ASICentrum
A COMPANY OF THE SWATCH GROUP

Synchronizace asynchronních signálů

ASICentrum, s.r.o.
Jakub Šťastný, Ph.D.
jakub.stastny@asicentrum.cz

Práce s asynchronními signály...

Digitální svět je jen abstrakce...

- **abstrakce odhlíží od detailů**
- **každá abstrakce potřebuje „rezervy"**
  - **musíme dodržovat „pravidla hry", pokud chceme abstrahovat od detailů**
- **podívejme se tedy na některé z „rezerv"**

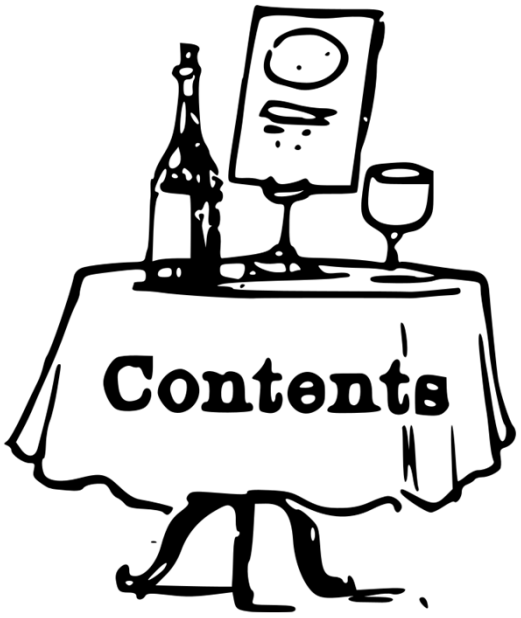
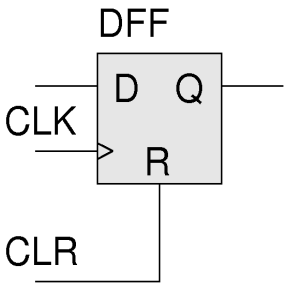


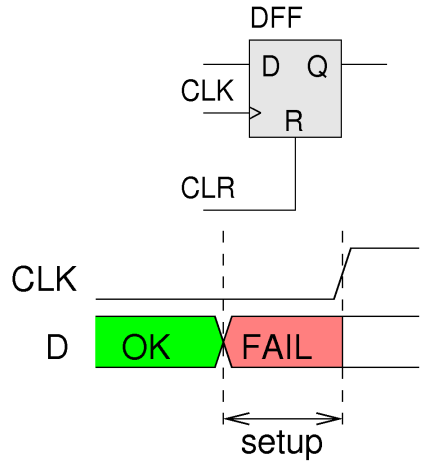
Table Of Contents

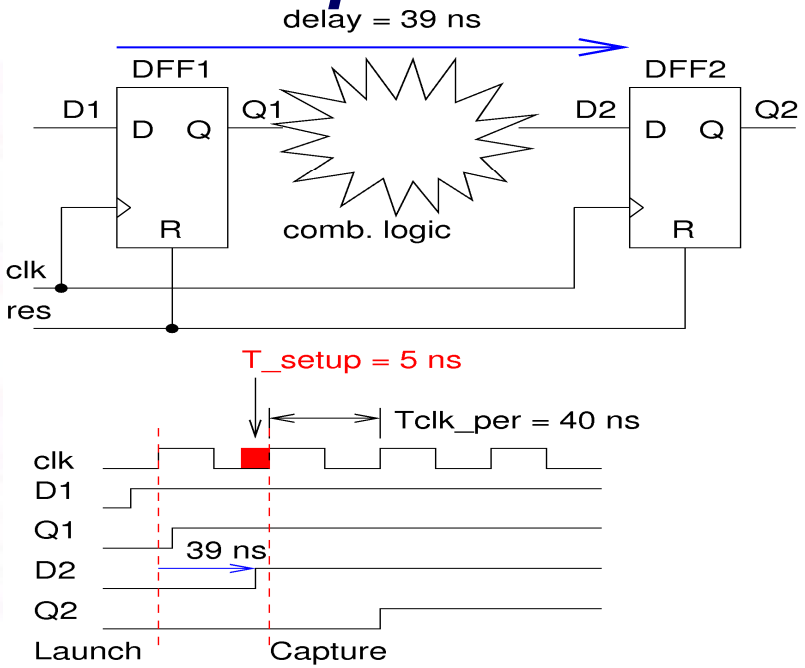
Časové parametry klopných obvodů a metastabilita



Předstih

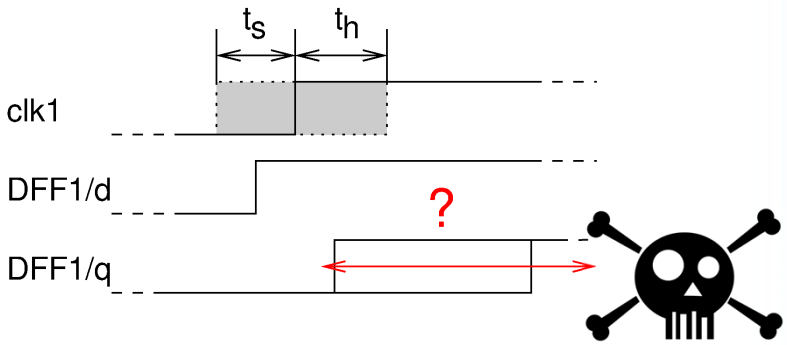
- **Setup time, T_{setup}**
- **jak vypadá?**
- **kdy si na něj musíme dát pozor?**
  - **tzv. *setup violation***





Co se stane při porušení?

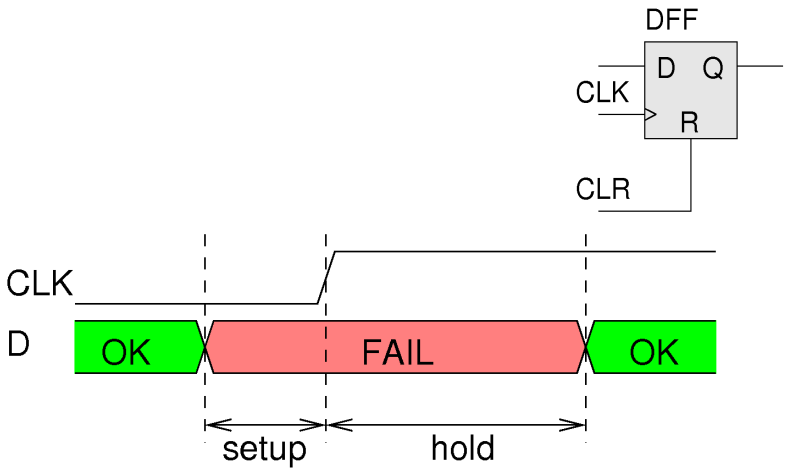
- **když se trefím do „setup-hold okna"**
- **dojde k tzv. metastabilitě**
- **registr se musí "rozhodnout", kam se překlopí**



Při porušení předstihu

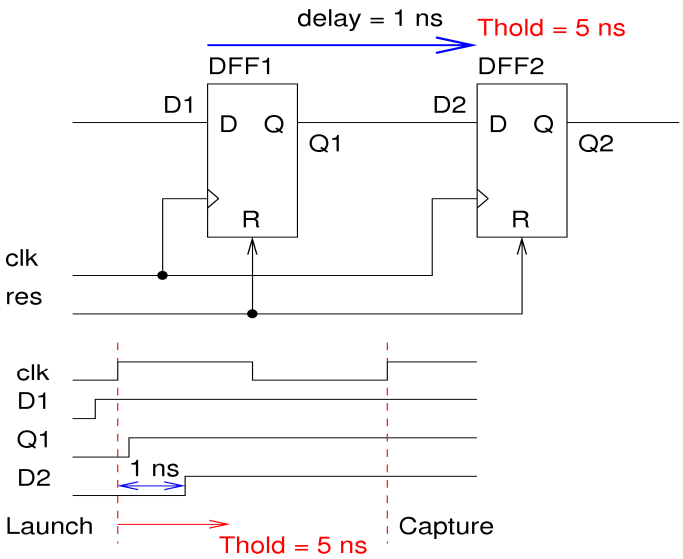
- **kombinační cesta je příliš pomalá vs. další hrana**

Přesah



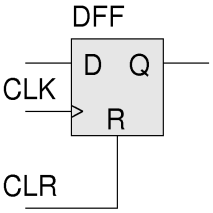
- **Hold time, T_{hold}**
- **jak vypadá, co to je?**
- **kdy si na něj musíme dát pozor?**
  - **tzv. *hold violation***
- **když je komb. cesta „moc rychlá" vs. stejná hrana**
- **jak porušení opravit v obvodu vpravo na DFF2/D2?**
- **a jak se registr bude chovat při porušení přesahu?**

Posuvný registr

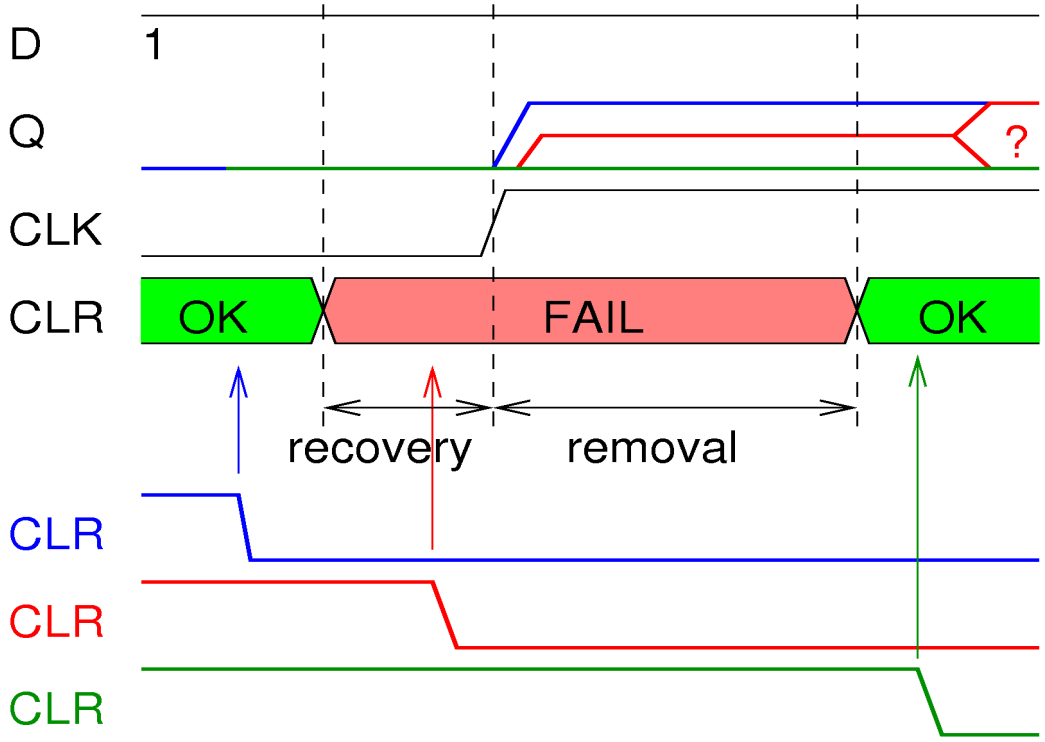


Zotavení po resetu

- **Reset recovery, Reset removal, *T_resrec, T_resrem***
- **kdy si na něj musíme dát pozor?**

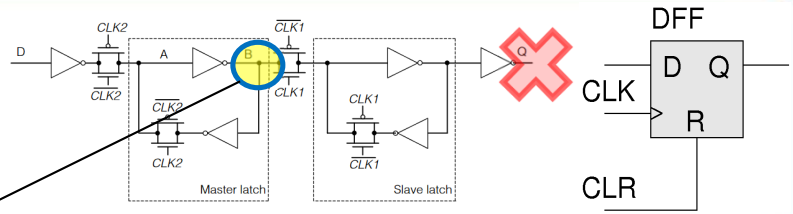


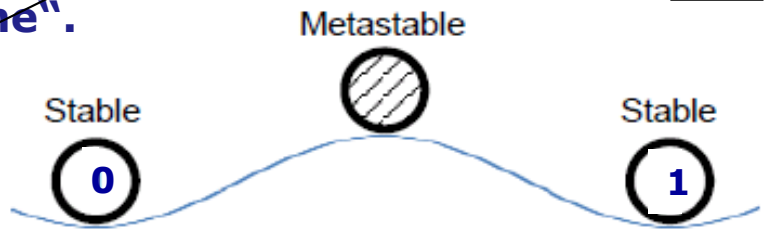
Asynchronní reset aktivní v log. 1

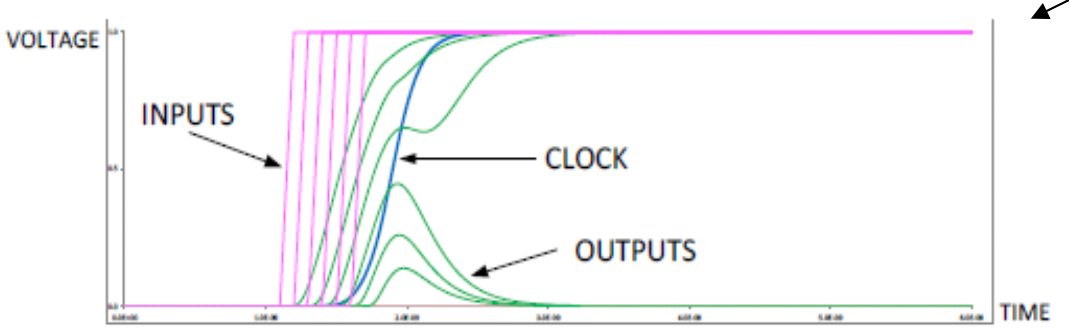


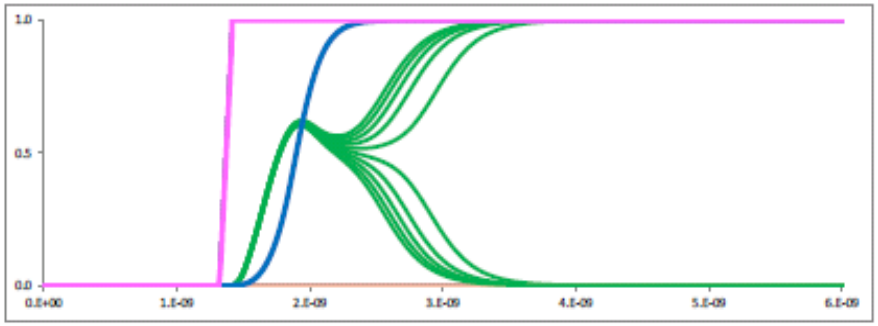
Jak vypadá metastabilita?

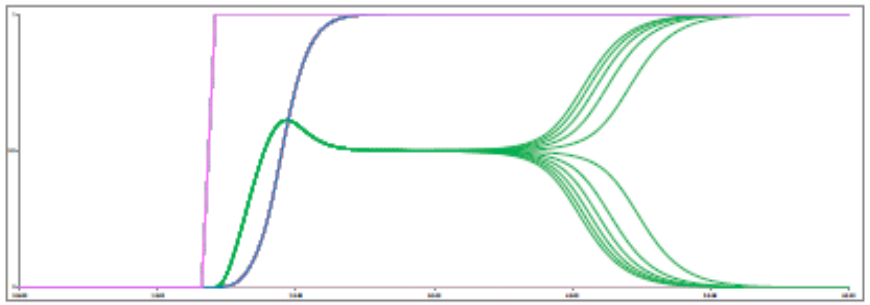
Při porušení časových parametrů registru dojde k metastabilnímu chování na výstupu registru.
Časové průběhy zachycují chování na výstupu „latche".











Empirical circuit simulations of entering metastability for the master latch in a DFF.
Charts show

- **multiple inputs D**
- **internal clock (!CLK2)**
- **multiple corresponding outputs Q (voltage *vs.* time).**

The input edge is moved in steps of 100ps, 1ps and 0.1fs in the top, middle and bottom charts, respectively .
Copied from [1]

Každá abstrakce vyžaduje „dodržení pravidel".

[1] Metastability and Synchronizers: A Tutorial. Ran Ginosar, IEEE Design & Test of Computers 28(5): 23-35 (2011)

Měření metastability (z literatury)

Článek *Chaney, T. J. and C. E. Molnar. Anomalous Behavior of Synchronizer and Arbiter Circuits. IEEE Transactions on Computers C-22 (1973): 421-422.*

Měření metastability R-S klopného obvodu (ECL logika, Motorola MC1016)

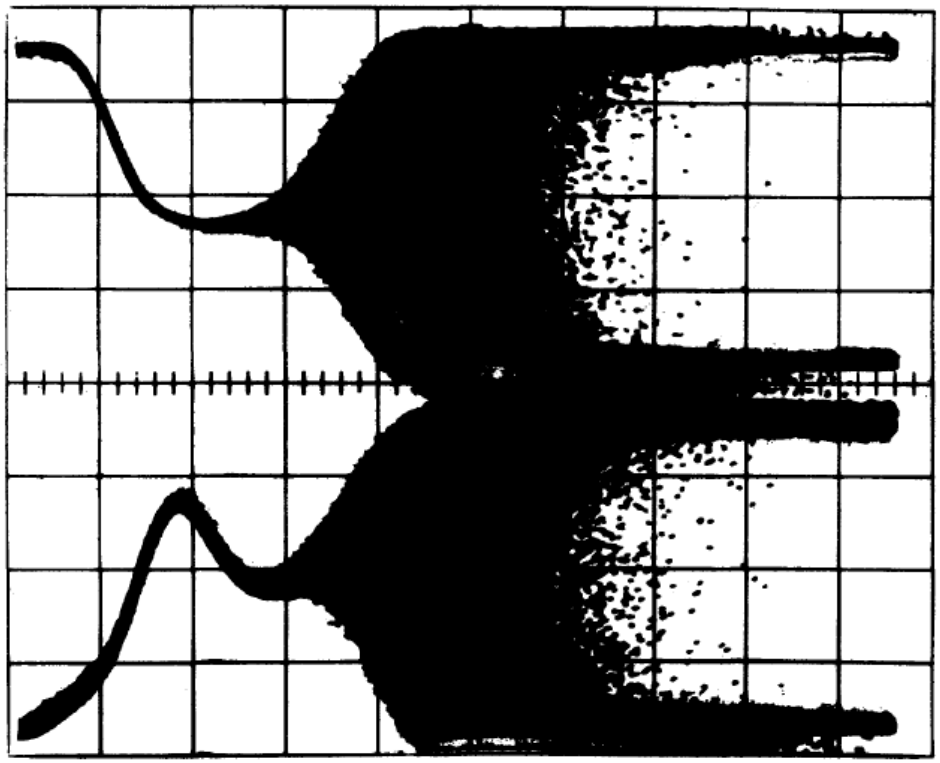


Fig. 1. Q and $\overline{Q}$ of ECL clocked R-S flip-flop with clock and data inputs changing simultaneously (5 ns/div, 0.25 V/div).

Kráceno z článku:

Discussion at a recent Workshop on Synchronizer Failures (Washington University, St. Louis, Mo.; April 27-28, 1972) has revealed that a number of computer systems are subject to significant rates of system failure that result from unreliable interactions between mutually asynchronous subsystems.

The conventional solution is to use these signals as inputs to a synchronizing element, typically a flip-flop, and to assume that the flip-flop output reaches a logically defined state within some maximum fixed time after the input occurs.

The authors have been aware for some time that this assumption is false, and that there is no fixed time interval sufficiently long to ensure that the flip-flop will, with probability one, reach a defined output state.

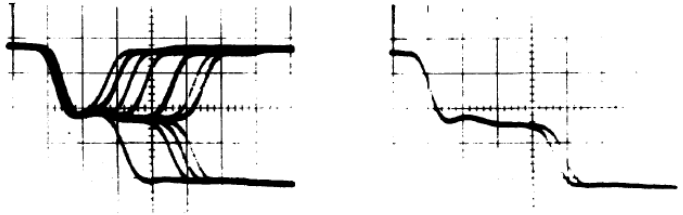


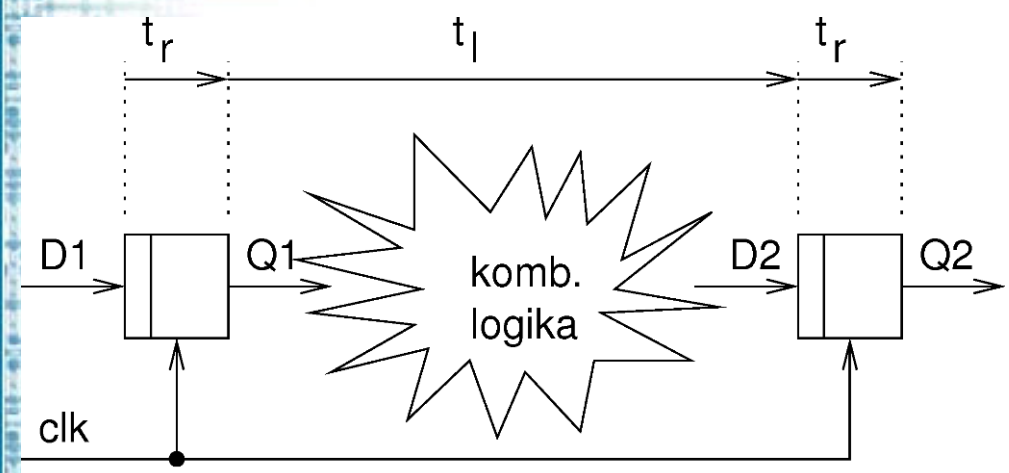
Fig. 2. Selected responses of ECL clocked R-S flip-flop to clock and data inputs changing simultaneously (10 ns/div, 0.2V/div).

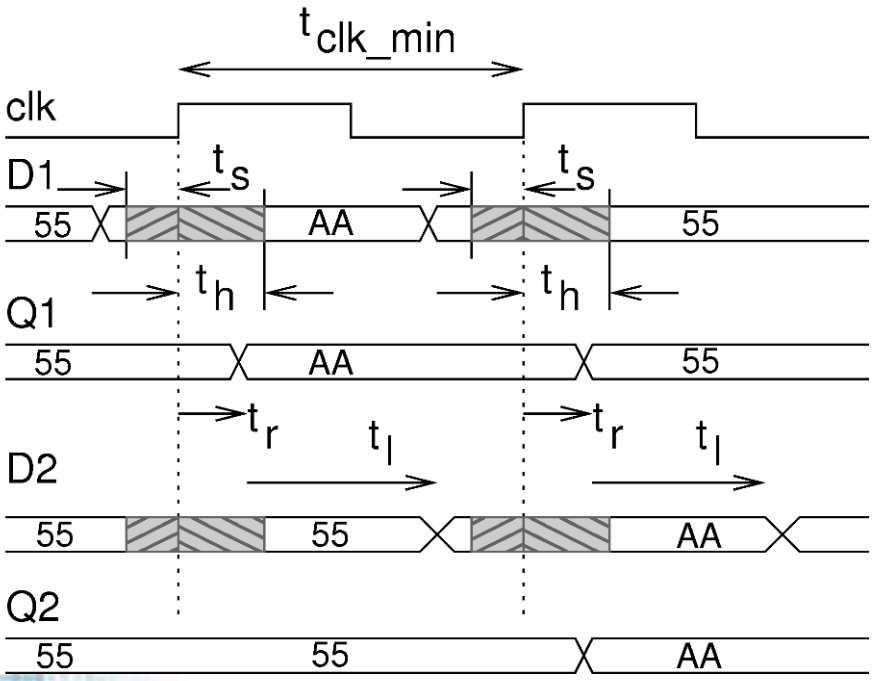
Nějaké dotazy?

Synchronní číslicový systém a hodiny

Zopakovali jsme si parametry, teď se podívejme, k čemu je to dál dobré...

Synchronní číslicový systém – jak pracuje?





Jak pracovat s předstihem a přesahem? Co musí platit, aby nedocházelo k jejich porušování? Jsme je schopni všude dodržet?

Zjednodušené vztahy mezi časovými parametry v rámci jedné hodinové domény

Minimální perioda hodin

- $t_{clk} \geq t_r + t_l + t_s$

Maximální zpoždění kombinační logiky, aby nedošlo k porušení předstihu:

- $t_l \leq t_{clk} - t_r - t_s$

Minimální zpoždění kombinační logiky, aby nedošlo k porušení přesahu:

- $t_l + t_r \geq t_h$

Při kontrole předstihu a přesahu ještě musíme uvažovat min-max zpoždění v logice, např. pro „setup check" maximální zpoždění na hradlech atd.

Synchronní obvod a více hodin

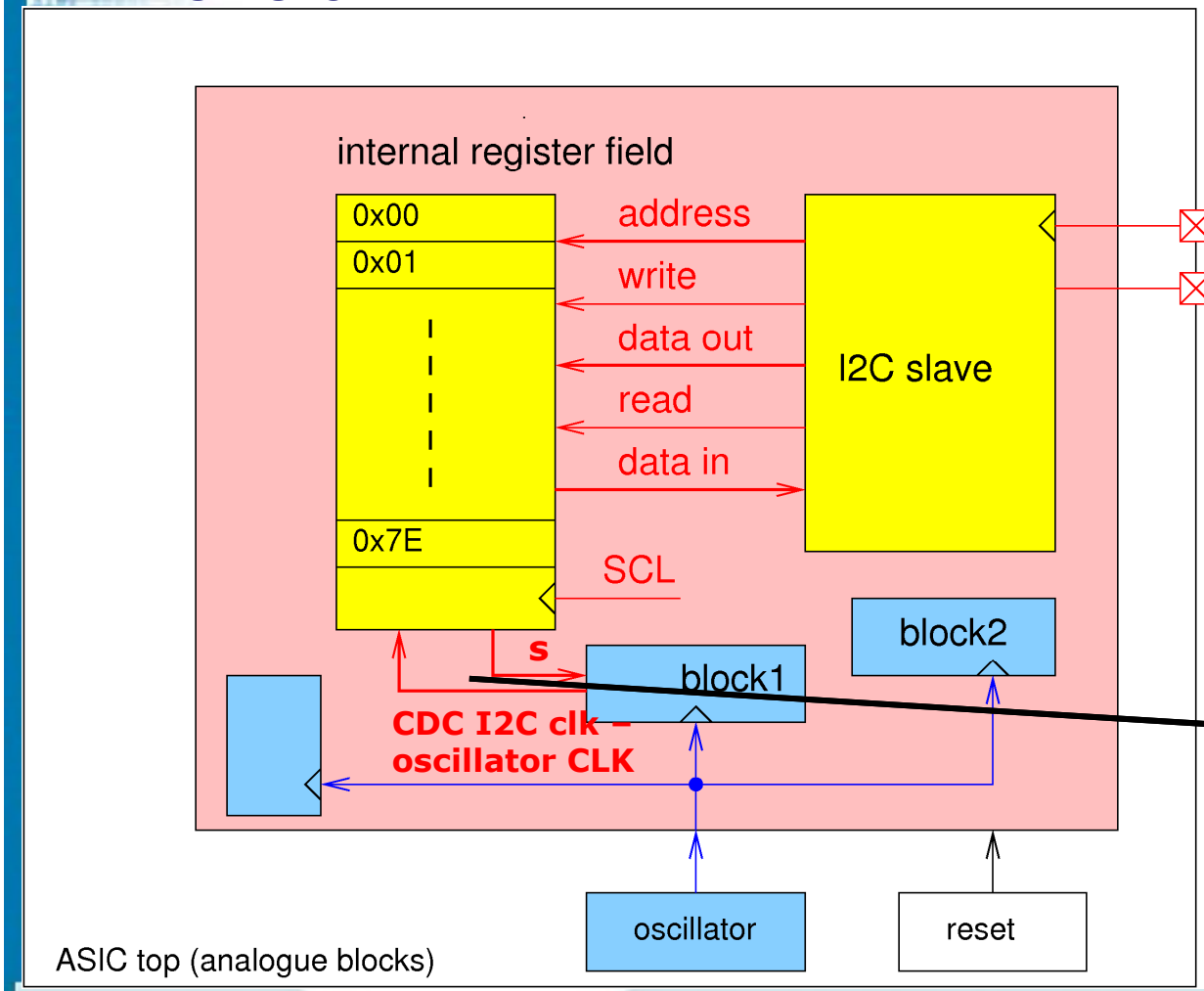
Ale jen jedny hodiny jsou vzácné...

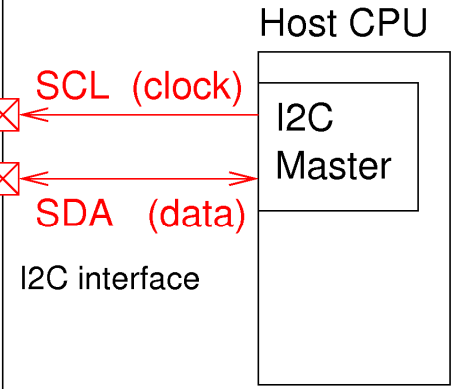
Příklad:

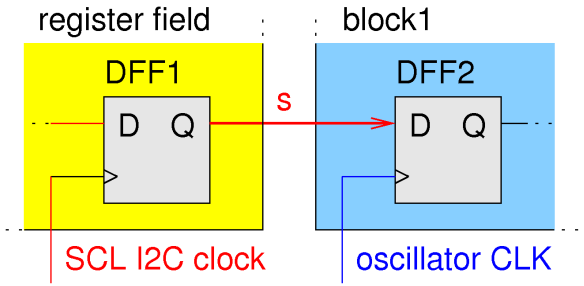
- **oscilátor běží stále**
- **SCL I2C hodiny běží jen někdy,**
- **mají proti oscilátoru náhodnou fázi a jinou frekvenci**

CDC: Clock Domain Crossing.
Červený signál s je

- **generovaný od jedněch hodin, př: SCL v I2C**
- **a vzorkovaný jinými hodinami, př: z interního oscilátoru**







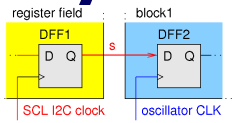
Jak tu zajistíme dodržení předstihu a přesahu?

Nějaké dotazy?

Jak to řešit?

Základní techniky pro resynchronizaci signálů přecházejících mezi doménami

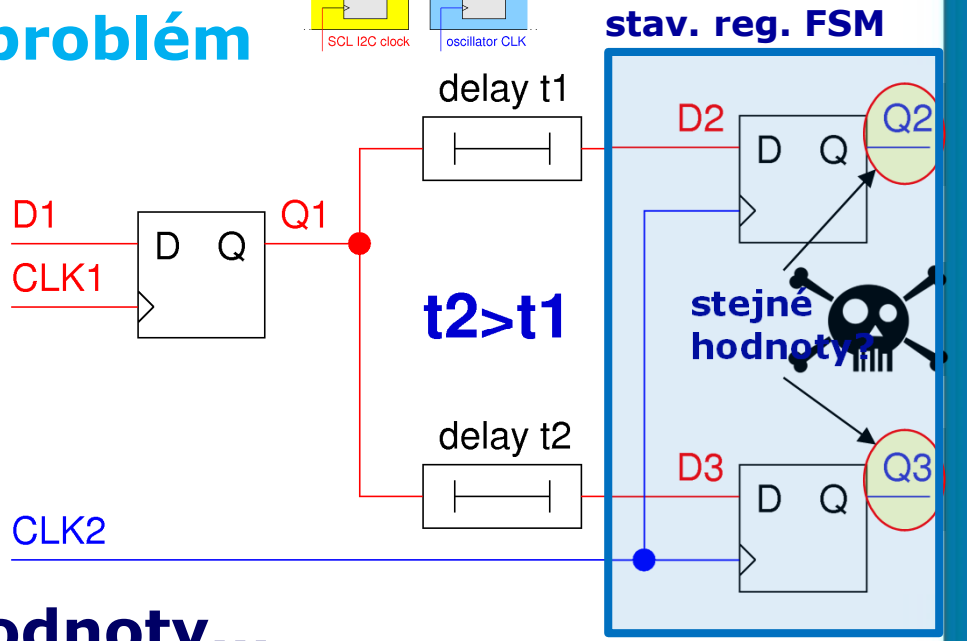
Synchronní obvod s asynchronními hodinovými doménami



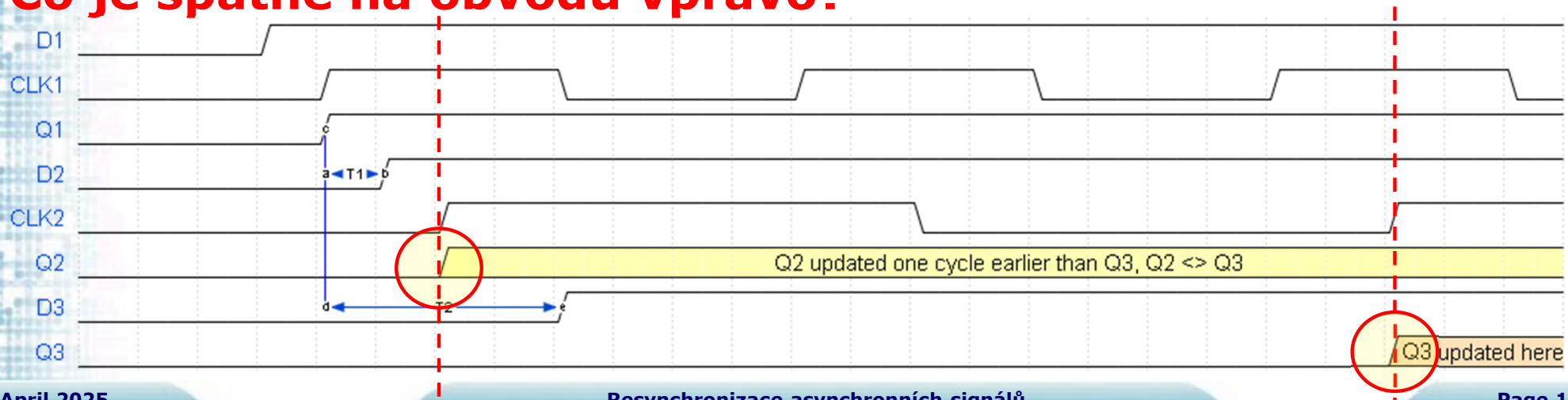
Nejobecnější případ – základní problém

CLK1 (SCL) a CLK2 (osc)

- vzájemně asynchronní
- odlišné frekvence
- odlišné fáze
- signál je zaveden „přímo“
- předpoklad: setup a hold = 0
- Q2 a Q3 by měly mít stejné hodnoty…



Co je špatně na obvodu vpravo?



Jak převést signál z jedné domény do druhé?

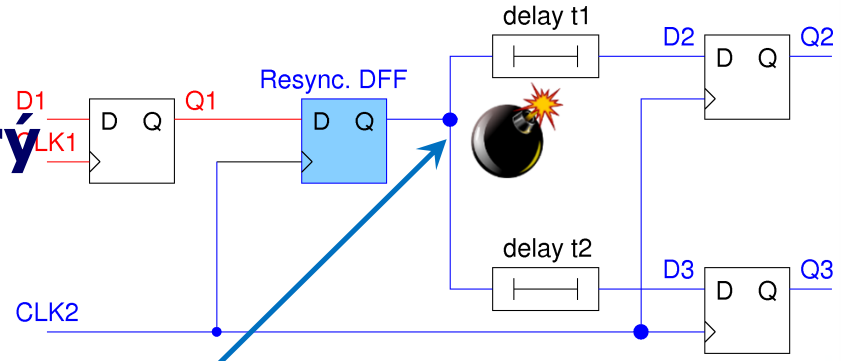
Aha! Tedy není možné asynchronní signál přímo zavést do registrů „v jádru obvodu“

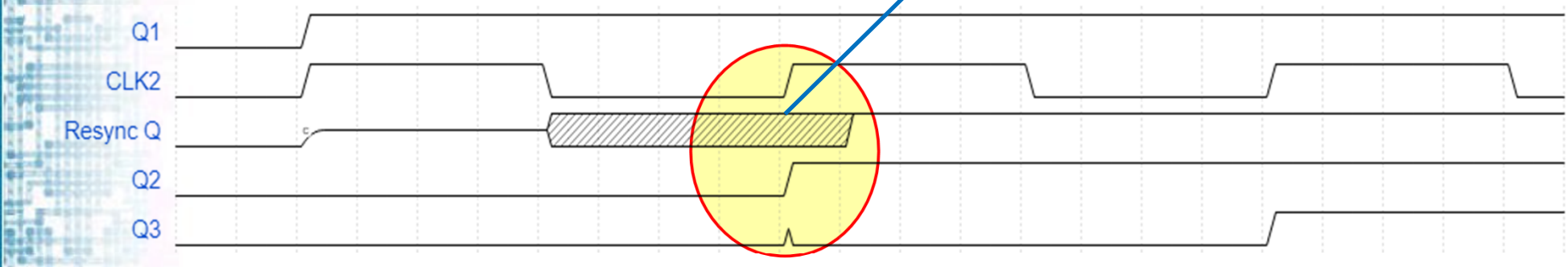
1. verze řešení:

do návrhu vložím jeden registr, který

- **„zaregistruje“ asynchronní signál**
- **... OK, a bude to stačit?**

NEBUDE! Jeden registr není dost!

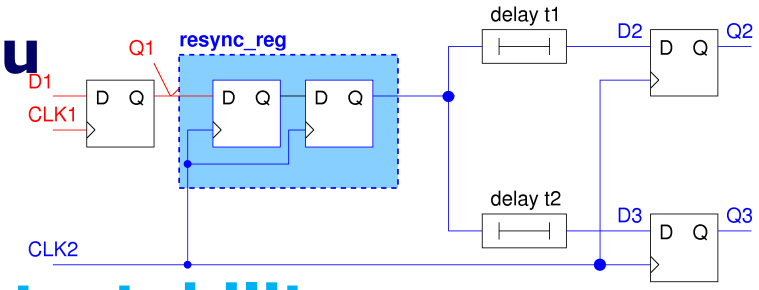




Jak převést signál z jedné domény do druhé?

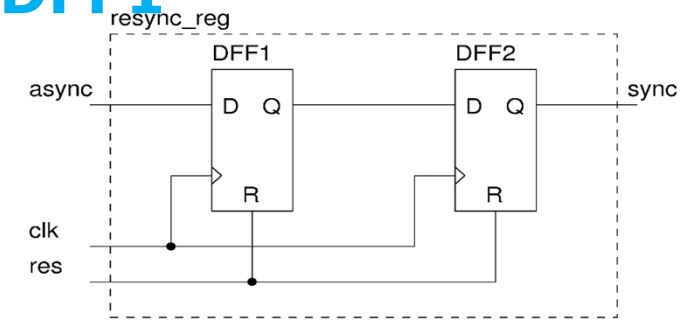
… → Koncept izolace metastability

1. do návrhu vložím jeden registr, který
   - „zaregistruje" asynchronní signál
   - počítám na něm s metastabilitou
   - a jeho výstup zavedu do
2. dalšího registru
   - na kterém pravděpodobnost metastability bude už výrazně redukovaná, protože máme o jeden cyklus více času na ustálení
   - předpokládáme, že zanedbatelná
   - odfiltruje mi zpoždění na výstupu DFF1

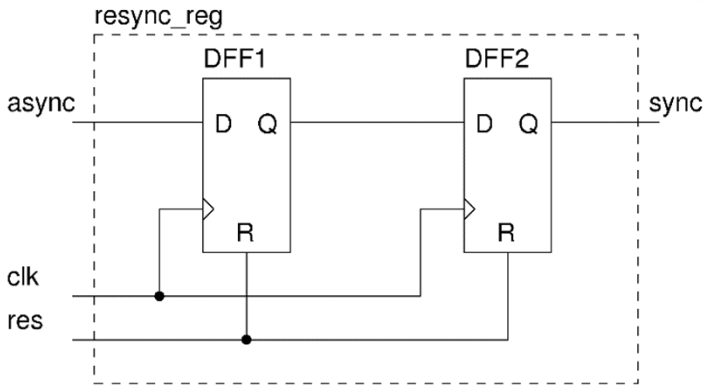


Vyhradím tak registry, na kterých „vzniknou problémy"

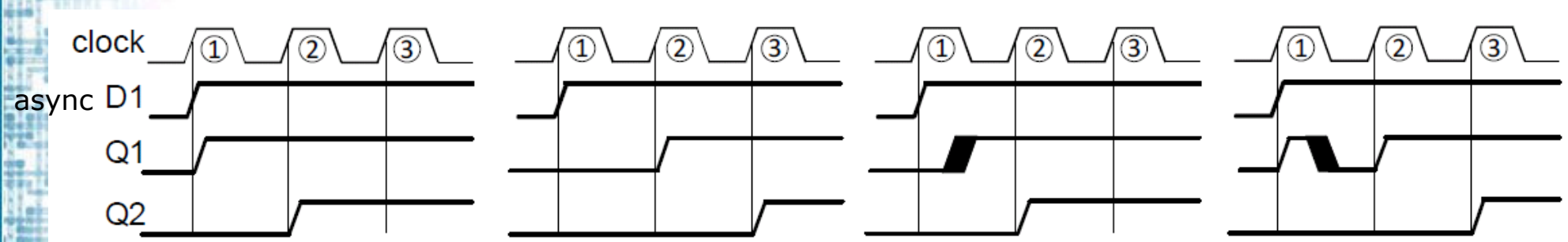
Získám základní synchronizační prvek: two flip-flop synchronizer



Jednobitový synchronizátor



Zpoždění je 2 nebo 3 hrany hodin
to může být omezující pro obvod (rychlost…)



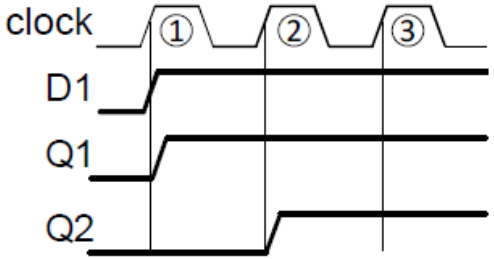
Musím ale dodržet několik podmínek:

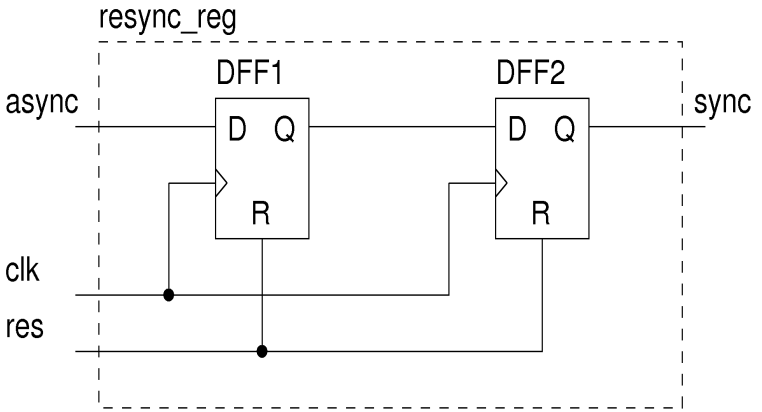
- **asynchronní signál se musí měnit „pomalu" oproti hodinám, kterými ho vzorkujeme**
- **asynchronní vstup musí být**
  - **!! bezhazardový (glitch-free) !! (namalovat na tabuli)**

All figures copied from [1].

A pravděpodobnost, že to bude fungovat?

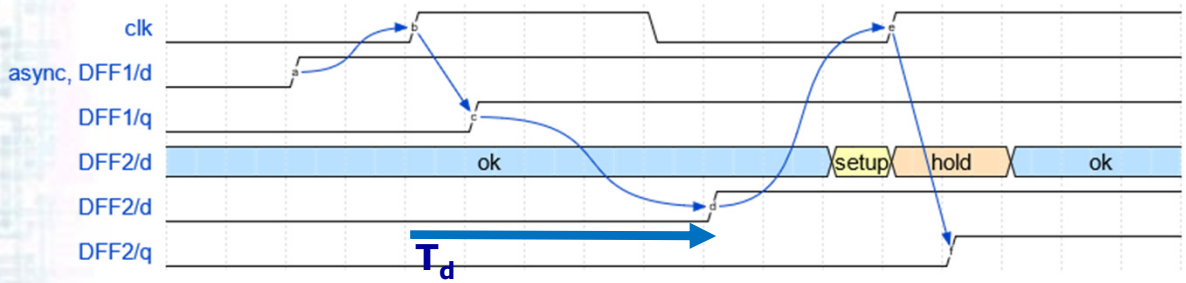
- **Předpoklad správné funkce:**
  - DFF2 už nesmí být metastabilní
  - nesmím porušit hold time na DFF2
- Ustálení je stochastické
  - Existuje nenulová pravděpodobnost, že DFF2 bude metastabilní
- Za zvláštních podmínek
  - nemusí stačit dva registry
  - může jich být potřeba více
- Zajímá nás, jak často může synchronizátor selhat
- To určuje parameter MTBF
  - *Mean Time Between Failures*
  - větší hodnota je lepší

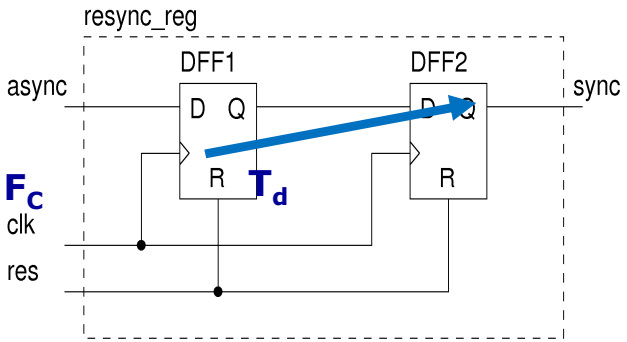




Odhad MTBF, na čem závisí?

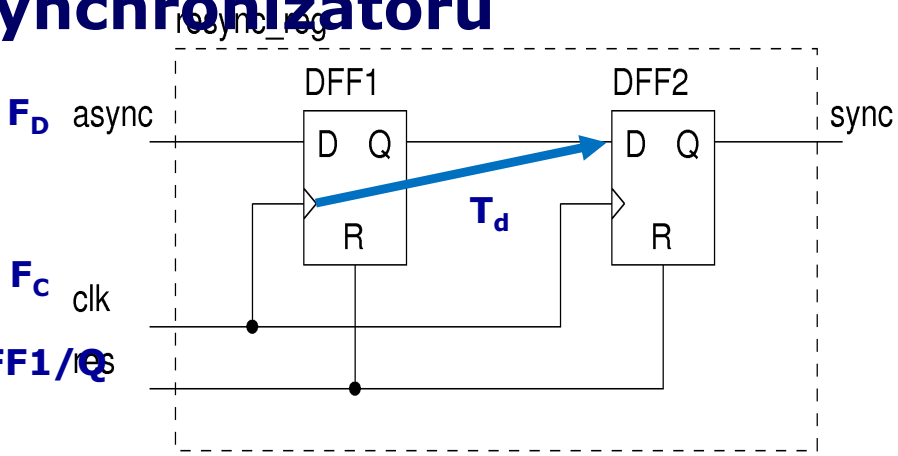
- na pracovní frekvenci logiky (F_C)
  - čím vyšší frekvence, tím kratší čas na ustálení, situace je horší
  - čím vyšší frekvence, tím je menší MTBF
- na četnosti změn asynchronního signálu (Fd)
  - čím více hran, které chceme synchronizovat, tím větší riziko
  - čím vyšší frekvence, tím je menší MTBF
- na zpoždění registrů a zpoždění na spoji mezi registry (T_d)
  - čím vyšší zpoždění, tím kratší čas na ustálení, situace je horší
  - čím vyšší zpoždění, tím je menší MTBF
  - vliv teploty a napájecího napětí
  - mezi registry nesmí být žádná hradla, jen spoj
    - nejvýše buffer (hold time fix)
  - a registry musí být „blízko sebe"
- na metastabilním okně registru
  - čím větší je „zakázaný" interval kolem hodin
    - tím větší je pravděpodobnost, že asynchronní vstup vyvolá metastabilitu
    - ...tím je menší MTBF





Odhad spolehlivosti synchronizátoru

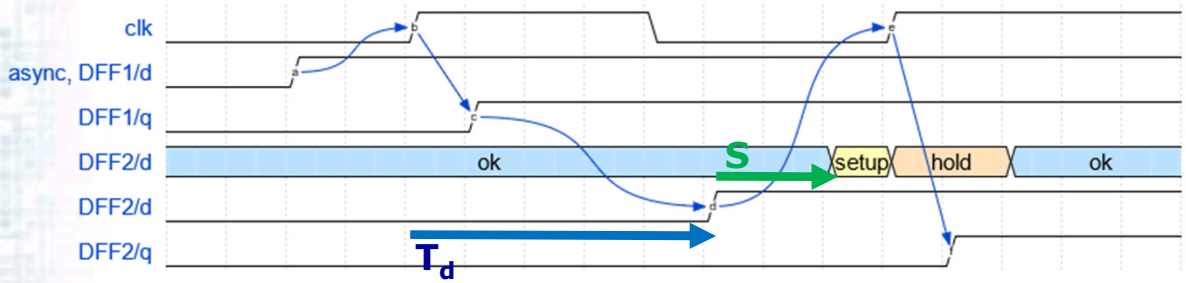
$$MTBF = \frac{e^{S/\tau}}{T_W F_C F_D}$$



Parametry ve vzorečku:

- *S* = maximální dostupný čas na ustálení
  - časový interval po kterém DFF2 navzorkuje DFF1/Q
  - zde perioda hodin mínus
    zpoždění DFF1,
    zpoždění spoje mezi DFF1/Q a DFF2/D a
    předstih DFF2.
  - $S = 1/F_c - T_d$
- τ = časová konstanta DFF
  - lze změřit, případně odhadnout, viz např. [2]
- *Tw* = metastabilní okno
  - šířka časového okna kolem aktivní hrany hodin,
  - kdy změna datového vstupu způsobí metastabilitu
- *Fc* = pracovní frekvence hodin
- *Fd* = frekvence asynchronních změn na asynchronním vstupu (vs. frekvence signálu!)
  - předpoklad: rovnoměrné rozložení událostí v čase

Vhodné hodnoty MTBF jsou o mnoho řádů větší, než je životnost produktu.



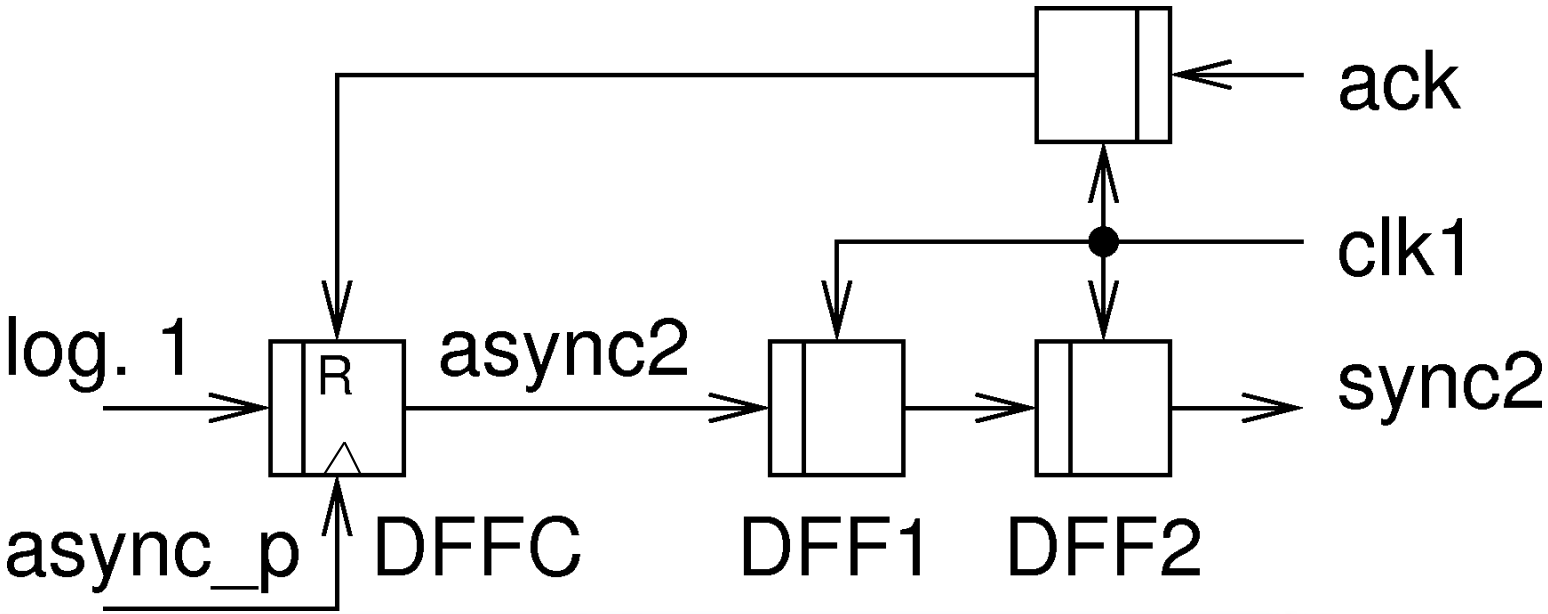
… Existují i jiné techniky řešení: pausible clocks …

Nějaké dotazy?

Další techniky pro ošetření CDC
… variace na „jednobitové téma“

Co když potřebujeme zachytit rychlé jevy?

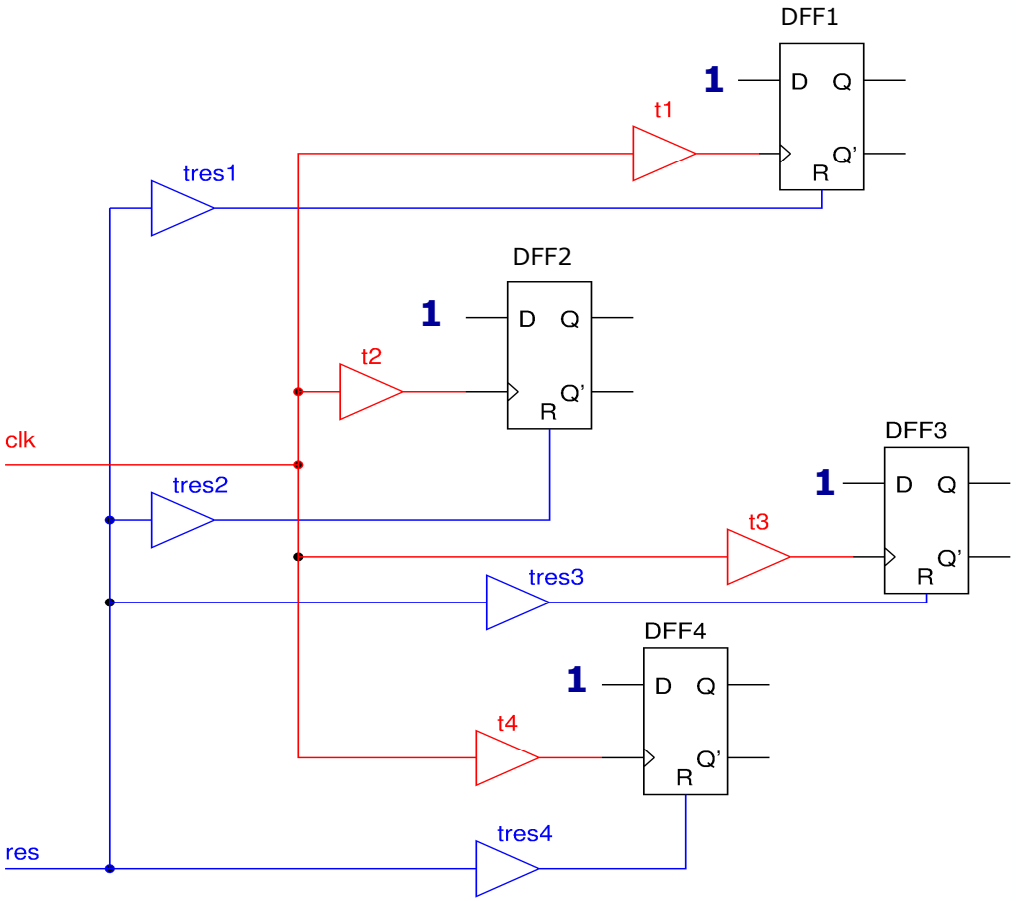
- Dvouregistrový synchronizátor
  - si poradí jen s pomalými signály…
- co když potřebujeme sledovat výskyt „krátkých pulzů?“
  - signál async_p je zaveden do hodin DFFC
  - DFFC reaguje na náběžnou hranu *async_p* zapsáním log. „1“
- … princip si namalujeme na tabuli ☺
- omezení frekvence pulzů, minimální šířka pulzu



Práce s asynchronním resetem

Asynchronní reset
- **rozvod resetu v obvodu**
- **reset recovery, removal**

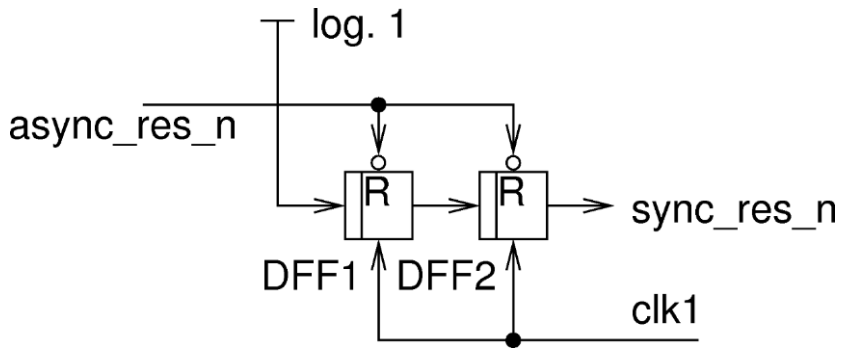
Všechny registry je třeba resetovat současně - reset uvolnit „naráz".



Synchronizace asynchronního resetu

Říkali jsme, že asynchronní reset nemůžeme „jen tak uvolnit"...

- **... potřebujeme dedikovaný obvod na synchronizaci resetu**
- **první problém: reset recovery/removal**
- **druhý problém: oblak registrů resetovaných jedním resetem**
- **Praktický obvod, který**
  - **jednu hranu propustí a**
  - **druhou synchronizuje**



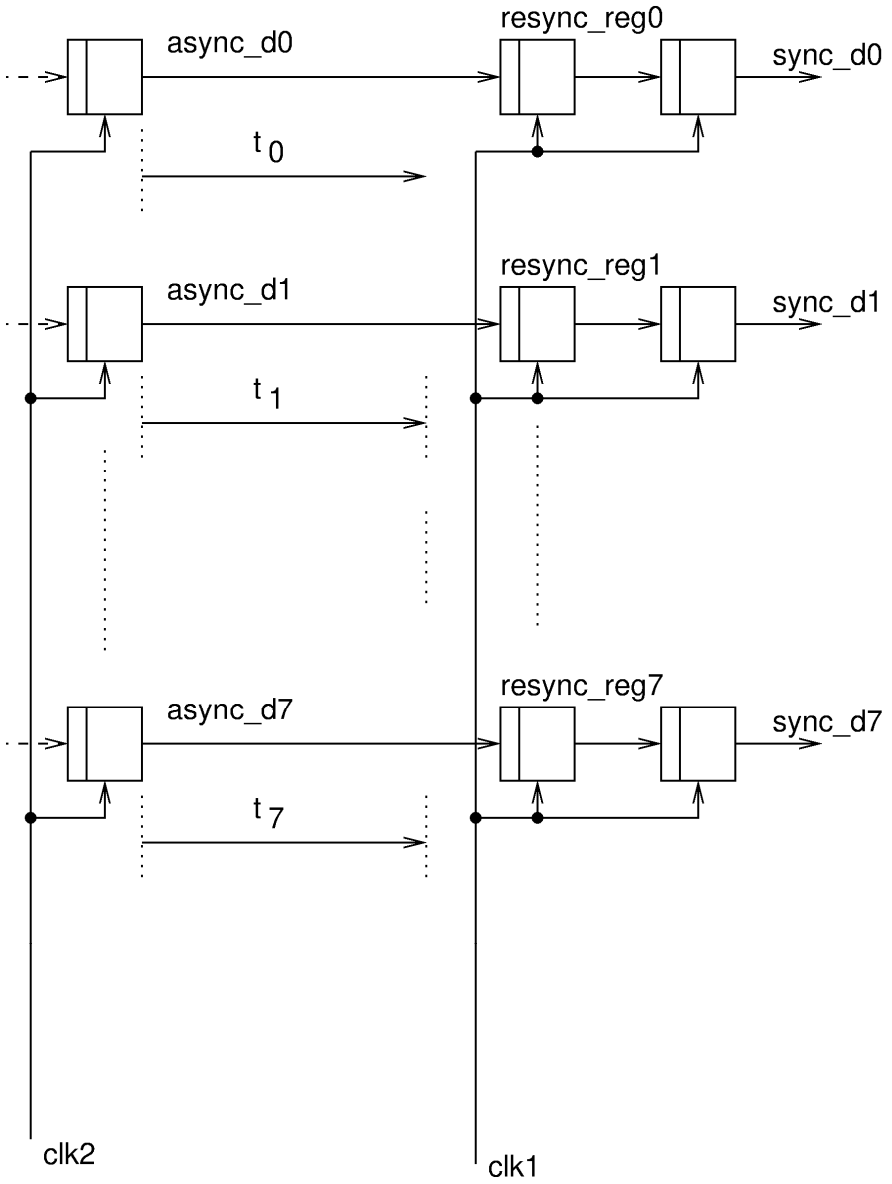
Nějaké dotazy?

Další techniky pro ošetření CDC
… synchronizace vícebitových signálů

Synchronizace vícebitové sběrnice

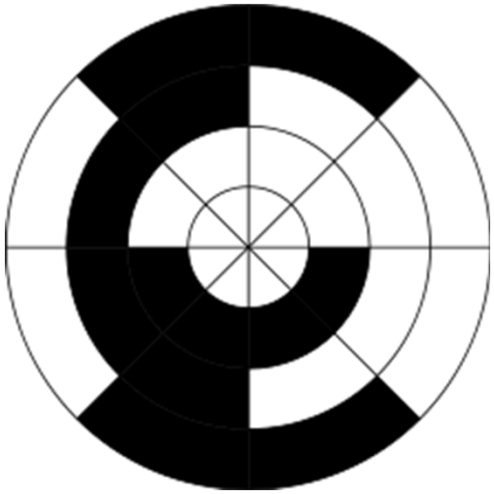
Naivní řešení

- **replikace synchronizeru**
- **nebude správně fungovat**
- **tušíte proč?**

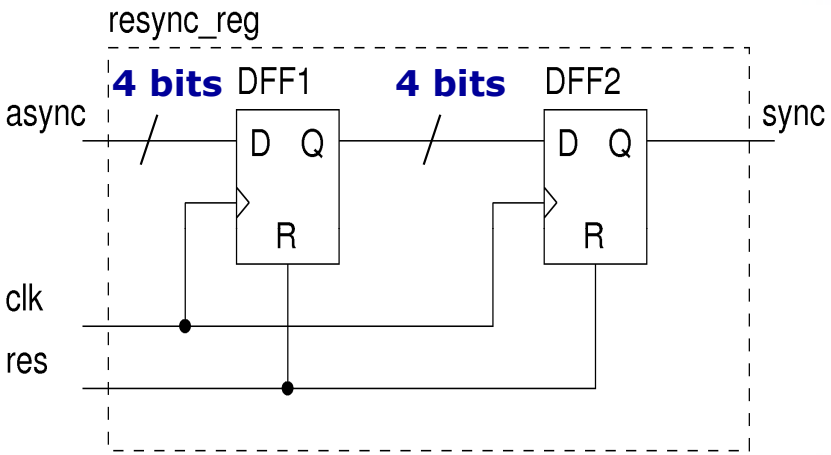


Synchronizace vícebitové sběrnice: sekvence slov

- řešení 1: Grayův kód
  - pro „sekvenční data“
  - přenos hodnoty čítače mezi doménami



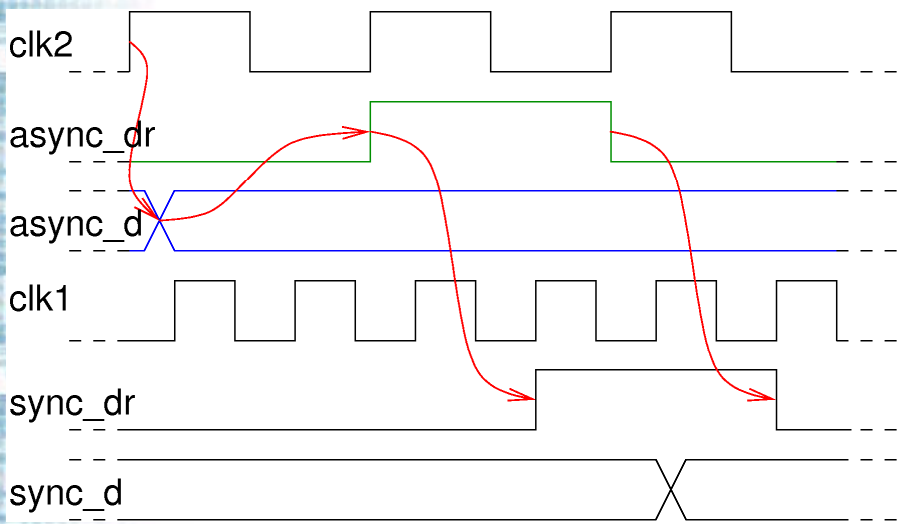
Source: Wikipedia
https://en.wikipedia.org/wiki/Gray_code

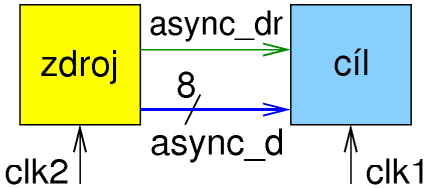


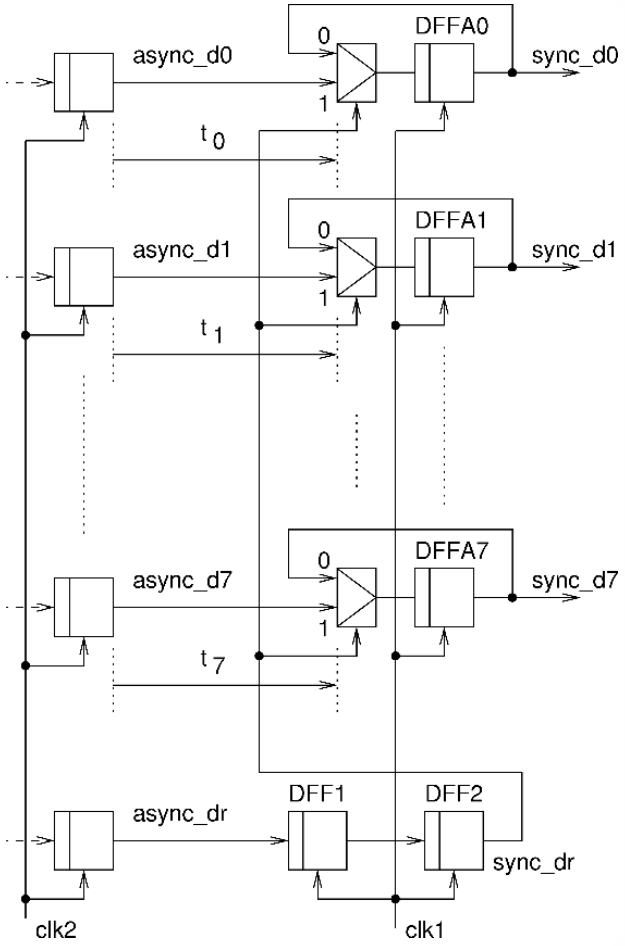
Synchronizace vícebitové sběrnice

Pokud nemůžeme použít Grayův kód...

- tzv. recirkulační synchronizér
- *recirculation synchronizer*
- zdroj musí být pořád „dost pomalý", abychom stihli správně navzorkovat indikaci platnosti dat





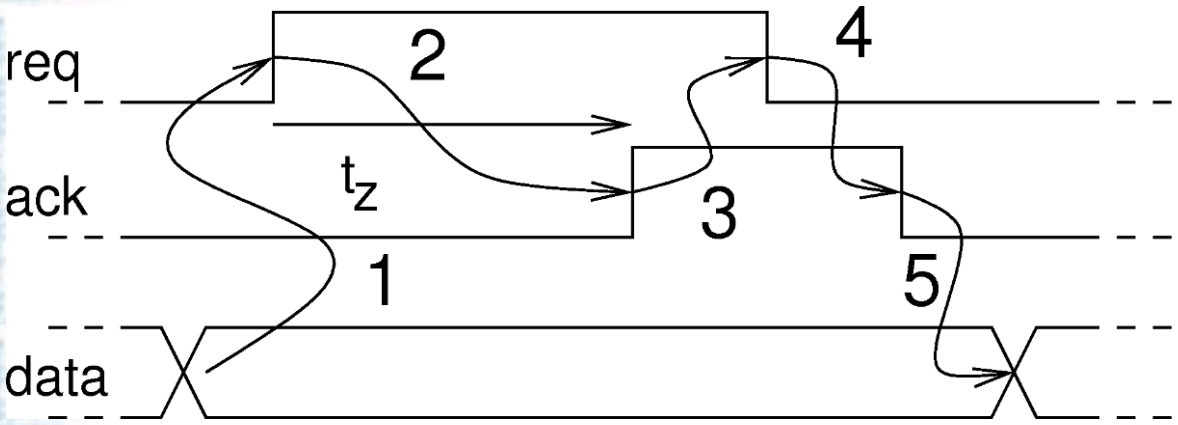


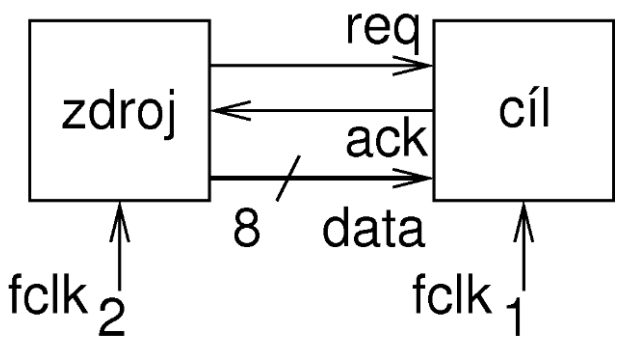
Synchronizace vícebitové sběrnice: handshake

Co když „cíl" nestíhá?

Zdroj a cíl/příjemce dat mohou mít velmi rozdílné rychlosti zpracování dat

- **řešení korespondenčním protokolem, „handshake"**
- **plně vázaný handshake**
- **oba bloky se vzájemně přizpůsobí**
- **nevýhoda: velká režie protokolu**
- **dá se zjednodušit**





Nějaké dotazy?

Asynchronní FIFO

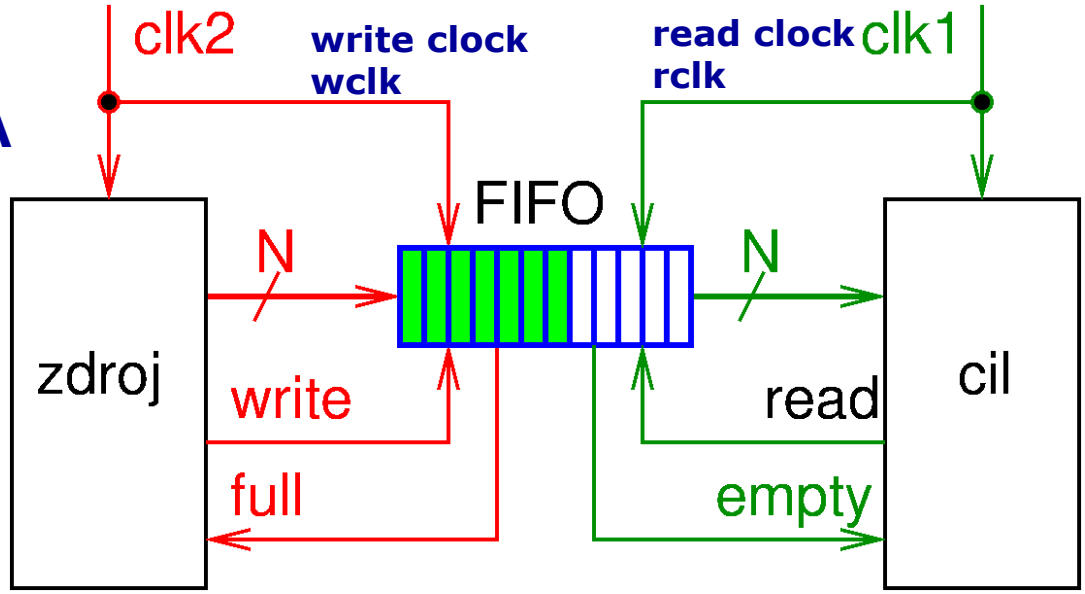
Řešení, které umožňuje
přenést libovolné množství dat
v libovolném kódu mezi doménami

Použití FIFO paměti

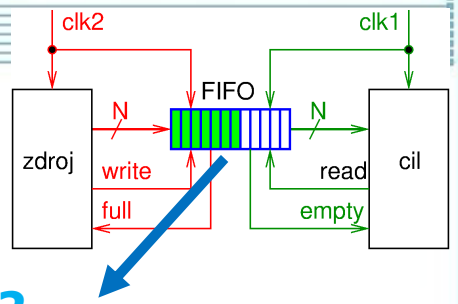
Zdroj
- zapisuje datová slova do FIFA
- dokud není plná
  - full = '1'
  - můžu zapisovat

Cíl
- čte datová slova z FIFA
- dokud není prázdná
  - empty = '1'
  - můžu číst

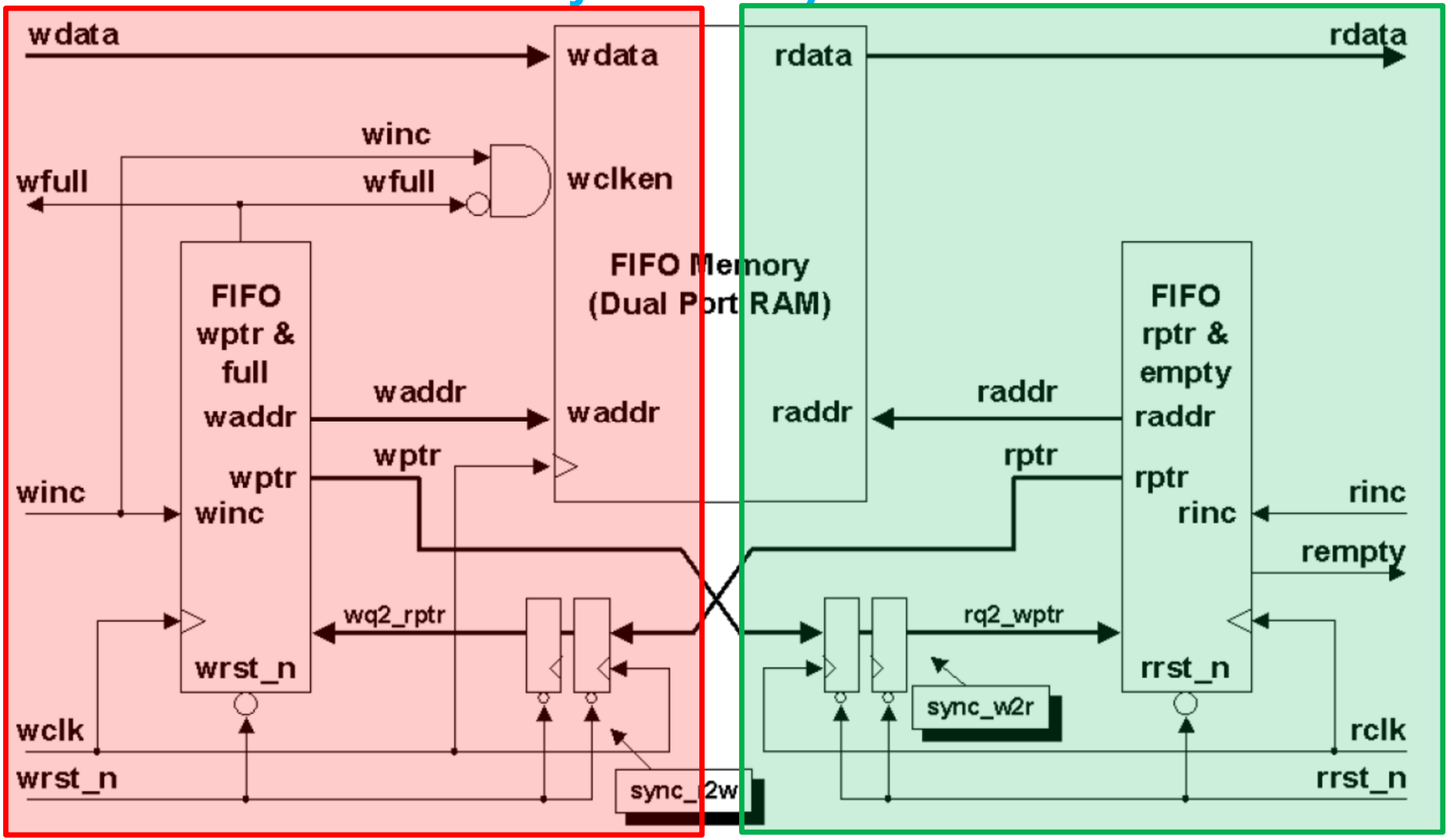
clk2 a clk1 mohou být vzájemně zcela asynchronní



Asynchronní FIFO



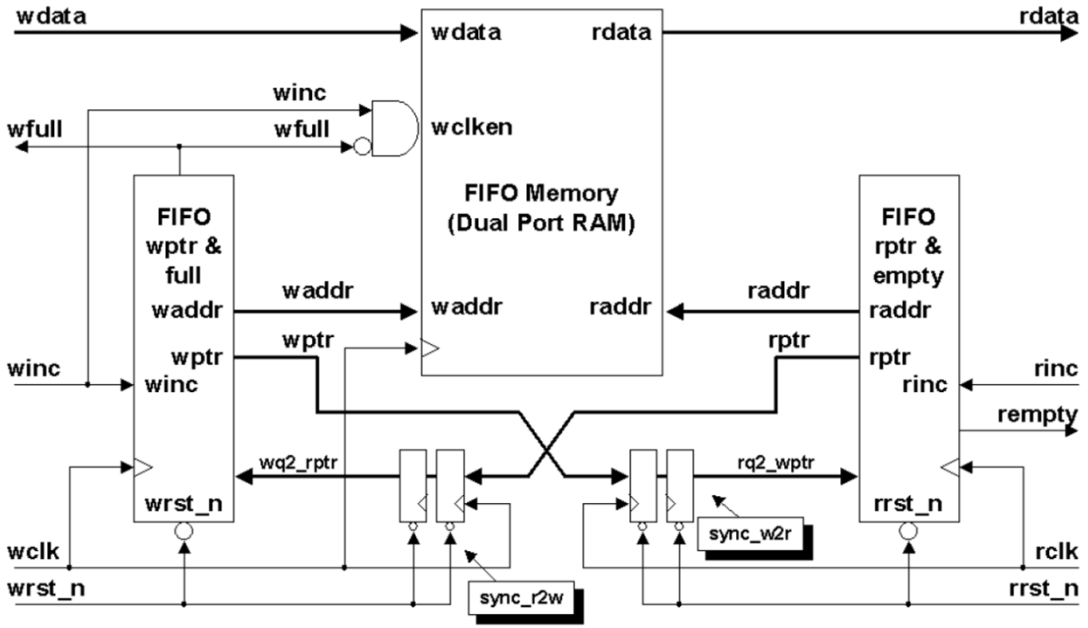
Princip fronty..., základní objasnění činnosti
Proč má každá strana svůj vlastní asynchronní reset?



Source: Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford Cummings, http://www.sunburst-design.com/

Indikace full/empty pro FIFO o 8 slovech

- Čtecí ukazatel RPtr
  - ukazuje na další slovo v paměti, které budeme číst (nejdéle zapsané)
- Zápisový ukazatel WPtr
  - ukazuje na další slovo v paměti, které budeme zapisovat (nejdéle prázdné)
- Po resetu bloku
  - RPtr = 0
  - WPtr = 0
  - FIFO indikuje stav „prázdné“

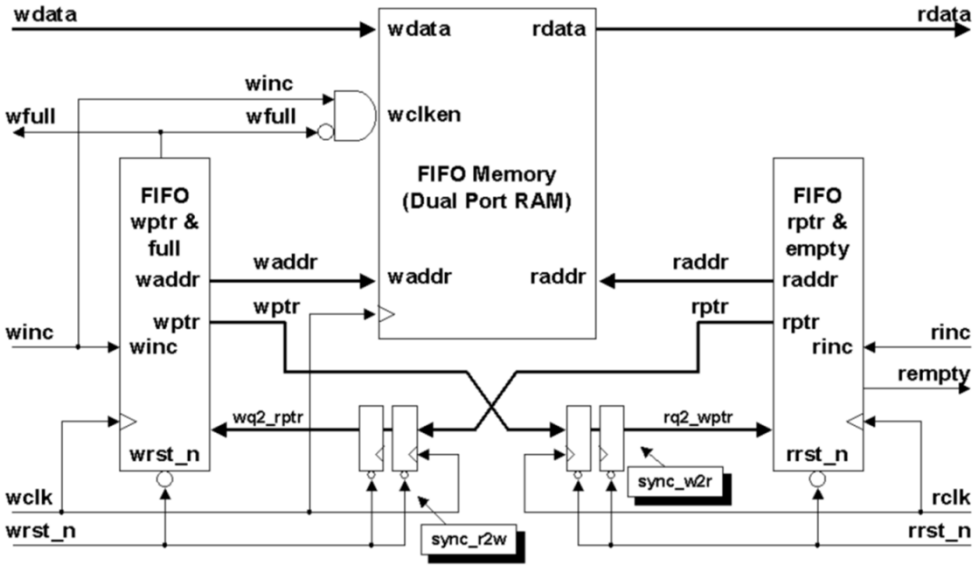


Source: Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford Cummings, http://www.sunburst-design.com/

Jak fungují ukazatele? (FIFO o 8 slovech)

WPtr (Bin)	RPtr (Bin)	Komentář
000	000	Write
001	000	Write
010	000	Write
…	…	Writes
111	000	Write
000	**000**	**Full, Read**
000	001	Read
000	010	Read
…	…	Reads
000	111	Read
000	**000**	**Empty, Write**
001	000	Write
010	000	Read
011	001	Read
011	**011**	**Empty**

- Po zápisu prvního slova
  - WPtr = 1
  - RPtr = 0
  - FIFO není ani prázdné, ani plné
- Když je plné po zápisu všech slov
  - opět WPtr = 0
  - RPtr = 0
- Kdy je FIFO prázdné?
  - když RPtr = WPtr

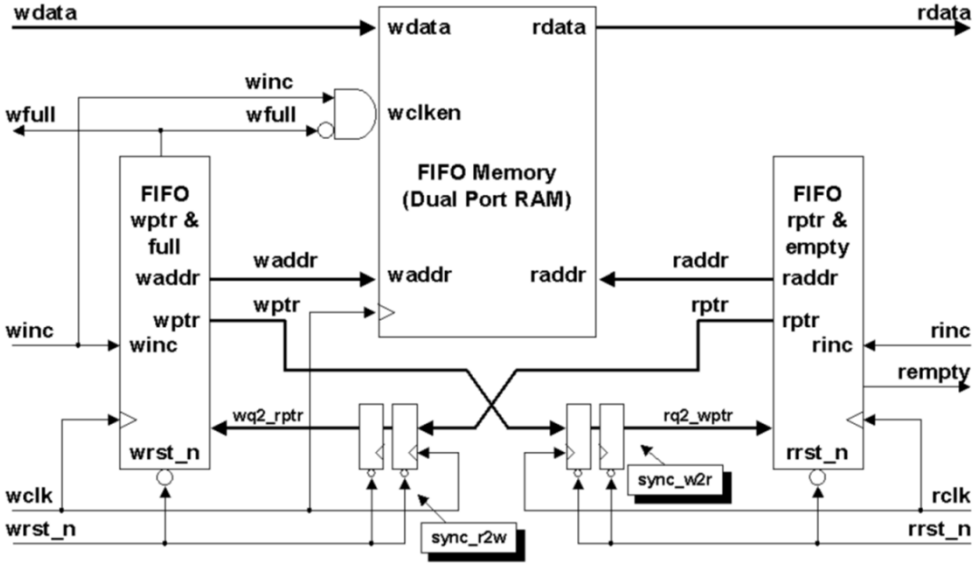


Source: Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford Cummings, http://www.sunburst-design.com/

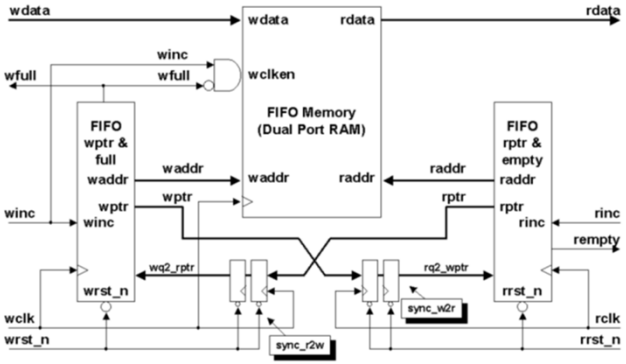
Jak fungují ukazatele + full/empty?

WPtr (Bin)	WPtr (Gray)	RPtr (Bin)	RPtr (Gray)	Komentář
000	0000	000	0000	Write
001	0001	000	0000	Write
010	0011	000	0000	Write
…	…	…	…	Writes
111	0100	000	0000	Write
000	**1100**	**000**	**0000**	**Full, Read**
000	1100	001	0001	Read
000	1100	010	0011	Read
…	…	…	…	Reads
000	1100	111	0100	Read
000	**1100**	**000**	**1100**	**Empty, Write**
001	1101	000	1100	Write
010	1111	000	1100	Read
011	1111	001	1101	Read
011	**1111**	**011**	**1111**	**Empty**

- **Kdy je FIFO prázdné?**
  - **když RPtr = WPtr**
- **Kdy je FIFO plné?**
  - **když RPtr = WPtr**
- **Jak to tedy rozlišit?**

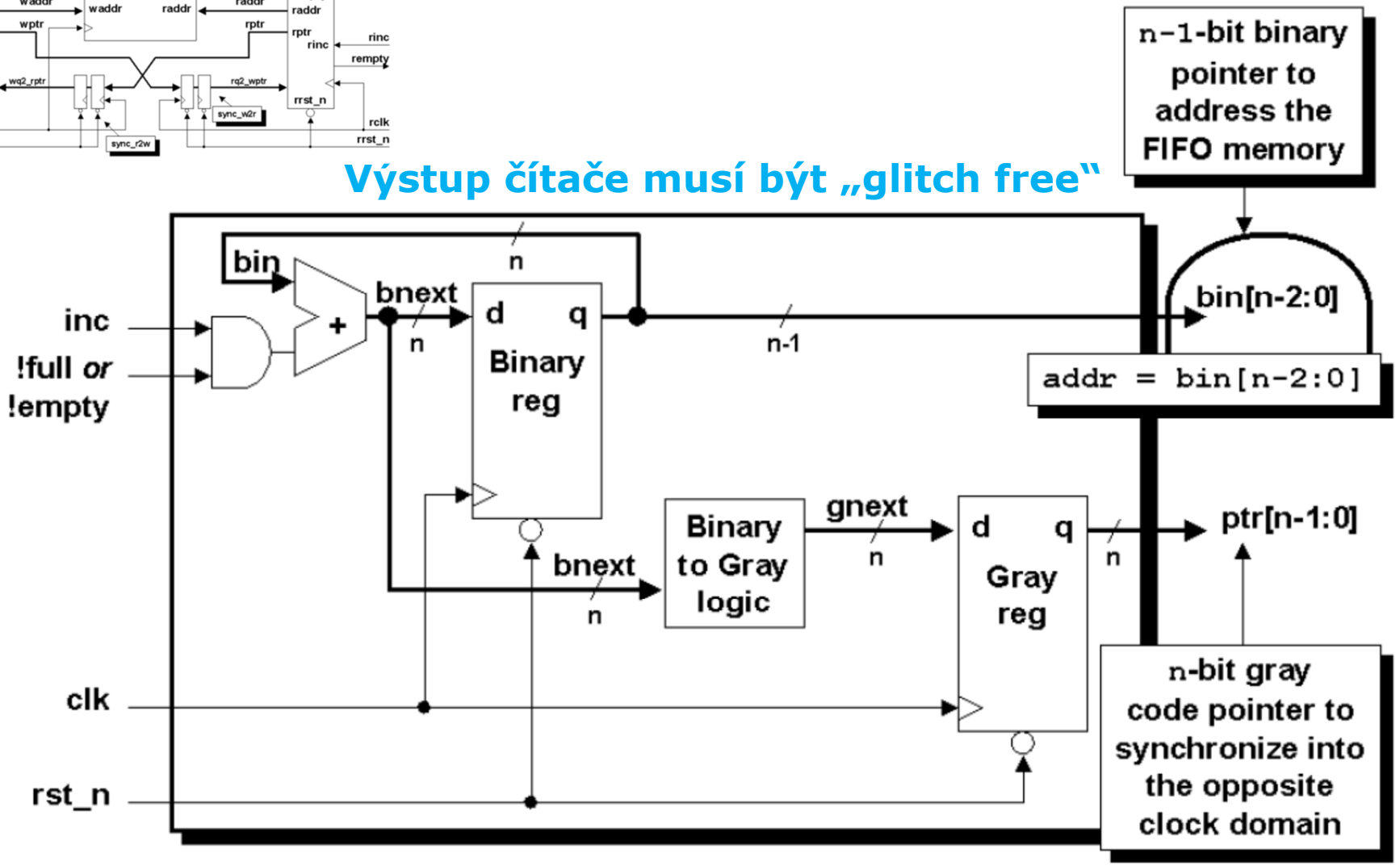


Source: Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford Cummings, http://www.sunburst-design.com/



Čítač v Grayově kódu

Výstup čítače musí být „glitch free"



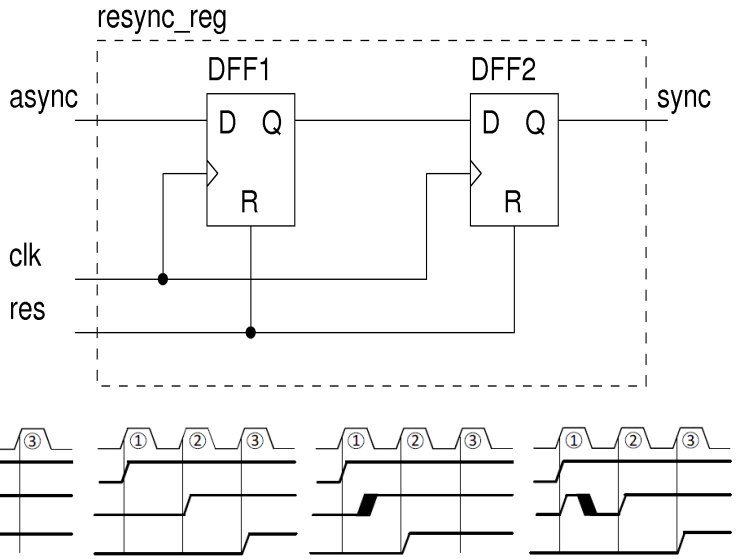
Source: Simulation and Synthesis Techniques for Asynchronous FIFO Design, Clifford Cummings, http://www.sunburst-design.com/

Nějaké dotazy?

Správná praxe pro RTL návrh

RTL návrh synchronizátoru: naivní přístup

```
sync_reg_proc: PROCESS(res, clk)
BEGIN
  IF (res = '1')  THEN
    clk_sync_reg1 <= '0';
    clk_sync_reg2 <= '0';
  ELSIF (clk = '1' AND clk'EVENT) THEN
      clk_sync_reg1 <= async;
      clk_sync_reg2 <= clk_sync_reg1;
  END IF;
END PROCESS sync_reg_proc;
```



- Takhle je to jednoduché…
- Hlavní nevýhody
  - zpoždění buňkou je vždy konstantní
  - synchronizátor nelze snadno „rozpoznat"
- Co bychom rádi…
  - reálné zpoždění v buňce (2 nebo 3 hrany hodin)
  - synchronizátor ve vlastní entitě → detekovatelnost
  - sémantická informace nesená strukturou návrhu

Řešení je dedikovaná buňka pro synchronizaci

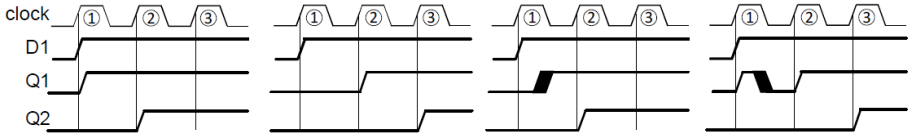
Příklad rozhraní:

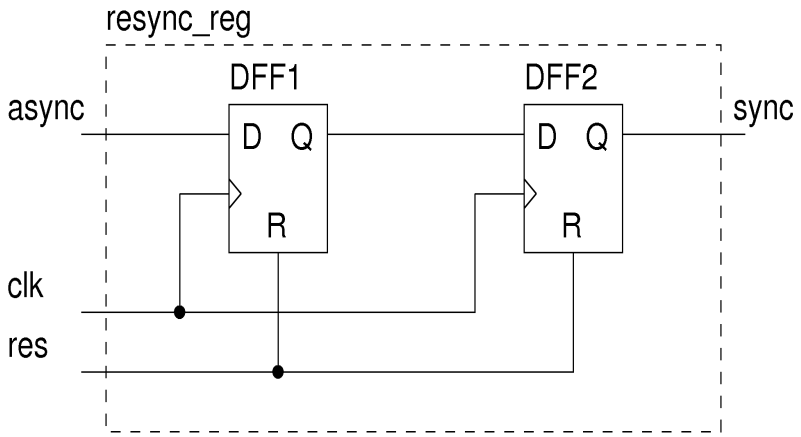
```
COMPONENT resync_reg
  GENERIC (
    prob_of_sync : real       := CRESYNC_PROB;
    -- probability of synchronization by the third edge
  );

  PORT (
    ck     : IN  std_logic;
    res    : IN  std_logic;

    async  : IN  std_logic;
    sync   : OUT std_logic
  );
END COMPONENT resync_reg;
```

Emulujeme toto chování:





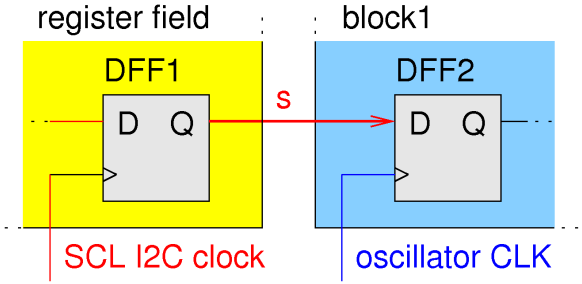
Určitou komplikací je implementace bloku tak, aby byl současně randomizovaný a syntetizovatelný.
Zkuste si to za domácí úkol.
Nápověda:

```
-- pragma synthesis_off
```

https://insights.sigasi.com/tech/list-known-vhdl-metacomment-pragmas/

Správná praxe pro RTL návrh

- největší riziko je **zapomenout na synchronizátor**
- pojmenovávejte signály podle zdrojových hodin
  - příklad:
    - signál *s*
    - generovaný z registru na hodinách *SCK*
    - se bude jmenovat *sck_s*



- při použití v registru hned vidíte, že máte problém
- v jednom bloku se pokuste mít registry jen na jedny hodiny
- používejte synchronizační buňku
  - pak rychle vidíte v návrhu co je a co není synchronizátor
  - snažší „timingchecksoff"
- … nástroje pro formální verifikaci CDC (a RDC) …

Malá omluva: Co jsem vám zatajil?

Ne všechno se mi sem vešlo, některé věci jsem v zájmu přehlednosti taktně zamlčel.

Pro ty zvídavější, kteří by si to chtěli sami promyslet následuje jejich seznam:

- **Další časové parametry DFF: clock pulse minimal width, reset pulse minimal width**
- **Měření metastability**
- **MTBF výpočet – jak stanovit parametry pro vzorec?**
- **další techniky předávání datového slova přes CDC**
  - **například „clock stopping"**
- **RTL – existují i jiné techniky identifikace synchronizátorů**
- **Jak psát správně constraints (SDC)….**
- **… a ještě pár dalších drobností**

Kdyby to mělo být všechno, byli bychom tu mnohem déle!

Citace, doporučené články

odkazy na literaturu o návrhu, další zdroje:
http://www.minimizedlogic.com

[1] Metastability and Synchronizers: A Tutorial. Ran Ginosar, IEEE Design & Test of Computers 28(5): 23-35 (2011)
http://webee.technion.ac.il/~ran/papers/Metastability%20and%20Synchronizers.posted.pdf
[2] Synchronization and Arbitration in Digital Systems. David J. Kinniment, Wiley 2007
[3] Characterization of a Flip-Flop Metastability Measurement Method. Antonio Cantoni, et al. IEEE Transactions on Circuits and Systems, Vol. 54, No. 5, May 2007
http://www.researchgate.net/publication/3451549_Characterization_of_a_Flip-Flop_Metastability_Measurement_Method
[4] Fourteen ways how to fool your synchronizer. Ran Ginosar, Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03)
http://www.researchgate.net/publication/4015785_Fourteen_ways_to_fool_your_synchronizer

Citace, doporučené články

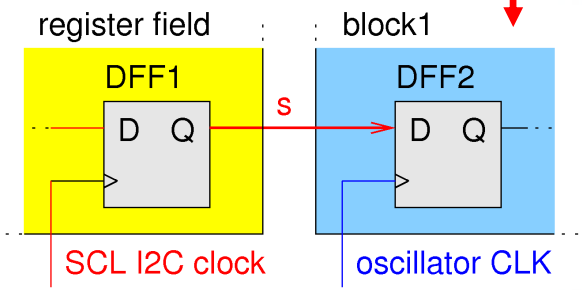
[5] Clifford Cummings. Synchronous Resets? Asynchronous Resets? I am so confused! How will I ever know which to use? SNUG 2003, Boston.
[6] Clifford Cummings. Asynchronous & Synchronous Reset Design Techniques - Part Deux. SNUG 2002. San Jose
[7] Clifford Cummings. Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs (clock naming...)
[8] Clifford Cummings. Simulation and Synthesis Techniques for Asynchronous FIFO Design
Všechny články viz http://www.sunburst-design.com/papers/
[9] Pluháček A: Návrh logiky počítačů, skriptum ČVUT 1995 (více o handshake)
[10] C. Dike and E. Burton, "Miller and noise effects in a synchronizing flip-flop," in IEEE Journal of Solid-State Circuits, vol. 34, no. 6, pp. 849-855, June 1999.
https://ieeexplore.ieee.org/document/766819
[11] Xilinx, Metastable Recovery in Virtex-II Pro FPGAs. XAPP094 (v3.0) February 10, 2005.

Použité zkratky

- ASIC - Application Specific Integrated Circuit
- CDC - Clock Domain Crossing
- DFF - D Flip-Flop
- DUT - Design Under Test
- I2C - Inter-Integrated Circuit bus
- LSB - Least Significant Bit
- MSB - Most Significant Bit
- MTBF - Mean Time Between Failures
- RDC - Reset Domain Crossing
- RTL - Register Transfer Level

Několik pojmů...

- **Hodinová doména** je část obvodu, jejíž registry jsou buzeny jen jedním hodinovým signálem.
- **V rámci jedné domény** jsou na všech registrech „zarovnané" hrany hodin
- O **mezidoménovém přechodu** (**clock domain crossing**) pak hovoříme tehdy, když signál buzený z registru v jedné doméně zavedeme na vstup registru v jiné hodinové doméně („z jiných hodin").
- **Synchronní domény** - signály buzené z jedné domény se stihnou vždy „ustálit před příchodem nejbližší události" na hodinovém signálu v druhé doméně. (není zcela přesně, ale to teď nevadí...)



Vzájemné vztahy hodinových domén

- Mesochronní domény (mesochronnous clocks)
  - hodinové signály mají stejnou frekvenci a konstantní fázový posuv
  - většinou není potřeba žádná zvláštní synchronizace
  - často se př. signál generuje z náběžné hrany v jedné doméně, vzorkuje na sestupnou v druhé doméně a jen se musí stihnout včas ustálit
- Asynchronní domény (asynchronnous clocks)
  - signály buzené z jedné domény se NEstihnou ustálit před příchodem nejbližší události na hodinovém signálu v druhé doméně
  - nejobecnější případ
- Domény se vzájemně výlučnými hodinami (mutually exclusive clock signals)
  - nikdy neběží současně

Děkuji za pozornost!

Stále hledáme nové kolegy:
www.asicentrum.cz