BZA Projekt

Posuvný registr s lineární zpětnou vazbou

Varianta 2 – implementace

Jan Suchánek <xsucha09@stud.fit.vutbr.cz>

3. dubna 2019

1 Úvod

V rámci projektu do předmětu Bezpečná zařízení jsem si zvolil jako cíl navrhnout a naimplementovat aplikaci, která by sloužila k demonstraci funkcionality Posuvného registru s lineární zpětnou vazbou. Největší přínos takové aplikace vidím ve využití jako učební pomůcky pro pomoc s pochopením funkcionality daného obvodu.

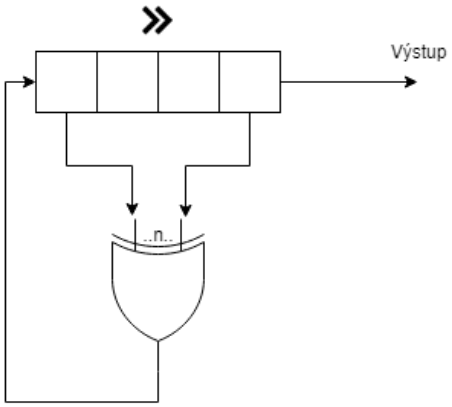
Výsledkem práce by měl být program s grafickým znázorněním modelovaného obvodu a možností jeho intuitivní modifikace. Rovněž by mělo být nad grafickým zobrazením modelu demonstrovat jeho průběh – tedy posun bitů v registru a „putování“ bitů v obvodu. Běžně nás při analýze LFSR zajímá proud bitů, který získáváme na výstupu a dále se využívá. Stejně přínosná může být i informace o stavu registru v daném kroku. Tyto informace by aplikace měla nabídnout. Bonusem práce by mohla být přitažlivá vizualizace, která by zaujala uživatele i případné posluchače při výkladu látky o tomto typu posuvných registrů.

V tomto dokumentu je přiblížen proces návrhu a implementace aplikace, v sekci 2 je popsána teorie kolem LFSR a jejího užití. Sekce 3 popisuje postup tvorby ukázkové aplikace a v sekci 4 je popsána obsluha, instalace a sestavení aplikace.

2 Posuvný registr s lineární zpětnou vazbou.

Posuvný registr s lineární zpětnou vazbou [Wik19], z anglického „Linear feedback shift register“, je číslicový obvod, který vychází z běžného posuvného registru a pomocí zpětných vazeb a funkce nad nimi upravuje svůj stav. Výhoda tohoto obvodu spočívá ve snadné a rychlé implementaci, jednak fyzicky

pomocí klopných obvodů, jednak programově. Schéma na obrázku 1 je jednoduché znázornění čtyřbitového LFSR. Stav registru můžeme napsat jako



Obrázek 1: Schéma LFSR

$R = (r_1, ..., r_n)$. Zpětná vazba bývá uvedena jako $T = (t_1, ..., t_n)$. Pro zápis zpětné vazby se ale většinou používá polynomiální zápis, například pro čtyřvitový registr na obrázku 1 se jedná o polynom $x^4 + x + 1$. Další alternativní zápis je pomocí matic.

Vlastnosti a užití LFSR

Využití LFSR je poměrně široké. Je schopny fungovat jako generátor pseudonáhodných čísel, také se dá výstupní bity použít pro proudové šifrování. Další oblast užití je i pro generování náhodného šumu. Výhody a nevýhody LFSR vyplývají z jeho návrhu. Hlavní výhodou je cena HW obvodu. Navíc kombinací více LFSR za menší cenu lze docílit vylepšení bezpečnostních vlastností a ušetření finančních prostředků.

Důležitým parametrem je perioda LFSR, což je počet kroků, než dojde k opakování stejného stavu registru. Perioda vychází z funkce zpětné vazby a pro dosažení maximální periody je potřeba, aby polynom popisující zpětnou vazbu registru byl primitivní. Výpočet maximální periody je pak $T_max = 2^n - 1$, kde n značí počet bitů registru. Pro ukázku, tabulka LFSR se zpětnou vazbou $x^2 + x + 1$ je ukázána na obrázku 2.

3 Návrh a implementace

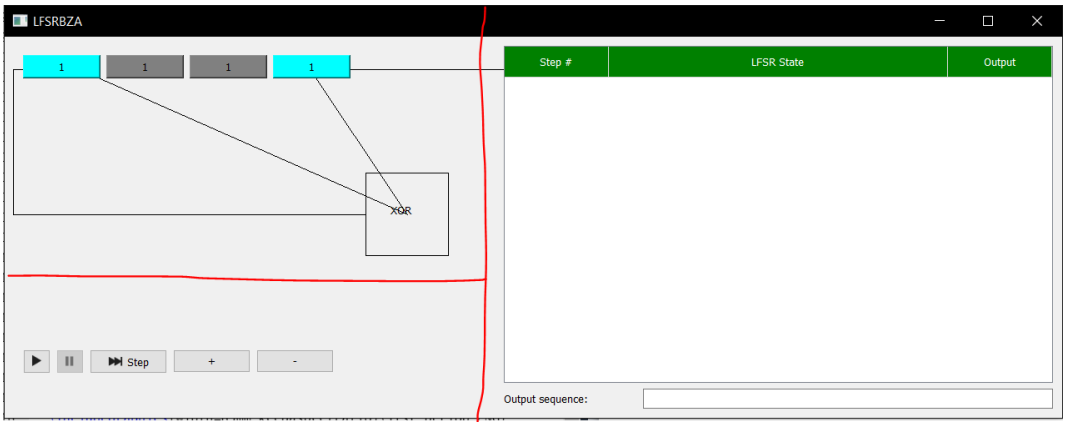
V této sekci je popsáno, jakým způsobem byla aplikace implementována a proč. Vypsány jsou také použité technologie a poukázáno na některé důležité části implementace.

0	[1, 1]	1
1	[0, 1]	1
2	[1, 0]	0
3	[1, 1]	1
4	[0, 1]	1
5	[1, 0]	0

Obrázek 2: Perioda dvoubitového LFSR

3.1 Návrh

Cílem bylo vytvořit okenní aplikaci, která umožňuje zobrazit definovaný model LFSR, nějakým způsobem ho ovládat a měnit a zobrazit všechny potřebné výstupy. Tímto jsou dány tři sekce aplikace, jak je vidět na fotce výsledku na obrázku 10.



Obrázek 3: Výsledná aplikace s naznačenými částmi

Registr je znázorněn zjednodušeně jako sada bitů, vodiče se nejlépe značí jednoduchými čarami. Ovladač aplikace by měl umožňovat krokování obvodu, jeho úpravy a také spuštění postupné simulace.

Technologie

Pro aplikaci jsme si stanovili vlastnosti, které by měla splňovat. Navíc k vlastnostem popsaným v úvodu je přidán ještě požadavek na multiplatformnost, což bývá pro okenní aplikace oříšek. Z toho vyplývá použitý Qt framework[Com19a], který umožňuje snadnou tvorbu GUI pro aplikace a navíc je přenositelný. Jako alternativa se jevilo vytvoření webové aplikace. Z osobních preferencí vyšel lépe Qt framework, protože jsem si chtěl vyzkoušet

práci s ním. Také k němu existuje kvalitní dokumentace, ze které jsem čerpal [Com19b].

Ve volbě programovacího jazyka hrál roli použítý framework. Původně je psán pro jazyk C++, ale v dnešní době již podporuje i Python. V tomto případě byl ale použit právě jazyk C++ pro možnost vytvoření malého jednoduchého binárního souboru.

4 Návod

V rámci návodu bych rád poukázal na potřebné prerekvizity pro sestavení aplikace a její případy užití.

Prerekvizity a instalace

Pro sestavení aplikace je potřeba mít nainstalován Qt Framework ve verzi 5.12.1 a novější. Ve windows je sestavení nejsnazší pomocí Qt Creator aplikace, kde stačí otevřít zdrojový projekt a aplikaci spustit. Případně je k dispozici přenositelná 32-bitová aplikace pro windows. Toto sestavení bylo testováno pod operačním systémem Windows 10. Instalace není nutná.

Použití

Obrázek 4: Ovládací panel

Ovládání aplikace je jednoduché, pomocí tlačítek „Play“ a „Pause“, znázorněných ikonkou se ovládá plynulá animace funkčnosti obvodu. Tlačítko „step“ provede právě jeden krok výpočtu, což obnáší vyčtení hodnot zpětné vazby, jejich XOR, posunutí registru, přidání nové hodnoty na vstup registru a zaznamenání výstupu a výchozího stavu. Tlačítka „+“ a „-“ slouží ke změně počtu bitů registru. Při této akci dojde ke kompletnímu resetu aplikace, vynuluje se výstupní proud bitů i tabulka předchozích kroků. Nastavení hodnot bitového registru je proveditelné pomocí myši. Levým kliknutím se přepíná hodnota bitu „1/0“ a pravým tlačítkem se aktivuje nebo deaktivuje zpětná vazba z daného bitu. Změnou zpětných vazeb opět dojde k resetu aplikace. Pro spuštění simulace je nutné mít nastavenou některý bit jako bit

se zpětnou vazbou, v aplikaci je tento bit značený modrou barvou a spojnicí ke XOR funkci. Při změně počtu bitů je potřeba tyto vazby opět nastavit.

5 Závěr

Aplikace napsaná v rámci tohoto projektu může sloužit jako pomůcka pro pochopení činnosti LFSR. V aplikaci je možné si ukázat, jak vlastně takový posuvný registr funguje, případně ji lze použít i pro simulaci LFSR s různým počtem bitů. I tak ale existuje prostor pro další vylepšení.

Vylepšení

Nejdůležitější část aplikace, která by si zasloužila vylepšení, je určitě celkový vzhled. Uživatelské rozhraní je sice jasné, ale nevypadá moc dobře na pohled a v ovládání chybí některé prvky, které by se mohly hodit. Jako příklad bych uvedl nastavení rychlosti simulace a animací, která je nastavena fixně. Ani není možné ji jednoduše změnit pomocí úpravy kódu, v tom vidím zásadní nedostatek. Poslední úpravu by si zasloužilo znázornění XOR funkce a cest k ní. Stane se, že občas vodiče prochází zkrz ostatní prvky.

Kód obecně by si zasloužil refaktoring. Občas se odchyluje od principů objektově orientovaného programování a není napsán úplně čistě. Zdrojové kódy jsou umístěny online v repozitáři `cesta-k-gitu`

Známé chyby

V aplikaci dochází při změně velikosti okna k defektům GUI. Nejvíce je to vidět při spuštení animace. Po změně velikosti neputují hodnoty správným směrem, ale směrem k původním pozicím před zvětšením nebo zmenšením okna. V dalším kroku, pokud opět nedojde ke změně, již vše vypadá normálně. Nejjednodušší způsob řešení tohoto problému by bylo deaktivovat změnu velikosti a přepočítávat ji automaticky podle počtu bitů v registru. Také při přidání většího počtu bitů dojde ke skrytí některých prvků a je nutná ruční korekce velikosti okna.

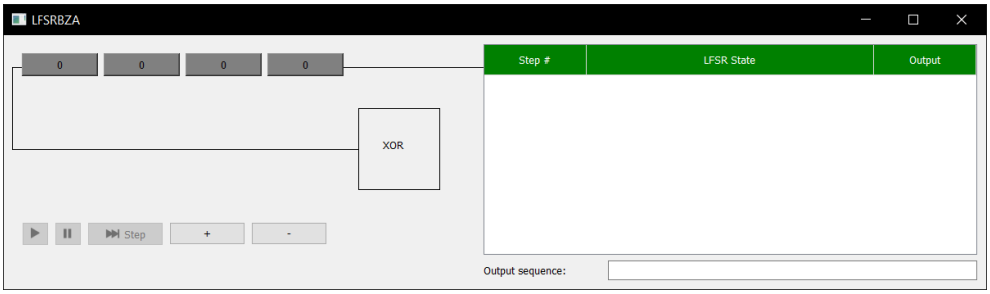
Reference

[Com19a] The Qt Company. Qt | cross-platform software development for embedded desktop, 2019. [https://www.qt.io/,Online; accessed 3-April-2019].

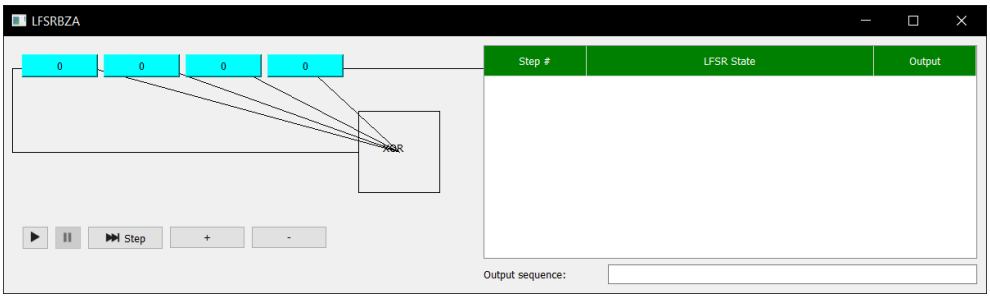
[Com19b] The Qt Company. Qt documentation, 2019. [https://doc.qt.io/,Online; accessed 3-April-2019].

[Wik19] Wikipedia. Linear-feedback shift register, 2019. [https://en.wikipedia.org/wiki/Linear-feedback_shift_register Online; accessed 3-April-2019].

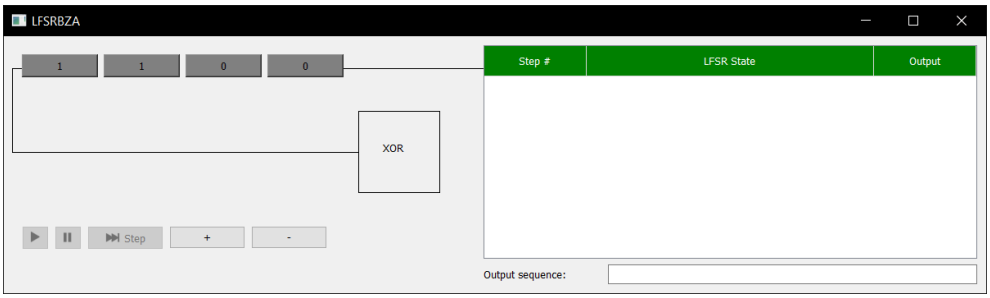
A Snímky aplikace



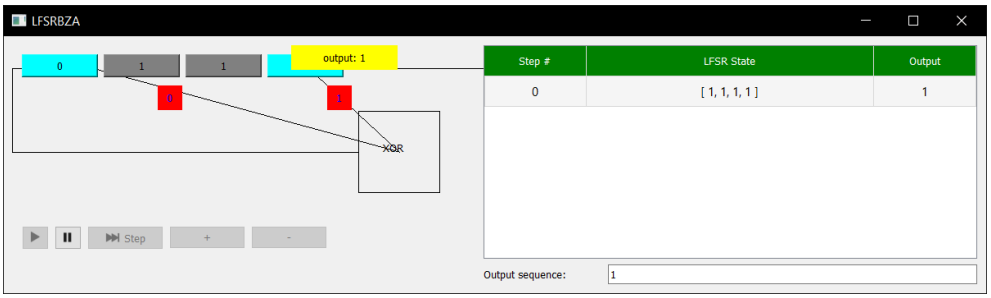
Obrázek 5: Po spuštění



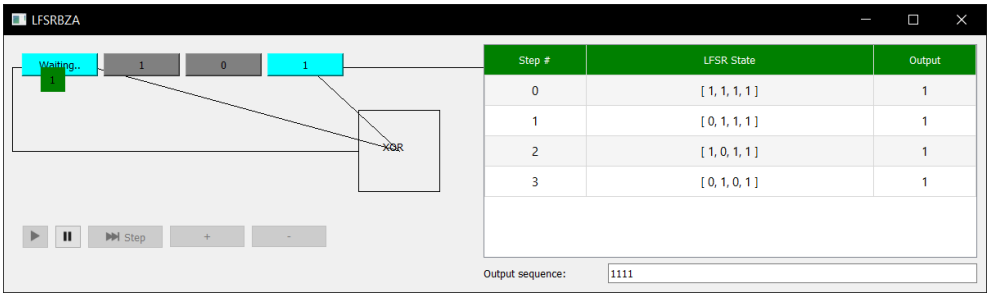
Obrázek 6: Nastavení zpětné vazby



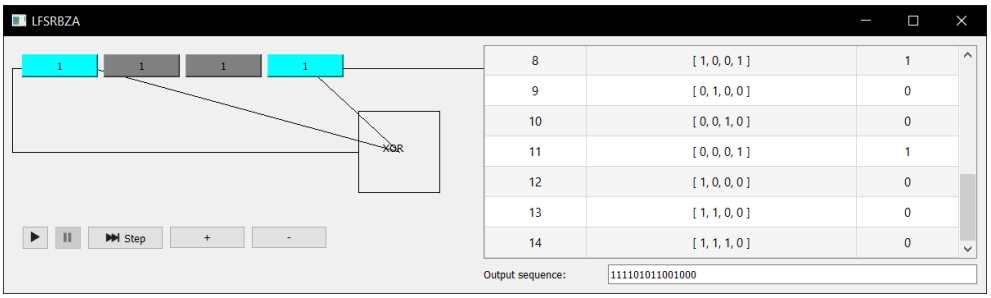
Obrázek 7: Nastaveny první dva bity



Obrázek 8: Po spuštění animace



Obrázek 9: Nastavení vstupního bitu na novou hodnotu



Obrázek 10: Po dokončení příkladu z přednášek, snímek 4