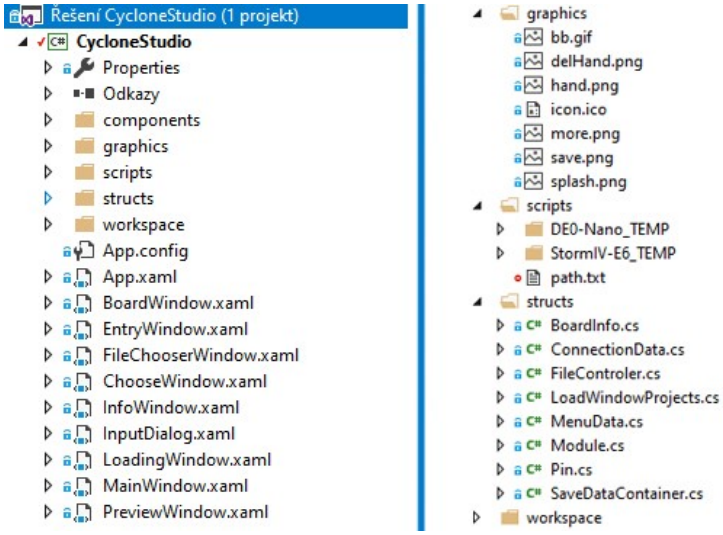
Nyní je zde popsána adresářová struktura projektu, kde jsou popsány jednotlivé složky, popřípadě obsah daných složek. V hlavní adresářové struktuře jsou také uloženy veškeré zdrojové kódy pro grafické třídy, kdy tyto třídy již byly popsány. První složkou v adresářové struktuře je složka components, kde jsou uložené všechny předdefinované Verilog moduly, které si aplikace při chodu načítá. Konkrétní obsah a popis této složky je již popsán ve třetí kapitole této práce, a tak zde je popis vynechán.

Složka graphics je úložné místo pro všechny obrázky (grafiky), které jsou v aplikaci použity jako různé grafické prvky. Jsou tu například obrázky pro úvodní načítací obrazovku aplikace (splash) nebo ikona samotné aplikace. Dále také dva obrázky (delHand a hand) pro ikony ovládacích tlačítek pracovní plochy.

Ve složce scripts jsou umístěné výchozí skripty pro sestavování a nahrávání do kitů. Je tu složka pro každý kit obsažený v aplikaci a také je tato složka místo pro textový soubor s cestou pro prostředí Quartus. Každá složka pro daný kit obsahuje dávkové soubory, tzv. bat soubory, pro spuštění, sestavení nebo nahrání obvodu na kit pomocí prostředí Quartus. Kromě těchto bat souborů jsou zde i konfigurační soubory kitu, které definují různé parametry a přiřazení pinů kitu. Obsahy těchto složek jsou neměnné a soubory se pouze kopírují.

Další složkou je složka structs, ve které se nacházejí veškeré datové a pomocné třídy aplikace a jejich popis je na začátku kapitoly 6.2.1. Poslední složkou je složka workspace, která slouží jako uložiště projektů. Každý projekt je uložen ve složce pod svým jménem. Do této složky jsou také kopírovány zdrojové Verilog moduly. To se hodí například při sestavování projektu, protože Verilog moduly jsou nutné při kompilaci obvodu a je tedy výhodné shromáždit všechny

potřebné soubory v jedné složce spolu se skripty a konfiguračními soubory. Výhoda pak tkví především v usnadnění odkazování na každý použitý modul a také v absenci obavy o možné poškození modulů nebo skriptů, protože se modifikují pouze kopie, a ne originální soubory.

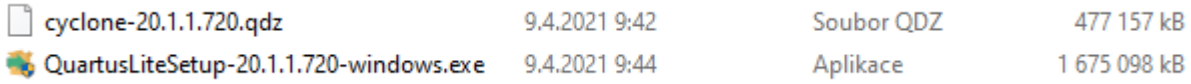


Obrázek 22 – Adresářová struktura

7 INSTALACE A ZPROVOZNĚNÍ APLIKACE

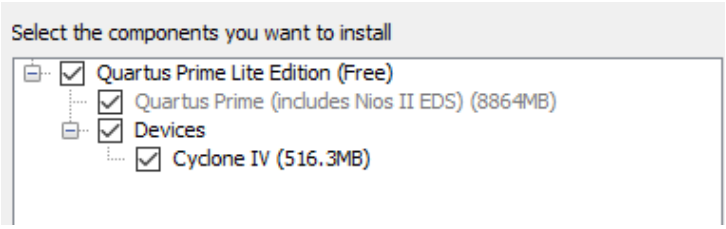
Tato část práce se věnuje instalaci a zprovoznění vyvíjené aplikace, což bude psáno formou návodu s ukázkami. Prvním krokem je stažení všech potřebných souborů, kdy kromě samotné vyvíjené aplikace je nutné stáhnout vývojové prostředí Intel Quartus. Ke správnému fungování postačuje Lite Edition, která je dostupná zdarma. Quartus je k dostání na stránkách společnosti Intel na adrese fpgasoftware.intel.com.

Ke stažení je nutné si založit účet na těchto stránkách, ale to nezabere více než pár minut a po přihlášení je už možné začít stahovat. Na stránkách je možné vybrat různé verze prostředí. Aplikace byla testována společně s prostředím Quartus Lite verze 18.1 a proto je doporučené si stáhnout tuto nebo novější verzi. Kromě samotného prostředí Quartus je také nutné stáhnout balíček pro podporu FPGA řady Cyclone IV.



Obrázek 23 – Ukázka souborů pro Intel Quartus

Pokud jsou oba soubory (instalační exe soubor a balíček pro podporu řady FPGA) stažené je nutné je umístit do stejné složky, aby byl balíček při instalaci nalezen a taktéž nainstalován. Instalace vývojového prostředí je stejná jako instalce jakéhokoliv jiného programu. Tedy zahájení instalace, akceptování podmínek a výběr složky pro instalaci. Po těchto krocích je nutné zkontrolovat, jestli byl nalezen balíček s podporou pro FPGA viz obrázek č. 24.

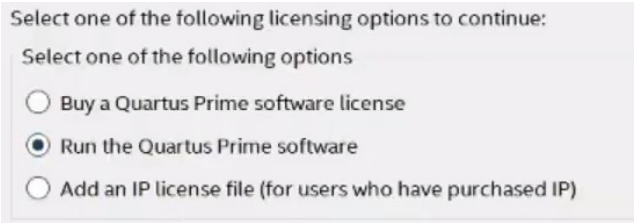


Obrázek 24 – Volba instalovaných komponent

Následně už dochází k samotné instalaci prostředí a doba instalace se pohybuje okolo několika minut. Po dokončení instalace prostředí Quartus se zobrazí okno ohlašující dokončení instalace a jsou na výběr další kroky. Kromě vytvoření zástupce na ploše a spuštění programu Quartus je také důležitá volba pro instalaci ovladačů pro USB Blaster, kterým se nahrávají soubory do vývojového kitu.

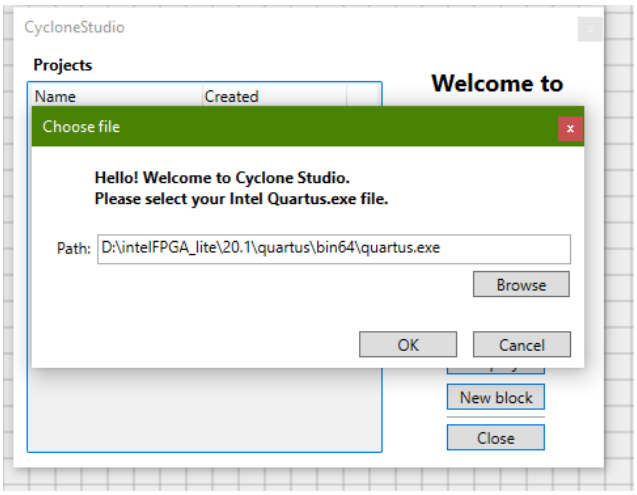
V případě, že uživatel nemá tyto ovladače nainstalované je nutné si je také instalovat do počítače. Instalace je opět jednoduchá a zabere pár minut. Nyní je Intel Quartus úspěšně nainstalovaný a zbývají poslední kroky k dokončení, kdy je ještě nutné při prvním spuštění

zakliknout správnou volbu u licencování produktu viz obrázek č. 25. Správná volba je druhá v seznamu s názvem Run the Quartus Prime software. Ostatní volby jsou pouze pro licencování placených verzí.



Obrázek 25 – Volby u prvního spuštění

Nyní je nainstalován veškerý potřebný software pro správné fungování vyvíjené aplikace. Samotnou aplikaci CycloneStudio není nutné nijak instalovat, ale pouze ji stačí stáhnout a umístit do libovolného adresáře. Nesmírně důležité je zachovat stejnou strukturu adresářů jakou má aplikace po stažení. Jde především o adresáře bin, components, graphics, scripts a workspace. Bez těchto adresářů nebude aplikace fungovat správně. Spouštěcí soubor aplikace je umístěn ve složce bin v podsložce release. Z tohoto exe souboru je možné si vytvořit zástupce, který pak lze umístit do libovolné složky. Při prvním spuštění aplikace se zobrazí okno pro zadání cesty k programu Quartus viz obrázek č. 26. Po zadání správné cesty je aplikace připravená k používání.



Obrázek 26 – Vybrání cesty k Intel Quartus

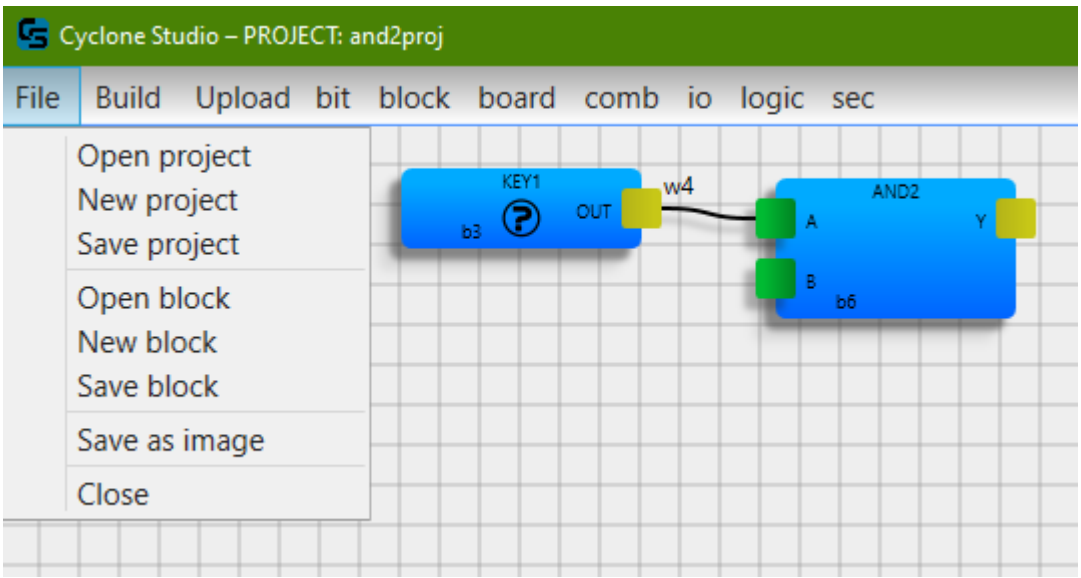
8 OVLÁDÁNÍ APLIKACE

Tato část práce je věnována popisu samotné vyvíjené aplikace. Je zde popsáno rozložení grafického prostředí a popis práce se samotnou aplikací. Dále zde jsou popsány návody na přidání vlastního Verilog modulu a také přidání vlastního vývojového kitu do aplikace. Následuje popis chování a základních pravidel pro správnou práci s aplikací, jako popis ovládání nebo rozdílů ve volbě projekt a blok. Dále jsou vysvětleny volby build a upload, a nakonec jsou popsány funkce a grafické rozložení některých dalších oken použitých v aplikaci.

8.1 Popis práce s aplikací

V této části je popsáno rozložení grafických prvků aplikace a práce s aplikací. Rozložení je jednoduché, rozděluje se na horní část, kde je umístěná lišta okna společně s hlavním menu a na spodní část, kde se nachází pracovní plocha, jak je vidět na obrázku č. 27. Lišta okna je použita jako informační oblast, kde se zobrazuje informace, jestli se právě vytváří projekt nebo blok a také je zde název projektu, či bloku, pokud již byl nějaký název zadán.

Pod lištou je hlavní menu aplikace. Z toho menu se provádí téměř všechny hlavní úkony v aplikaci. Konkrétně se jedná o vytváření nových projektů nebo bloků, jejich ukládání a načítání, exportování pracovní plochy jako obrázku, také tu jsou položky pro sestavení a nahrání obvodu do kitu a nakonec se přes menu přidávají moduly na pracovní plochu z dynamicky vytvářených položek menu.



Obrázek 27 – Menu a moduly v aplikaci

V položce menu s názvem File jsou umístěny položky starající se o správu projektu nebo bloku. Tyto položky jsou rozdělené pomocí čáry do pomyslných skupin. Pomocí položek

začínajících slovem Open se vyvolává okno pro otevření uloženého projektu nebo bloku. U položek New se vytváří nová čistá pracovní plocha a u Save položek dochází k vyvolání okna pro zadání jména, které je nutné pro uložení projektu nebo bloku.

V případě, že si uživatel aplikace bude chtít vytvořit screenshot nakresleného obvodu může tak učinit rovnou z aplikace pomocí položky Save as image. Také je zde položka Close pro zavření aplikace. Všechny tyto zmíněné položky jdou ovládat pomocí klávesnice, kdy po zmáčknutí tlačítka Alt je možné vyvolávat položky v menu pomocí jejich prvních písmen. Například pro zavření aplikace je možné zmáčknout Alt, poté klávesu f pro otevření položek ve volbě File a následně tlačítko c, které vyvolá položku Close a tím se zavře aplikace.

Dalšími položkami v menu jsou volby Build a Upload, které slouží, jak název napovídá, k sestavení obvodu a nahrání obvodu do kitu. Na tyto položky stačí pouze kliknout a vše se následně vykoná automaticky, jenom v případě, že není vybrán kit je nutné ho před sestavením vybrat.

Přidávání modulů na pracovní plochu se také provádí přes menu, kdy jsou moduly rozděleny do různých podmenu. Pro samotné přidání na pracovní plochu je nutné kliknout na příslušnou položky v menu, která reprezentuje požadovaný modul a poté je na pracovní ploše přidána grafická podoba zvoleného modulu. Některé moduly lze přidávat opakovaně a některé pouze jednou, tyto pravidla jsou popsány v další kapitole.

Důležitou součástí aplikace je vizualizace samotných modulů, které jsou reprezentovány jako modré obdélníky nebo čtverce. Každý modul má stejnou šířku a výšku má každý modul podle počtu pinů, jak je vidět na obrázku č. 27. Na levé straně modulu jsou umístěny vstupní piny modulu, ty jsou reprezentovány pomocí zelených čtverců. Na pravé straně modulu jsou umístěny výstupní piny a jedná se také o čtverce, ale pro přehlednost mají žlutou barvu. Každý z pinů má své označení, které je umístěno vždy vedle příslušného pinu.

Také každý modul má své označení, a to rovnou dvojího typu. Prvním je název modulu, který je umístěný uprostřed na horní části modulu a obsahuje název podle toho o jaký modul se jedná, například to může být AND2, OR3 nebo KEY0. Tyto názvy nejsou žádné identifikátory a mohou se tedy na modulech opakovat. Druhým typem označení je unikátní identifikátor modulu, který je platný v rámci jednoho projektu nebo bloku. Je umístěný na dolní straně modulu blíže levé straně a vždy začíná na písmeno **b** následované číslicí. Tyto identifikátory jsou generovány aplikací postupným vkládáním modulů.

Některé moduly mohou obsahovat symbol otazníku v kolečku. Toto označení signalizuje, že se jedná o modul reprezentující komponentu na kitu, u které je možné zobrazit její pozici na fotografii kitu. Zobrazení umístění se vyvolá stisknutím prvého tlačítka myši na modulu s tímto

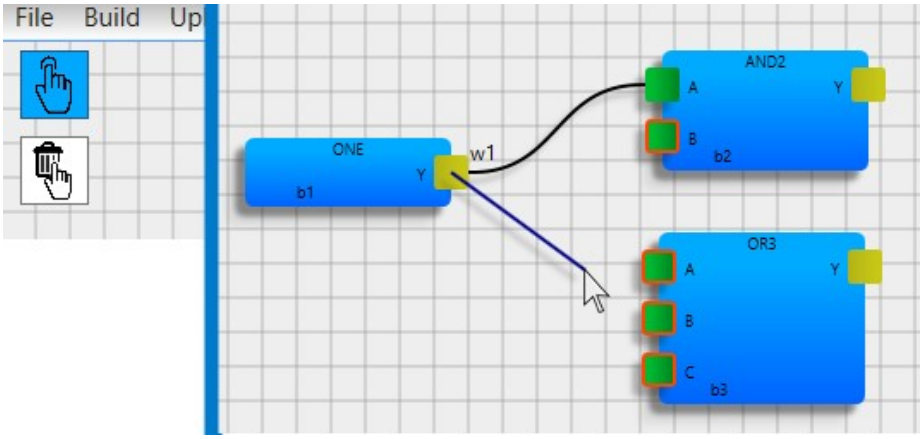
symbolem. V případě, že se jedná o moduly pinů s označením jedna až deset, může být nejdříve zobrazeno okno pro výběr kitu a to v případě, že ještě nebyl vybrán konkrétní kit pro projekt nebo blok. Hlavní ovládání modulů se provádí pomocí levého tlačítka myši, ovšem podrobnější popis pro posouvání, přidávání, odebírání a propojování je popsán v dalších kapitolách.

Jak bylo zmíněno, tak pod menu se již vyskytuje samotná pracovní plocha aplikace. Na tuto plochu jsou vytvářeny moduly, s kterými je pak možné pracovat. V levém horním rohu pracovní plochy jsou umístěny dvě přepínací tlačítka pro volby posouvání nebo propojování (obrázek ruky) a druhé pro volbu mazání modulů nebo vodičů a sběrnic (obrázek ruky s košem).

V případě, že je zakliknuté (je modré) horní přepínací tlačítko je možné moduly přesouvat po pracovní ploše pomocí tzv. drag and drop operace, kde je možné každý modul držením levého tlačítka myši přesouvat libovolně po pracovní ploše a po uvolnění levého tlačítka zůstane modul na nové pozici. Také je možné propojovat piny modulů a to tak, že se klikne levým tlačítkem myši na pin, ze kterého se má vytvořit spojení a poté se opět klikne levým tlačítkem na koncový pin. Zásady pro propojování pinů jsou popsány v další kapitole.

Pro možnost mazání modulů nebo propojení je nutné zakliknout dolní přepínací tlačítko. Poté je možné mazat pouhým kliknutím levého tlačítka na příslušný modul nebo spojení, které má být smazáno. Před smazáním modulu nebo vodiče je uživatel dotázán pomocí dialogového okna, zda chce objekt skutečně smazat.

Pokud je možné provádět nějakou akci s některým z modulů nebo pinů jsou po najetí myši označeny červeným ohraničením. Při vytváření propojení jsou piny, do kterých je možné toto propojení vést, označeny podobně jako po najetí myši, jak je vidět na obrázku č. 28. Aplikace také umožnuje, aby si uživatel mohl zvětšit velikost modulů na pracovní ploše pomocí držení tlačítka ctrl a scrolováním kolečka myši.



Obrázek 28 – Ukázka tlačítek a tvorby spojení

8.2 Přidání vlastního Verilog modulu

Pokud by si uživatel chtěl přidat vlastní Verilog modul do aplikace je to možné, ale je nutné dodržet některá pravidla. Nejdůležitějším pravidlem je dodržení správného umístění nového modulu. Ten musí být umístěn ve složce components a to buď do některé z již existujících složek, kromě složek block a board, nebo do nově vytvořené složky. Po restartování aplikace bude nový modul umístěn v menu dle složky umístění. V případě nové složky, bude v menu aplikace po restartování automaticky přidána nové položka menu s názvem složky a veškerými moduly, které složka obsahuje.

Dále je důležité dodržet formátování kódu Verilog modulů, jako u předdefinovaných modulů. Jde především o formát hlavičky moduly, ze které jsou zpracovávány informace pro správné vykreslení modulu. Vlastní modul musí začínat slovem **module**, to je následováno mezerou a názvem samotného modulu, přičemž název musí obsahovat prefix **c** (např. cMujModul). Po názvu modulu následují kulaté závorky ukončené středníkem, ve kterých může být napsán seznam vstupních a výstupních pinů. Piny mohou být dvojího typu, a to buď vstupní (**input wire**) anebo výstupní (**output wire**). I piny mají určený formát psaní, nejdříve je uveden typ pinu, následuje mezera a pak samotný název pinu. Každý pin je nutné oddělit pomocí čárky. Ukázková hlavička je vidět na obrázku č. 29.

Nakonec je důležité, aby název souboru byl stejný jako název modulu definovaný v hlavičce, a také aby měl příponu souboru .v jinak by modul nebyl aplikací rozpoznán.

```
1  module cModuleName(input wire PinOne, output wire PinTwo);
```

Obrázek 29 – Hlavička modulu

8.3 Přidání vlastního vývojového kitu

Aplikace umožňuje přidání vlastního vývojového kitu, ale pro správné fungování je třeba dodržet následující podmínky a pravidla. Zkráceně je nutné vytvořit příslušné Verilog moduly pro daný kit, vytvořit scripty pro sestavení a nahrání obvodu do kitu, přidat fotografie do složky graphics a upravit některé existující moduly.

Dále je nutné dodržet stejné pojmenovávání, aby veškeré názvy obsahující název kitu byly vždy napsány zcela stejně. Jedná se především o název složky kitu ve složce board, název fotografie kitu, název složky obsahující skripty ke kitu, kde u této složky musí být postfix _TEMP a také název samotných skriptů. Příklad správného pojmenování je uveden v tabulce č. 6.

Moduly reprezentující komponenty na kitu je nutné umístit do složky s názvem kitu a tuto složku umístit do podsložky board ve složce components. Moduly musí být napsané stejně jako

již existující moduly komponent jiných kitů. Každý modul má své vstupní a výstupní piny, ovšem v aplikaci jsou typicky zobrazována pouze kity na vstupní nebo na výstupní straně modulu. To, které piny budou v aplikaci graficky viditelné, se nastavuje pomocí speciálního atributu hidden, jehož popisu se věnuje kapitola 3.3 v této práci. Tyto schované piny jsou používány při sestavování, kdy jsou propojovány s fyzickými piny.

Pro zprovoznění fungování funkcí sestavení a nahrávání je nutné vytvořit ve složce scripts příslušnou složku a do ní zkopírovat obsah z již přidaného kitu, který je nutné následně upravit. Mezi první úpravy patří správné přejmenování souborů s příponami qsf, qpf a qws. Dále je nutné v souborech build.bat a upload.bat změnit název projektu (set PROJECT=) na název přidávaného kitu. Název projektu je také nutné změnit v souboru s příponou qpf.

Nejvíce práce je u souboru s příponou qsf. Tento soubor je nutné z velké části upravit a přepsat podle přidávaného kitu. Je nutné upravit informace o samotném přidávaném kitu (položky FAMILY, DEVICE atd.), ale především je nutné přiřadit fyzické piny kitu k názvům skrytých pinů Verilog modulů, které mají být použity k daným účelům. Aplikace předpokládá, že kit bude mít aspoň deset vstupních a deset výstupních fyzických pinů. Jedná se o moduly ve složce io, které obsahují v názvu číslovku. Je nutné názvy skrytých pinů z těchto modulů také přiřadit, jinak nebudou pro přidávaný kit funkční. Názvy fyzických pinů a jejich účel je nutné zjistit k dokumentace kitu. Pro připomenutí i názvy pinů musí být napsány zcela identicky jako jsou napsány v hlavičce příslušných Verilog modulů.

Název kitu	MujKit
Název složky v boards	MujKit
Název fotografie v graphics	MujKit.jpg
Název složky ve scripts	MujKit_TEMP
Název skriptů	MujKit.qsf, MujKit.qpf, MujKit.qws
Úprava dávkových souborů	set PROJECT=MujKit

Tabulka 6 – Pojmenování u nového kitu

V případě, že uživatel bude chtít, aby fungovala funkce ukázky komponenty na fotografii kitu, je nutné do kódu každého nového modulu přidat speciální atribut position, tak jak je to popsáno v kapitole 3.3. Zároveň je nutné přidat fotografii ve formátu jpg do složky graphics, přičemž ideální rozlišení fotografie je okolo 500x500 pixelů. Pro zprovoznění ukázky také u prvních deseti vstupních a výstupcích pinů je nutné přidat kit do atributu position u těchto

předdefinovaných modulů. Souřadnice komponent na fotografii je možné zjistit pomocí různých grafických nástrojů, které zobrazují pozici vůči osám x a y.

8.4 Základní pravidla

V této kapitole jsou popsána pravidla pro práci s aplikací, je popsán způsob provádění různých úkonů v aplikace jako je přidávání a odebírání modulů, propojování pinů, kdy je popsáno, jaké piny je možné propojovat a kdy to možné naopak není. Dále jsou popsány módy aplikace pro tvorbu projektu a pro tvorbu bloku. Nakonec jsou popsány volby build a upload a také některá okna aplikace.

8.4.1 Přidávání a odebírání modulů

Přidávání modulů se provádí kliknutím na příslušnou položku v horním menu, čímž dojde k vytvoření modulu na pracovní ploše. Většinu modulů lze vytvářet na pracovní ploše opakovaně. Jedná se o moduly ze skupin: bit, block (pokud neobsahuje modul reprezentující komponentu kitu viz. dále), comb, logic a sec. Naopak moduly reprezentující komponenty, které jsou v menu pod položkami board a io, se mohou vyskytovat v projektu, ale i v použitých blocích vždy pouze jednou. Takovýto modul po přidání na pracovní plochu v menu zešedne a nedá se již přidat.

Dále je tu omezení na použití pouze jednoho vývojového kitu v rámci projektu a popřípadě i bloků, které jsou v rámci projektu použity. Výběr kitu se provádí intuitivně tím, že je na pracovní plochu přidán modul konkrétního kitu, čímž dojde k zablokování možnosti přidání jakýchkoliv dalších modulů, které nepatří do zvoleného kitu. Pro změnu vybraného kitu stačí odstranit všechny moduly z původního kitu a poté je aplikací umožněno přidat moduly kitu jiného.

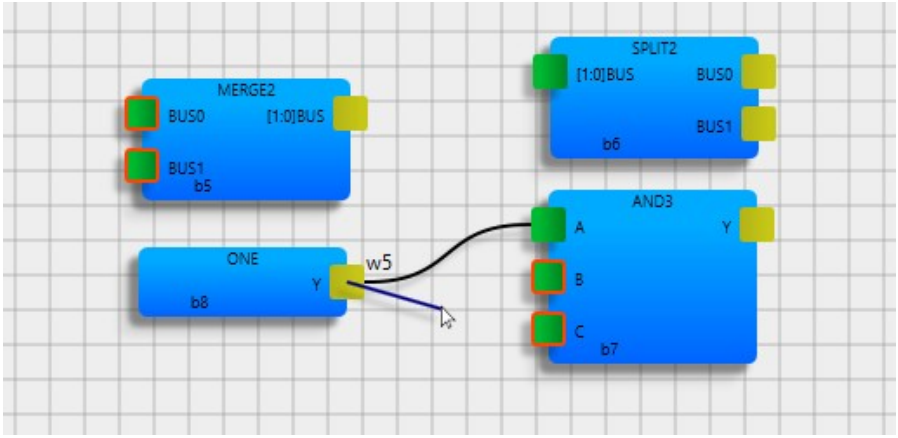
Přidání modulu z uživatelského bloku do projektu je možné pouze v případě, že blok používá stejný vývojový kit, jako je použit v projektu, kam se přidává, nebo nepoužívá žádný kit. V případě, že blok obsahuje některé komponenty z kitu, nesmí být tyto moduly již použity v rámci projektu. Pokud nejsou aplikací nalezeny žádné kolize v různých kitech nebo použitých modulech jsou po přidání bloku v menu deaktivovány všechny použité moduly, které reprezentují komponenty. Odstranění jakéhokoliv modulu je jednoduché a nejsou tu žádné podmínky a stačí pouze klikat na zvolené moduly při zvoleném přepínači aplikace na mazání.

8.4.2 Propojování pinů

Pro propojování pinů jsou stanovená jasná pravidla, které aplikace hlídá a nedovolí jejich porušení. Hlavním pravidlem je, že je možné propojit pouze piny, které jsou opačného typu. Tedy vždy lze propojit pouze vstupní a výstupní pin.

Dalším pravidlem je, že na jeden vstupní pin může být připojen pouze jeden výstupní pin, ovšem výstupní piny mohou být napojeny na více vstupních pinů. Pokud je z nějakého pinu možné vést nové propojení má na sobě po najetí kurzoru myši červené ohraničení. Propojení je možné začít jak kliknutím na vstupní, tak na výstupní pin. Po kliknutí na pin aplikace označí červeným ohraničením všechny piny, na které je možné se připojit.

Samotné piny také mohou být dvojího typu, a to klasický vodič nebo sběrnice. Tyto piny nelze navzájem propojovat. Vždy je možné propojit pouze piny stejného typu a u sběrnic navíc lze propojit pouze piny se stejnou šířkou rozhraní sběrnice. Toto ovšem hlídá samotná aplikace a není možné vytvořit nevalidní propojení.



Obrázek 30 – Ukázka nabízených propojení

8.4.3 Bloky

Uživatelské bloky jsou opakovatelně použitelné obvody, které se dají využívat v rámci projektů pro zjednodušení výsledného schématu obvodu. V aplikaci je možné vytvářet nové bloky a ukládat nebo otevírat již uložené bloky. Pokud je vytvářen blok je tato informace napsána v liště aplikace.

V režimu tvorby bloku není možné provádět sestavení nebo nahrání obvodu do kitu, ovšem na rozdíl od projektu je tu možné vytvářet vlastní piny, ať už vstupní nebo výstupní, které jsou používány jako vstupní nebo výstupní body do nebo z bloku. Tyto vlastní piny jsou po použití v projektu viditelné jako piny modulu. Každý vlastní pin je nutné vhodně pojmenovat v textovém dialogu, který aplikace zobrazí.

V rámci bloku je možné používat všechny předdefinované moduly, dle pravidel popsaných v kapitole 8.4.1. Pro použití v rámci projektu je nutné, aby byl blok validní, což znamená, že všechny použité moduly mají propojené všechny vstupní piny. V opačném případě se blok uloží, ale není možné ho použít v rámci projektu.

8.4.4 Projekty

V tomto režimu aplikace je již možné obvod sestavit a nahrát do kitu, ale není zde možné vytvářet vlastní piny. I v tomto případě je v horní liště aplikace zobrazeno, že se právě kreslí obvod v režimu projekt. Projekty je možné vytvářet, ukládat i nahrávat již uložené. Je zde možné používat pouze předdefinované moduly nebo validní uživatelské bloky. Bloky je možné v jednom projektu používat opakovaně pouze v případě, že nepoužívají některou z komponent kitu.

8.4.5 Časté chyby

Mezi nejčastější chyby, které by se mohly při vytváření obvodu vyskytnou patří především vynechání zapojení pinu, díky kterému nepůjde buď sestavit projekt nebo nepůjde použít blok v rámci projektu. Všechny vstupní piny musí být vždy zapojené, aby byl výsledný obvod funkční.

Další častou chybou, která se může vyskytnout je, že pro změnu kitu je třeba smazat veškeré moduly z kitu jiného. Většinou jsou všechny takovéto moduly označené symbolem otazníku, ovšem některé jako například modul pro hodinový signál takovýto symbol nemají a může je pak být snadné přehlédnout.

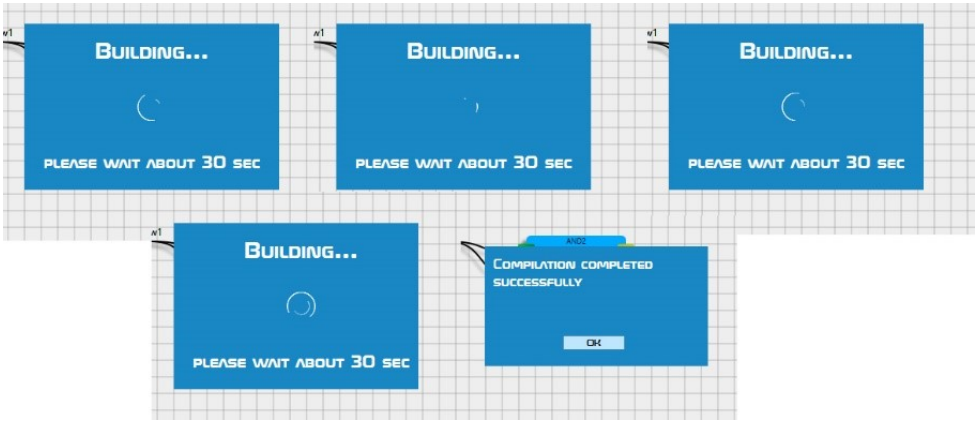
Pokud došlo k zadání špatné cesty k vývojovému prostředí Quartus je nutné smazat soubor path.txt, který je uložen ve složce s aplikací CycloneStudio ve složce scripts. Po tomto úkonu se aplikace při příštím spuštění opět zeptá na cestu a je možné je již zadat správně. Důležitým poznatkem je také to, že zmáčknuté tlačítko může posílat signál logická nula nebo logická jedna a konkrétní zapojení tlačítka záleží na vývojovém kitu.

8.5 Build a upload

Pro sestavení a nahrání obvodu do kitu je nutné mít nainstalovaný Intel Quartus a zadat správnou cestu k tomuto prostředí. Sestavení obvodu je možné ve chvíli, kdy jsou zapojeny všechny vstupní piny použitých modulů. V opačném případě aplikace na tuto skutečnost upozorní pomocí dialogového okna s informací, který blok nemá zapojené piny.

Pokud je obvod v pořádku je spuštěné sestavování obvodu pomocí prostředí Intel Quartus, které běží na pozadí a v samotné aplikaci je zobrazeno animované okno, které se ukončí po

skončení sestavování. Tato operace trvá v závislosti na použitém počítači minimálně 30 vteřin a na slabších sestavách může být délka této operace i v řádu jednotek minut. Po úspěšném sestavení je možné nahrát obvod do kitu. I v tomto případě je tato akce prováděna na pozadí a v aplikaci je opět zobrazena načítací obrazovka. Nahrání do kitu je kratší záležitost, která obvykle trvá okolo deseti vteřin.



Obrázek 31 – Ukázka načítací obrazovky

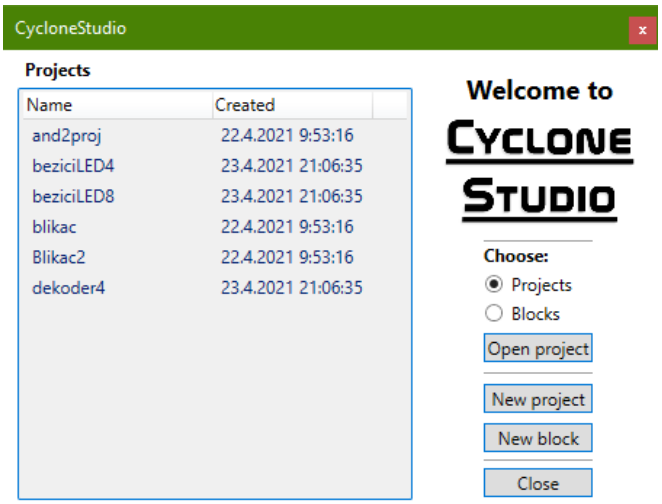
8.6 Okna aplikace

Nyní jsou popsány a ukázány některá vybraná okna použitá v rámci aplikace. Jedná se o okno, které se zobrazuje vždy po zapnutí aplikace. Dále bude popsáno okno, které slouží pro textový vstup od uživatele a jako poslední je popsáno okno pro zobrazování komponentů na fotografii vývojového kitu.

8.6.1 Vybrání projektu nebo bloku

Toto okno je výchozí okno aplikace, které se zobrazí vždy po zapnutí a jedná se o jakousi úvodní obrazovku. V levé části okna je výpis seznamu dostupných projektů nebo se po překliknutí přepínačů pod textovým polem s nápisem Choose zobrazí seznam dostupných bloků.

Při změně z projektu na blok a opačně se také mění některá pojmenování v okně. Například na obrázku č. 32 je vidět nápis Project nad seznamem projektů a nápis na tlačítku pro otevření projektu je Open project. Pokud je, ale zakliknutá možnost blocks, změní se horní nápis na Blocks a i nápis na tlačítku se změní na Open blocks. Tyto změny jsou dělány z důvodu vyšší přehlednosti a intuitivnosti aplikace. V seznamu je kromě jména projektu či bloku napsáno také datum a čas vytvoření dané položky.

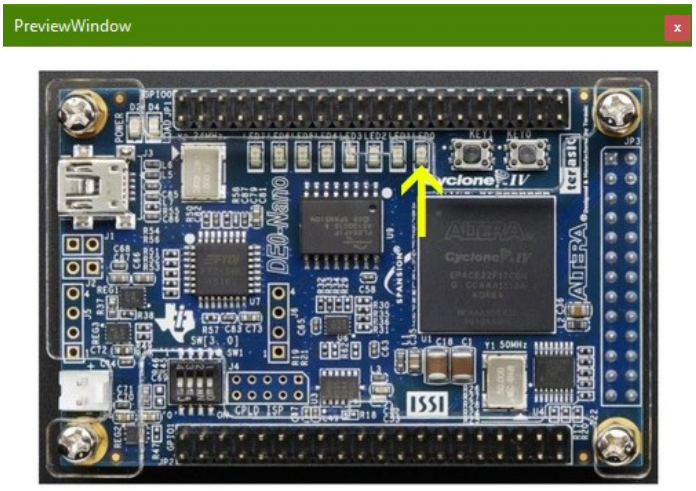


Obrázek 32 – Dialog pro výběr projektu nebo bloku

Položka se vybírá zakliknutím vybraného projektu, čímž dojde k jeho označení. Označená položka se dá libovolně změnit a pro otevření je pak nutné kliknout na tlačítko Open project/block umístěné pod přepínači. Dále je také možné z této úvodní obrazovky vytvořit nový projekt nebo nový blok opět pomocí samostatných tlačítek s nápisem New projet a New block. Poslední tlačítko Close pak slouží k ukončení celé aplikace.

8.6.2 Ukázka komponent na kitu

Pro zobrazení fotografie kitu slouží speciální okno nazvané PreviewWindow. Jedná se o užitečnou pomůcku pro začátečníky, pomocí které je usnadněné případné hledání komponent na kitu. Především je nápomocné při identifikaci vstupních a výstupních pinů na 40 pinových rozhraních, které tyto kity obsahují. V tomto okně je zobrazena fotografie vybraného kitu společně se žlutou šipkou, která ukazuje na vybranou komponentu. Toto okno je voláno zmáčknutím pravého tlačítka nad modulem se symbolem otazníku.

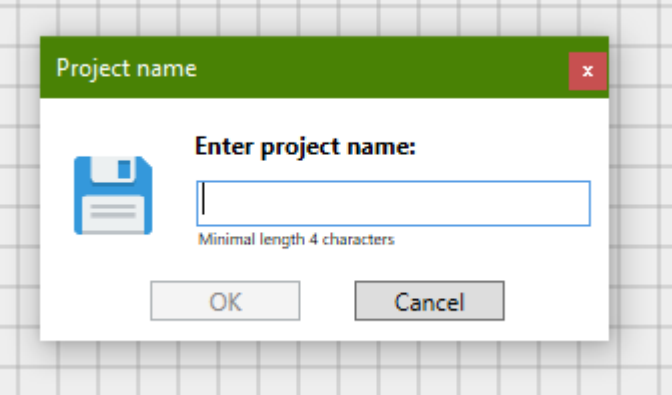


Obrázek 33 – Okno ukázky kitu

8.6.3 Input dialog

Toto okno slouží k získání textové informace od uživatele. Slouží pro pojmenovávání projektů, bloků nebo vlastních pinů. Obsluha tohoto okna je velice jednoduchá, stačí pouze napsat požadovaný text do textového vstupu okna. Je tu jediná podmínka a to, aby měl text více než čtyři znaky jinak není možné stisknout tlačítko OK a je možné pouze zavřít toto dialogové okno.

Nápisy na tomto okně jsou také přizpůsobovány, dle aktuálního využití dialogového okna. Jedná se zde o nápis v liště okna a o nápis nad textovým vstupem informující jaký text je po uživateli požadován.



Obrázek 34 – Dialog pro textový vstup

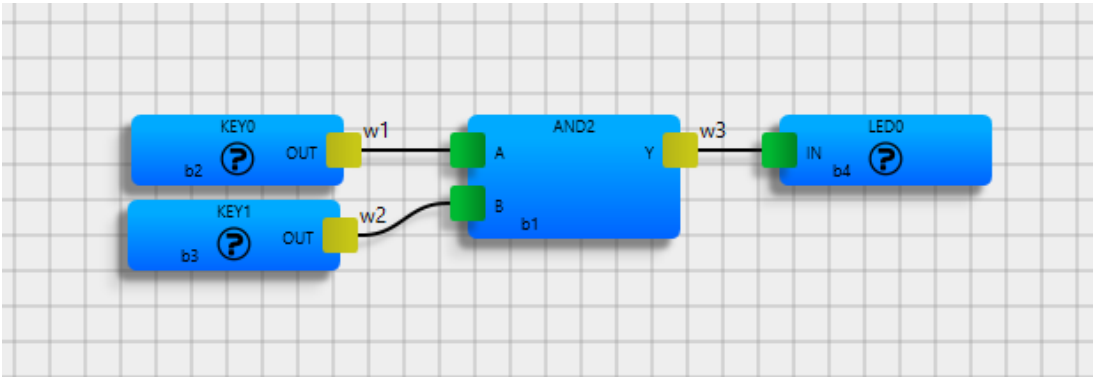
9 UKÁZKOVÉ ÚLOHY

Poslední kapitola této práce je věnována ukázkám možných zapojení obvodů v aplikaci CycloneStudio. Jsou tu tři ukázky od jednoduché až po složitější zapojení obvodů. Ukázky jsou popsány a ukázány na obrázcích. Všechny ukázky jsou také součástí výsledné aplikace.

9.1 Ukázka 1: logický AND s LED diodou

V první ukázce se jedná o projekt, který je pojmenovaný and2proj a patří mezi nejjednodušší možný obvod. Uživatel si tímto způsobem může otestovat funkci základních logických obvodů, v tomto případě pravdivostní tabulku dvouvstupého hradla AND.

Stisknuté tlačítko prezentuje logickou nulu, nestisknuté logickou jedničku. Pokud svítí LED dioda tak signalizuje logickou jedničku, pokud nesvítí tak signalizuje logickou nulu. V tomto případě bude tedy dioda svítit v okamžiku kdy není zmáčknuté žádné tlačítko a v případě, že bude zmáčknuté jedno nebo obě tlačítka, dioda svítit nebude. Na základě podobného obvodu lze takto otestovat pravdivostní tabulky různých logických hradel.



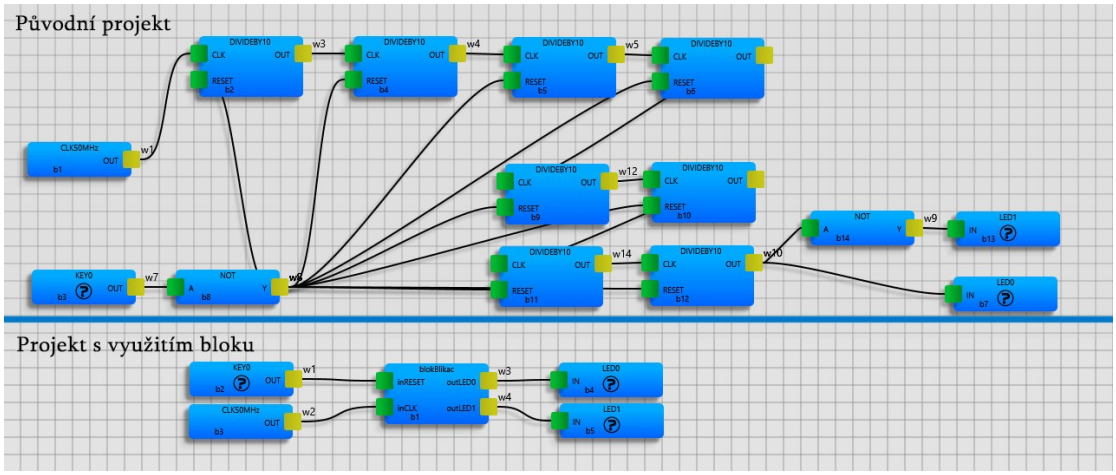
Obrázek 35 – Ukázka zapojení hradla AND

9.2 Ukázka 2: signalizace na železničním přejezdu

Druhá ukázka je uložená v projektu s názvem blikac a jedná se o trochu složitější obvod, kdy ukázka představuje světelnou signalizaci na železničním přejezdu. Zapojení obsahuje dvě LED diody, kdy v jednom okamžiku svítí pouze jedna. Toho je docíleno použitím negace pomocí hradla NOT. Délka svitu diody je dána frekvencí oscilátoru na kitu, které generuje signál a jeho hodnota se mění právě dle frekvence. Vzhledem k příliš vysoké hodinové frekvenci oscilátoru (50 MHz) je nutné tuto frekvenci snížit s využitím děliček 10 (modul DIVIDEBY10). Tyto děličky jsou zapojeny do kaskády a výsledná frekvence je pak 0,5 Hz.

Celé zapojení, lze zjednodušit s použitím uživatelského bloku, kdy celou logiku umístíme právě do tohoto jednoho bloku (blikacblok). Výsledný projekt (blikac2) tak bude jednodušší a přehlednější, viz obrázek č. 36. Tlačítko KEY0 slouží pro reset jednotlivých děliček, proto

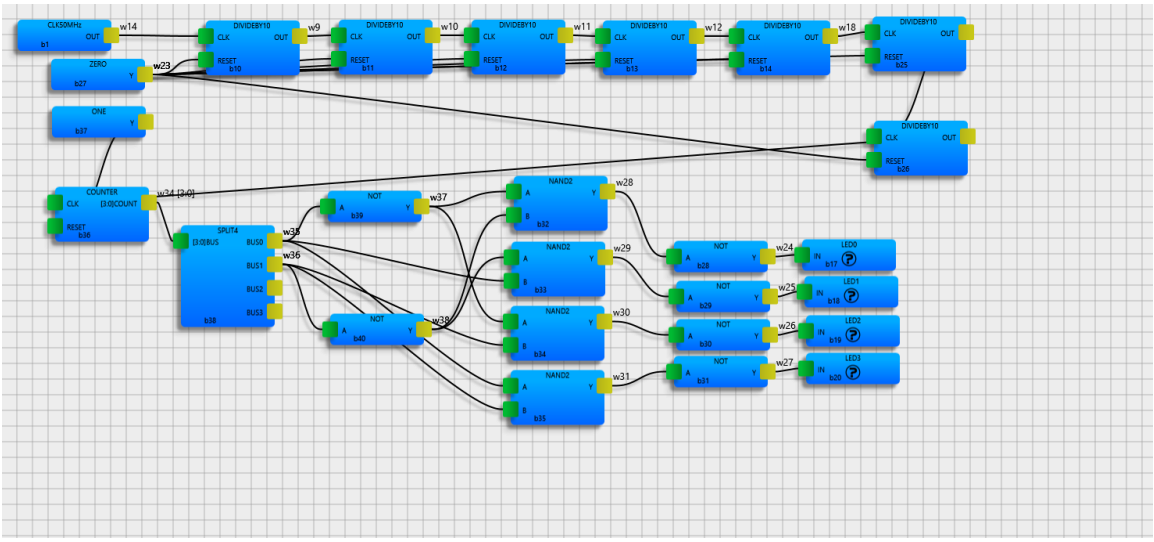
jsou na toto tlačítko připojeny všechny resetovací vstupy modulů. Jelikož děličky potřebují pro svou činnost logickou nulu, je použito hradlo NOT pro negaci stavu tlačítka.



Obrázek 36 – Ukázka světelné signalizace

9.3 Ukázka 3: běžící světlo

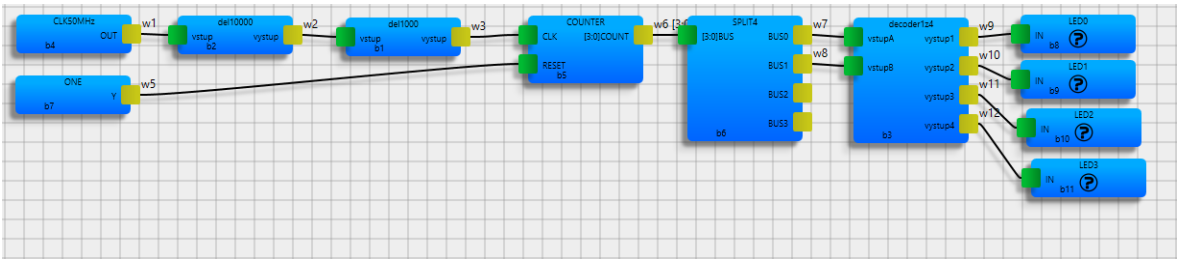
Třetí ukázka je v projektu s názvem dekoder4 a je zaměřena na vytvoření tzv. běžícího světla, kdy se postupně rozsvěcuje LED dioda v řadě LED diod, což vytváří zmiňovaný efekt běžícího světla. Projekt je navržen pro čtyři LED diody. U většiny podobných návrhů je potřeba podstatně menší hodinová frekvence, proto je vhodné vytvořit blok jako je dělička frekvence použitá i u předchozí ukázky. Frekvence oscilátoru je zde snížena na 5 Hz. K přepínaní mezi diodami pak slouží hradla NOT a NAND, která dekódují hodnoty vycházející z počítadla hodinového signálu (COUNTER). Blok Counter obsahuje čítač hodinových pulsů, který je 4bitový a nabývá hodnot od 0 do 15. Výstup tohoto čítače je definován jako sběrnice. Proto je následně použito bloku SPLIT4, kde tato sběrnice je převedena na jednotlivé samostatné vodiče. Pro využití dekodéru jsou použity nejnižší dva bity.



Obrázek 37 – Projekt dekoder4

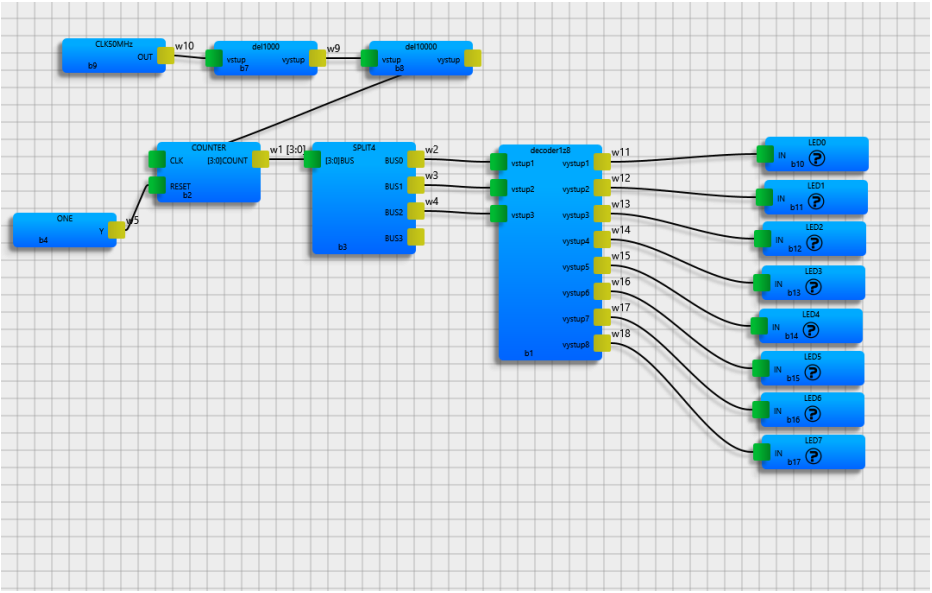
Jak je patrné z nakresleného schématu, vzhledem ke složitosti návrhu je lepší použít uživatelské bloky. Proto jsou v rámci ukázky vytvořeny dva uživatelské bloky děličky, kde první má dělící poměr 1:1000 (blok del1000) a druhá 1:10000 (blok del10000). V obou těchto blocích jsou kaskádově zapojeny děličky DIVIDEBY10. Tyto bloky pak jsou využity pro zjednodušení návrhu zapojení.

V rámci projektu beziciLED4 jsou pak tyto bloky využity a celkový návrh zapojení je následně přehlednější. V rámci dalšího zjednodušení následuje vytvoření bloku nazvaného decoder1z4, do kterého je schována logika přepínaní svitu mezi čtyřmi LED diodami.



Obrázek 38 – Projekt beziciLED4

Pro využití všech LED diod na vývojovém kitu je vytvořen blok decoder1z8, který je použit v rámci projektu beziciLED8. Výsledné schéma pak bude podstatně přehlednější, vzhledem k složitosti zapojení dekodéru pro přepínání mezi osmi LED diodami.



Obrázek 39 – Projekt beziciLED8

ZÁVĚR

Cílem této diplomové práce bylo vytvoření funkční aplikace umožňující vytvářet a propojovat logické obvody zapojení s využitím grafického rozhraní. Aplikace nese název CycloneStudio a je navržená, aby sloužila uživatelům jako pomůcka pro seznámení s logickými operátory a sekvenčními obvody. Tyto vytvořená zapojení je pak možné vyzkoušet na vývojovém kitu z řady Intel Cyclone IV.

V současné chvíli aplikace podporuje dva vývojové kity a to DE0-Nano a StormIV-E6. V době psaní práce šlo zakoupit pouze první jmenovaný a u druhého pouze jeho podobné verze. Naštěstí to nebyl problém, protože díky návrhu aplikace je možné přidat i vlastní kit.

Aplikace plně splňuje veškeré stanovené cíle, je také graficky povedená a působí moderním a hezkým dojmem. Uživatel si může vytvářet vlastní projekty a znovupoužitelné bloky. Prostředí Intel Quartus je v aplikaci používáno pouze k sestavení a nahrání projektů do vývojového kitu. Velký přínos je, že veškeré ovládání je intuitivní, přehledné, a navíc aplikace hlídá správné propojení pinů čímž zaručuje, aby byla syntaxe výsledného kódu správná a tím usnadňuje učení s logickými a sekvenčními obvody. Ve výsledku se jedná o aplikaci určenou pro úplné začátečníky, ve které je možné se začít učit základy obvodů a kitů. Aplikace navíc dokáže poradit s identifikací komponent na vývojovém kitu.

V budoucnu bych chtěl aplikaci rozšířit o simulaci, kdy by si uživatel mohl vyzkoušet funkci navrženého zapojení. Uživatel by si mohl v tomto režimu nadefinovat logické úrovně na jednotlivých vstupech a na výstupech každé komponenty by byly zobrazeny výsledné logické úrovně. Jako další rozšíření plánuji vytvoření uživatelské komponenty pro přímé vkládání Verilog kódu. Tuto komponentu by mohl využívat zkušenější uživatel pro seznámení s jazykem Verilog. Tento kód by uživatel tvořil pomocí průvodce, který by zajišťoval správnou syntaxi jazyka. Jako poslední zlepšení programu plánuji vytvořit průvodce pro přidání nového vývojového kitu do aplikace. Tento průvodce by po uživateli požadoval vyplnit formulář na základě, kterého by pak přidal nový vývojový kit do aplikace.

Pro rozšíření této aplikace plánuji vytvořit jednoduché webové stránky, kde by si mohli zájemci o programování FPGA (od firmy Intel) tuto aplikaci stáhnout. Zatím je aplikaci možné stáhnout z portálu GitHub na adrese /karliklukas/CycloneStudio. Dále bych na stránkách umístil podrobný manuál pro ovládání aplikace. Plánuji na webové stránce také vytvořit základní kurz pro programování v jazyce Verilog s odkazy na podobně tematicky zaměřené webové stránky. Osobně se domnívám, že by tato aplikace mohla sloužit na středních školách při výuce logických obvodů a vhodně by doplňovala teorii o praktickou část.

POUŽITÁ LITERATURA

[1] ŠŤASTNÝ, Jakub. *FPGA prakticky: Realizace číslicových systémů pro programovatelná hradlová pole.* Praha: BEN - technická literatura, 2010. ISBN 9788073002619.

[2] DANĚK, Martin. Programovatelná hradlová pole – FPGA. *Automa – časopis pro automatizační techniku* [online]. 2006(2) [cit. 2021-03-30]. Dostupné z: https://automa.cz/cz/casopis-clanky/programovatelna-hradlova-pole-fpga-2006_02_30930_672/

[3] Programovatelná logika I: Přehled PLD. *AbcLinux* [online]. 2012, 18. 12. 2012 [cit. 2021-03-31]. ISSN 1214-1267. Dostupné z: https://www.abclinuxu.cz/blog/digital_design/2012/12/programovatelna-logika-i-prehled-pld

[4] Jak se píše procesor. *AbcLinux* [online]. 2005, 8. 7. 2005 [cit. 2021-03-31]. ISSN 1214-1267. Dostupné z: https://www.abclinuxu.cz/clanky/programovani/jak-se-pise-procesor

[5] MALÝ, Martin. *Data, čipy, procesory: vlastní integrované obvody na koleni*. Praha: CZ.NIC, z.s.p.o., 2020. CZ.NIC. ISBN 978-80-88168-56-0.

[6] What Is a Hardware Description Language (HDL)? *All about circuits* [online]. 2020, 11. 3. 2020 [cit. 2021-03-31]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/what-is-a-hardware-description-language-hdl/

[7] Getting Started with the Verilog Hardware Description Language. *All about circuits* [online]. 2017, 29. 12. 2017 [cit. 2021-03-31]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/getting-started-with-the-verilog-hardware-description-language/

[8] Intel® FPGAs. *Intel* [online]. 2020 [cit. 2021-03-31]. Dostupné z: https://www.intel.com/content/www/us/en/products/programmable/fpga.html

[9] 7 Series: Product selection guide. *Xilinx* [online]. 2020 [cit. 2021-03-31]. Dostupné z: https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf

[10] *Xilinx* [online]. 2020 [cit. 2021-04-01]. Dostupné z: https://www.xilinx.com/

[11] *Lattice Semiconductor* [online]. 2020 [cit. 2021-04-01]. Dostupné z: https://www.latticesemi.com/

[12] Cyclone® IV FPGAS Features: 2020. *Intel* [online]. [cit. 2021-04-01]. Dostupné z: https://www.intel.com/content/www/us/en/products/programmable/fpga/cyclone-iv/features.html

[13] DE0-Nano Development and Education Board. *Terasic* [online]. 2013 [cit. 2021-04-02]. Dostupné z: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=593&PartNo=2#section

[14] Altera FPGA Board ALTERA Cyclone IV EP4CE6 FPGA Development Kit. AliExpress [online]. 2015 [cit. 2021-04-02]. Dostupné z: https://www.aliexpress.com/item/32758408263.html?spm=2114.search0104.3.2.3ef23f4dW0glk6&ws_ab_test

[15] Verilog Tutorials and Examples. *Nandland* [online]. 2015 [cit. 2021-04-04]. Dostupné z: https://www.nandland.com/verilog/tutorials/index.html

[16] Google Trends. *Google* [online]. 2020 [cit. 2021-04-05]. Dostupné z: https://trends.google.com/trends/explore?q=%2Fm%2F0bwsp,%2Fm%2F0h3vb

[17] Icestudio: A real gamechanger in the world of Open Source FPGAs [online]. 2020 [cit. 2021-04-10]. Dostupné z: https://icestudio.io/

[18] Download Center for FPGAs: Quartus Prime Lite Edition. *Intel* [online]. 2021 [cit. 2021-04-10]. Dostupné z: https://fpgasoftware.intel.com/20.1.1/?edition=lite&platform=windows

[19] Introduction to Visual Studio. *GeeksForGeeks* [online]. 2019, 4.8.2019 [cit. 2021-04-07]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-visual-studio/

[20] Visual Studio 2019. *Microsoft* [online]. 2020 [cit. 2021-04-07]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

[21] C# - Overview. Tutorialspoint [online]. 2017 [cit. 2021-04-07]. Dostupné z: https://www.tutorialspoint.com/csharp/csharp_overview.htm

[22] C# and its Features. *C# Corner* [online]. 2017, 15.3.2017 [cit. 2021-04-07]. Dostupné z: https://www.c-sharpcorner.com/article/C-Sharp-and-its-features/

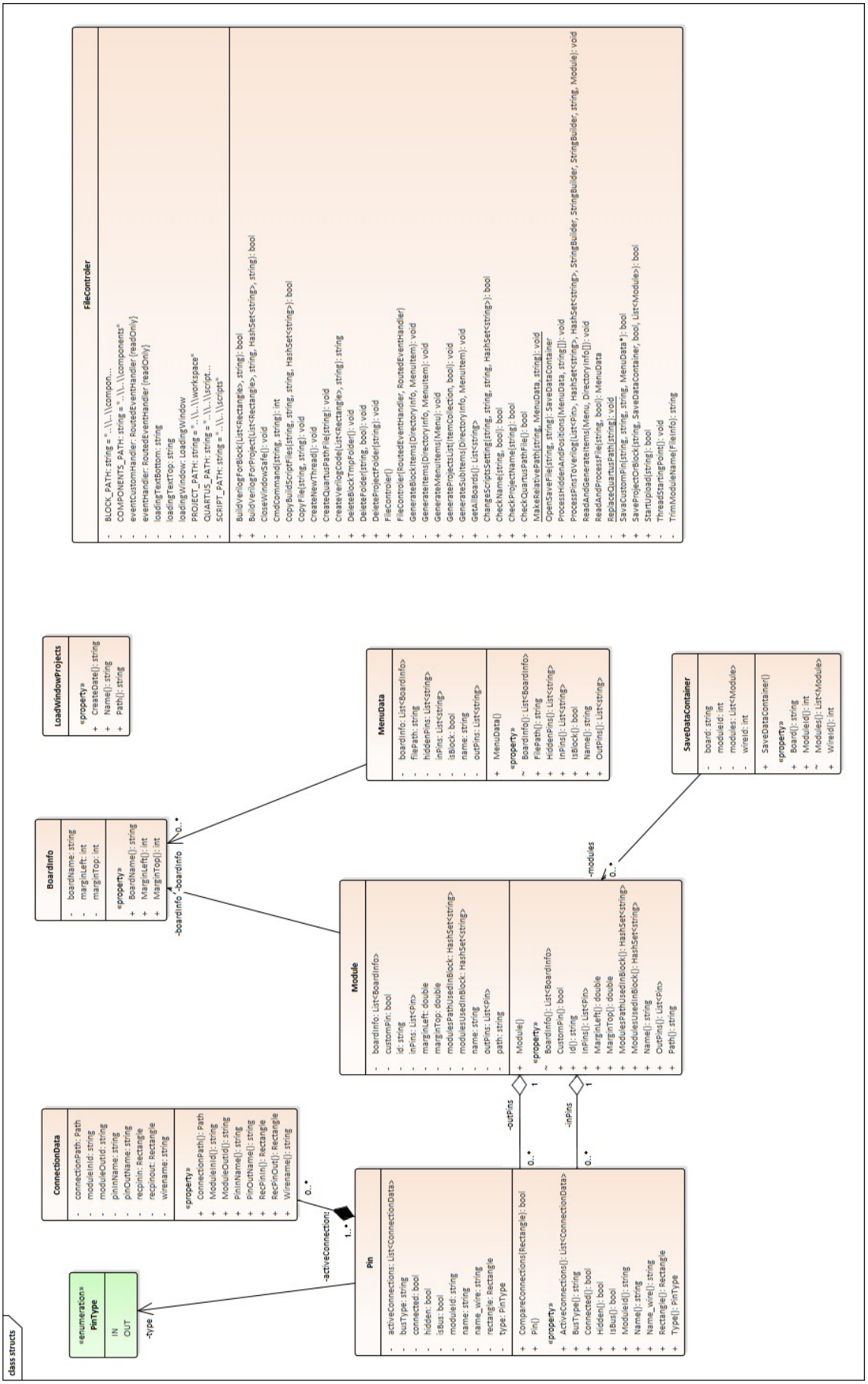
[23] WPF - Overview. *Tutorialspoint* [online]. 2018 [cit. 2021-04-07]. Dostupné z: https://www.tutorialspoint.com/wpf/wpf_overview.htm

[24] Úvod do WPF. *ITnetwork* [online]. 2017 [cit. 2021-04-07]. Dostupné z: https://www.itnetwork.cz/csharp/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace

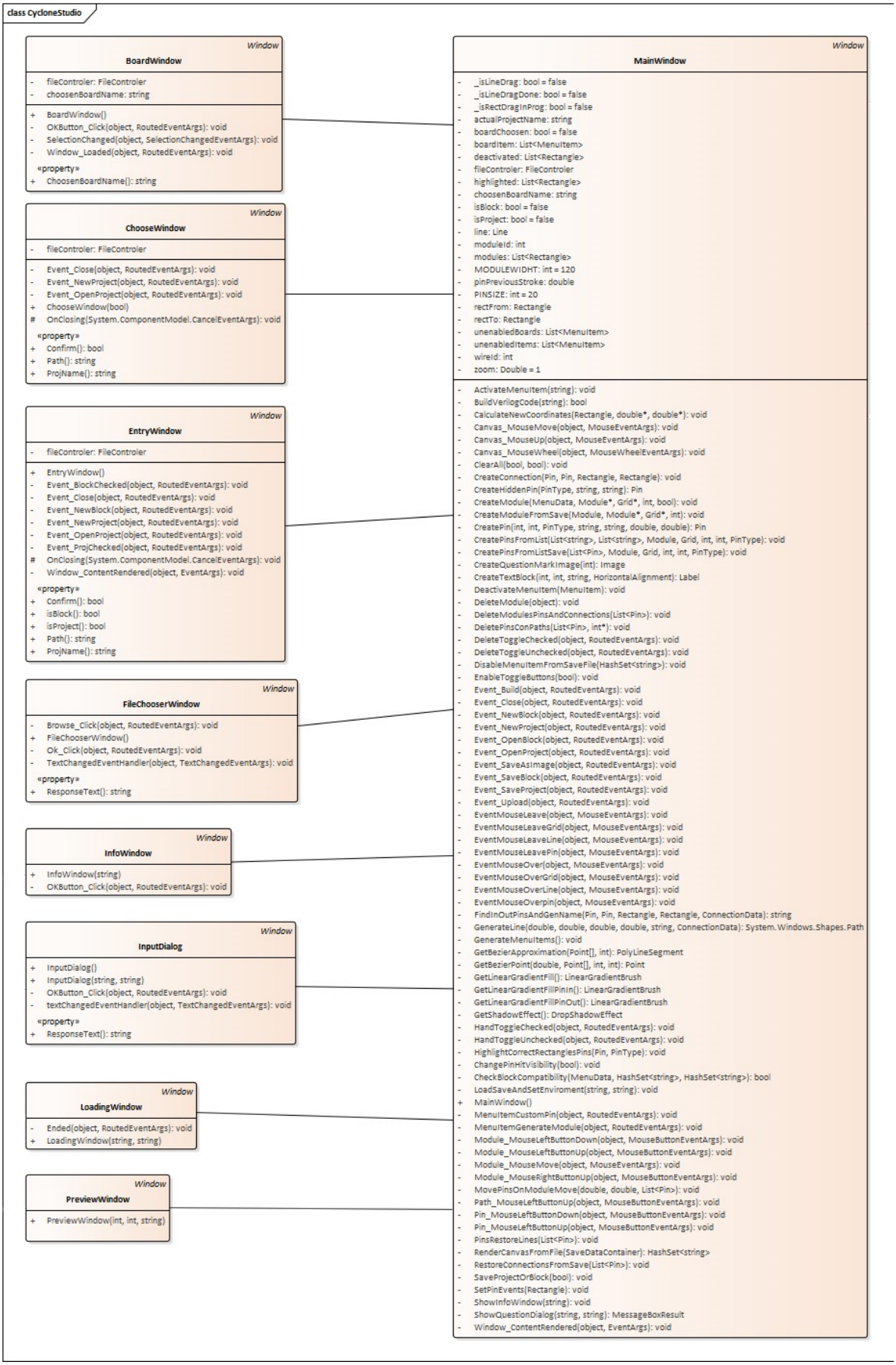
[25] PEDEMKAR, Priya. Winforms vs WPF. EDUCBA [online]. 2018 [cit. 2021-04-07]. Dostupné z: https://www.educba.com/winforms-vs-wpf/

PŘÍLOHY

PŘÍLOHA A – DIAGRAM DATOVÝCH TŘÍD



PŘÍLOHA B – DIAGRAM GRAFICKÝCH TŘÍD



PŘÍLOHA C – ZDROJOVÉ KÓDY A DOKUMENTY

K práci jsou přiloženy veškeré zdrojové kódy aplikace spolu se samospustitelným exe souborem a základními Verilog komponentami.