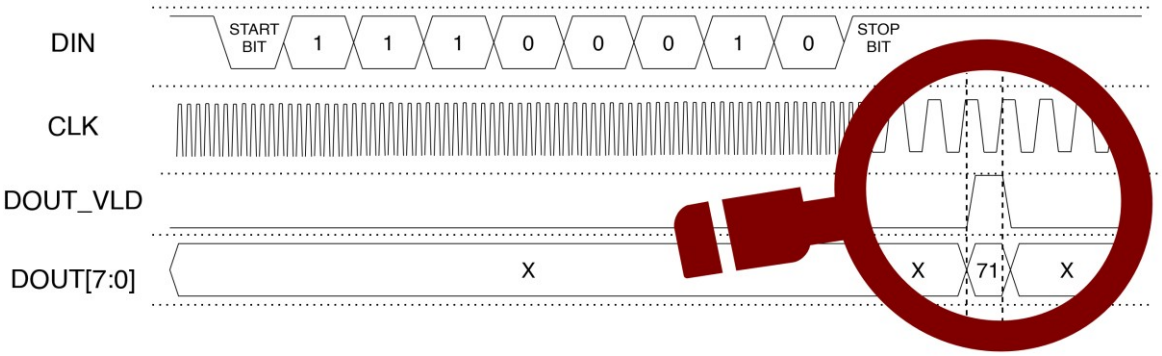
První část projektu UART – Návrh obvodu

Navrhněte obvod pro příjem datových slov po asynchronní sériové lince (UART_RX).

- Vycházejte ze základních informací o fungování a zpracování asynchronní sériové komunikace uvedených v následující kapitole.
- Uvažujte vstupní tok dat v pevném formátu: jeden START bit, 8 bitů dat, jeden STOP bit, zasílaných rychlostí 9600 baudů za sekundu. Přijímací obvod bude pracovat na 16x vyšší frekvenci (signál CLK) ve srovnání s přenosovou rychlostí jednotlivých datových bitů. Vašim úkolem bude snímat datové bity uprostřed přenášeného intervalu (viz. obrázek 3).
- Obvod UART_RX bude přijímat jednotlivé bity na vstupním datovém portu DIN, provede jejich de-serializaci a výsledné 8-bitové slovo zapíše na datový port DOUT. Platnost datového slova na portu DOUT potvrďte nastavením příznaku DOUT_VLD na úroveň logické 1 po dobu jednoho taktu hodinového signálu CLK. Příklad časového diagramu ukazujícího očekávaný průběh signálů na vstupně/výstupních portech komponenty UART_RX je znázorněn na obrázku 1.



Obrázek 1. Příklad časového průběhu na vstupech a výstupech obvodu UART_RX

- Jednotlivé části datové cesty obvodu bude potřeba ovládat skrze konečný automat (*Finite State Machine*). Sestavte si nejprve graf přechodů tohoto automatu.
- Při návrhu nezapomeňte ošetřit asynchronní vstup do synchronní sítě obvodu UART_RX pro redukci možných metastabilních stavů.

Volitelně vytvořte a odevzdejte technickou zprávu, která bude obsahovat:

- Jméno, příjmení a login.
- Schéma architektury navrženého obvodu UART_RX na úrovni RTL a její stručný popis.
- Nákres grafu přechodů konečného automatu a jeho stručný popis.

Ukázku formátu a obsahu výstupní zprávy naleznete v příloze. Rozsah vaší zprávy z první části projektu by neměl překročit dvě strany formátu A4. Zprávu odevzdejte ve formátu PDF jako soubor s názvem *zprava.pdf* skrze informační systém, termín pro první část projektu.

Před odevzdáním do informačního systému se prosím ujistěte, že finální podoba zprávy obsahuje všechny potřebné náležitosti. Zkontrolovanou verzi zprávy odevzdejte prostřednictvím informačního systému nejpozději do data uvedeného v informačním systému u termínu pro první část projektu. Pozdější odevzdání nebude bráno v úvahu.

Asynchronní sériová komunikace

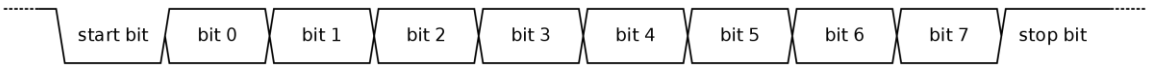
Asynchronní sériová komunikace se stala základním způsobem přenosu dat mezi počítači a periferními zařízeními. V současné době se používá zejména v oblasti vestavěných systémů. Pro přenos dat mezi dvěma uzly stačí při sériové komunikaci jeden datový vodič, po kterém jsou postupně zasílány jednotlivé datové bity. Asynchronnost komunikace pak znamená, že přenášené bity nejsou synchronizovány žádným dodatečným signálem jako je např. hodinový signál CLK. Přijímač je schopen rozpoznat příchozí bity a jejich synchronizaci na základě použitého zakódování na datovém vodiči.

Komunikační linka (vodič) je vždy před začátkem přenosu libovolného vícebitového slova (obvykle bajtu) nastavena na úroveň logické 1. Přenos vícebitového slova pak začíná tzv. START bitem s hodnotou logické 0. Odvysílání START bitu, tedy přechod datové linky z hodnoty logické 1 do 0, umožní přijímači spolehlivě identifikovat okamžik začátku přenosu.

Za START bitem jsou následně odvysílány jednotlivé bity datového slova od významově nejnižšího bitu (LSB) po významově nejvyšší bit (MSB). Za posledním bitem datového slova následuje jeden nebo více tzv. STOP bitů, které jsou vždy nastaveny na úroveň logické 1.

Za STOP bitem může začít přenos dalšího datového slova, začíná se opět START bitem. Všimněte si, že STOP bit předchozího datového slova v kombinaci se START bitem dalšího slova umožňují spolehlivou detekci začátku nového přenosu (přechod z log. 1 do 0).

Příklad přenosu 8-bitového datového slova s jedním STOP bitem je znázorněn na obrázku 1.

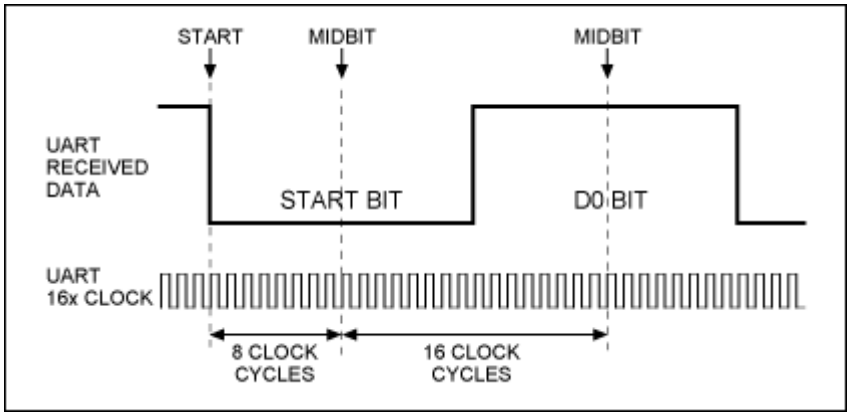


Obrázek 2. Příklad sériového přenosu 8-bitového datového slova s jedním STOP bitem

Pro spolehlivé rozpoznání jednotlivých bitů přenášeného datového slova na straně přijímače je potřeba nejen identifikovat začátek přenosu (přechod z logické 1 do 0), ale také vědět na jaké rychlosti komunikace probíhá. Vysílač i přijímač se proto musí nejprve nastavit na stejnou přenosovou rychlost.

Přenosová rychlost se udává v počtu přenesených baudů za sekundu, přičemž jeden baud odpovídá v tomto případě jednomu bitu. Základní a také nejčastěji používanou přenosovou rychlostí je rychlost 9600 baudů za sekundu. Pokud uvažujeme přenos 8-bitových datových slov ohraničených jedním START bitem a alespoň jedním STOP bitem (celkem 10 bitů), potom jsme schopni na rychlosti 9600 baudů přenášet až 960 bajtů za sekundu (9600/10).

Aby přijímač spolehlivě identifikoval hodnoty (logické úrovně) přenášených datových bitů je navíc doporučeno, aby tento obvod pracoval na 16x větší frekvenci, než je vybraná přenosová rychlost. Každý datový bit by následně měl být snímán uprostřed časového intervalu pro přenos bitů, jak je naznačeno na obrázku 2. Specifikace doporučuje při 16x větší vzorkovací frekvenci snímat logickou úroveň vstupních dat v 7., 8. a 9. hodinovém cyklu a jako výsledný bit použít majoritu z těchto tří hodnot. Pro jednoduchost se však zde spokojme s hodnotou nasnímanou na konci 8. hodinového cyklu označenou jako MIDBIT.



Obrázek 3. Příklad vzorkování datového bitu uprostřed[1]

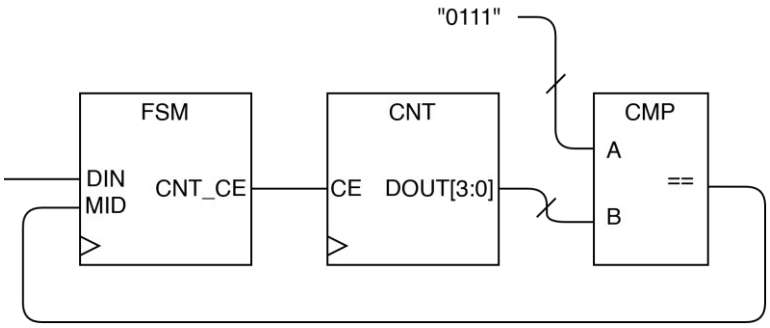
[1] Převzato z https://electronics.stackexchange.com/questions/207870/uart-receiver-sampling-rate

Příloha: Výstupní zpráva (Ukázka)

Jméno:
Login:

Architektura navrženého obvodu (na úrovni RTL)

Schéma obvodu



Poznámky:

- Pro přehlednost CLK a RST signály ve schématech uvádíme, ale nemusíme zapojovat.
- Několik jednobitových D-KO můžete pro přehlednost spojit do jednoho vícebitového registru, pokud tedy sdílí všechny kontrolní signály jako CLK, RST, nebo CE.
- Jednotlivé vodiče můžete spojovat do vícebitových sběrnic.

Popis funkce

Stručný slovní popis struktury a funkce obvodu (max. polovina strany A4).

Druhá část projektu UART – Implementace a ladění

1. Pro vypracování druhé části projektu budete potřebovat nástroje pro simulaci a syntézu číslicových obvodů popsaných v jazyce VHDL. Vyžadovány jsou konkrétně open-source nástroje GHDL a GTKWave. Více informací o tom, jak nástroje získat a používat je uvedeno v následující kapitole.
2. Z informačního systému si stáhněte ZIP archiv zdrojových souborů a seznamte se s jeho obsahem. V archivu naleznete hlavně tyto tři VHDL zdrojové soubory:
   - *uart_rx.vhd* – zdrojový soubor v jazyce VHDL obsahující definici rozhraní (entity) komponenty UART_RX a prázdnou architekturou na doplnění vaši implementace,
   - *uart_rx_fsm.vhd* – zdrojový soubor v jazyce VHDL, který bude sloužit pro popis konečného automatu řídicího ostatní komponenty vašeho obvodu UART_RX,
   - *testbench.vhd* – zdrojový soubor v jazyce VHDL, který reprezentuje ukázkové zapojení pro testování základní funkcionality vámi navržené a implementované komponenty UART_RX v prostředí simulace.
3. Navržený RTL obvod z první části projektu implementujte v jazyce VHDL a uložte do předpřipraveného souboru *uart_rx.vhd*. Kód odpovídající konečnému automatu vložte pro přehlednost do samostatného souboru *uart_rx_fsm.vhd*.
4. Proveďte syntézu a simulaci vašeho VHDL kódu spuštěním připraveného skriptu *uart.sh*. Skript využívá program GHDL pro analýzu, syntézu a simulaci VHDL kódu, programem GTKWave pak zobrazuje průběh simulace (tzv. WaveForm). Na základě textových výstupů programu GHDL v terminálu nejdřív opravte nalezené syntaktické a sémantické chyby a simulaci opakujte. Nakonec zkontrolujte průběh simulace v grafickém okně s ohledem na správnou funkčnost implementovaného obvodu.
5. Doplňte technickou zprávu z první části projektu o snímek obrazovky (Print Screen) programu GTKWave s ukázkou časového průběhu simulace zachycujícího přenos alespoň jednoho datového slova. Jestliže jste se při implementaci odchýlili od vašeho původního návrhu z první části projektu, upravte také nákresy RTL architektury obvodu a grafu přechodů konečného automatu. Finální verze zprávy by měla souhlasit s vytvořenou implementací. Zachovejte formát a obsah zprávy podle ukázky v příloze.
6. Výstupy druhé části projektu budou tvořit:
   - doplněné zdrojové soubory v jazyce VHDL (*uart_rx.vhd* a *uart_rx_fsm.vhd*),
   - soubor *zprava.pdf* s výstupní zprávou ve formátu PDF.

   Všechny tři uvedené soubory zabalte do ZIP archivu s názvem ***<login>.zip***. Archiv odevzdejte skrze informační systém, do termínu pro druhou část projektu.

Před odevzdáním vašeho archivu do informačního systému si ho prosím otestujte skrze zveřejněnou sadu testovacích skriptů připravených v souboru *test.zip*. Podrobný návod na otestování naleznete v přiloženém README souboru.

Otestovaný archiv odevzdejte prostřednictvím informačního systému nejpozději do data uvedeného v informačním systému u termínu pro druhou část projektu. Pozdější odevzdání nebude bráno v úvahu.

Simulace a syntéza obvodu

Můžete si zvolit jednu ze tří následujících možností překladového prostředí pro vypracování a testování druhé části projektu.

Virtuální stroj

Pro účely vypracování projektu jsme pro vás připravili obraz disku virtuálního stroje, kde jsou nainstalovány potřebné nástroje. Obraz obsahuje zejména instalace GHDL a GTKWave. Naistalován je také výukový program LogiSim, který byl zmiňován na přednáškách.

Archiv (2,3 GB) s obrazem disku virtuálního stroje si stáhněte z privátních stránek FITkitu. VMDK soubor (7,3 GB) z archivu vybalte pomocí aplikace 7z.

Pro spuštění virtuálního stroje si nainstalujte volně dostupný program VirtualBox. Při tvorbě VM použijte nastavení Ubuntu (64-bit), alespoň 1024 MB paměti, stažený VMDK disk.

Lokální instalace

Nástroje GHDL a GTKWave jsou dostupné jako open-source bez potřeby jakékoli licence pro jejich používání. Možná je proto jejich jednoduchá přímá instalace na vašem stroji.

Na operačních systémech typu **Linux** by měla být instalace možná přímo standardním balíčkovacím nástrojem (např. *apt*, *yum*, *pacman*). Nainstalujte si nástroj GHDL (balíček *ghdl*) alespoň ve verzi 1.0.0 a nástroj GTKWave (balíček *gtkwave*) alespoň ve verzi 3.3. Ujistěte se, že cesta k nástrojům je součástí proměnné prostředí PATH a tedy je možné přímo v terminálu volat příkazy *ghdl* a *gtkwave*.

Na operačních systémech typu **Windows** a **MAC OS** je potřebná u obou nástrojů ruční instalace nebo překlad ze zdrojových souborů. Pro GHDL jsou dostupné instalační archivy a návod instalace. Pro GTKWave je dostupný archiv pro Windows a archiv pro MAC OS. Na Windows si navíc nainstalujte Bash terminál například skrze prostředí Cygwin, MinGW nebo WSL. Ujistěte se, že cesta k nástrojům je součástí proměnné prostředí PATH a tedy je možné přímo v terminálu volat příkazy *ghdl* a *gtkwave*.

Překladový server

Na fakultě je skrze SSH spojení dostupný server *fitkit-build.fit.vutbr.cz*, na kterém jsou potřebné nástroje GHDL a GTKWave již nainstalovány. Pro vzdálené zobrazení grafického okna programu GTKWave je potřeba mít u připojení správně nastavený X11 forwarding.

Server je navíc umístěn v doméně fit.vutbr.cz a umožňuje připojení jen přímo ze sítě fakulty nebo vzdáleně skrze fakultní VPN. Postupujte proto dle návodu uvedeného na následujících stránkách a vždy před přístupem k serveru mimo síť fakulty si VPN spojení aktivujte.

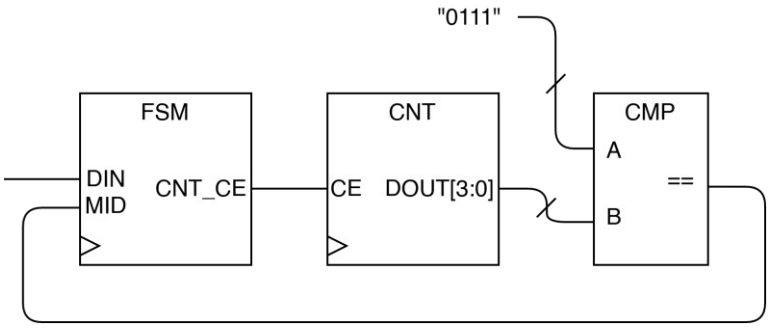
Uvědomte si také, že v ročníku vás je několik stovek a sdílený server nemusí zvládnout nápor všech studentů najednou. Upřednostněte proto jednu z předešlých dvou variant a na vzdálený server se plně nespoléhejte. Nemusí být dostupný nebo plně responzivní zejména v čase těsně před termínem odevzdání projektu.

Příloha: Výstupní zpráva (Ukázka)

Jméno:
Login:

Architektura navrženého obvodu (na úrovni RTL)

Schéma obvodu



Poznámky:

- Pro přehlednost CLK a RST signály ve schématech uvádíme, ale nemusíme zapojovat.
- Několik jednobitových D-KO můžete pro přehlednost spojit do jednoho vícebitového registru, pokud tedy sdílí všechny kontrolní signály jako CLK, RST, nebo CE.
- Jednotlivé vodiče můžete spojovat do vícebitových sběrnic.

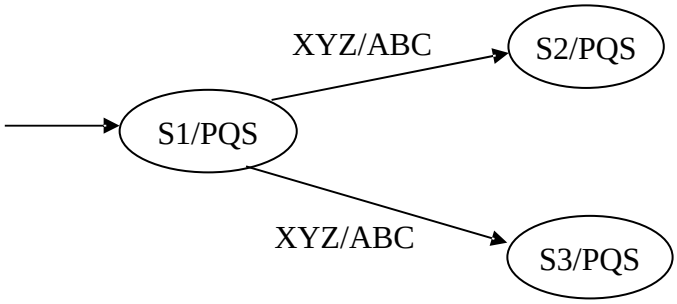
Popis funkce

Stručný slovní popis struktury a funkce obvodu (max. polovina strany A4).

Návrh automatu (Finite State Machine)

Schéma automatu

Legenda:
- Stavy automatu: S1, S2, S3
- Vstupní signály: X, Y, Z
- Mealyho výstupy: A, B, C
- Moorovy výstupy: P, Q, S



Poznámky:
- Použijte vhodné názvy pro stavy, vstupní a výstupní signály tak, aby byl snáze pochopitelný jejich význam.
- Za vstupně/výstupní signály XYZ, ABC a PQS dosaďte do grafu přímo hodnoty 0, 1 nebo X (don't care).
- Signály CLK a RST neuvádíme mezi vstupy automatu ani je nekreslíme do schémat.
- Automat může vhodně kombinovat jak Mealyho, tak Moorovi výstupy.
- Pokud je vstupním signálem vektor bitů, můžete s ním na hranách pracovat jako s vektorem.
- Připomeňte si konvence pro kreslení grafu automatu probírány na přednáškách.
- Nezapomeňte na označení počátečního stavu automatu.

Popis funkce

Stručný slovní popis funkce automatu (max. polovina strany A4).

Snímek obrazovky ze simulací

Zde prosím vložte obrázek (snímek obrazovky) z nástroje GTKWave, který demonstruje funkčnost vašeho obvodu na úrovni simulací. Zachyťte prosím přenos alespoň jednoho datového slova v okně Wave. Pro přehlednost můžete obrázek orientovat např. na šířku stránky A4.

Třetí část projektu UART – Testování v hardware

Analýza a překlad hardware

1. Připraveny jsou tři možnosti jak realizovat překlad hardwarové architektury UART modulu a jeho zapojení na procesor ARM. Vybrat si je možné jeden z následujících postupů:

   a. Je možné použít libovolný **počítač v CVT** a na něm nainstalovaný nástroj Vivado v OS Windows. Spuštění nástroje je možné z příkazové řádky (Win+R, cmd, Enter). Po zobrazení okna CMD je potřebné nastavení cesty, v CVT se to provádí příkazem:
   `PATH=Q:\fitkit\Vivado2018\Vivado\2018.2\bin;%PATH%`
   Posléze je možné nástroj již spouštět jednoduše voláním *vivado*. Při každém novém otevření CMD je však potřeba tuto cestu znovu nastavit.

   b. Připraven je obraz disku **virtuálního stroje** na privátních stránkách FITkitu, konkrétně jde o archiv *Vivado_2020.1.7z*. VMDK soubor vybalte pomocí nástroje 7z a použijte pro virtuální stoj v programu VirtualBox. Při tvorbě virtuálního stroje použijte nastavení Ubuntu (64-bit), alespoň 2048 MB paměti, stažený VMDK disk.

   c. Nástroj Vivado v správné verzi je možné **nainstalovat lokálně** na svůj počítač. Z privátních stránek FITkitu si stáhněte instalační archiv pro Linux nebo Windows, rozbalte jej a spusťte instalaci. V nastaveních instalace můžete vypnout podporu DocNav, Cable Drivers a všech FPGA čipů kromě Zynq-7000 rodiny. Po dokončení instalace přidejte podsložku *Xilinx/Vivado/2020.1/bin* do globální proměnné PATH a pro fungování programu *make* na Windows také *Xilinx/Vivado/2020.1/gnuwin/bin*.

2. Z informačního systému si stáhněte ZIP archív zdrojových souborů a rozbalte ho. Podle použité verze nástroje Vivado zvolte složku, se kterou budete dále pracovat. V CVT je dostupná verze 2018.2, jinak (VM nebo vlastní instalace) je použitá verze 2020.1. Seznamte se s obsahem složky, naleznete v ní hlavně tyto části:

   a. Složku *ip_cores* obsahující specifické moduly (IP cores). Konkrétně zde jde jenom o modul *simple_uart_1.0* implementující kompletní zjednodušenou UART periferii.

   b. Složku *overlay* obsahující implementaci hlavní architektury zapojení hardwarového obvodu pro FPGA čip včetně jejího ovládání skrze třídu řadiče v jazyce Python.

   c. Program *uart_echo.py* implementující jednoduchou aplikaci pro přenos dat skrze UART modul zapojený v hardwarovém overlay.

3. Přesuňte se do složky *ip_cores* a otevřete *.xpr* soubor Vivado projektu pro editaci modulu *simple_uart_1.0*. Otevření je možné z GUI nástroje Vivado skrz možnost *Open Project* nebo voláním následujícího příkazu z CMD: `vivado edit_simple_uart_v1_0.xpr`. Analyzujte vnitřní architekturu modulu pomocí volby *Open Elaborated Design*. Všimněte si zejména zapojení zatím prázdných *UART_RX* a *UART_TX* komponent, které jsme právě v projektu nebo na cvičení implementovali.

4. Z informačního systému si stáhněte svůj *xlogin00.zip* archiv s odevzdanou implementací komponenty *UART_RX* z druhé části projektu. Ve složce *ip_cores/simple_uart_1.0/hdl* nahraďte prázdné soubory *uart_rx.vhd* a *uart_rx_fsm.vhd* právě touto vaší implementací.

5. Podobně jako v předešlém bodě nahraďte také soubory *uart_tx.vhd* a *uart_tx_fsm.vhd* svou vlastní implementací kontroléru vysílací strany, nebo použijte vzorovou implementaci zveřejněnou v informačním systému.

6. V GUI nástroje Vivado proveďte *Reload* otevřeného schématu pro načtení právě změněných souborů. Všimněte si, že komponenty *UART_RX* a *UART_TX* již nejsou prázdné. Kliknutím na symbol + v jejich horním rohu můžete prozkoumat architekturu těchto implementací tak jak jí pochopil nástroj. Následně prozkoumejte zvenku viditelné parametry celého IP modulu v dialogu otevřeném volbou *Package IP*. Zastavte se v sekci *Addressing and Memory* kde jsou definovány ze softwaru přístupné registry. Změnu implementace části modulu nakonec potvrďte volbou *Re-Package IP* na konci dialogu.

7. Po zavření nástroje Vivado se přesuňte do složky *overlay*. Zde proběhne překlad FPGA architektury realizující propojení UART IP modulu s ARM procesorem platformy Zynq. Spusťte z CMD připravené překladové TCL skripty nástrojem *make* nebo přímo příkazy:
   ```
   vivado -mode batch -source build_blockdesign.tcl -notrace
   vivado -mode batch -source build_bitstream.tcl -notrace
   ```
   Zatímco běží překlad, pokračujte následující kapitolou, posléze se sem vraťte.

8. Po skončení překladu se seznamte s architekturou implementovaného napojení UART modulu na čipu Zynq. Projděte si zejména:

   a. Hardwarový Vivado projekt vytvořený ve složce *overlay/inc_uart* v průběhu překladu. Otevření podobně jako dříve, pomocí *.xpr* souboru přes GUI volbu *Open Project* nebo voláním příkazu z CMD: `vivado inc_uart.xpr`. Náhled na použité zapojení IP modulů poskytuje volba *Open Block Design*. Náš jednoduchý UART modul je propojený s procesním systémem čipu Zynq pomocí AXI sběrnice a je také připojen na dva externí signály.

   b. Definování mapování externích signálů na konkrétní piny pouzdra čipu Zynq v souboru *overlay/inc_uart.xdc*. RXD sériový vodič je připojen na pin W14 a TXD sériový vodič na pin Y14. Podle dokumentace platformy PYNQ-Z2 od výrobce jsou tyto piny čipu Zynq připojeny na piny 1 a 2 portu PMODB.

   c. Implementaci Python tříd v souboru *overlay/inc_uart.py* pro ovladač UART periferie a její zapojení v hardwarové architektuře. Třída *UartCtrl* implementuje ovladač a zapouzdřuje specifika nízko úrovňového ovládání komponenty do lehčeji uchopitelných funkcí. Třída *INCUartOverlay* reprezentuje celkovou architekturu zapojení, v tomto případě zde není definováno nic navíc.

9. Po úspěšném dokončení překladu jsou ve složce *overlay* vytvořeny soubory *inc_uart.bit* a *inc_uart.hwh* s konfigurací pro programovatelnou FPGA logiku čipu Zynq.

Zprovoznění kitu PYNQ-Z2

Kit PYNQ-Z2 včetně základních doplňků si můžete půjčit z knihovny fakulty. Doplňky kitu by měli zahrnovat minimálně micro-SD kartu s nahraným PYNQ-Z2 obrazem OS, USB-kabel, a RJ45 síťový kabel. Pro potřeby této demonstrace je navíc potřeba jeden DuPont M-M kablík. Při organizovaném laboratorním cvičení z předmětu INC všechny tyto pomůcky studentům zajistí cvičící.

K platformě PYNQ existuje oficiální portál s veškerou potřebnou dokumentací. Zde se také nachází sekce speciálně věnovaná zprovoznění PYNQ-Z2 kitu. Postup je popsán formou videa nebo v textu sekcí odkazované stránky. Základní zprovoznění zahrnuje zejména:

1. Základní zprovoznění a zapnutí kitu:
   video 1:54 až 4:01 a 6:04 až 7:01,
   nebo textové sekce *Board Setup* a *Turning On the PYNQ-Z2*.

2. Zapojení USB kabelu pro přístup k příkazové řádce skrz sériovou linku:
   video 9:28 až do konce,
   nebo textové sekce *Opening a USB Serial Terminal*.

   Poznámka: Na počítači v CVT si stáhněte program *putty* z odkazu v textové sekci. Stahujte přímo *.exe* soubor, který je možné spustit bez instalace.
   Poznámka: Na Linuxu ve VM spusťte *putty* jako root a zvolte sériový port */dev/ttyUSB1*. Nezapomeňte také při použití VirtualBoxu předat zprávu nad USB zařízením *Xilinx TUL USB* do rukou běžícího virtuálního stroje.
   Poznámka: Jestli po připojení sériové linky vidíte prázdné okno terminálu, stiskněte Enter.

3. Připojení Ethernet kabelu pro přenos souborů a webový přístup:
   video 4:01 až 6:04 a 7:01 až 9:28,
   nebo textové sekce *Network connection*.

   Poznámka: Na počítači v CVT je nástroj Vivado spouštěn ze síťového disku Q. Před odpojením síťového kabelu a připojením PYNQ-Z2 dokončete používání nástroje a nachystejte si potřebné výsledky překladu.

Teď byste měli mít kit PYNQ-Z2 připraven k použití. Za zmínku stojí, že kromě demonstrační aplikace připravené v následující kapitole jsou přímo na kitu dostupné i jiné ukázkové aplikace ve formě Jupyter Notebooků. Přístupné jsou skrze síťové připojení ke kitu z webového prohlížeče počítače.

Spuštění na kitu

1. Vytvořte si zložku v souborovém systému na SD kartě kitu PYNQ-Z2. Nakopírujte do této složky následující soubory (podsložku *overlay* v cestě zachovejte):
   - *uart_echo.py* s implementací demonstrační aplikace,
   - *overlay/inc_uart.py* s implementací tříd pro přistup k hardwarové architektuře,
   - *overlay/inc_uart.bit* s přeloženým binárním souborem konfigurace FPGA,
   - *overlay/inc_uart.hwh* s popisem přeložené architektury a její komponent.

2. Připojte se *putty* terminálem ke kitu PYNQ-Z2 a získejte root práva (*su* s heslem *xilinx*).

3. Spusťte demonstrační aplikaci pro UART komunikaci voláním *python3 uart_echo.py*. Aplikace nejdřív nahraje konfiguraci do FPGA a inicializuje přenosy. Pak je možné zasílat a přijímat zprávy přes sériové datové vodiče UART sběrnice. Co napíšete do konzole a potvrdíte klávesou Enter se odešle na TXD vodič. Co je přijato na RXD vodiči se automaticky čte a vypisuje do konzole. Kombinace *Ctrl+C* běh aplikace ukončuje.

4. Propojte DuPont kablíkem piny 1 a 2 PMODB portu, na které jsou z FPGA vyvedeny RXD a TXD vodiče UART sběrnice. Port PMODB je na desce pojmenován, nachází se vedle čtyř tlačítek. Piny 1 a 2 jsou první dva zprava v horním řádku portu. Propojením RXD a TXD vzniklo takzvané *loopback* zapojení, všechno co vysílací strana odesílá na RXD pin se zapojeným vodičem vrací na přijímací stranu TXD pinem. Cokoliv je teď v demonstrační aplikaci odesláno by se mělo vzápětí přijmout a vypsat, každý řádek tedy vidíte v konzoli dvakrát.

5. Rozpojte *loopback* propoj své UART sběrnice a propojte se se sousedem. Pozor, vždy propájejte piny do kříže. Tedy svůj pin 1 (RXD) napojte na pin 2 (TXD) druhého kitu a svůj pin 2 napojte na pin 1 druhého kitu. Teď by mělo být možné posílat si zprávy přes UART mezi kity. Při spuštěné demonstrační aplikaci na obou kitech je v konzoli jedného vypsáno cokoli co bylo druhým odesláno a opačně.