Tento článek je upraveným původním rukopisem textu publikovaného v časopise DPS Elektronika A-Z:
J. Šťastný. Implementace čítačů v číslicových systémech 3, DPS Plošné spoje od A do Z, no 5, pp. 18-20, 2011.
Oproti původnímu textu byla upravena cvičení (s ohledem na změny v prvním dílu).

Implementace čítačů v číslicových systémech 3

Jakub Šťastný
ASICentrum, s.r.o.
FPGA Laboratoř, Katedra teorie obvodů FEL ČVUT Praha

1 Úvod

V předchozích článcích byly shrnuty základní vlastnosti čítačů, implementace a výhody a nevýhody všech běžně používaných synchronních čítačů. Poslední část je věnována implementaci asynchronního čítače (*ripple counteru*) a shrnutí parametrů předvedených konstrukcí.

V celém textu označujeme počet registrů udržujících stav čítače jako N, počet stavů jako N_s. Jako f_{clk_max} označujeme maximální dosažitelnou pracovní frekvenci čítače, $T_{clk\_min}=1/f_{clk\_max}$ je pak minimální perioda hodinového cyklu.

Zkratkou MHVS budeme označovat počet současně se měnících bitů na sběrnici na výstupu čítače – maximální Hammingovu vzdálenost [2] dvou sousedních stavů čítače.

2 Asynchronní binární čítač

Doposud jsme prezentovali jen synchronní řešení obvodu čítače. U synchronního čítače se – zjednodušeně řečeno – překlápí všechny registry současně, všechny registry udržující stav čítače mají jednotný zdroj hodin. V asynchronním čítači se oproti tomu mění stavy jednotlivých registrů „jeden po druhém“. Asynchronní binární čítač (*ripple counter*) je ze všech prezentovaných typů čítačů nejmenší pro daný počet stavů. Také dosahuje nejmenší spotřeby a je schopen pracovat na nejvyšší možné f_{clk_max}. Jednoduše ho lze zkonstruovat jako čítající nahoru, či dolů, problematičtější je ale čítání s jiným krokem, než 1. Výhoda malé spotřeby a plochy zabrané obvodem je nicméně vyvážena asynchronním klopením registrů v čítači a složitějším zkracováním čítaného cyklu. Obtížnější je i detekce stavů.

Příklad 1 ukazuje asynchronní čítač čítající nahoru, sekvence stavů je uvedena v obrázku 1. Všimněte si, že každý registr užívá jako zdroj hodin předchozí registr v čítači. To posléze vede na zpoždění v překlápění jedotlivých registrů, jak je možné vidět v obrázku 2. Mezi překlopením registrů 0 a 2 zde uplyne cca 1,5 ns a na výstupu čítače se vystřídají stavy 011, 010, 000 a 100. Popsaný jev se v angličtině nazývá *ripple* (česky bychom řekli řetězová reakce, nebo dominový efekt) a dal čítači jméno. Pro větší počet registrů čítače a vyšší hodinové frekvence se snadno může stát, že ustálení celého čítače nestihne proběhnout do konce hodinové periody. S popsaným chováním je potřeba počítat a skutečné chování asynchronního čítače po implementaci zkontrolovat. Na druhou stranu je ale kritická cesta v asynchronním čítači velmi krátká a tak čítač může typicky pracovat na velmi vysokých hodinových frekvencích bez ohledu na počet jeho registrů. Proto ho bývá užitečné užít například pro čítání událostí s velmi vysokou frekvencí. Zde nám nemusí vadit delší doba ustálení než je perioda hodin, jen je potřeba správně implementovat vzorkování čítače (např. stav čítače číst jen je-li zastaven).

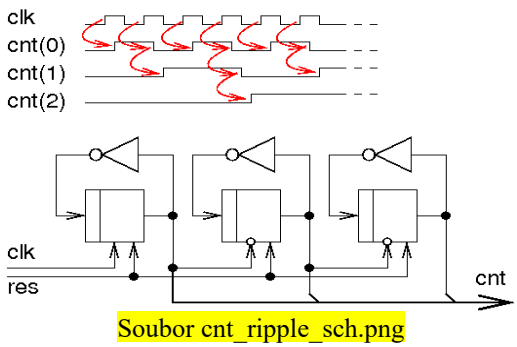
Implementace asynchronního čítače čítajícího dolů by byla obdobná s tím rozdílem, že jednotlivé registry musí klopit na náběžnou a nikoliv spádovou hranu výstupu předchozího registru.

Asynchronní čítače nejsou příliš vhodné pro programovatelná hradlová pole právě kvůli generování vlastních hodinových signálů; díky obtížnějšímu nastavování čítače (*preload*), které se musí dělat asynchronně a složitější detekci stavů v případě potřeby zkrácení čítacího cyklu ani nejsou doporučenými konstrukcemi pro začínající návrháře. Naopak, při návrhu číslicových obvodů pro zákaznické integrované obvody jsou výhody asynchronního čítače (malá plocha a malá spotřeba) naprosto nedocenitelné.

Případná modifikace asynchronního čítače pomocí ECO změny může být dosti obtížná kvůli časovému chování obvodu.

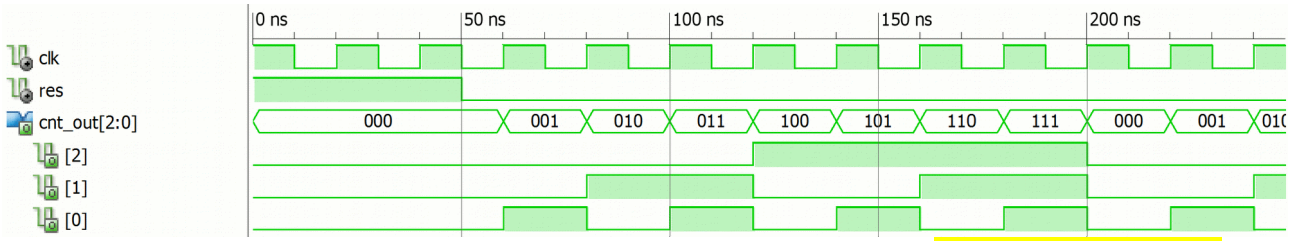
```
ARCHITECTURE rtl_ripple OF cnt IS
  SIGNAL cnt_d : std_logic_vector (N-1 DOWNTO 0);
  SIGNAL cnt_q : std_logic_vector (N-1 DOWNTO 0);
  SIGNAL clk_i : std_logic_vector (N-1 DOWNTO 0);
BEGIN
  g_rp:FOR index IN 0 TO N-1 GENERATE
    reg_cnt : PROCESS (clk_i(index), res)
    BEGIN
      IF res='1' THEN
        cnt_q(index) <= '0';
      ELSIF clk_i(index)'EVENT AND clk_i(index)='0' THEN
        cnt_q(index) <= cnt_d(index);
      END IF;
    END PROCESS reg_cnt;
  END GENERATE g_rp;

  clk_i   <= cnt_q(N-2 DOWNTO 0)&NOT(clk);
  cnt_d   <= NOT(cnt_q);
  cnt_out <= cnt_q;
END ARCHITECTURE rtl_ripple;
```

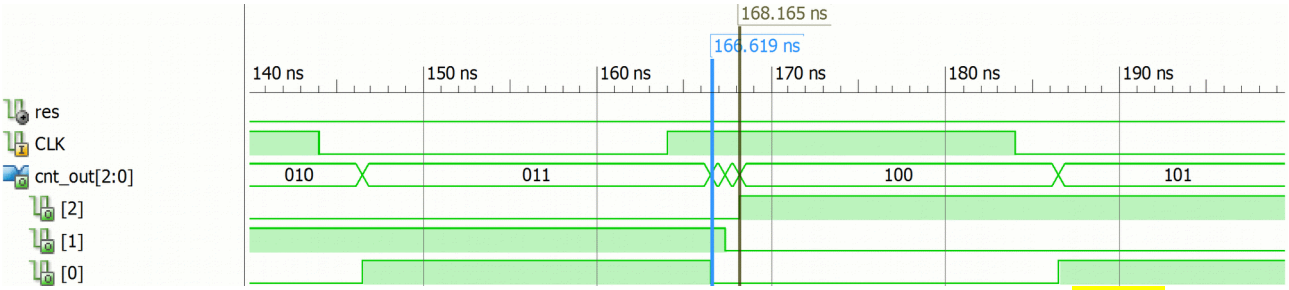


Soubor cnt_ripple_sch.png

Příklad 1: Asynchronní čítač - RTL kód a schéma.



Obrázek 1: Příklad běhu čítače, sekvence stavů 000, 001, 010, 011, 100, 101, 110, 111. Soubor cnt_ripple_rtl.png.



Obrázek 2: Příklad běhu čítače, simulace na hradlové úrovni, detail přechodu mezi stavy 011 a 100. Soubor cnt_ripple_gate.png.

3 Shrnutí a závěr

Tabulka 1 obsahuje teoretické odhady implementačních parametrů jednotlivých typů čítačů; kvantitativní srovnání parametrů jednotlivých čítačů po implementaci na programovatelném hradlovém poli xc5vlx30-3ff324 je uvedeno v tabulce 2. Z tabulky můžeme učinit následující závěry:

- **U malých čítačů** (zde příklad pro 8 stavů) téměř nehraje roli, jaký typ čítače zvolíme. Sčítačky v binárním a Grayově čítači jsou natolik optimalizované a redukované, že jejich zpoždění jsou zanedbatelná proti zpožděním na spojích v obvodu. Johnsonův čítač má proti čítači 1 z N handicap v nutnosti vložit invertor do cesty mezi výstupem druhého a vstupem nultého stavového registru čítače; tato cesta je omezující pro jeho maximální hodinovou frekvenci.
- **Pro středně velké čítače** (zde příklad pro 64 stavů) platí celkem dobře to, co jsme uvedli výše. Johnsonův čítač a čítač 1 z N mají téměř shodné maximální hodinové frekvence (minimální perioda se liší jen o 100 ps, vyšší rychlost Johnsonova čítače je zde zjevně způsobena lepším rozmístěním a propojením jeho logických prvků). Oba jsou rychlejší než binární a Grayův čítač. Mírně abnormální vyšší rychlost Grayova čítače proti binárnímu je způsobena dobrou optimalizací kombinační logiky generující příští stav a šikovnějším rozmístěním a propojením. Rozdíl mezi oběma bloky je tak také v řádu 100 ps v délce kritické cesty.
- **U velkých čítačů** (1024 stavů pro binární, Grayův, 200 pro 1 z N a 400 pro Johnsonův) se stává použití

Johnsonova čítače a čítače 1 z N nevýhodné. Zabírají příliš mnoho logických prvků, nicméně jsou oba schopny pracovat na nejvyšší hodinové frekvenci. Grayův čítač vidíme, že je nyní mnohem pomalejší, než binární díky komplexnější logické funkci pro generování příštího stavu.

- **Asynchronní čítač** je na první pohled světem sám pro sebe. Musíme rozlišit dva možné způsoby jeho užití. Užijeme-li ho pro *čítání událostí*, není při správném návrhu třeba vyžadovat jeho ustálení do příchodu další hrany na hodinovém signálu. Pak můžeme čítač ve všech případech provozovat až na frekvenci cca 1,4GHz. Budeme-li chtít asynchronní čítač užít jako náhradu běžného čítače, je nezbytné aby došlo k ustálení čítače před příchodem další hrany hodin; odpovídající perioda a frekvence hodin je uvedena v tabulce v závorce. Zde je patrný lineární nárůst zpoždění s rostoucím počtem registrů v čítači; zpoždění na registr je zhruba 0,6 ns.

Závěrem bychom čtenáře rádi upozornili na internetovou stránku [4], na které je k dispozici balíček s přílohou k článku se všemi zdrojovými kódy čítačů.

Čítač	LUTů	DFF	Délka kritické cesty
Binární	$O(\log_2 N_s)$	$\log_2 N_s$	$O(\log_2 N_s)$
Johnsonův	1	$N_s/2$	1
Grayův	$O(\log_2 N_s)$	$\log_2 N_s$	$O(\log_2 N_s)$
1 z N	0	N_s	1
LFSR	$O(1)$	$\log_2 N_s$	$O(1)$
Asynchronní	0	$\log_2 N_s$	1

Tabulka 1: Srovnání parametrů jednotlivých čítačů. Zápis O(n) znamená "řádově n", délka kritické cesty je v počtech obecných kombinačních hradel v cestě.

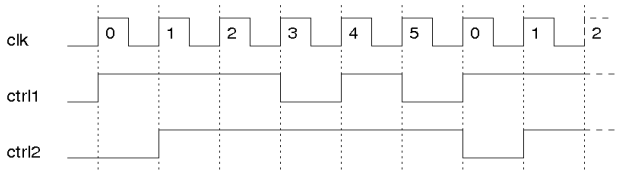
Čítač	N_s	LUT	DFF	f_{clk_max} [MHz]/T_{clk_min} [ns]
Binární	8	3	3	1470 MHz/0,68 ns
	64	6	6	934 MHz/1,07 ns
	1024	10	10	843 MHz/1,186 ns
Grayův	8	3	3	1282 MHz/0,78 ns
	64	6	6	961 MHz/1,04 ns
	1024	17	10	534 MHz/1,87 ns
Johnsonův	8	1	4	1010 MHz/0,99 ns
	64	1	32	1123 MHz/0,89 ns
	400	1	200	877 MHz/1,14 ns
1 z N	8	0	8	1351 MHz/0,74 ns
	64	0	64	1010 MHz/0,99 ns
	200	0	200	800 MHz/1,25 ns
Asynchronní	8	3	3	1470 MHz/0,68 ns (608 MHz/1,64 ns)
	64	6	6	1492 MHz/0,67 ns (284 MHz/3,52 ns)
	1024	10	10	1470 MHz/0,68 ns (158 MHz/6,31 ns)

Tabulka 2: Parametry jednotlivých typů čítačů.

4 Cvičení

1. Implementujte v prostředí ISE univerzální binární čítač z příkladu 1 v prvním dílu, použijte stejné FPGA jako v našich příkladech. Jak závisí maximální hodinová frekvence na *N*? Spusťte statickou časovou analýzu pro *N=16*, zjistěte maximální hodinovou frekvenci, na které čítač může pracovat.

2. Navrhněte generátor sekvence z obrázku 3 podle schématu uvedeného v obrázku 1 v prvním dílu s použitím binárního, Johnsonova, 1 z N, LFSR i Grayova čítače a binárního asynchronního čítače. Srovnejte velikosti implementací a dosažitelné f_{clk_max}, pozorujte v simulaci na hradlové úrovni chování výstupů *ctrl1* a *2* bloku. Pro který čítač je na nich nejvíce a pro který nejméně zákmitů?



Obrázek 3: Sekvence generovaných stavů čítačem.
Soubor cviceni.eps.

3. Implementujte asynchronní binární čítač čítající dolů.

5 Doporučená literatura

[1] Jakub Šťastný. FPGA prakticky, BEN Praha 2011.

[2] Hamming Distance. http://en.eikipedia.org/wiki/Hamming_distance

[3] Milan Křemeček. Implementace základních logických funkcí na FPGA, bakalářská práce, FPGA Laboratoř, Katedra teorie obvodů FEL ČVUT v Praze, 2006.

[4] Jakub Šťastný. FPGA návrh. http://minimizedlogic.sweb.cz/