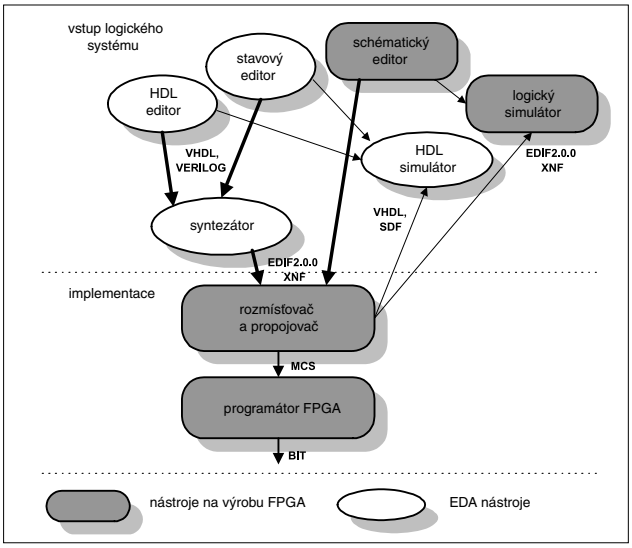
Metody návrhu systémů na bázi FPGA

Úvod

Ve třetím dílu série článků o programovatelných logických obvodech bude nastíněna metodika návrhu systémů realizovaných právě pomocí FPGA. Současně budou zmíněny i příslušné návrhové nástroje (software CAD), které se v současné době pro tento návrh používají a bez kterých si již dnes tuto oblast lze jen těžko představit. Pro návrh složitých zákaznických obvodů je velmi důležitá problematika tzv. maker IP (Intellectual Property), která je poměrně nová, pro mnohé neznámá a představuje významnou etapu v průběhu vývoje obvodu. Na závěr tohoto dílu byla zařazena problematika programování a konfigurace obvodů FPGA.



Obr. 1 Rozdělení CAD nástrojů

Všechny součástky bývají označovány jako zákaznické integrované obvody – ASIC (Application Specific Integrated Circuits), jejichž vývoj a výroba se provádí pro jednoho konkrétního zákazníka a nelze je koupit v běžné obchodní síti. Způsob realizace hardware může být dvojí. Buď můžeme použít programovatelný obvod – tedy součástku předem vyrobenou (např. CPLD nebo FPGA – viz celý tento seriál článků) nebo nechat vyrobit součástku pomocí zákaznických masek a složitých technologických postupů přímo na křemíku. Postupem času se termínem obvod ASIC začaly označovat pouze ty součástky, které se takto „programují“ během výrobního procesu maskami, tj. klasické obvody ASIC, tradiční obvody ASIC, maskami programované zákaznické obvody apod. S touto terminologií se setkáte nejen v tomto článku, ale i ve všech současných materiálech týkajících se zákaznických integrovaných obvodů a návrhových prostředků určených pro jejich vývoj.

Vývoj metod návrhu systémů s použitím FPGA

Stejně jako samotné součástky FPGA, tak i příslušné nástroje a postup návrhu procházejí prudkým vývojem. V minulosti při relativně malé složitosti návrhů, které měly být realizovány obvody FPGA, pro zadání logického systému zcela vyhovovalo použití schématu (tzv. návrhový vstup). Kromě toho ještě existovala možnost popisu zadání pomocí jednoduchých jazyků, např. ABEL. Pochopitelně i návrhové prostředky byly na tyto metody orientované a byly dodávány přímo výrobci součástek FPGA společně s nástroji pro implementaci. Na *obr. 1* je naznačeno jak použití, tak i rozdělení prostředků CAD.

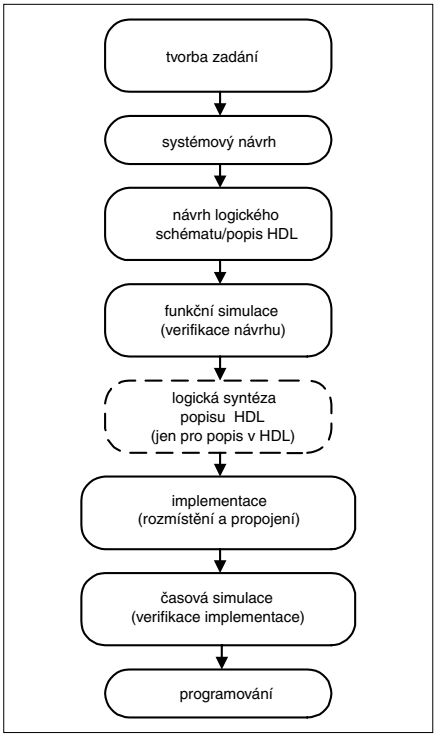
Protože oblast návrhu obvodů ASIC čelí podobným problémům (vždy s předstihem), začal návrh obvodů na bázi FPGA přejímat jak metodiku, tak postupně i návrhové nástroje. Prvním případem byla postupná náhrada již nedostačujícího schématického vstupu normalizovanými jazyky vyvinutými pro popis logických systémů (tzv. HDL – Hardware Description Language). V Evropě se rozšířil zejména jazyk VHDL, mimo Evropu získává značnou oblibu i Verilog. S tímto pochopitelně souvisí i nástroje pro tvorbu, simulaci i syntézu těchto jazyků. Především simulátory tak společně slouží jak pro návrh ASIC, tak pro FPGA. Příkladem může být ModelSim (nástroj firmy Model Technology – Mentor Graphics) nebo VSS (Synopsys). V případě nástrojů pro syntézu ovšem probíhá vývoj opačný. Syntezátory pro ASIC nelze pro FPGA efektivně použít, protože algoritmus syntézy pro součástky FPGA musí být přímo optimalizován pro jejich vnitřní architekturu. Tento problém řeší firma Mentor Graphics tím, že ve svém nástroji Leonardo Spectrum sdružuje optimalizované algoritmy syntézy jak pro FPGA, tak i pro ASIC. Praktické zkušenosti návrhářů s tímto produktem ukazují na velmi dobře zvládnutý proces. Vhodnému řešení kompletního návrhového prostředí se bude věnovat jedno z dalších pokračování seriálu o programovatelných obvodech. Příkladem bude FPGA Advantage (nové označení programového balíku Packaged Power).

Dalším velmi výrazným směrem vývoje je tendence používat spíše nástroje EDA (Electronic Design Automation) jiných firem, což představuje ústup od jednoúčelových softwarových prostředků dodávaných přímo výrobci součástek FPGA. Např. firma Xilinx klade stále větší důraz na program Alliance, jemuž prostředky pro návrh, verifikaci a syntézu dodávají Mentor Graphics, Synopsys, Cadence, případně další firmy. Implementační nástroje ovšem pochopitelně zůstávají doménou firem Xilinx, Altera, Actel atd.

Modely metod návrhu systémů

Tradiční postup při návrhu systémů na bázi FPGA (design flow), který je uveden na *obr. 2*, se často nazývá „vodopád“ (waterfall). V tomto modelu návrh postupuje po krocích z jedné fáze do druhé a již se nevrací zpět.

Pro návrhy složitých systémů (100 K hradel a více), kdy na projektu pracují desítky návrhářů rozčleněných do skupin, se jednoznačně jeví jako vhodnější model „spirála“ (spiral) na *obr. 3*. Většina fází se opakuje několikrát s postupně se upřesňujícím a zlepšujícím řešením úvodního zadání, jedná se tedy o iterativní proces. Tento model byl pochopitelně nejdříve nasazen v návrzích ASIC. V současné době však již i technologie obvodů FPGA umožňuje realizovat natolik složité návrhy, že je pro ně rovněž nutné použít model „spirála“.



Obr. 2 Postup návrhu FPGA – metoda „vodopád“

Etapy návrhu

Tvorba specifikací je první a velmi důležitá část návrhového procesu. Provádí se obvykle opakovaně a postupně se upřesňuje tak dlouho, až je možné zahájit tvorbu kódu HDL na úrovni RTL (Register and Transfer Level). Správně vytvořená specifikace usnadňuje vlastní návrh a stává se i základem konečné dokumentace hotového obvodu.

Je též třeba zmínit fakt, že existují dva základní způsoby tvorby specifikací:

– *Formální specifikace* používá některého ze specifických jazyků vyvinutých pro

tyto účely (např. VSPEC) nebo běžného psaného textu.V současné době ještě není rozšířená, ale v dlouhodobém horizontu slibuje nesporné výhody.

– *Spustitelná specifikace* představuje abstraktní model navrhovaného obvodu. Pro vyšší úroveň specifikace se typicky používají jazyky C, C++ a SDL. Popis HDL je pak vhodný pro nižší úroveň.

Specifikace musí postihnout funkci navrhovaného systému, časování (zejména z hlediska vstupů a výstupů obvodu), rozhraní se SW, odhad velikosti návrhu, pouzdra a rozložení vývodů.

Teprve po vypracování specifikace je možné zahájit návrh systému. Jestliže je již součástí specifikace i spustitelný model, lze rovnou přistoupit k jeho odlaďování a zpřesňování. Důležitým okamžikem je následné rozdělení řešení zadání na část softwarovou (SW) a hardwarovou (HW), obě části se pak řeší samostatně. Je samozřejmě třeba ověřit zároveň jejich součinnost, proto se provádí i souběžná simulace HW/SW (HW/SW co-simulation).

Následuje návrh logického schématu nebo v současné době spíše popis HDL na úrovni RTL, která je již syntetizovatelná. Zde je důležité respektovat zásady, které zabezpečí, že návrh HDL je snadno syntetizovatelný, čitelný, modifikovatelný a vhodný pro opakované použití.

Je třeba podotknout, že je velmi obtížné definovat takovou sadu zásad či pravidel, která by plně postihovala problematiku tvorby kódu VHDL, neexistuje dosud žádná norma.

Nicméně za základ takovéto normy lze považovat ve [3] kapitolu 5 – RTL Coding Guidelines. Autoři zde rozdělili zásady návrhu do několika sekcí, jak je patrno z následujícího odstavce.Vzhledem k rozsahu tohoto článku jsou uvedeny pouze hlavní pravidla.

Základní zásady tvorby HDL kódu:

– konvence pojmenovávání v návrhových prvků (např. bloků, entit, signálů, procedur, funkcí apod.),
– konvence pojmenovávání pro podporu VITAL (standard pro modelování knihovních prvků různých technologií),
– pravidla pro tvorbu záhlaví a komentářů ve zdrojových souborech VHDL.

Zásady pro použití struktur VHDL:

– tvorba kódu s ohledem na opakované použití (portability),
– nutnost používat pouze takové typy signálů, které jsou definované normami IEEE,
– nutnost používat technologicky nezávislé knihovny VHDL stavebních prvků.

Zásady pro hodinové a resetovací signály:

– minimalizovat počet hodinových domén, každá musí být zdokumentována (zejména externí časové požadavky, případný PLL, apod.),
– pokud to není nezbytně nutné, vyhnout se „hradlování“ hodinových signálů,
– obvod by měl mít vlastní (asynchronní) vývod power-on-reset, samostatný HW a SW reset.

Návrh pro syntézu:

– doporučuje se používat signály místo proměnných,
– nepoužívat, pokud je to možné, konstrukce VHDL, která popisuje paměťový element typu latch (asynchronní klopný obvod),
– popis konečných automatů má být rozložen do dvou procesů.

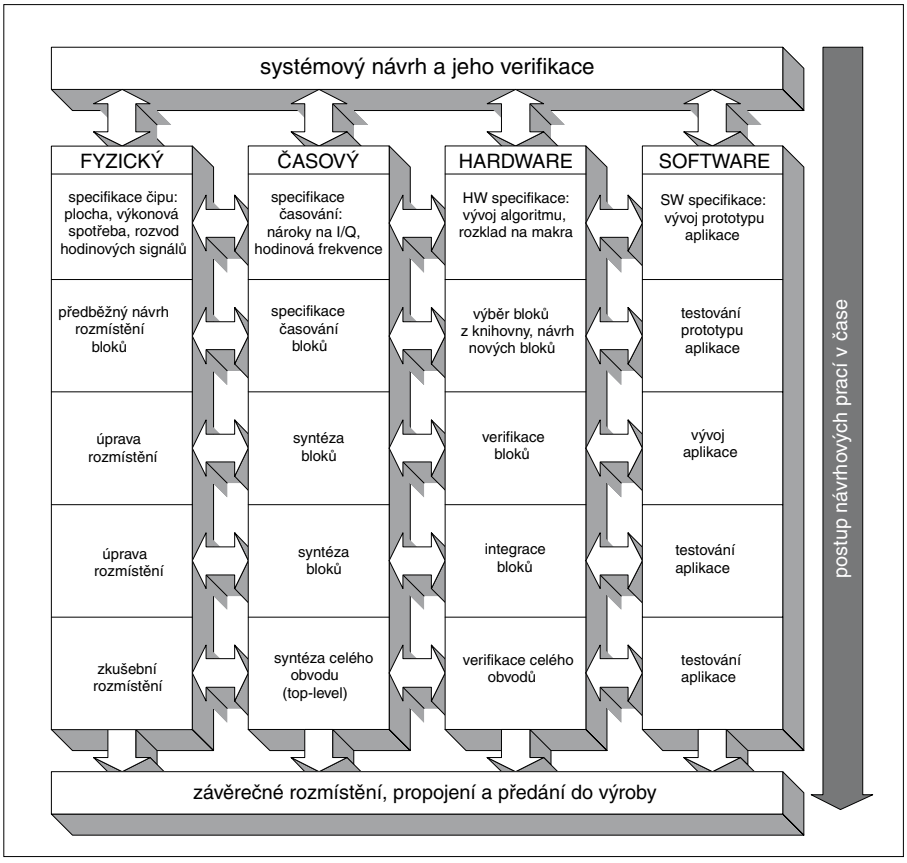
Dělení návrhu pro syntézu:

– nutné pro urychlení běhu syntézy, získání lepších výsledků,
– všechny výstupní signály z bloku mají být registrovány,
– provázané logické struktury mají být v jediném bloku,
– struktury s různými funkcemi je třeba rozdělit do různých bloků,
– neumisťovat tzv. „glue logic“ na nejvyšší úrovni návrhu.

Návrh s pamětmi:

– adresové a datové registry a logika WE mají být odděleny,
– řídicí logika paměti pak může pracovat jak s asynchronním, tak synchronním typem paměti.

V případě návrhu velmi složitého logického systému se v podstatě nelze obejít bez použití tzv. maker IP. Jedná se vlastně o předem navržené bloky, které je možné zaintegrovat do celého logického systému, a velmi výrazně tak zkrátit dobu návrhu.



Obr. 3 Postup návrhu FPGA – metoda „spirála“

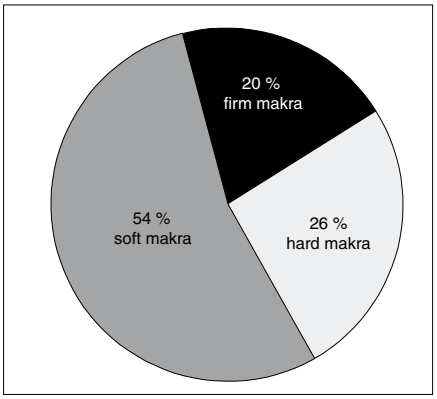
Další etapou návrhu je ověření navrženého systému pomocí simulátoru. Volba metody verifikace opět závisí na složitosti obvodu. U malých systémů, u nichž jako vstup často slouží logické schéma, lze většinou vystačit s kontrolou grafických průběhů simulátorů. Pokud je ale návrh proveden pomocí popisu HDL (což je u velkých návrhů nutností), používá se metoda verifikace pomocí tzv. „test bench“. Test bench v podstatě umožňuje emulovat podmínky, ve kterých bude navrhovaný obvod pracovat. Protože vstupy a výstupy pro test bench jsou pouze v textové podobě, lze snadno provést porovnání očekávaných výsledků verifikace se skutečnými. Takto je možné vytvářet sadu testů a spouštět je v dávkách pomocí tzv. skriptů.

Návrhy HDL je po verifikaci nutno syntetizovat. Syntéza je vlastně konverze úrovně popisu RTL na úroveň hradel zvolené technologie. Syntezátory FPGA navíc musí respektovat architekturu čipu a základních stavebních bloků. Dříve bylo možno tuto úlohu pomocí v té době dostupných programových prostředků špatně zvládnout. V současné době jsou však výsledky natolik uspokojivé, že již většinou není nutný ruční zásah.

Po syntéze je třeba návrh implementovat do zvolené součástky FPGA příslušné technologie. Nástroje, které provádějí mapování, rozmísťování a propojování vstupního „netlistu“, jsou stále doménou výrobců součástek FPGA. Dosahují takových kvalit, že „ruční“ zásah např. ve fázi propojování je zcela výjimečný. Výsledkem implementačních nástrojů je pak již datový soubor, který slouží k naprogramování obvodu nebo konfigurační paměti.

Jelikož během funkční verifikace nebyly časové poměry na čipu zohledněny, je nyní nutné provést tzv. časovou verifikaci. Informace o reálném zpoždění hradel a propojo-

vacích vodičů jsou uloženy v tzv. souboru SDF, který vzniká během implementace. Výsledky časové verifikace návrhář porovnává s funkční verifikací. Často se ovšem stává, že výsledkem porovnání výsledků obou simulací není úplná shoda. Je to způsobené tím, že porovnání probíhá na úrovni výstupních vektorů z obvodu, které jsou obvykle generovány s periodou nejrychlejšího hodinového signálu. A protože zpoždění některých datových signálů může být i větší, než je tato perioda, nelze dosáhnout absolutní shody. Přesto však platí, že pokud je funkce obvodu před a po implementaci stejná, lze výsledek porovnání prohlásit za úspěšný. K odstranění těchto potíží ve vyhodnocování je třeba posunout porovnávání výsledků z nejnižší úrovně (tj. vektorové) na úroveň vyšší. Např. u telekomunikačních obvodů je běžné porovnávání přijatých/vyslaných dat ve formě znaků.



Obr. 4 Typy IP maker

Všechny tyto metody vyhodnocování verifikací dokáže realizovat vhodně navržený test bench.

Makra a duševní vlastnictví

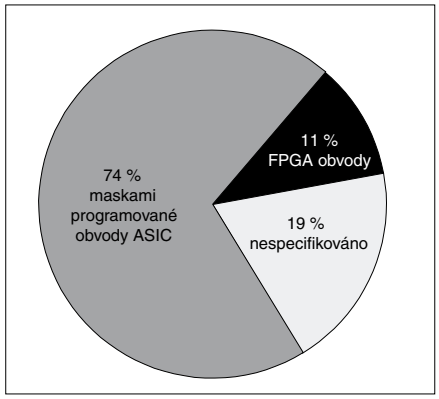
Velkou výhodou použití FPGA je kratší doba návrhu systému v porovnání s tradičními obvody ASIC. Ovšem s růstem složitosti obvodů FPGA roste i složitost návrhů, které je možno na FPGA implementovat. Současné moderní rodiny FPGA, jako např. Virtex firmy Xilinx nebo Apex firmy Altera, zvládnou integraci systémů o velikostech až 4 milióny hradel. I když je takové složitosti možno dosáhnout pouze v případě, že podstatnou část obvodu zabírá paměť, lze konstatovat, že současné součástky FPGA umožňují realizovat obvody integrující celé systémy na jediném čipu (SoC – System on a Chip). To znamená, že např. procesor včetně periférií a paměti může být implementován v jednom obvodu FPGA.

V případě návrhu obvodů takových složitostí již nelze vystačit s tradičním návrhovým procesem popsaným dříve. Jedinou cestou je „složit“ obvod z již hotových stavebních bloků, které byly navrženy za účelem opakovaného použití – bývají označovány různými termíny jako např. makra, makra IP, makrobloky, jádra, cores, IP cores, reusable cores, atp. Vždy to však znamená jediné – části obvodů se přímo nenavrhují, ale používají se již hotové a pouze se spojují dohromady. Jádrem může být například aritmetická jednotka (ALU), sériové rozhraní nebo i celý 32bitový procesor či rozhraní PCI. Zdrojem těchto jader může být i vlastní knihovna dodavatele maker IP. Makra se ovšem během krátké doby stala předmětem čilého obchodování – v mnohých případech se totiž vzhledem k časové náročnosti nevyplatí určité makro vyvíjet vlastními prostředky, takže je vhodnější je raději koupit včetně veškeré dokumentace. Oba hlavní producenti FPGA mají vlastní knihovny maker, která je možné zakoupit – Xilinx LogiCore a AllianceCore, Altera Megafunction Partners Program. Na druhé straně ale existují i firmy, které produkují makra technologicky nezávislá – např. Mentor Graphics Inventra. Na trhu jsou tak dnes k dispozici ekvivalenty většiny běžných standardních součástek a nejpoužívanějších procesorů a rozhraní.

Velmi oblíbené je u různých firem z důvodu odlišení jejich vlastních produktů označování maker spojením slova CORE s jiným slovem nebo zkratkou – např. LogiCore, ASICore, INIcore apod. – ve většině případů se z těchto spojení stávají chráněné obchodní značky. Existuje již několik návrhářských firem, které jsou specializovány pouze na návrh takovýchto makrobloků. Pro ně se vžívá anglické označení IP provider. Trh s bloky IP ve světě teprve postupně vzniká. Jedná se ovšem o trh velmi rychle rostoucí – zvláště v případě jader FPGA. Současně vzniká i další typ firem, které ve svých katalozích nabízejí produkty dodavatelů maker IP. Např. firma Design & Reuse ve svém katalogu nabízí přibližně 5 000 bloků IP od zhruba 80 dodavatelů. *Obr. 4 a 5* prezentují údaje právě v katalogu této firmy a velmi dobře odrážejí situaci na celosvětovém trhu s bloky IP.

Pokud využití jader někomu připomíná návrh desky ze součástek, má pravdu. Rozdíl je ovšem v tom, že vše se neodehrává na desce plošných spojů, ale na čipu a jádro není fyzickým produktem. Jedná se o virtuální produkt – informaci nebo, chcete-li, duševní vlastnictví (Intellectual Property – IP). Příslušné makro je možné získat například prostřednictvím Internetu. Samozřejmostí je vybavení každého jádra simulačním modelem a detailní dokumentací, jinak by jej nebylo možné použít.

Nejčastěji se makra rozdělují podle toho, v jaké formě se vyskytují, což v podstatě určuje okamžik, kdy vstupují do návrhového procesu.



Obr. 5 Podíl IP maker v jednotlivých technologiích

Soft makra – v tomto případě je dodáván popis jádra v jazycích VHDL nebo Verilog. Je obvykle technologicky nezávislý a teoreticky může být syntetizován do libovolné technologie.

Firm makra – v případě „firm“ jádra je dodáván „netlist“ v dané technologii. V této podobě se makra pro FPGA vyskytují nejčastěji.

Hard makra – v případě klasických obvodů ASIC je výsledkem návrhu hotová topografie čipu (layout), např. ve formátu GDS II. Jádra pro FPGA se zpravidla do této kategorie neřadí, i když do ní lze začlenit případy, ve kterých jsou dodávány vnitřní soubory pro rozmisťovací a propojovací program.

Na *obr. 4* je vyznačeno procentuální zastoupení jednotlivých typů maker, v jakém se dnes vyskytují na trhu. *Obr. 5* ukazuje na jednoznačnou převahu makrobloků pro klasické zákaznické obvody.

varianty testovacích desek pro obvody XILINX najdete na http://www.asicentrum.cz/ASICs/Aplikace/Texty/xilpcb.htm

Obr. 6 Příklad testovacího přípravku pro programovatelné součástky

Jaký typ makra je nejvhodnější?

Na tuto otázku neexistuje jednoduchá odpověď. Soft jádra umožňují relativně největší míru flexibility, při jejich použití je však nutné projít celým návrhovým procesem. V případě FPGA je také vhodné zjistit, zda-li dané jádro bylo do FPGA implementováno. Teoreticky je sice možné provést syntézu jádra navrženého pro tradiční technologii ASIC do FPGA, dosažená rychlost a velikost obvodu však nemusí být optimální. Je známo, že pro dosažení vysoké rychlosti v FPGA je

třeba vynaložit výrazně více úsilí a návrh provádět se znalostí cílové technologie. Hranice, od které je při návrhu třeba brát v úvahu cílovou architekturu FPGA, je dána požadovanou frekvencí. Ta se liší pro jednotlivé rodiny FPGA a neustále se posouvá výše. Např. hodinová frekvence 50 MHz je pro obvody řady Xilinx Spartan téměř na hranici možností – návrh takového obvodu vyžaduje již hodně zkušeností a pečlivosti. Oproti tomu stejný návrh lze v řadě Virtex realizovat poměrně snadno. V této rodině je ale možné realizovat i obvody s hodinovou frekvencí přesahující 100 MHz.

„Firm jádra“ a „Hard jádra“ jsou optimalizována pro danou technologii FPGA a hodinovou frekvenci. Výše zmíněné problémy jsou tedy již vyřešeny. Nevýhodou ovšem je, že návrhový proces s nimi musí zacházet jako s černými skříňkami a je obtížné v nich cokoli měnit (u hard maker je to obvykle nemožné). Protože kvalita rozmístění výrazně ovlivňuje rychlost obvodu, dodávají se s „netlistem“, pomocné soubory s direktivami pro rozmísťovací a propojovací program.

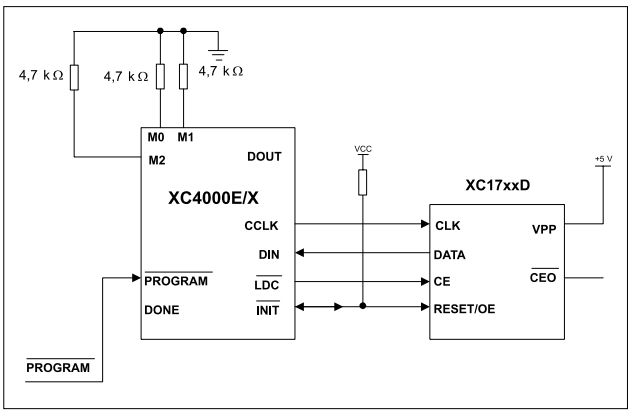
Aby bylo již navržené a verifikované makro znovu použitelné, musí návrh splňovat určitá kritéria jak po stránce technické, tak dokumentační. V tomto směru probíhá celosvětový standardizační proces, který je zastřešen organizací VSIA (Virtual Socket Initiative Association). Pro přípravu na tento trend je již nyní vhodné navrhovat obvody tak, aby jejich části byly použitelné i v dalších systémech. Očekává se, že využití maker je jedinou možností, jak v budoucnu v rozumném čase navrhovat složité obvody v klasických technologiích, případně i FPGA. A čas je v elektronickém průmyslu bezesporu jeden z nejdůležitějších faktorů.

Programování obvodů FPGA

Po ukončení návrhu systému na bázi programovatelného obvodu a simulaci jeho funkce nastává etapa jeho ověření v reálném zapojení nebo alespoň v některém speciálním testovacím přípravku (např. *obr. 6*). V každém případě je nutné pomocí dat získaných během návrhu naprogramovat nebo nakonfigurovat konkrétní programovatelný obvod.

Způsob programování (konfigurace) obvodu závisí na zvoleném typu obvodu a jeho architektuře. V současné době nejvíce používané režimy programování a konfigurace jsou následující:

– přímé programování obvodu v univerzálním nebo jednoúčelovém programátoru,



Obr. 7 MASTER SERIAL – mód konfigurace obvodu XILINX řady XC4000

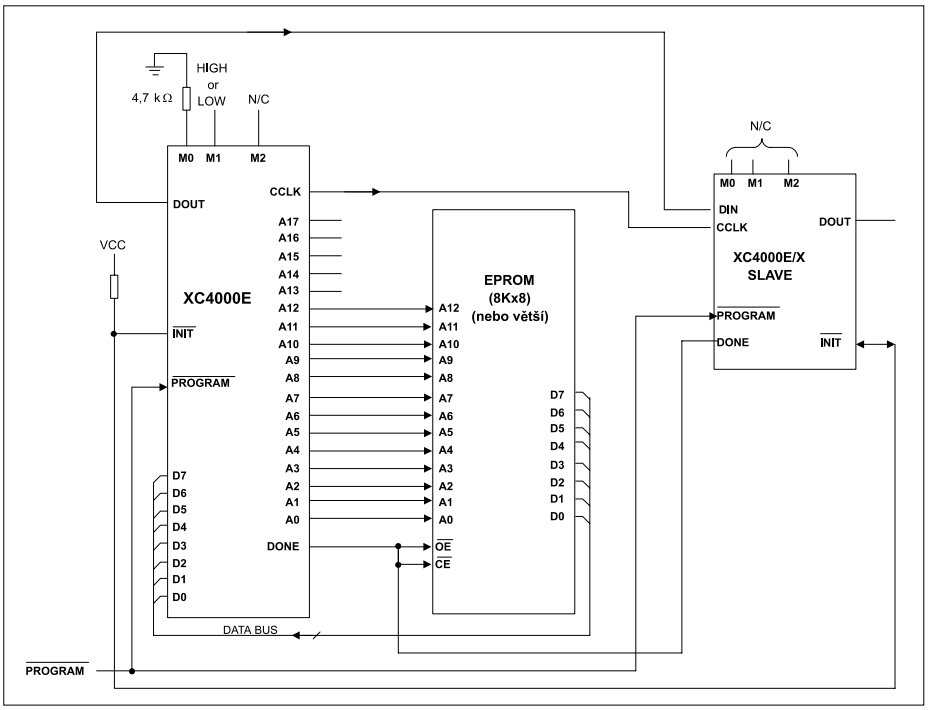
– programování obvodů přímo v systému (ISP – In System Programming),
– konfigurace obvodu z externí konfigurační paměti.

Přímé programování obvodů je velmi jednoduché, programovací soubor je finálním produktem návrhového systému. Tento režim se nejčastěji používá pro jednodušší typy programovatelných obvodů. Jeho nevýhodou je nutnost použití speciálních patic pro různá pouzdra programovatelných obvodů, dostupnost softwarového vybavení pro konkrétní typ programátoru a v neposlední řadě obtížná možnost případného přeprogramování obvodu po jeho předchozím zapájení do desky plošných spojů. Navíc nelze vyloučit poškození kontaktů vícevývodových pouzder při manipulaci během jejich programování.

Druhá často používaná možnost je programování již zapájených obvodů na desce plošných spojů (ISP). Tato varianta byla nejdříve použita u programovatelných obvodů firmy Lattice a později se začala používat i u obvodů dalších firem. Jedná se nejčastěji o sériový přenos programovacích dat z prostředí návrhového systému na počítači do programovaného obvodu. Využívá se dnes hlavně u programovatelných obvodů typu CPLD.

Většina velkých programovatelných obvodů vyráběných na bázi paměťové technologie ESRAM využívá třetí výše uvedenou možnost – konfiguraci z externí paměti. Každá skupina obvodů od různých výrobců má svoje doporučené konfigurační zapojení, ale princip je vždy téměř totožný. Programovatelný obvod je předepsaným způsobem propojen s konfigurační pamětí, která je předtím v běžném univerzálním programátoru naprogramována pomocí souboru, který je opět výsledným produktem návrhového systému. Po zapnutí napájecího napětí se data automaticky přenesou z konfigurační paměti do vnitřní paměti obvodu FPGA, která zajišťuje správnou konfiguraci logických komponentů uvnitř obvodu a jejich vzájemné propojení.

Možností propojení obvodu FPGA a konfigurační paměti je vždy několik, dělí se buď v závislosti na tom, jestli si hodinový kmitočet pro vyčítání dat z konfigurační paměti generuje přímo obvod FPGA či nikoliv nebo jestli je jako konfigurační paměť použita paměť sériová nebo paralelní. Paměti se mohou používat PROM, tak i EPROM či EEPROM. Většinou se reprogramovatelné konfigurační paměti používají ve fázi vývoje, kde často dochází ke změnám funkce obvodu. Po odladění zapojení je konfigurační soubor naprogramován do paměti typu OTP (One Time Programming).



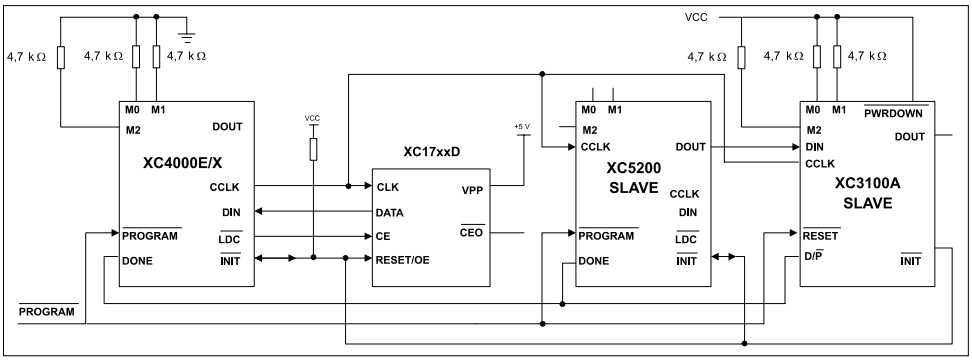
Obr. 8 MASTER PARALLEL mód konfigurace obvodu XILINX

Jako příklad konfiguračního zapojení obvodu FPGA znázorňuje *obr. 7* schéma módu *MASTER SERIAL* konfigurace obvodu Xilinx řady XC4000. Tento režim využívá možnosti generace hodinového signálu obvodem FPGA a jako konfigurační paměť je použita sériová paměť typu XC17xx.

Dalším používaným konfiguračním režimem je konfigurace z externí paralelní paměti, nejčastěji typu EPROM nebo EEPROM. Na *obr. 8.* je opět zobrazena možnost konfigurace více obvodů FPGA najednou s tím, že další konfigurované obvody v řetězci jsou

již nastaveny do konfiguračního módu *SLAVE*, neboli hodinová frekvence konfiguračních dat tohoto obvodu je generována prvním obvodem v řetězci (výstup *CCLK*).

Do konfigurační paměti je možné uložit konfigurační data více programovatelných obvodů FPGA a z jedné paměti je pak možné konfigurovat několik obvodů, a to dokonce i různých typů a velikostí (samozřejmě od jednoho výrobce). Na *obr. 9*. je znázorněna současná konfigurace obvodu FPGA Xilinx řady XC4000 konfigurovaného v módu *MASTER SERIAL* a obvodů Xilinx řady XC5200 a XC3100A v módu *SLAVE SERIAL*. Konfigurační data pro všechny obvody jsou opět uložena v sériové paměti Xilinx typu XC17xxD.



Obr. 9 Konfigurace více obvodů XILINX různých typů

Každý z různých režimů konfigurace má svoje výhody i nevýhody, volí se vždy ten, který nejlépe vyhovuje systému, ve kterém obvod pracuje. Záleží na době konfigurace, na počtu nevyužitých vývodů obvodu FPGA a na místě na plošném spoji pro konfigurační paměť. Konfigurace z klasické paměti je rychlejší, ale samotná paměť zabírá na plošném spoji více místa než paměť sériová a navíc je nutné ji s obvodem FPGA propojit více vývody.

Závěr

Použití obvodů FPGA je v každém případě velmi perspektivní a v moderních systémech se s nimi téměř vždy setkáme. Jejich výhodou není jen úspora místa v systému, ale také snížení spotřeby a především urychlení doby vývoje systému – zvláště při použití ověřených maker IP. Ačkoliv se návrhové prostředky EDA a implementační nástroje výrobců programovatelných obvodů neustále zdokonalují, roste jejich výkon a počet plně automatizovaných operací, závisí kvalita a rychlost návrhu vždy na dlouholetých zkušenostech a praxi návrháře.

Obvody FPGA byly dříve používány především do prototypových a ověřovacích sérií. Dnes mohou v určitých aplikacích díky novým moderním řadám a při splnění určitých podmínek nahradit klasické zákaznické obvody. Ve většině případů ale pro specializovaná návrhová centra a v konečné fázi pro zákazníka představuje konečné řešení klasický – z hlediska parametrů a funkce maximálně optimalizovaný – zákaznický obvod. Od obvodů FPGA k obvodům ASIC programovaným maskami vede také jednoduchá cesta nazývaná konverze – o této metodě převodu kompletně navrženého a ověřeného systému se dozvíte v některém z dalších pokračování tohoto seriálu článků.

Ing.Luboš Hradecký, Ing.Miloš Bečvář
Ing.Petr Slavík

LITERATURA:

[1] http://www.xilinx.com
[2] http://www.altera.com
[3] Keating, M.,Bricaud, P.: Reuse Methodology Manual, Kluwer Academic Publisher, Boston, Doedrecht, London, 1998
[4] Xilinx: Programmable Logic Data Book 1999
[5] Xilinx: Core Solution DataBook 2/1998
[6] http://www.design-reuse.com
[7] http://www.vsi.org