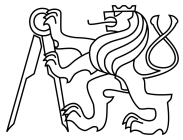
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Emulátor bezkontaktní čipové karty v FPGA
Student:	Bc. Stanislav Jeřábek
Vedoucí:	Ing. Jiří Buček
Studijní program:	Informatika
Studijní obor:	Projektování číslicových systémů
Katedra:	Katedra číslicového návrhu
Platnost zadání:	do konce letního semestru 2015/16

Pokyny pro vypracování

Prostudujte normu ISO/IEC 14443 o čipových kartách s vazbou na blízko. Navrhněte a realizujte emulátor bezkontaktní čipové karty podle výše uvedené normy. Zařízení bude podporovat množinu protokolů vybraných po konzultaci s vedoucím práce. Pro rádiové rozhraní použijte vhodný přípravek se speciálním obvodem, např. PN532. Zařízení otestujte v součinnosti s bezkontaktní čtečkou.

Seznam odborné literatury

Mezinárodní norma ISO/IEC 14443

L.S.

doc.Ing. Hana Kubátová, CSc.
vedoucí katedry

prof.Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 27. ledna 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Diplomová práce

Emulátor bezkontaktní čipové karty v FPGA

Bc. Stanislav Jeřábek

Vedoucí práce: Ing. Jiří Buček

10. května 2016

Poděkování

Rád bych touto formou poděkoval vedoucímu práce, Ing. Jiřímu Bučkovi, za trpělivost se mnou a Lukášovi Koštenskému za zapůjčení tabletu pro testování. Dále děkuji rodině, přátelům a všem ostatním, kteří mi dodávali skvěle vyvážený mix podpory a popohánění.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 10. května 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Jeřábek, Stanislav. *Emulátor bezkontaktní čipové karty v FPGA*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá emulací komunikace bezkontaktních čipových karet typu A podle normy ISO/IEC 14443. Odlišnost tohoto emulátoru spočívá v téměř stoprocentní implementaci v hardwaru – FPGA – v jazyce VHDL.

Klíčová slova FPGA, Emulace, Bezkontaktní čipová karta, ISO/IEC 14443, VHDL

Abstract

This diploma thesis describes implementation of contactless chip card emulator compliant with ISO/IEC 14443. It differs in way of emulation when almost everything is done in hardware – FPGA – using VHDL language.

Keywords FPGA, Emulation, Contactless Chip Card, ISO/IEC 14443, VHDL

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Čipové karty dnes mají spoustu využití. Jejich obliba i přes jistá bezpečnostní rizika nadále roste a pokud se nic zásadně nezmění, dočkáme se v blízké budoucnosti jistě další spousty způsobů jejich využití. Ačkoliv se čipové karty těší největší oblibě v Evropě a Asii, jejich popularita roste globálně. [10]

Využívají se často jako zaměstnanecké identifikační průkazy, zákaznické věrnostní průkazy, bezpečnostní průkazy, úložiště biometrických informací nebo například pro zabezpečení přístupu na internet a internetových transakcí. Dále se využívají ve spoustě komerčních aplikací jako třeba v bankovnictví, při platbách poplatků za parkování, mýtného či jízdného v samoobslužných dopravních systémech [11].

Stále nejčastějším využitím čipových karet jsou SIM karty v mobilních telefonech. Ačkoli jejich počet v České republice již neroste, stále připadá na každého člověka v ČR více než jedna SIM karta [12]. Za SIM kartami jenom lehce zaostává další specifický typ čipových karet – karty platební. Těch je v ČR přes 10 milionů [13].

První karty vzhledově podobné těm, jak je známe dnes, vydal Diner's Club počátkem 50. let 20. století. Nejednalo se však o karty čipové, ale pouze plastové kartičky popsané identifikačními údaji. S přibývajícími uživateli těchto plastových identifikačních karet rostly také nároky na bezpečnost z důvodu ochrany před podvody a manipulováním údajů. Proto v 70. letech vyvinula Mezinárodní asociace letecké dopravy magnetický proužek o kapacitě 210 bitů na palec. S myšlenkou spojení úložiště dat a logiky do jednoho kusu křemíku přišel v roce 1970 Japonec Dr. Kunitaka Arimura. Myšlenka zakomponování integrovaných obvodů do identifikačních karet si nechali patentovat již v roce 1968 němečtí vynálezci Jurgen Dethloff a Helmut Grotrupp. [10]

Za otce čipových karet je považován Ronald Moreno, který v roce 1974 umístil čip na plastikovou kartu a vynalezl zařízení, které jej dokáže přečíst. Za první inteligentní kartu je pak považována karta CP8 společnosti Bull z března 1979. Ta obsahovala 1 kB programovatelné paměti a procesorové jádro 6805 od společnosti Motorola. Na této kartě byla paměť od procesoru oddělená, což se

později ukázalo jako bezpečnostní riziko. Spojit obě části do jednoho čipu však umožnil technologický pokrok až v letech 80. V té době se čipové karty dočkaly velkého rozšíření v podvědomí lidí, a to když se masivně rozšířily přednabité karty do veřejných telefonů. Dnes jsou čipové karty nejčastěji využívány jako SIM karty v mobilních telefonech, což započalo v roce 1995. Dále čipové karty najdeme v mnoha odvětvích a různých způsobech využití, přičemž se zdá, že jejich expanze bude nadále pokračovat. [10]

Cíl práce

Zásadní rozdíle při využívání čipových karet spočívá ve způsobu provedení transakce. Karta primárně slouží jako nosič a její UID (unikátní sériové číslo) primárně pro účely řešení možných kolizí více karet v poli jedné čtečky. V druhém případě se celá transakce odehrává čistě mezi kartou a čtečkou, přičemž čtečka pomocí přístupových klíčů komunikuje s aplikacemi uloženými na kartě a provádí potřebné změny přímo v datech na kartě uložených. Některé systémy však UID využívají jako identifikátor karty a poté celou kartu čistě jako identifikátor. V tomto případě nezabezpečeného systému čtečka přečte UID, které je přístupné komukoli se čtecím zařízením v blízkosti, a poté ve spolupráci se vzdáleným serverem provede požadovanou transakci. V takovém případě karta slouží pouze jako elektronický identifikátor pro zjednodušení a urychlení obsluhy. Tento způsob je častý v případě pevně umístěných systémů, jako jsou například zámky u dveří či pokladny. První a bezpečnější způsob je dokonce snazším řešením při využití v podmínkách se stíženým přístupem k internetu – tedy transakčnímu serveru. Typickým použitím jsou elektronické peněženky pro platby v prostředcích hromadné dopravy.

Cílem práce je vytvořit emulátor bezkontaktní čipové karty typu A s vazbou na blízko podle normy ISO/IEC 14443 [5] implementovaný v FPGA. Motivací je fakt, že hardware umožňuje přesnější časování a tím dokonalejší emulaci oproti řešení softwarovému. Mojí osobní motivací je po dosavadních zkušenostech s technologií bezkontaktních čipových karet získat zkušenost novou a pokročilejší. Dosud jsem se s technologií čipových karet setkal jako programátor čtecích zařízení nebo zabezpečených modulů.

V celé práci popisuji protokol typu A, který bude implementovaný ve výsledném emulátoru. Protokol pro karty typu B implementovat nebudu a proto ho ani nepopisuji. Jedním z důvodů neimplementace komunikace typu B je ten, že použitý přípravek PN532 Breakout Board [6], který je použit jako anténa a zmíněn již v zadání práce, tento typ komunikace nepodporuje. Vytvořený emulátor bude mít konfigurovatelné UID a měl by tak být schopen podvrhnout identitu jiného uživatele v systémech, kde je identita ověřována pouze pomocí UID čipu. Podmínkou podvrhnutí identity samozřejmě zůstává nutnost znát UID emulovaného čipu.

Kapitola 1

Analýza

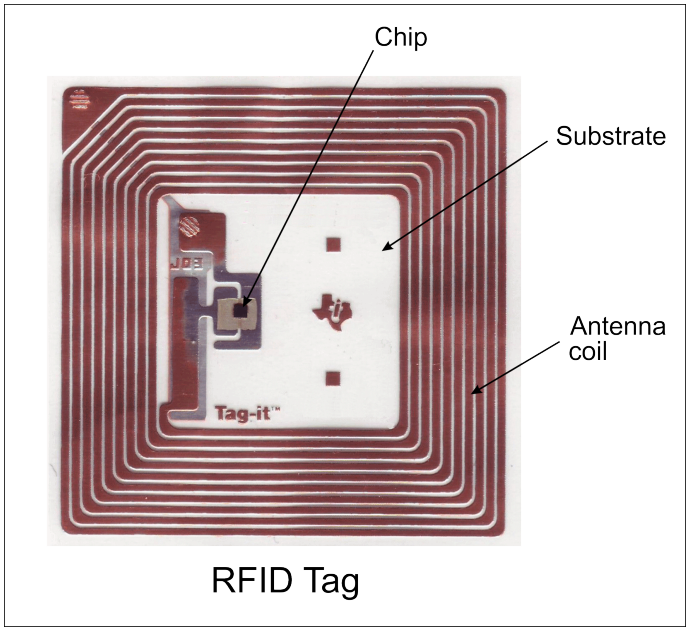
Bezkontaktní čipové karty vznikly v 90. letech ve společnosti Mikron, která je pojmenovala MIFARE (zkratka z Mikron FARE Collection System) a zaregistrovala stejnojmennou obchodní značku. Nyní licence patří společnosti NXP Semiconductors. Tato technologie se nazývá technologií karet s vazbou na blízko. Název technologie je výstižný, neboť čtečky bezkontaktních čipových karet indukují elektromagnetické pole krátkého dosahu. Po vložení karty do tohoto pole se na anténě (cívce po obvodu plastové karty) indukuje napětí a následně obvodem teče elektrický proud, který napájí čip na kartě. Jak vypadá taková čipová karta uvnitř je vidět na obrázku 1.1. Technologie karet s vazbou na blízko je popsána v ISO/IEC normě 14443 [5], kterou v rámci této práce implemetuji pomocí obvodu v FPGA.

1.1 Existující podobná řešení

Nejen k emulaci bezkontaktních čipových karet slouží také zařízení zvané Proxmark3 [2] na obrázku 1.2. Toto zařízení pracuje nejen s vysokofrekvenčními ale i nízkofrekvenčními kartami. Umí se chovat jako čtečka i jako karta nebo jen odposlouchávat komunikaci reálné karty se čtečkou. Toto zařízení se skládá z mikrokontroléru a FPGA, přičemž FPGA slouží ke zpracování signálu a předání výsledku mikrokontroléru [2].

Dalším obdobným obvodem je *Simple NFC* [3] na obrázku 1.3. Ten se skládá pouze z mikrokontroléru, antény a pár základních součástek. Zvládá emulovat NFC – Near Field Communication – tagy fungující dle normy ISO/IEC 14443 [5] až po její třetí část.

Posledním a asi nejpokročilejším zařízením je ChameleonMini na obrázku 1.4 z projektu Chameleon [4]. Toto zařízení umí emulovat karty dle výše zmíněné normy jako ostatní, ale navíc také umí emulovat kompletně některé specifičtější druhy karet včetně jejich paměťových bloků. Zařízení lze také použít k různým útokům na technologii bezkontaktních karet. Projekt stále běží a za-



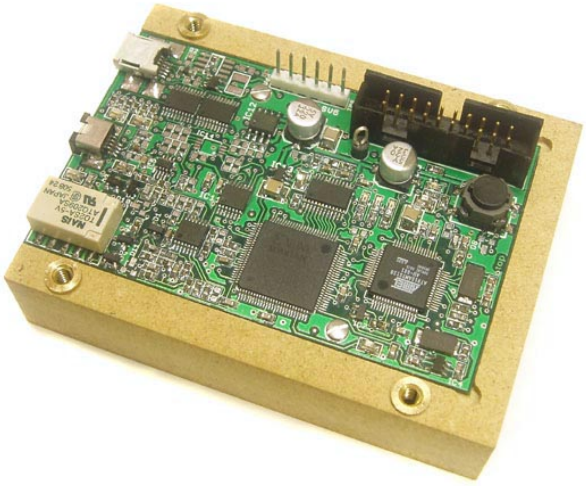
Obrázek 1.1: Bezkontaktní čipová karta bez „obalu“ (převzato z [1]).

čátkem roku 2016 se na jeho další rozvoj formou crowdfundingu povedlo vybrat osminásobek cílové částky.

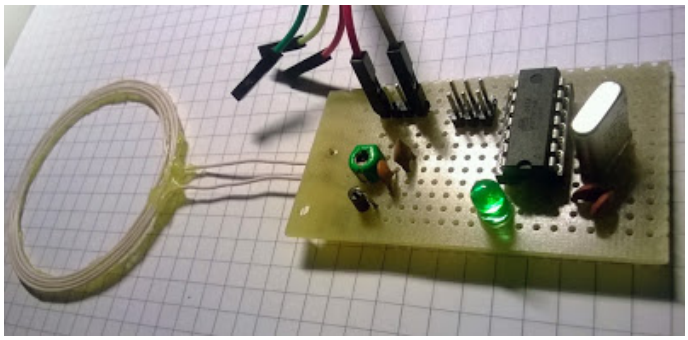
Můj přístup k problému bude odlišný, jelikož použiji přípravek PN532 k elementárnímu zpracování elektromagnetického pole na signálovou úroveň. FPGA potom bude emulovat celý protokol samostatně a předávat pouze pokyny k modulaci pole zátěží. Na rozdíl od předchozích řešení se tedy téměř veškerá činnost systému bude odehrávat v hardwaru. Hardware umožňuje přesnější časování a tím dokonalejší emulaci oproti řešení softwarovému.

1.2 Norma ISO/IEC 14443

Norma ISO/IEC 14443 [5] popisuje protokol pro komunikaci s vysokofrekvenčními čipy s vazbou na blízko. Čtečka vytváří elektromagnetické pole o nosné frekvenci 13,56 MHz (v normě uváděna jako *fc*). Toto pole slouží jako jediný zdroj napájení pro bezkontaktní čipy. Čtečka současně tímto elekromagnetickým polem přenáší data, a to vytvářením tzv. pauz. Pauza je přerušení tvorby elektromagnetické pole na krátkou dobu. Tato doba musí být dostatečně krátká na to, aby tento výpadek nezpůsobil reset bezkontaktní karty. Komunikace pak připomíná sériový protokol *RS–232*, kdy po první pauze – *začátku komunikace* – následují data a celý přenos je ukončen *koncem komunikace*. Podobným způsobem pak odpovídá karta čtečce, akorát namísto vytváření pauz v nosné frekvenci moduluje toto pole proměnnou zátěží.



Obrázek 1.2: Zařízení Proxmark3 (převzato z [2]).



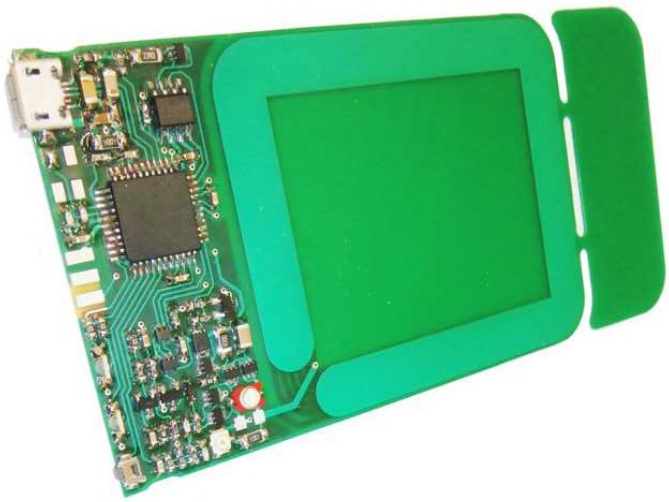
Obrázek 1.3: Zařízení SimpleNFC (převzato z [3]).

1.2.1 Části normy ISO/IEC 14443

[5] Celá norma se dělí na čtyři části:

- První část popisuje fyzikální charakteristiky karet. Určují především požadavky na rozměry, odolnost vůči ultrafialovému a rentgenovému záření, odolnost vůči ohýbání a také elektromagnetické podmínky, v jakých má karta fungovat. Jelikož při emulaci karty v FPGA používám jako anténu přípravek PN532, považuji elektromagnetické požadavky pro fungování za splněné.

- Druhá část popisuje jak minimální a maximální hodnotu nemodulova-



Obrázek 1.4: Zařízení ChameleonMini (zdroj [4]).

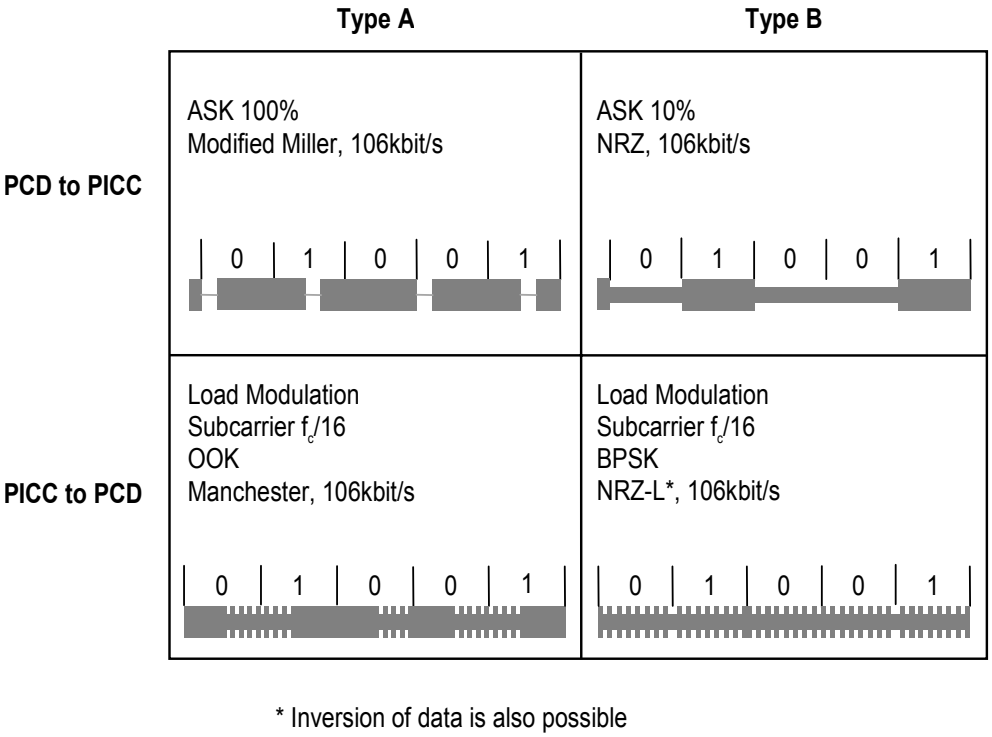
Název	Znak v ISO	Hodnota	Použití
Nosná	fc	13,56 MHz	Zdroj a příkazy pro kartu
Bit rate	fc/128	106 kbit/s	-
Pomocná nosná	fs	847,5 MHz	Modulace zátěží

Tabulka 1.1: Přehled používaných frekvencí

ného pracovního pole v A/m, tak také hodnoty pro amplitudu modulace zátěží. Dále popisuje veškeré frekvence použité při komunikaci na obrázku 1.1 a podrobné časování všech použitých signálů v protokolu.

- Třetí část normy popisuje příkazy použité k navázání komunikace mezi čtečkou a kartou. Určuje kódování příkazů a odpovědí, stavy karty, postup při výběru karty a řešení kolize mezi více kartami a další.

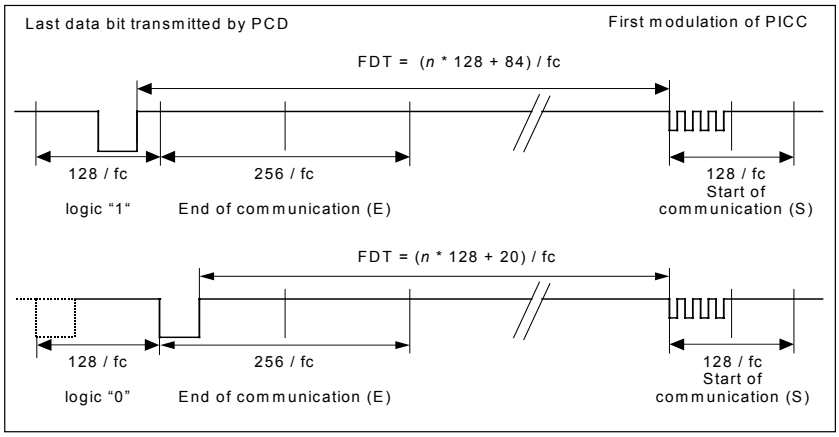
- Čtvrtá část normy popisuje navázání komunikace s kartou na vyšší úrovni. Vyšší úroveň komunikace je z pohledu normy již čistě proprietární. Je známo několik takových vyšších protokolů v závislosti na typu karty (MIFARE Classic, MIFARE Ultralight, MIFARE DESFire,...), avšak emulace těchto protokolů nejen vzhledem k proprietárnosti a časové náročnosti není součástí této práce.



Obrázek 1.5: Amplituda elektromagnetického pole při komunikaci pro typy A i B (převzato z [5]).

1.3 Přenos dat

Celá komunikace mezi kartou a čtečkou probíhá striktně v režimu master/slave, kde master je čtečka a slave jsou všechny karty v jejím elektromagnetickém poli. Vždy, když karta přijme nějaký příkaz, na který by měla odpovědět, pak počká přesně definovanou dobu a začne vysílat odpověď. Tuto dobu lze vyčíst z obrázku 1.6. Dodržení přesného časování je nutné především kvůli řešení kolizí více karet v poli jedné čtečky, což podrobněji popisuje samostatná podkapitola 1.3.5. Zda karta má odpovědět nebo ne, závisí nejen na typu příkazu a jeho dalším obsahu, ale i stavu karty. Norma určuje, v jakém stavu má karta na jaký příkaz reagovat. Například na jeden příkaz odpovídají karty jak uspané, tak právě „oživené“ přítomností v elektromagnetickém poli čtečky, na jiný zase pouze karty právě přidané do pole čtečky. Typickým případem, kdy karta na příkaz z nějakého důvodu neodpovídá, je výběr karty s jiným UID.



Command type	*n* (integer value)	FDT	
		last bit = (1)b	last bit = (0)b
REQA Command WUPA Command ANTICOLLISION Command SELECT Command	9	1236 / *fc*	1172 / *fc*
All other commands	[3] 9	(*n* * 128 + 84) / *fc*	(*n* * 128 + 20) / *fc*

Obrázek 1.6: Určení časové prodlevy mezi přijetím příkazu a začátkem vysílání odpovědi (převzato z [5]).

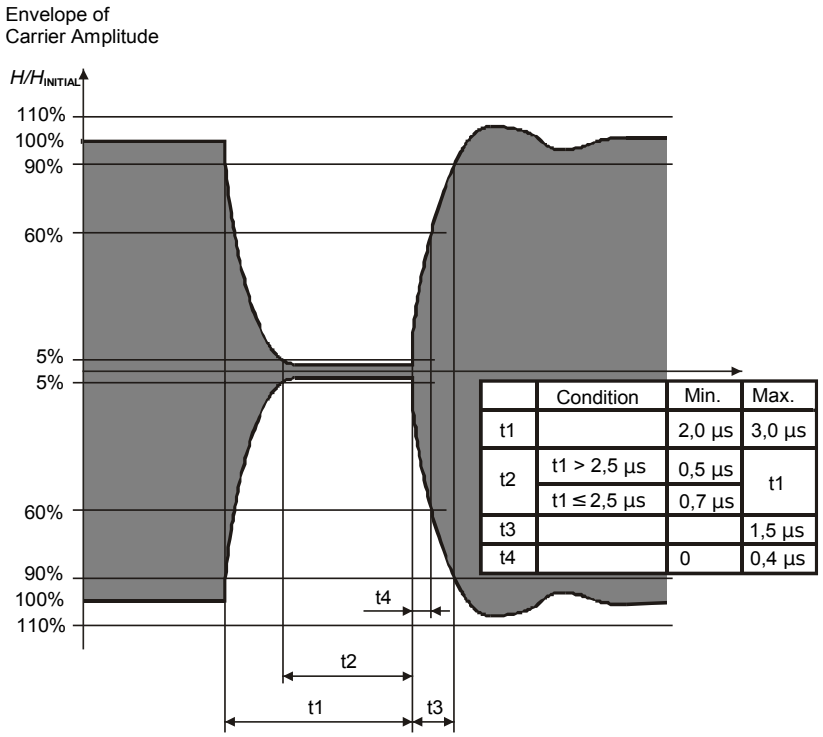
1.3.1 Přenosové sekvence

Na rozdíl od protokolu RS–232 nemůže čtečka na delší dobu přestat vysílat nosnou frekvenci z důvodu napájení čipové karty tímto polem. Ve vysílání nosné frekvence tvoří pouze pauzy, jejichž pozice v rámci bitového intervalu určuje hodnotu daného bitu. Přenosová rychlost protokolu typu A dle normy ISO/IEC 14443 [5] je 128–krát menší, než frekvence nosná (přibližně 106 kbaud/s). Perioda jednoho bitu je potom přibližně 9440 ns. Pauza včetně fade-in a fade-out pak trvá 2–3,4 µs, jak je zobrazeno na obrázku 1.7.

Čipová karta čtečce předává informace pomocí modulování nosné frekvence zátěží. Zátěž vytváří při frekvenci 16–krát nižší, než je frekvence nosná (847,5 kHz) viz obrázek 1.5. Modulaci zátěží pak karta provádí tak, aby amplituda elektromagnetického pole byla vždy na začátku modulace na nižší z obou hodnot.

Pro přenos od čtečky ke kartě rozeznáváme tři různé sekvence:

- sekvence X – pauza po polovině trvání bitu
- sekvence Y – po celou dobu bitu nosná frekvence bez modulace



Obrázek 1.7: Požadavky na průběh pauzy ve vysílání pole (převzato z [5]).

- sekvence Z – pauza na začátku bitu

Tyto sekvence pak mají následující použití:

- logická 1 – sekvence X
- logická 0 – sekvence Y s těmito dvěma výjimkami
  1. V případě, kdy po sobě následuje více logických 0, pak je pro druhou a každou další použita sekvence Z.
  2. Pokud ihned po začátku přenosu následuje logická 0, pak jsou všechny logické 0 až do první logické 1 vyjádřeny sekvencí Z.
- začátek komunikace – sekvence Z
- konec komunikace – logická 0 (Y či Z) následovaná sekvencí Y
- žádná informace – alespoň dvě po sobě jdoucí sekvence Y

Pro přenos dat od karty ke čtečce rozeznáváme tyto sekvence:

- sekvence D – nosná frekvence je modulována zátěží po dobu první poloviny bitu
- sekvence E – nosná frekvence je modulována zátěží po dobu druhé poloviny bitu
- sekvence F – nosná frekvence není modulována po dobu jednoho bitu

Tyto sekvence pak mají následující použití:

- logická 1 – sekvence D
- logická 0 – sekvence E
- začátek komunikace – sekvence D
- konec komunikace – sekvence F
- žádná informace – žádná modulace

1.3.2 Podoba přenášených datových bloků

Všechny příkazy jsou zahájené *začátkem komunikace* – zahajovací sekvencí – a ukončeny *koncem komunikace* – ukončovací sekvencí. Vždy také platí, že data jsou přenášena v rámci jednotlivých bytů vždy on nejméně významného bitu. Další vlastnosti se mohou lišit.

Příkazy od čtečky ke kartám jsou přenášeny ve 3 různých formátech:

1. Krátký rámec – Jde o pouhých 7 bitů. Tento rámec používají příkazy REQA a WUPA.
2. Antikolizní rámec – Zvláštní rámec pro příkaz výběru karty s antikolizní smyčkou – SELECT (Select Command, Type A). Podrobněji popsán v jiné podkapitole 1.3.5.
3. Standardní rámec – Používá se pro všechny ostatní příkazy. Skládá se z jednotlivých bytů doplněných o lichou paritu. Standardní rámec je dále doplněn o CRC v délce 2 bytů, též doplněných o liché paritní bity. Podoba CRC je popsána v jiné podkapitole 1.3.4. Používá se také pro příkaz výběru SELECT v případě, kdy je odesíláno celých 5 bytů UID.

Odpovědi karty čtečce jsou přenášeny také ve 3 formátech:

1. Rámec pro ATQA – Tímto způsobem je přenášena pouze odpověď ATQA. Jedná se o 16 bitů, jejichž kódování udává obrázek 1.8.
2. Antikolizní rámec – Tímto rámcem odesílá karta část svého UID. Podrobněji popsán v jiné podkapitole 1.3.5.

3. Standardní rámec – Používá se pro všechny ostatní odpovědi. Jeho podoba je shodná se standardním rámcem pro přenos příkazu čtečky ke kartě popsaným v předchozí odrážce 3 3.

MSB LSB

b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1
RFU				Proprietary coding				UID size bit frame		RFU	Bit frame anticollision				

b8	b7	Meaning
0	0	UID size: single
0	1	UID size: double
1	0	UID size: triple
1	1	RFU

b5	b4	b3	b2	b1	Meaning
1	0	0	0	0	bit frame anticollision
0	1	0	0	0	bit frame anticollision
0	0	1	0	0	bit frame anticollision
0	0	0	1	0	bit frame anticollision
0	0	0	0	1	bit frame anticollision

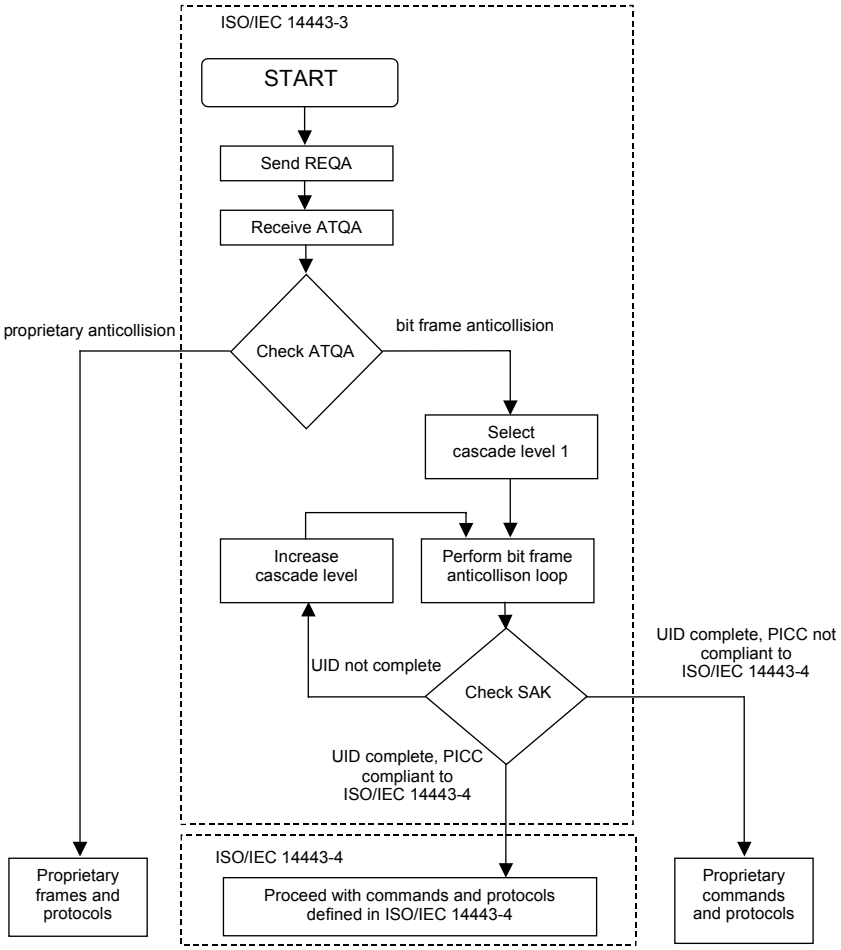
Obrázek 1.8: Kódování odpovědi ATQA – Answer To Request, Type A (převzato z [5]).

1.3.3 Životní cyklus karty

Celá komunikace probíhá striktně podle modelu master/slave. Čtečka jako master neustále vysílá nosnou frekvenci a periodicky také příkaz REQA (Request Command, Type A). Jakmile se v poli vytvářeném čtečkou nachází karta, měla by (s výjimkou stavu HALT) odpovědět svým ATQA (Answer To Request, Type A). Poté následuje další komunikace vždy příkazem čtečky a na většinu z nich také odpovědí od karty.

Jakmile karta vstoupí do elektromagnetického pole vytvářeného čtečkou, kterým je současně napájen, začne odposlouchávat provoz v tomto poli. Typická komunikace karty spočívá v odpovědi karty na požadavek REQA od čtečky. Poté následuje výběr karty zahrnující také antikolizní smyčku, která řeší přítomnost více karet v poli čtečky najednou. Výběru karty a antikolizní smyčce se věnuje samostatná podkapitola 1.3.5. Jakmile je karta úspěšně vybrána, odešle čtečce odpověď SAK (Select AcKnowledge, Type A) a očekává další příkazy dle normy ISO/IEC 14443 části 4 [5]. Tato část normy popisuje navázání komunikace s kartou na tomto vyšším protokolu. Další instrukce krom navázání samotného jsou již čistě proprietární, a proto čtvrtá část normy není v této práci implementována. Komunikaci v tomto odstavci znázorňuje diagram 1.9.

Podrobněji životní cyklus karty a příkazy popisují diagram stavů karty 1.10 s přiloženou tabulkou vysvětlující některé použité symboly 1.2.



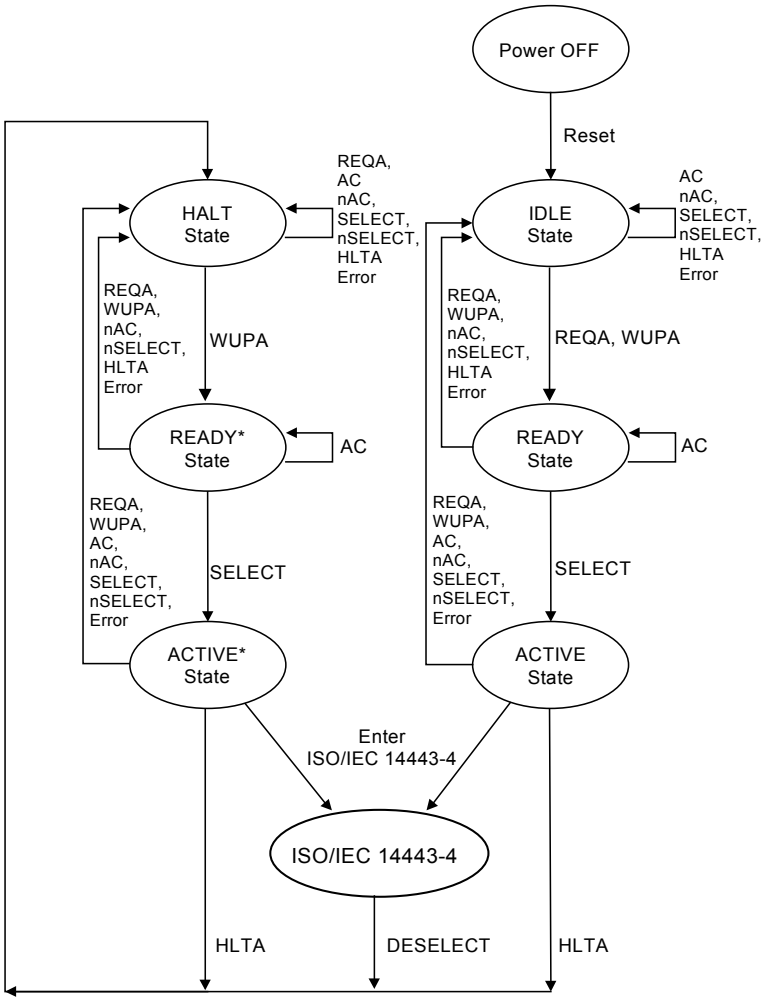
Obrázek 1.9: Sekvence výběru karty čtečkou (převzato z [5]).

1.3.3.1 Příkaz REQA

Tento příkaz používá čtečka na úvod komunikace se všemi kartami. Na tento příkaz karta v IDLE stavu odpovídá ATQA a přechází do READY stavu. Příkaz má hodnotu 0x26.

1.3.3.2 Příkaz WUPA

Na tento příkaz nové karty v poli reagují stejně jako na REQA. Navíc na něj stejným způsobem reagují karty uspané – ve stavu HALT. Příkaz má hodnotu 0x52.



Obrázek 1.10: Přechodový diagram stavů karty (převzato z [5]).

1.3.3.3 Příkaz ANTICOLLISION

Tento příkaz čtečka používá k výběru karty. Tento příkaz může obsahovat část UID, nikdy však UID celé. Vybrané karty odpovídají zbytkem svého UID v dané úrovni.

1.3.3.4 Příkaz SELECT

V podstatě se jedná o zvláštní podobu příkazu ANTICOLLISION, kdy však naopak musí obsahovat celých 5 bytů UID. Vybraná karty, případně karty, odpovídají pomocí SAK a přechází do stavu ACTIVE.

Symbol	Význam
AC	Antikolizní smyčka s UID karty
nAC	Antikolizní smyčka s jiným UID
SELECT	Karta vybrána příkazem SELECT
nSELECT	Příkazem SELECT vybrána jiná karta (jiné UID)
DESELECT	Příkaz definovaný ve 4. části ISO/IEC 14443 [5]
Error	Detekována chyba při komunikaci

Tabulka 1.2: Symboly použité v diagramu stavů karty 1.10 (převzato z [5])

1.3.3.5 Příkaz HLTA

Tento příkaz uspává dříve úspěšně vybranou kartu. Uspaná karta nijak neodpovídá a přechází do stavu HALT. Příkaz má hodnotu 0x50, 0x00.

1.3.3.6 Odpověď ATQA

Tuto odpověď používá karta na příkazy REQA a WUPA. Odpověď obsahuje další užitečné informace jako například velikost UID karty. Strukturu podrobně znázorňuje obrázek 1.8.

1.3.3.7 Odpověď ANTICOLLISION

Tato odpověď obsahuje zbytek bitů UID v dané úrovni, která v předchozím dotazu neodeslala čtečka.

1.3.3.8 Odpověď SAK

Tuto odpověď používá karta, pokud byla vybrána pomocí příkazu SELECT. Odpověď obsahuje informaci o úplnosti UID (poslední potřebné úrovni výběru) a podpoře čtvrté části normy – proprietárního protokolu. Strukturu podrobněji zobrazuje obrázek 1.11.

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
x	x	x	x	x	1	x	x	Cascade bit set: UID not complete
x	x	1	x	x	0	x	x	UID complete, PICC compliant with ISO/IEC 14443-4
x	x	0	x	x	0	x	x	UID complete, PICC not compliant with ISO/IEC 14443-4

Obrázek 1.11: Struktura odpovědi SAK (převzato z [5]).

1.3.4 CRC

CRC komunikačního protokolu pro karty typu A se počítá ze všech předchozích bytů bez paritních bitů v daném rámci. Výsledek má délku 2 bytů a přidává se včetně paritních bitů na konec rámce. Samotný výpočet CRC se provádí tak, jak je uvedeno v normě ISO/IEC 13239 [14] se dvěma výjimkami. Zaprvé je počáteční hodnota CRC pro výpočet 0x6363, zadruhé výsledek po výpočtu není invertován.

Pro ověřování výpočtu CRC mojí entitou napsanou v jazyce VHDL jsem použil příklady z ISO/IEC 14443 [5]. Pro lepší pochopení samotného výpočtu CRC jsem pak použil zdrojový kód programovacího jazyka C přiložený k normě ISO/IEC 14443 [5] a dalšími náhodnými daty proti tomuto programu ověřil správnost mojí VHDL entity.

1.3.5 Výběr karty a antikolizní smyčka

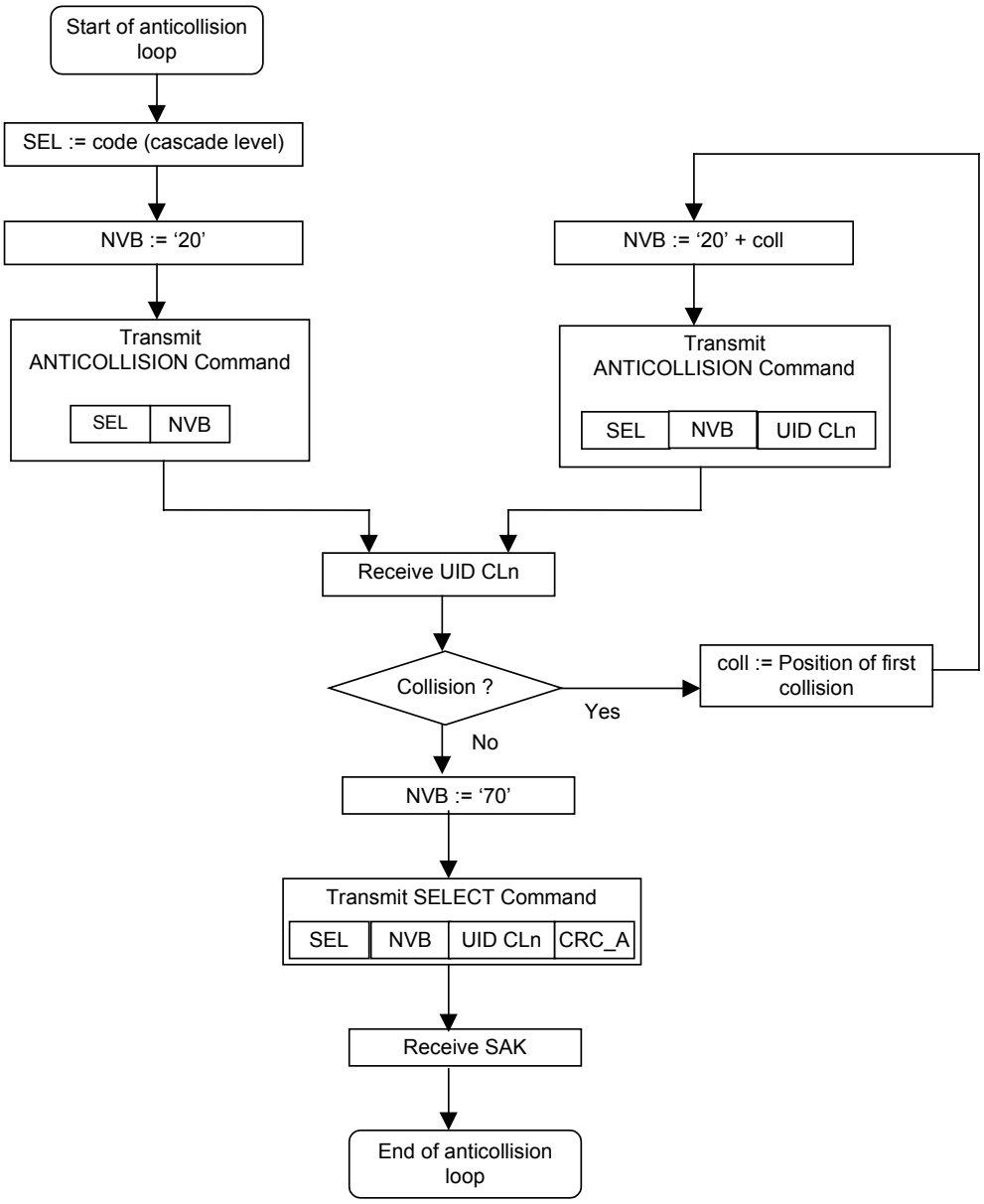
Pro výběr dané karty včetně antikolizní smyčky se používá příkaz SELECT. Celý proces funguje na principu, kdy každá karta má svoje vlastní UID – Unique identifier. Existují 3 různé velikosti UID:

1. Single – 4 byty
2. Double – 7 bytů
3. Triple – 10 bytů

Existují i výjimky z unikátnosti UID, kdy má karta náhodně generované UID a na správci systému je, aby zajistil unikátnost v rámci používaného systému. Tento princip se podobá privátním rozsahům IP adres a obdobně je určen zvláštní hodnotou, kterou je 0x08 v nultém bytu UID. Takto vygenerované UID je vždy velikosti single.

Celý proces výběru karty může být v závislosti velikosti UID karet přítomných v poli rozdělen až na 3 úrovně (při velikosti triple), které jsou zobrazeny na obrázku 1.14. V každé úrovni nejprve čtečka vyšle samotný příkaz SELECT bez jediného bitu UID. Všechny karty, které ještě v této úrovni mají UID (záleží na délce UID), odpoví 5 byty dle svého UID. Jelikož všechny karty odpovídají současně, projeví se současná odpověď jedné karty logickou jedničkou a jiné karty logickou nulou modulací nosné frekvence po dobu celého bitu. Takto čtečka pozná, že se v poli nachází více karet s různým UID v dané úrovni. Postup výběru karty lépe znázorňuje diagram 1.12.

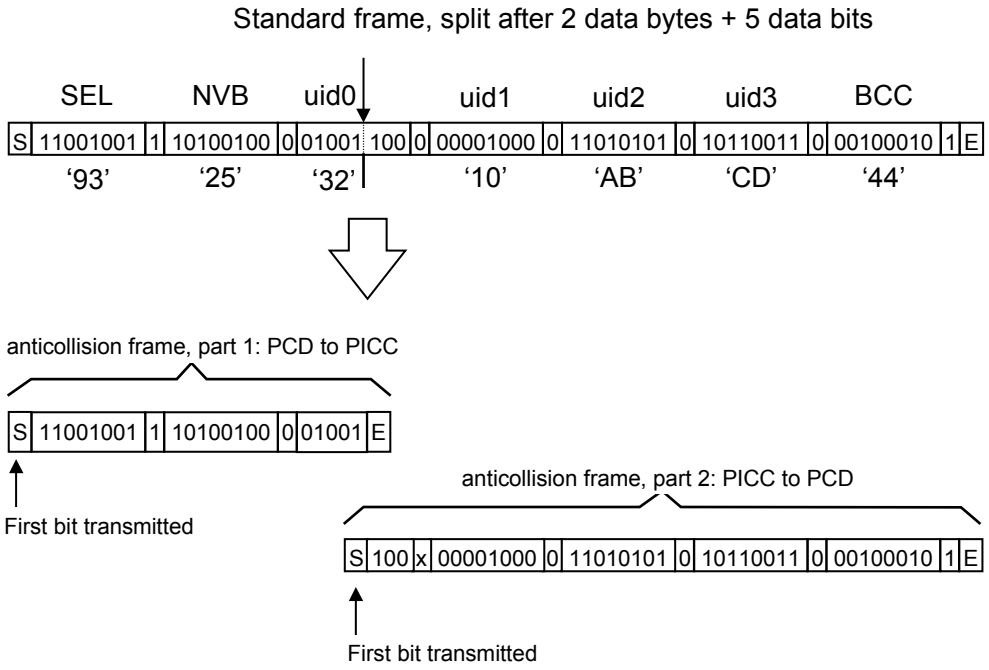
Jakmile čtečka detekuje kolizi, vysílá znovu příkaz SELECT v dané úrovni, avšak tentokrát doplněný o část UID. Přesněji je podoba vysílaných rámců během antikolizní smyčky na obrázku 1.13. Vysláním počátečních společných bitů a poté buď logické nuly či jedničky vybere čtečka ze jednu ze dvou skupin odpovídajících karet. Takto vybrané karty odpoví zbývající částí UID v dané



Obrázek 1.12: Diagram antikolizní smyčky (převzato z [5]).

úrovni. Tento postup se opakuje, dokud existuje v obdržených UID v dané úrovni kolize.

Výběr karty dle UID v dané úrovni je ukončen příkazem SELECT doplněným o celých 5 bytů UID z dané úrovně, který je také doplněn o CRC. Na tento příkaz reaguje vybraná karta (vybrané karty v případě se shodným UID na nižší úrovni) odpovědí SAK 1.11 – Select AcKnowledge, Type A. Příznaky



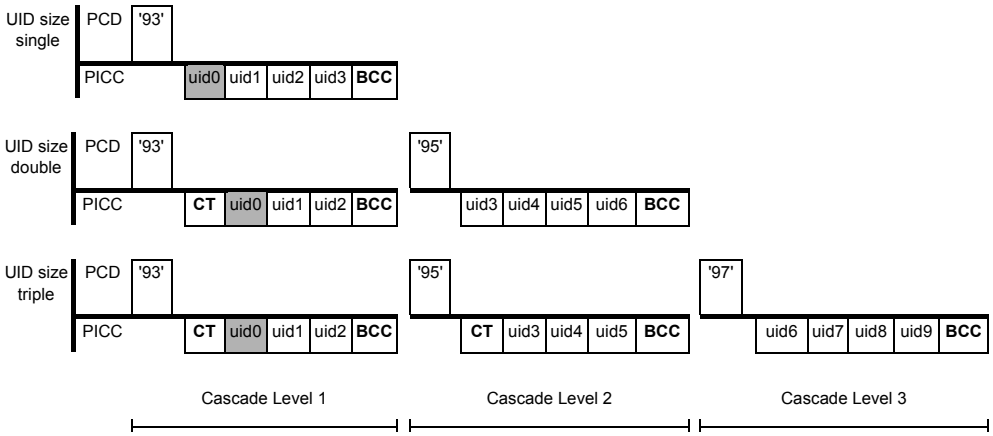
Obrázek 1.13: Ukázka antikolizního rámce (převzato z [5]).

v odpovědi SAK poté určují mj. to, zda má čtečka pokračovat ve výběru dle UID na další úrovni nebo je již UID kompletní.

Podoba 5 bytů odvozených z UID je podrobněji na obrázku 1.14. CT – Cascade tag – má hodnotu 0x88 a označuje fakt, že UID karty v dané úrovni nekončí. Tuto hodnotu by tedy neměl mít nultý byte UID o velikosti single (4 byty) a třetí byte UID o velikosti double (7 bytů), aby nedocházelo ke kolizi s delšími UID. BCC je potom redundantní byte sloužící ke kontrole, který se vypočítá jako XOR předchozích 4 bytů UID.

Na dalším obrázku 1.15 je příklad dvou různých karet v poli jedné karty a celého procesu výběru karty včetně antikolizní smyčky.

Hodnota příkazu SELECT závisí na úrovni výběru a jeho hodnoty zobrazuje tabulka. Druhým bytem příkazu je NVB – Number of Valid Bits, Type A – které určuje počet odesílaných bitů UID čtečkou. Významnější nibble NVB určuje počet odeslaných bytů (včetně příkazu SELECT a samotného NVB), méně významný nibble potom počet bitů k celým bytům navíc, jak lze vyčíst z obrázku 1.16.



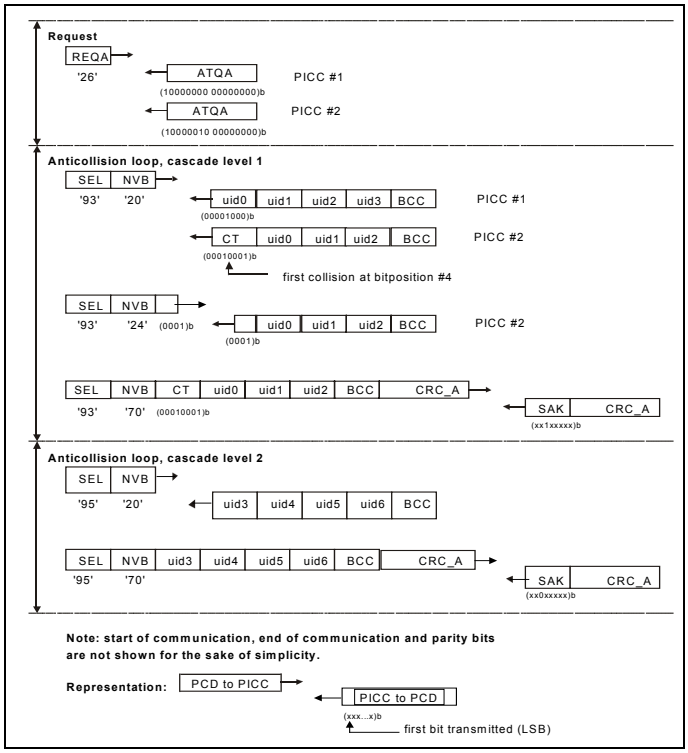
Obrázek 1.14: Struktura 5 bytů odvozených z UID na jednotlivých úrovních (převzato z [5]).

1.4 Specifika FPGA a PN532

Při návrhu emulace bezkontaktní čipové karty na FPGA je třeba vzít v potaz několik zásadních rozdílů. FPGA oproti bezkontaktní kartě nemá anténu pro příjem signálu. Skrze anténu je klasická bezkontaktní karta také napájena elektromagnetickým polem, přičemž FPGA má vlastní zdroj napájení. Na přítomnost vlastního napájení FPGA je třeba myslet především z toho důvodu, že čtečka může vypnutím tvorby elektromagnetického pole na delší dobu všechny karty zresetovat. Emulátor v FPGA tedy bude muset při delším výpadku nosné frekvence zresetovat svoje vnitřní stavové automaty, aby napodobil chování reálné karty.

1.4.1 Charakteristika PN532 [6]

Jako anténa bude použit přípravek PN532 Breakout board [6] zobrazený na obrázku 1.17. Přípravek PN532 umí komunikovat po sériové lince a emulovat jak čtečku tak i bezkontaktní kartu. Současně také umí fungovat jako pouhé rádiové rozhraní pro bezdrátovou komunikaci dle ISO/IEC 14443 [5] – tzv. mód *virtuální karty*. Použitím přípravku PN532 bude dosaženo všech požadavků uvedených v první části normy ISO/IEC 14443, což vede k možnosti soustředit se pouze na implementaci samotného protokolu uvnitř FPGA. V módu virtuální karty komunikuje PN532 pomocí dvou signálů (`SIGOUT` a `SIGIN`) se SAM (Secure access module) [15] – samotným čipem. Ke komunikaci antény se samotným čipem slouží rozhraní dle obrázku 1.18. Jelikož v případě této práce nebude PN532 sériovou linkou připojený k žádnému počítači, nemá volitelný signál CLAD sloužící k informování hosta o dokončené transakci žádný



Obrázek 1.15: Ukázka antikolizní smyčky (převzato z [5]).

smysl. FPGA a PN532 budou tedy propojeny 4 dráty, a to komunikačními signály `SIGIN` a `SIGOUT` a 2 dráty napájecími.

1.4.2 Požadavky na FPGA

Pro výběr desky s FPGA není mnoho omezujících podmínek. Požadavky pro implementaci emulátoru do dané desky jsou tyto:

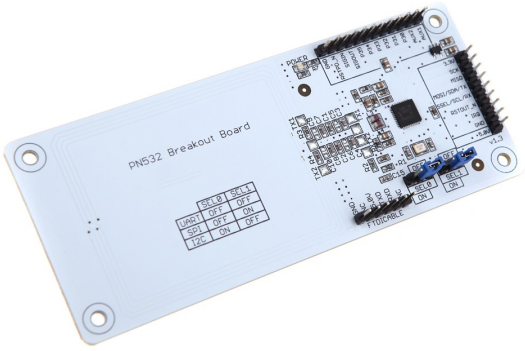
- Alespoň 2 vstupně/výstupní signály.
- Hodinový krystal o frekvenci alespoň 27,15 MHz.
- Dostatek vnitřních obvodů FPGA pro nahrání návrhu.

Požadovaná hodinová frekvence musí být dle Nyquist–Shannonova teorému [16] alespoň o něco vyšší, než dvojnásobek vzorkované frekvence (v tomto případě vzorkujeme 13,56 MHz). Vyšší frekvence oscilátoru na FPGA nám však poskytuje možnost přesnějšího vzorkování, a tím pádem přesnější reakce, a také rychlejšího výsledného návrhu. Rozhodl jsem se pro návrh použít

b8	b7	b6	b5	Meaning
0	0	1	0	Byte count = 2
0	0	1	1	Byte count = 3
0	1	0	0	Byte count = 4
0	1	0	1	Byte count = 5
0	1	1	0	Byte count = 6
0	1	1	1	Byte count = 7

b4	b3	b2	b1	Meaning
0	0	0	0	bit count = 0
0	0	0	1	bit count = 1
0	0	1	0	bit count = 2
0	0	1	1	bit count = 3
0	1	0	0	bit count = 4
0	1	0	1	bit count = 5
0	1	1	0	bit count = 6
0	1	1	1	bit count = 7

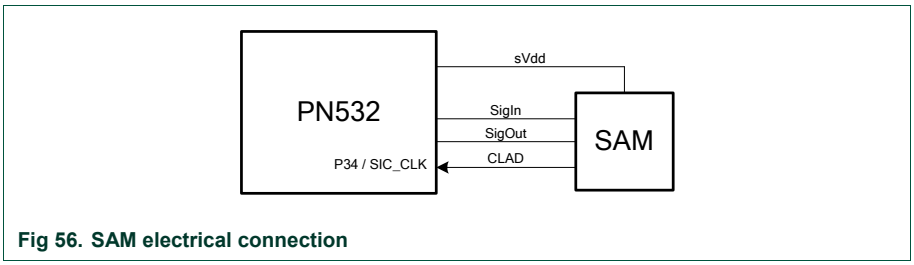
Obrázek 1.16: Kódování NVB – Number of Valid Bits, Type A (převzato z [5]).



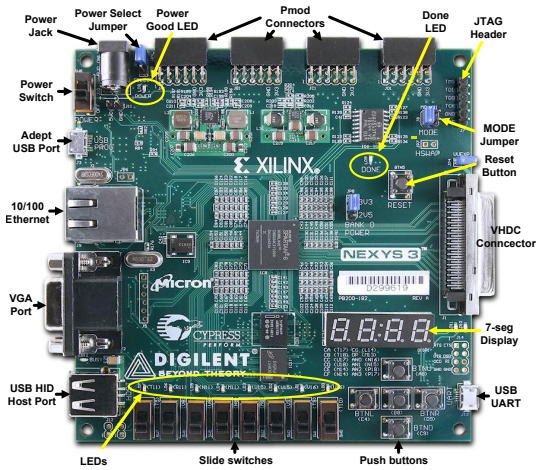
Obrázek 1.17: Přípravek PN532 Breakout Board.

FPGA od výrobce Xilinx z rodiny Spartan6 [17] na desce Nexys3 [7] na obrázku 1.19. Tato deska má krystalový oscilátor s frekvencí 100 MHz, který je tedy s přehledem dostačující. Dalším z důvodu výběru této desky byla možnost zapůjčení desky pro tvorbu diplomové práce z fakulty a také osobní zkušenost přímo s touto deskou. Problematiku Nyquist–Shannonova teorému poměrně populárně přibližuje například tato stránka [18].

Softwarová výbava pro FPGA od firmy Xilinx obsahuje také generátor různých IP core [9], přičemž jedním z IP core je *Clocking wizard*. *Clocking wizard* umožňuje snadno vygenerovat modul, který ze signálu krystalového oscilátoru vytváří hodinové signály o požadovaných frekvencích. Bylo by vhodné takto vytvořit hodinový signál o stejné frekvenci, jako je bitová rychlost protokolu (106 kbit/s), a také signál s pomocnou nosnou frekvencí, která se používá k modulaci pole zátěží. Nosnou frekvenci (13,56 MHz) v FPGA používat není vůbec třeba používat, jelikož z této frekvence bude emulátor pouze získá-



Obrázek 1.18: Zapojení PN532 ve virtual card módu (převzato z [6]).



Obrázek 1.19: Zapojení PN532 ve *virtual card* módu (převzato z [7]).

vat data pomocí vzorkování. Tento signál je nutné ihned za vstupem signálu `SIGOUT` do FPGA pomocí digitálního filtru filtrovat do logické nuly v době pauzy a do logické jedničky po dobu oscilace vstupního signálu.

Kapitola 2

Návrh a realizace

Po dohodě s vedoucím práce bude realizována emulace komunikace karet typu A dle normy ISO/IEC 14443 [5] na úrovni, která končí třetí částí normy. Část čtvrtá realizována nebude vzhledem k existenci mnoha různých proprietárních protokolů splňující tuto velmi obecnou část normy. V případě, že by se realizace takto omezeného emulátoru dařila nad očekávání, bude přistoupeno k realizaci emulátoru zvoleného protokolu dle čtvrté části normy. Komunikace karet typu B emulována nebude především z důvodu nekompatibility přípravku PN532 [6] s tímto protokolem.

Během realizace bylo nutné nakonfigurovat přípravek PN532 [6] do *virtual card* módu. Poté už bylo zapotřebí pouze zpracovat v FPGA vstupní signál dle protokolu a správným způsobem generovat signál výstupní. Největší překážkou v tomto úkolu byla práce s více hodinovými doménami, přičemž situaci ještě znepřehledňovaly časové požadavky komunikačního protokolu.

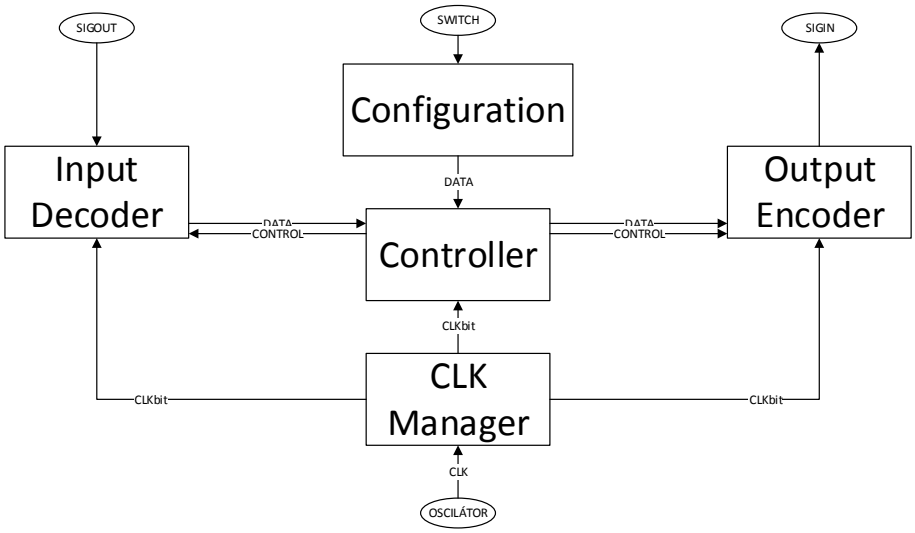
Pro prvotní návrh realizace jsem použil představu, že budu implementovat entity postupně s jejich rostoucí mírou abstrakce. Prvotní představa o de facto odpovídajících entitách na straně přijímaného i vysílaného signálu a jednoho řadiče, který bude na abstraktní úrovni transakce zpracovávat je znázorněna na obrázku 2.1.

2.1 Použitý hardware

Během tvorby diplomové práce jsem použil následující hardware (většina je ho zachycena na obrázku 2.2):

PN532 Breakout board

Tento přípravek [6] slouží jako anténa pro FPGA emulující čip samotný. Současně provádí se signálem z elektromagnetického pole elementární úpravy 1.4.1. Přípravek nepodporuje komunikaci čteček a karet typu B dle normy ISO/IEC 14443 [5].



Obrázek 2.1: Původní zjednodušená představa o finální podobě realizace.

FPGA Nexys3

FPGA zvolené jako cílová platforma pro emulaci karty [7].

Osciloskop Rhode & Schwarz RTO 1024

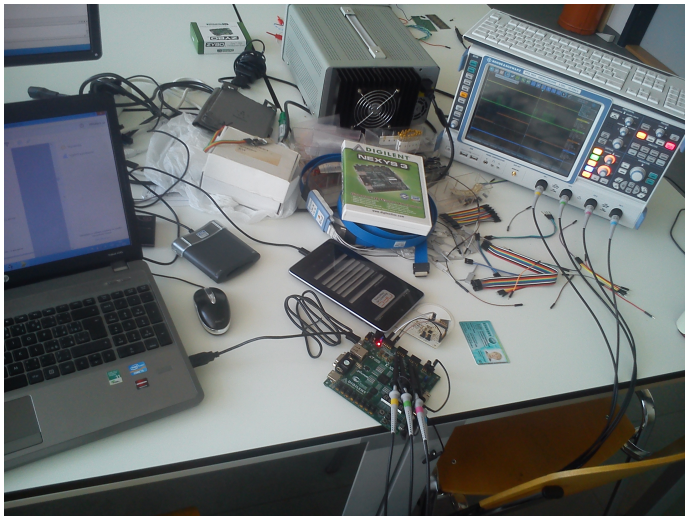
Osciloskop [8], na kterém jsem prováděl konečné testování emulátoru i průběžné testování jednotlivých kroků včetně empirického zjišťování chování některého hardwaru. Současně jsem ho použil jako zdroj některých obrázků.

Čtečka Omnikey Cardman 5321

Čtečka karet [19] použitá při testování emulátoru. Čtečka byla použita pouze pro generování elektromagnetického pole a komunikaci s emulátorem. Důvodem je především fakt, že pro čtení informací z karty je mnohem snazší použít aplikaci v telefonu či tabletu vybaveného NFC technologií a operačním systémem Android.

Tablet Google Nexus 7C ME370TG

Tablet [20] použitý při testování emulátoru. Na rozdíl od čtečky Omnikey Cardman byl použit k vyčtení dat odesílaných emulátorem.



Obrázek 2.2: Pracovní stůl plný používaného hardware.

Různé čipové karty dle standartu ISO/IEC 14443 typu A

Během testování jsem použil různé typy tzv. tagů – tedy čipových karet v různých podobách (karta, emulátor karty, klíčenka, apod.). Tyto karty jsem přečetl pomocí aplikace v tabletu a poté testoval chování různých systémů k mému emulátoru při zadání údajů vyčtených z těchto karet.

2.2 Použitý software

Během tvorby diplomové práce jsem použil následující software:

Cutecom

Jedná se o grafický terminál pro sériovou linku [21] pro UNIXové operační systémy. Použil jsem ho pro konfiguraci přípravku PN532.

ISE Design Suite 14.2

Jedná se o vývojové prostředí od společnosti Xilinx Inc. Použil jsem verzi 14.2 WebPACK Edition [22]. Studentsky licencovanou verzi používám pro návrh do FPGA v jazyce VHDL.

Digilent Adept

Program Adept [23] slouží k nahrávání konfigurace do FPGA obvodů. Používám verzi 2.4.2. Byl vytvořen společností Digilent Inc.

NetBeans IDE 8.1

NetBeans IDE [24] je vývojové prostředí od NetBeans.org, ve kterém jsem napsal program pro výpočet CRC a také pro generování části testbenche jedné entity.

NFC TagInfo

Jedná se o aplikaci pro operační systém Android od vývojáře NFC Research Lab [25]. Tuto aplikaci jsem nainstaloval na tablet Nexus, abych mohl testovat chování mého emulátoru.

Libnfc

Opensource knihovna pro práci s bezkontaktními kartami a čtečkami [26]. Použil jsem ji k prvotnímu vyzkoušení funkčnosti přípravku PN532.

Graphviz

UNIXový program pro tvorbu diagramů [27]. Použil jsem ho ke tvorbě přechodových diagramů konečných automatů v emulátoru.

Microsoft Visio 2013

Tento grafický editor [28] jsem použil pro tvorbu blokového schématu a jednoduchého schématu návrhu.

2.3 Konfigurace PN532

Přípravek PN532 v této diplomové práci slouží jako anténa pro FPGA emulující čip samotný. Přípravek je konfigurovatelný přes sériovou linku. Na začátku realizace jsem chtěl jako první krok otestovat samotnou funkčnost přípravku na svém počítači. K otestování funkčnosti jsem využil volně dostupnou UNIXovou knihovnu libnfc [26]. Po připojení k počítači jsem funkčnost přípravku otestoval pomocí ukázkových příkladů, které jsou součástí instalace knihovny. Přípravek fungoval jak v režimu čtečky, kdy po přiložení karty k němu vypsal typ karty do příkazové řádky, tak také v režimu emulace karty, kdy po přiložení čtečky ukázkový program pro emulaci karty úspěšně doběhl do konce.

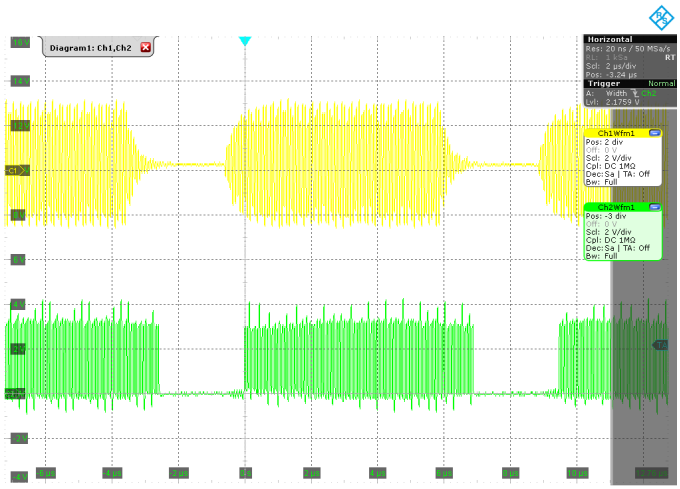
Abych docílil přepnutí přípravku PN532 do režimu rádiového rozhraní – *virtual card* módu, použil jsem příkaz pro konfiguraci přípravku dle manuálu [6], přičemž jsem pro Timeout a vnitřní přerušení IRQ zvolil hodnoty 0x00. Přerušení IRQ slouží k informování hosta o dokončení transakce, přičemž sám přípraven PN532 je o tomto informován pomocí CLAD line od čipu nebo vypršením Timeoutu od začátku transakce. Vzhledem k tomu, že ve finální implementaci nebude přípravek připojen sériovou linkou k žádnému

hostu, tento systém nevyužívám ani nepřipojuji k FPGA nepovinnou CLAD line.

Jelikož z dokumentace k přípravku PN532 [6] není zcela jasné, jak přesně by signály `SIGIN` a `SIGOUT` měly vypadat, rozhodl jsem se to zjistit empiricky. Testováním jsem zjistil následující:

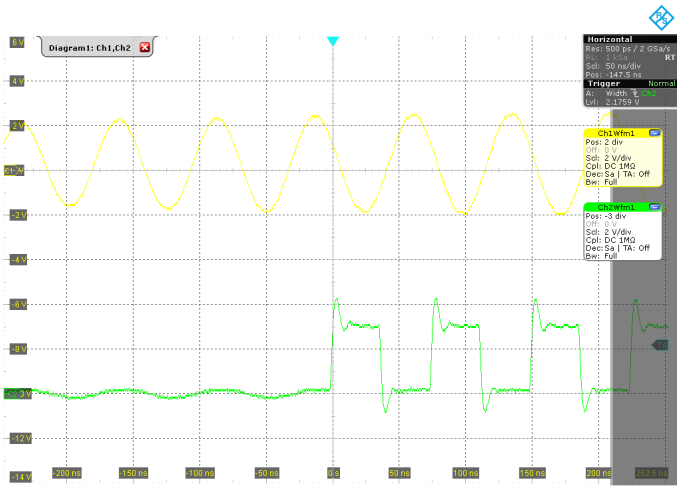
- Na signálu `SIGOUT` je upravený signál pole vysílaného čtečkou. Úprava signálu spočívá v zaprvé usměrnění, kdy signál o nosné frekvenci osciluje pouze od nuly do 3,3 V. Druhá úprava, kterou PN532 [6] provádí, spočívá v interpretaci zeslabování nosné frekvence před pauzou jako klasické nosné frekvence a naopak interpretaci zesilování nosné frekvence po pauze jako pauzy samotné. Navíc je signál upraven z tvaru funkce sinus do tvaru obdélníkového.
- Na úroveň logické jedničky resp. logické nuly na signálu `SIGIN` PN532 [6] reaguje úpravou zátěže tak, že při logické jedničce je amplituda elektromagnetického pole větší, jak lze vidět na obrázku 2.5. Tato změna samozřejmě z důvodu elektrických kapacit přítomných v systému zcela neodpovídá obdélníkovému tvaru dle obrázku 1.5 v ISO/IEC 14443 [5].

Zjištění uvedená výše lépe znázorňují obrázky 2.3 2.4 pořízené osciloskopem [8] během testování.



Obrázek 2.3: Úpravy na signálu `SIGOUT` oproti nosné frekvenci provedené PN532 [6] (pořízeno na [8]).

Pomocí osciloskopu jsem po nastavení přípravku testoval, jak se mění signál `SIGOUT` po přiblížení přípravku ke čtečce (na obrázku 2.3). Poté jsem nastavil trigger osciloskopu tak, aby ke spuštění došlo při nulové úrovni po



Obrázek 2.4: Úpravy na signálu `SIGOUT` oproti nosné frekvenci provedené PN532 [6] v přiblížení (pořízeno na [8]).

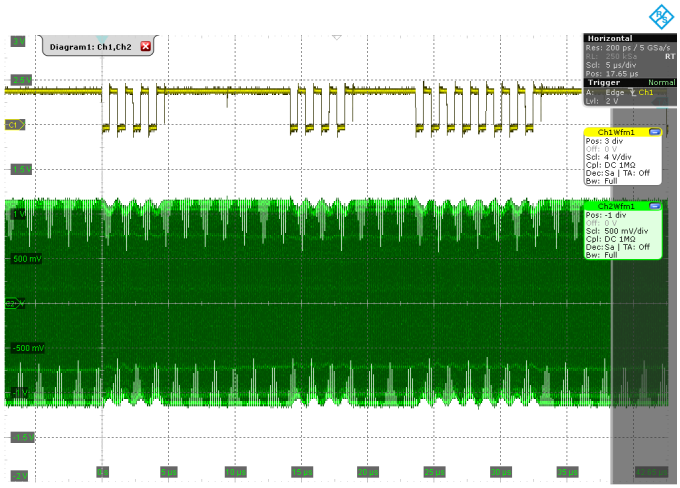
dobu alespoň 600 ns. S trochou snahy se z naměřených údajů na obrázku 2.6 dala vyčíst podoba příkazu REQA popsaného v podkapitole 1.3.3.1.

Jak jsem si ověřil vyzkoušením, přípravek PN532 si pamatuje konfiguraci i po odpojení napájení. Jakmile je tedy nakonfigurován do módu *virtual card*, nemusí být pro účely této práce nadále připojen sériovou linkou k hostu. Také z důvodu usnadnění práce minimalizací nutných připojených součástek jsem přípravek PN532 připojil pouze k FPGA, přičemž používám 2 dráty pro napájení a dva dráty pro datové signály `SIGOUT` a `SIGIN`.

2.4 Implementace emulátoru do FPGA

Pro samotnou implementaci emulátoru do FPGA Nexys3 [7] jsem zvolil jazyk VHDL z důvodu největších zkušeností s tímto jazykem pro popis hardwaru. Pro návrh struktury implementace jsem zvolil přístup zdola nahoru. Implementoval jsem tedy postupně entity od těch nejblíže signálům od PN532 s nejnižší mírou abstrakce až po ty s nejvyšší mírou abstrakce používající mnou zadefinované datové typy. Jak na straně přijímaného signálu z pohledu FPGA (`SIGOUT` signál z PN532), tak i na straně vysílaného signálu (`SIGIN`), jsou obdobné entity zpracující vždy signály na dané úrovni abstrakce a převádějící do úrovně další. Na rozhraní vstupu a výstupu je potom stavový automat, který zpracovává přijímané příkazy a podle nich také připravuje vysílané odpovědi.

V návrhu se také nachází další podpůrné entity nutné pro správnou funkčnost entit obecně popsaných výše jako manažer hodinových signálů, vstupní digitální filter, konfigurační entita a další.

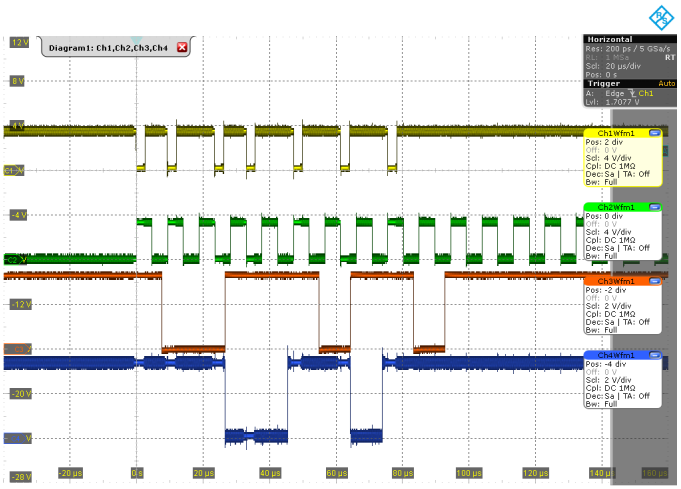


Obrázek 2.5: Modulace pole čtečky v závislosti na hodnotě signálu `SIGIN` (pořízeno na [8]).

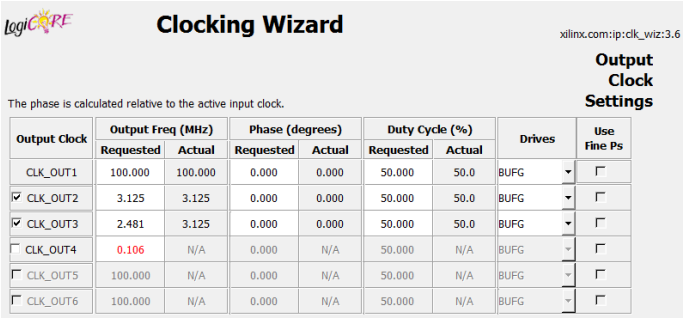
2.4.1 Entita CLKmanager

Pro implementaci entity, která bude ze signálu krystalového oscilátoru odvozovat všechny potřebné hodinové signály, jsem chtěl použít *Clocking wizard*. *Clocking wizard* je obdoba IP core jinak známého jako DCM – Digital Clock Manager. Generování takových entit dle zadaných parametrů podporuje softwarová výbava ISE [22], přímo jeho část pro generování IP core [9]. Pro emulátor potřebuji hodinový signál o stejné frekvenci, jako je bitová rychlost protokolu (106 kbit/s) a dále signál o pomocné nosné frekvenci, která se používá k modulaci pole zátěží (847,5 kHz). Signál pro modulaci nebude sloužit jako hodinový signál rozváděný ke klopným obvodům, ale jako kombinační signál určující výstup.

Clocking wizard pro desku FPGA na desce Nexys3 bohužel umožňuje vytvořit hodinový signál o frekvenci minimálně 3,125 MHz, a pro moji práci je tedy nepoužitelný, což je vidět na obrázku 2.7. Oba výše zmíněné signály tedy budu muset generovat pomocí svoji vlastní entity. Vzhledem k pravidlu pro návrh do FPGA, kdy klopné obvody musí mít hodinový vstup buď přímo z krystalu či právě DCM, nemohu tento mnou vytvořený de facto hodinový signál jako hodinový použít. Pomocí hranového detektoru jsem tedy z tohoto „hodinového signálu“ vytvořil pouze signál, který se jednou za periodu bitové rychlosti octne v logické jedničce na dobu jedné periody krystalu na desce. Tento signál vedu ke klopným obvodům jako Clock Enable, přičemž na hodinový vstup je napojen hodinový signál z krystal na desce o frekvenci 100 MHz. Takový rozvod „hodinového signálu“ *CLKbit* je vidět na obrázku 2.8.



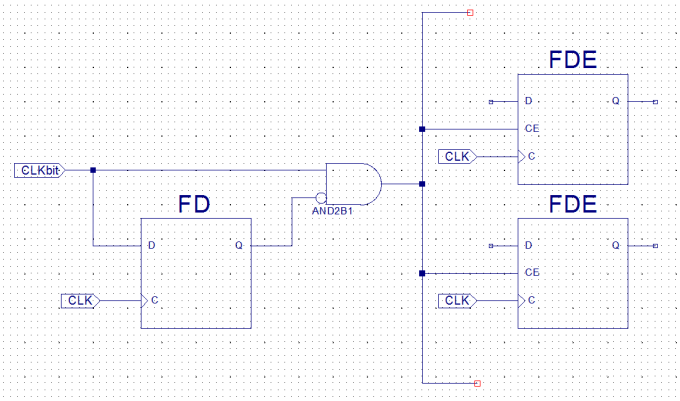
Obrázek 2.6: Příjem příkazu REQA (pořízeno na [8]). CH1 = Vstupní signál; CH2 = CLKbit před hranovým detektorem



Obrázek 2.7: Obrazovka nastavení pro Clock wizard s ukázkou minimální požadované frekvence (pořízeno z [9]).

Stejný problém s nemožností tvorby v IP core bloku nastává také s frekvencí pomocnou nosnou, která se používá k modulaci pole zátěží. Zde však tento signál neřídí klopné obvody, pouze určuje jako signál na vstupu kombinační logiky výstup na signál `SIGIN` přípravku PN532. Nosnou frekvenci (13,56 MHz) v FPGA vůbec nepoužívám. Tuto frekvenci pouze ihned za vstupem signálu `SIGOUT` do FPGA pomocí digitálního filtru filtruji do logické nuly v době pauzy a logické jedničky po dobu oscilace vstupního signálu. To je popsáno v podkapitole 2.4.7.

Samotná tvorba těchto frekvencí začíná v jednom případě na pokyn řídícím signálem od následující entity – CLKcontrol, což se děje při startu vysílání



Obrázek 2.8: Rozvod hodinového signálu CLKbit.

odpovědi. Druhým a posledním případem, kdy začíná entita generovat signály, je v okamžiku, kdy se na výstupu entity FrequencyFilter objeví logická nula, tedy při začátku pauzy v poli čtečky. V tu chvíli začíná přijímání příkazu.

Blokové schéma celého emulátoru lze vidět na obrázku 2.9.

2.4.2 Entita CLKcontrol

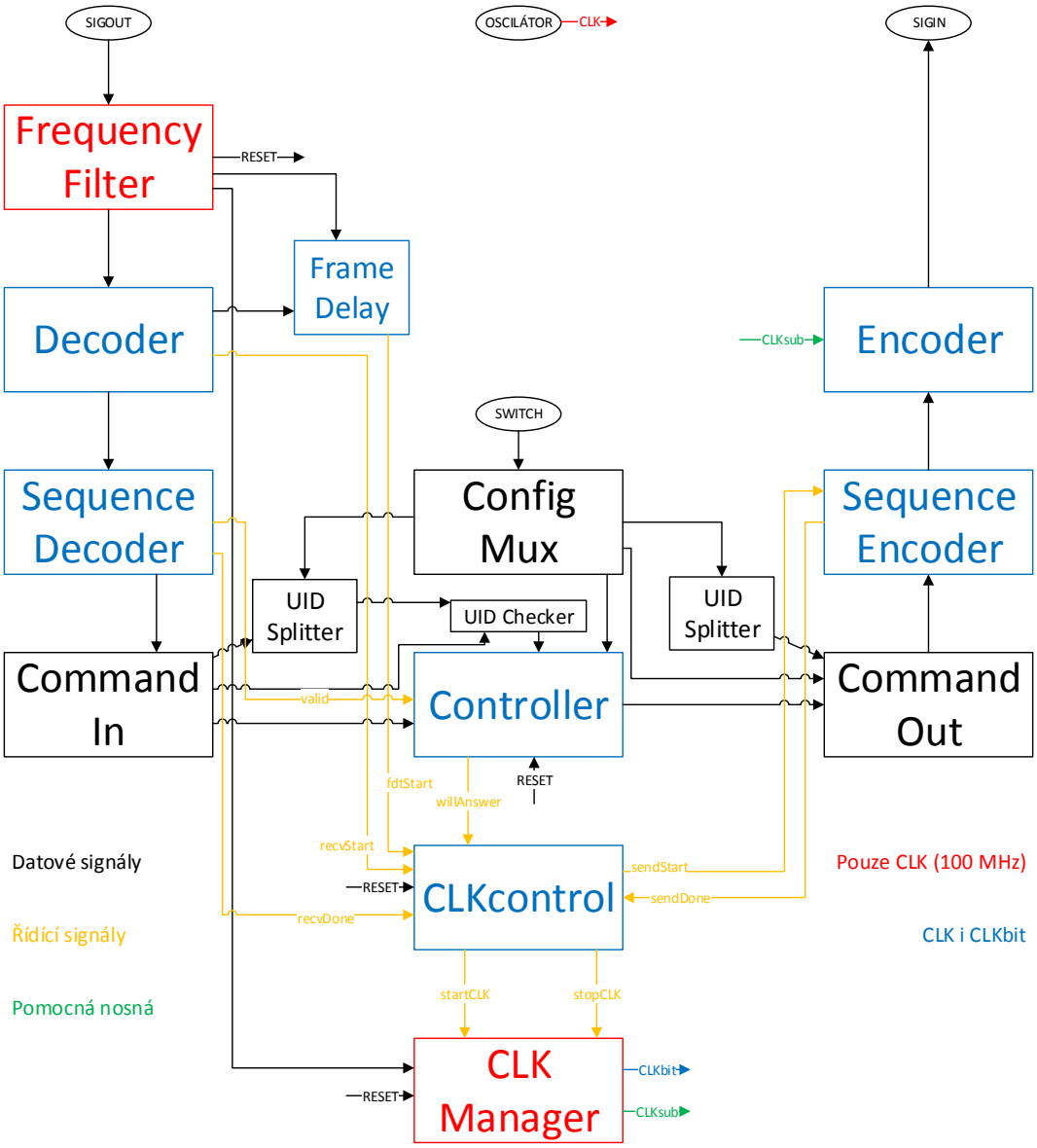
Jelikož čtečka vysílá příkaz v blíže nespecifikovanou dobu, přičemž musí dodržet pouze minimální rozestupy mezi jednotlivými příkazy, je nutné umět zastavit a znovu spustit hodinové signály o bitové rychlosti a frekvenci pro modulaci zátěží. Také je nutné umět v přesně definovanou dobu po ukončení příjmu příkazu znovu spustit obvody pro generování signálu modulace pole – signál `SIGIN`. Vzhledem k nutnosti implementovat dříve uvedené se přímo nabízí neimplementovat žádný signál zakazující práci určených obvodů, ale zpracováváni signálu zastavit jednoduše přerušením hodinového signálu, resp. nechat signál Clock Enable delší dobu v logické nule.

Entita CLKcontrol tedy na základě řídících signálů z ostatních entit řídí start odesílání odpovědi (entitu SequenceEncoder popsanou v podkapitole 2.4.13) a také startuje a zastavuje generování hodinových signálů entitou CLKmanager, která je popsána v podkapitole 2.4.1.

Tato entita také obsahuje stavový automat pro jednotlivé příkazy, tedy zpracování příkazu a odeslání odpovědi. Automat je zobrazený na obrázku 2.10.

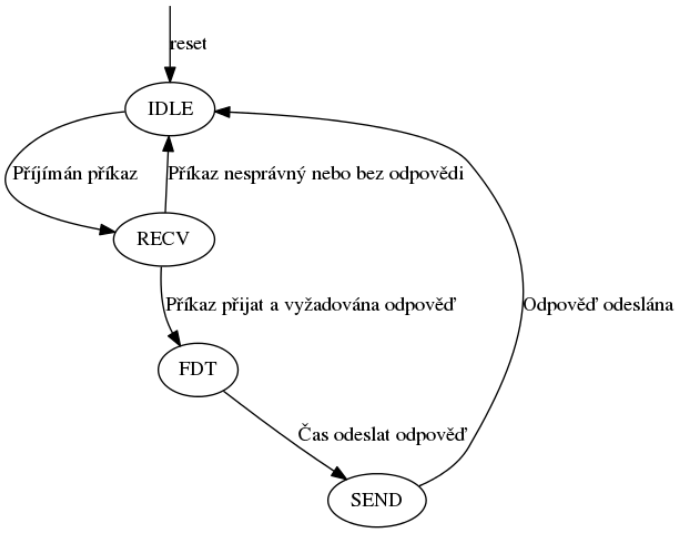
2.4.3 Entita FrameDelay

Entita FrameDelay slouží k přesnému načasování začátku vysílání odpovědi dle normy ISO/IEC 14443 [5]. Tato hodnota je uvedena na obrázku výše 1.6. V mé implementaci na obrázku uvedenou hodnotu $n = 9$ používám nejen tam,



Obrázek 2.9: Blokové schéma emulátoru

kde je to povinné, ale při všech příkazech. Vzhledem k implementaci v hardwaru není nejmenší problém tento požadavek splnit. Požadovaná hodnota je i při odečtení sekvence ukončující přenos včetně posledního datového bitu vysoko přes 50 ms. Moje implementace přitom je schopná začít vysílat odpověď již 3 periody hodinového signálu – 30 ns – po detekci sekvence konce příkazu.



Obrázek 2.10: Přechodový diagram konečného automatu v entitě CLKcontrol

2.4.4 Entita ConfigMux

Entita ConfigMux slouží ke konfiguraci emulátoru. Vzhledem k velmi širokým možnostem UID – čísla karty – oproti omezeným možnostem snadno měnitelných vstupů desky Nexys3 [7] – v tomto případě především přepínačů – jsem ponechal konfiguraci emulátoru přímo ve formě zdrojového souboru. To má sice tu nevýhodu, že pro změnu konfigurace je třeba emulátor znovu syntetizovat v prostředí ISE Design Suite [22], ale prostor UID, jehož velikost je mezi 2^{80} a 2^{81} možnostmi, neposkytuje možnost konfigurace přímo přepínači. Za úvahu by stálo rozšíření o možnost konfigurace přes sériovou linku, jelikož na desce je přítomen micro USB konektor, který právě sériovou linku emuluje.

V entitě se nachází 4 stejné bloky skupiny, přičemž každá skupina obsahuje:

- UIDsize – Udává velikost UID. Možnosti jsou SINGLE (4 byty), DOUBLE (7 bytů) a TRIPLE (10 bytů).
- UID – Pole 10 bytů pro UID samotné. V případě kratšího UID nejsou byty na vyšších pozicích použity.
- SAKC – Byte, jehož hodnota má být odeslána jako odpověď SAK 1.3.3.8 při poslední úrovni UID.
- ATQA – 16 bitů, které má karta odpovědět na příkazy REQA 1.3.3.1 a WUPA 1.3.3.2. Strukturu ukazuje obrázek 1.8.

- confOnlyUID – Tato hodnota říká, zda konstanty SAKC a ATQA mají být použity ('0') nebo ne ('1'). Pokud je hodnota jedna, pak se použijí hodnoty dle normy ISO/IEC 14443 [5]. Během testování jsem však zjistil, že některé bity v normě určené jako rezervované pro budoucí použití jsou již použity a implementace ve čtečkách s nimi počítají. Problém je podrobněji popsán v jiné podkapitole 3.2.1.

Konfigurační údaje pro jednu kartu (v ukázce kartu č. 0) mohou vypadat například takto:

```
constant UIDsize0 : UIDsizeType := DOUBLE;
constant UID0 : dataType(0 to 9) := (X"01",X"23",X"45",X"67",
        X"89",X"ab",X"cd",X"00",X"00",X"00");
constant SAKC0 : byte := X"08";
constant ATQA0 : STD_LOGIC_VECTOR (15 downto 0) := X"0044";
constant confOnlyUID0 : std_logic := '0';
```

Jak napovídá název entity, obsahuje také multiplexor, který multiplexuje právě konfigurační konstanty. Podle nastavení přepínačů 7 a 6 (dva nejvíce vlevo) potom deska emuluje hodnoty v konstantách. Entita vnímá vstup přepínačů jako dvoubitové binární číslo, přičemž přepínač číslo 7 je významnější bit, a na výstupu multiplexoru jsou pak konstanty uvedené ve skupině s daným číslem (0–3).

2.4.5 Entita UIDchecker

Entita UIDchecker, jak název napovídá, porovnává UID přijaté v příkazu SELECT s UID karty uloženým v konfiguraci. Na výstupu je pak jediný bit, který určuje, zda se shodují od začátku až po bit určený dalšími vstupy – počtem celých bytů a zbylých bitů. Na vstupu entity není ani v případě konfigurace celé UID, ale pouze pětice bytů odpovídající dané úrovni výběru. Určení těchto 5 bytů na základě aktuální úrovně a celého UID provádí následující entita – UIDsplitter.

2.4.6 Entita UIDsplitter

Entita UIDsplitter určuje pětici bytů přijímaných a vysílaných v příkazu SELECT na základě celého UID karty. Tato entita má také zvláštní vstup *commInActive*, který určuje, zda má daná namapovaná komponenta být citlivá na signál *commIn* – přijímaný příkaz – nebo *commOut* – odesílaná odpověď. Tato entita je totiž v návrhu použita dvakrát. Jednou kvůli porovnání, zda je čtečkou vybírána právě emulovaná karta, a podruhé pro určení bitů, které mají být odeslány v odpovědi na neúplný příkaz SELECT.

2.4.7 Entita FrequencyFilter

Tato entita je hned za vstupním signálem `SIGOUT`. S frekvencí krystalu na desce (100 MHz) kontroluje úroveň signálu `SIGOUT` a na svůj výstup dává informaci o tom, zda je v poli nosná frekvence či nikoli. Pauzu detekuje ve chvíli, kdy pět po sobě jdoucích hodnot na vstupu je v logické nule. Právě pět po sobě jdoucích takových hodnot znamená, že je na vstupu přítomná buď pauza nebo se PN532 nenachází v poli čtečky. Tento fakt vychází z toho, že polovina periody nosné frekvence je zhruba 37 ns. Jakmile naměříme 5 po sobě jdoucích hodnot v rozmezí 40 ns, víme jistě, že se nejedná o hodnoty naměřené v „nulových půlperiodách“ generovaných na signál `SIGOUT` nosnou frekvencí pole čtečky dle obrázku 2.4.

Dále entita FrequencyFilter generuje signál resetu, a to ve chvíli, kdy naměří logickou nulu alespoň 500–krát po sobě. K resetu tedy dojde, kdy je na signálu `SIGOUT` logická nula nepřetržitě alespoň 5 µs. Takto napodobuji případ skutečné bezkontaktní karty, která nemá vlastní zdroj napájení a při nepřítomnosti v poli karty se tak přirozeně vypne/zresetuje.

2.4.8 Entita Decoder

Entita Decoder má na vstupu signál z předchozí entity – FrequencyFilter. Vstupem je tedy logická nula, právě ve chvíli, kdy je na signálu `SIGOUT` přítomna pauza. Entita mění výstup ve frekvenci bitové rychlosti, přičemž klopné obvody řízené krystalovým oscilátorem z desky o frekvenci 100 MHz testují úroveň vstupního signálu. Podle toho, zda byla pauza přítomna na začátku bitu, po polovině bitu či vůbec, entita vysílá na výstup odpovídající sekvenci – Z, X či Y. Entita výstup nastavuje před náběžnou hranou „bitových hodin“, jelikož přibližně ve 3/4 doby bitu již víme, o jakou sekvenci se jedná. Dle by totiž do této doby už musela být přítomna případná pauza po polovině bitu. Tím se šetří zpozdění jedné periody oscilátoru na desce mezi entitami Decoder a SequenceDecoder.

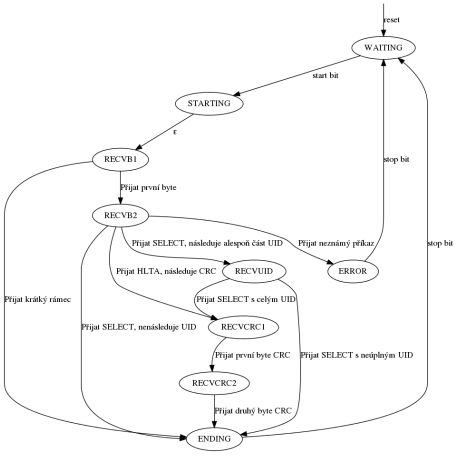
2.4.9 Entita SequenceDecoder

Entita pracuje na frekvenci o bitové rychlosti protokolu a na vstupu má vždy jednu sekvenci z entity Decoder. Tyto sekvence si entita střádá do bufferu a většinou po celých bytech potom do svých datových výstupů. Výjimkou jsou příkazy REQA a WUPA, které používají krátký rámec a skládají se z pouhých 7 bitů, tedy ani ne jednoho bytu.

Na výstupu entity jsou řídící signály oznamující konec příjmu příkazu – detekci *konce komunikace* a určující validitu přijatého příkazu. Aby byl příkaz vyhodnocen jako validní, musí odpovídat bity liché parity u všech bytů a také CRC, pokud je v příkazu přítomno. K detekci tohoto slouží dvě subentity – komponenty popsané dále. Na výstupu entity je dále informace o tom, zda

přijatý rámec byl krátký či jiný (standardní nebo antikolizní), a samozřejmě samotný příkaz a případně také přijaté UID – to vše na úrovni bytů a bitů.

Entita obsahuje stavový automat zobrazený na diagramu 2.11.



Obrázek 2.11: Přechodový diagram konečného automatu v entitě SequenceDecoder

2.4.9.1 Entita SeqIntoByte

Entita SeqIntoByte je komponentou entity SequenceDecoder a na vstupu má buffer posledních devíti přijatých sekvencí (X,Y,Z). Entita slouží zaprvé k převedení vstupních sekvencí na logické hodnoty a následně do mnou definovaného typu „byte“. Tento typ se ničím neliší od klasického bytu, jak ho známe, a je definován jako *std_logic_vector (7 downto 0)*. Druhou funkcí této entity je kontrola parity k danému bytu.

2.4.9.2 Entita CRCcalc

Entita slouží k výpočtu CRC na základě vstupních dat. Na vstupu má dva řídící signály – *start* a *reset* – a jeden datový o velikosti jednoho bytu. Entita také obsahuje 16–bitový klopný obvod obsahující aktuální hodnotu CRC a na výstup posílá 16–bitovou hodnotu CRC po aplikaci vstupního bytu na aktuálně uloženou hodnotu. Výstupní vypočítané CRC tedy reaguje čistě kombinačně – ještě před náběžnou hranou hodin – na změnu vstupního bytu. Tento způsob jsem zvolil z důvodu úspory času oproti čekání na výpočet po náběžné hraně hodin – začátku dalšího bitu.

Na aktivní signál *reset* CRCcalc reaguje nastavením vnitřní hodnoty na počátečních 0x6363. Jakmile je aktivní signál *start*, tak při začátku dalšího

bitu entita zapíše hodnotu aktuálně vysílanou na výstup do klopného obvodu – provede jednu rundu výpočtu s aktuální vstupní hodnotou.

2.4.10 Entita CommandIn

Entita Command je čistě kombinační a pouze převádí přijaté bity na hodnoty abstraktních datových typů. Konkrétně určí, zda byl přijatý příkaz REQA, WUPA, SELECT (včetně úrovně výběru), HLTA či neznámý. Také nastavuje počet přijatých bytů a bitů v rámci příkazu SELECT.

2.4.11 Entita Controller

Entita Controller je samotný řadič emulované čipové karty. Pracuje při frekvenci rovné bitové rychlosti protokolu. Na základě přijatých příkazů na abstraktní úrovni, informacích o jejich validitě a informaci o shodě přijatého UID s UID emulovaným reaguje dle protokolu v normě ISO/IEC 14443 [5]. Nastavuje na výstupy zda má karta vůbec odpovídat a případně jakým způsobem a samozřejmě obsahuje stavový automat odpovídající stavům a přechodům popsaným v diagramu 1.10.

2.4.12 Entita CommandOut

Entita CommandOut odpovídá entitě CommandIn na straně vysílaného signálu a je také složena pouze z kombinační logiky. Z abstraktně zadefinovaných odpovědí karty na vstupech , kterými jsou typ odpovědi (ATQA, odpověď na neúplný SELECT s úrovní výběru a odpověď SAK – úplný SELECT), případně pořadí bitu UID, od kterého se má začít při odpovědi určuje entita na svých výstupech počet a podobu bytů/bitů připravených k odeslání.

Pro potřeby odpovědi SAK, kterou je jako jedinou dle třetí části normy ISO/IEC 14443 [5] nutné doplnit o CRC, má tato entita komponentu CRCcalc podobně jako entita SeqIntoByte. V tomto případě má komponenta CRCcalc připojeny pouze datové vstupy a výstupy, hodinové a řídící signály jsou připojeny k zemi – logické nule. Toto si mohu dovolit, jelikož odpověď SAK je jednobytová a na změnu vstupu komponenta reaguje ihned.

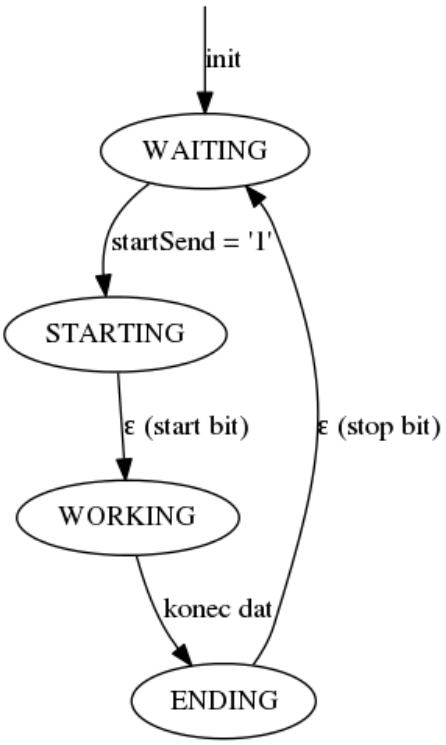
Nakonec jsem se vzhledem k přílišné složitosti entity CRCcalc pro velmi omezené specifické využití s entitě CommandOut rozhodl entitu CRCcalc nepoužít. Pouze jsem vzal kód z entity CRCCalc, vložil do entity CommandOut a signály, které v případě entity CommandOut nabývaly jen jedné hodnoty, nahradil konstantami.

2.4.13 Entita SequenceEncoder

Entita pracuje na frekvenci rovné bitové rychlosti protokolu. Jakmile je na vstupu aktivní signál *start*, začne odesílat data na vstupu, jejichž množství – počet bytů a bitů – je určeno dalšími vstupy. Entita vůbec neřeší, o jaký příkaz

se jedná. Pouze surová data v logických úrovních jedna a nula mění způsobem určeným normou na sekvence D a E na výstupu. Jediná práce s daty spočívá v přidání liché parity na konec každého bytu a případných počátečních bitů. Pokud odpověď vůbec nějaké počáteční bity – neúplný byte – má, pak se určitě jedná o bity UID v antikolizním rámci, přičemž jejich lichá parita je dle normy pro čtečku *don't care*. Po odeslání všech dat nastaví entita výstup na sekvenci F.

Entita obsahuje stavový automat zobrazený na diagramu 2.12.



Obrázek 2.12: Přechodový diagram konečného automatu v entitě Sequence-Encoder

2.4.14 Entita Encoder

Entita encoder přeměňuje výstupní sekvence D, E a F na správnou podobu signálu `SIGIN` přípravku PN532 [6]. Entita pracuje především na frekvenci rovné bitové rychlosti. Obsahuje však navíc čítač, který čítá s frekvencí 100 MHz. Na základě vstupu – sekvence k vysílání – entita vždy na začátku bitu nastaví dva pomocné signály určující, zda má být v dané polovině signál modulován

Stav automatu	**Hodnoty LED diod**
Stav karty	LED (2:0)
IDLE	000
READY	001
READY*	010
ACTIVE	011
ACTIVE*	100
HALT	101
Stav transakce	LED (5:3)
IDLE	001
RECV	010
FDT	100
SEND	111

Tabulka 2.1: Hodnoty na LED diodách reprezentující stavy automatů

zátěží. Podle čítače – polohy v rámci bitu – a těchto pomocných signálů je pak na výstup multiplexována buď logická jednička nebo negace signálu o frekvenci pomocné nosné určené k modulaci. Negace je použita z toho důvodu, že signál startuje pomocné nosné frekvence začíná vždy logickou jedničkou, ale pole během modulace má podle normy začínat nižší hodnotou.

2.4.15 Entita Main

Entita main, jak název napovídá, plní především funkci provázání všech dříve zmíněných entit. Kromě naportování komponent entita obsahuje pouze generování dvou řídících signálů a testovací výstupy na LED diody desky. Těmito řídícími signály jsou *recvStar*t – začátek příjmu – a *sendDone* – konec odesílání. Tyto signály jsou jednoduše odvozeny z aktuálně přijímané resp. vysílané sekvence. LED diody blikají v závislosti na stavu karty z entity Controller a na stavu transakce z entity CLKcontrol. Tabulka s hodnotami na LED diodách dle stavů automatů je zde 2.1.

2.4.16 Definované typy

Pro usnadnění implementace jsem využil možnosti definovat ve VHDL vlastní datové typy. Toto bylo užitečné především při práci na úrovni abstrakce jednotlivých příkazů. Definované typy uvádí tabulka 2.2. Pro tyto definice jsem použil zvláštní soubor *types.vhd*.

Název typu	Struktura typu
outSeq	(D,E,F)
inSeq	(X,Y,Z)
frameType	(SHORT,STAND.,BITOR.,UNKNOWN)
inSeqStream	array (integer range <>) of inSeq
byte	array (7 downto 0) of std_logic
CRC_A	array (0 to 1) of byte
UIDtype	array (0 to 4) of byte
commType	(0 to 1) of byte
dataType	array (integer range <>) of byte
COMMINtype	(REQA,WUPA,HLTA,SELANTI1(,2,3),ERR)
COMMOUTtype	(ATQA,SELANTI1(,2,3),SAKC,SAKINC,ERR)
UIDsizeType	(SINGLE,DOUBLE,TRIPLE)
PICCstate	(IDLE,READY(,R*),ACTIVE(,A*),HALT)
transactionState	(IDLE,RECV,FDT,SEND)

Tabulka 2.2: Definované vlastní datové typy ve VHDL

2.4.17 Oprava WUPA

Při testování během realizace jsem narazil na problém s přijímáním příkazu WUPA. Po vyvedení různých signálů na výstupy desky Nexys3 a jejich prohlížení na osciloskopu [8] jsem postupně zjišťoval, kde chyba vzniká. Zjistil jsem, že při detekci krátkého rámce nemám v době vyhodnocování jeho hodnoty na vstupu entity CommandIn ještě přenesený poslední bit z entity SequenceDecoder. Problém jsem vyřešil tím, že jsem mezi tyto dvě entity přidal jeden řídící signál nazvaný *willBeWUPA*, který je v logické jedničce právě když je detekován krátký rámec a v příštím taktu by se na výstupu entity CommandIn objevila hodnota WUPA. De facto v entitě SequenceDecoder testuji zvláštní případ hodnot, které v entitě CommandIn zpracovávám obecně. Jelikož mám v entitě SequenceDecoder dané hodnoty k dispozici o takt dřív, mohu tuto skutečnost signalizovat pomocí výše zmíněného řídícího signálu.

2.4.18 Oprava SAK

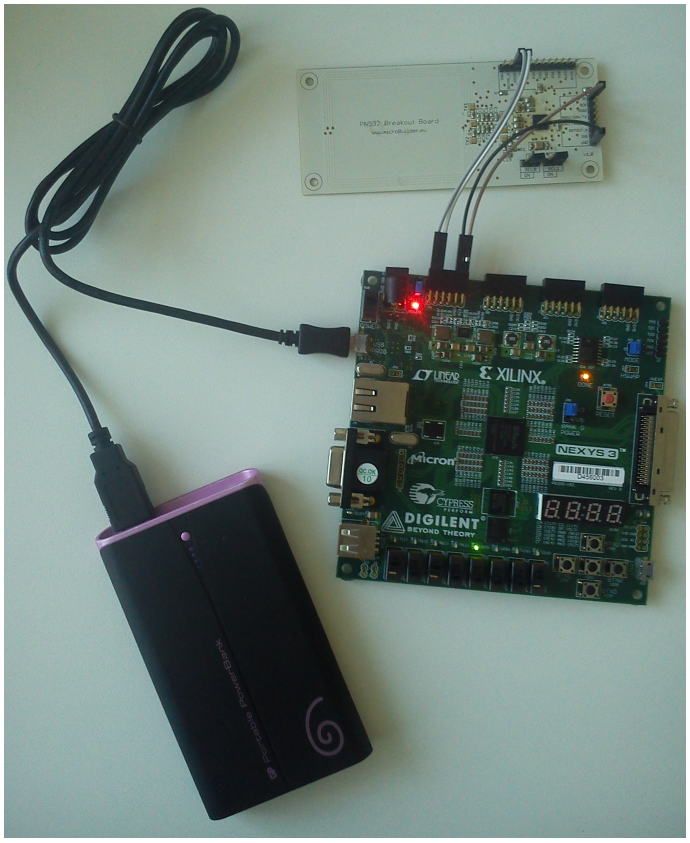
V reakci na zjištění během testování vůči karetním systémům ČVUT, které podrobněji popisuje jiná podkapitola 3.2.1, jsem musel provést drobné úpravy v entitě ConfMux. Ty spočívají v rozšíření konfigurovatelných informací také o podobu odpovědí SAK a ATQA. Současně s tím byla entita rozšířena o řídící signál, zda se mají tyto hodnoty použít či ne, jelikož tyto signály jsou na výstupu bez ohledu na hodnoty konstant. V případě, že je hodnota konstanty *confOnlyUIDx* na úrovni logické jedničky, pak emulátor použije hodnoty vyplývající pouze u normy ISO/IEC 14443 [5]. Tyto hodnoty však dle novějších

standardů oznamují, že karta je typu Mifare Ultralight [29] [30], což vede k dále zmíněným problémům.

2.4.19 Shrnutí

Výsledná realizace zabírá v FPGA čipu na zvolené desce velmi málo dostupných prostředků. Deska Nexys3 je osazena čipem z rodiny Spartan6 s označením XC6SLX16 v pouzdru CSG324C [7]. Nejvyšší procento využití má parametr využitých sliců, kterých návrh využívá 14 % z dostupných. Celý návrh by se tak s přehledem vešel jak do co se prostředků týče mnohem menšího FPGA – například XC3S50A [31] z rodiny Spartan-3A. Celý návrh by se také vešel i do fyzicky mnohem menšího FPGA čipu, například XC6SLX4 [17] z rodiny Spartan6 v pouzdru CPG196 o fyzických rozměrech 8x8 mm.

Zmenšení desky s FPGA by bylo vhodné také z toho důvodu, že v současnosti se právě deska Nexys3 největší částí celého systému. Při testování mě začínalo vadit, že s sebou musím brát notebook nejen jako programátor, ale také jako zdroj napájení pro FPGA. Proto jsem konfiguraci – bitstream – nahrál do paměti na desce s FPGA a jako zdroj použil zapůjčenou power banku. V současnosti tedy celý systém pro emulaci karty vypadá takto 2.13.



Obrázek 2.13: Současná podoba celého emulátoru.

Kapitola 3

Testování

Celý návrh jsem poměrně podrobně testoval během celé fáze realizace. Po dokončení realizace, kdy byly používané čtečky úspěšně schopné provést s emulovanou kartou příkaz SELECT, jsem se rozhodl emulátor otestovat v praxi – na systémech používaných na ČVUT. Při tomto testování jsem narazil na spoustu problémů spojených především s novější implementací protokolu ve čtečkách, které zohledňují další dokumentace různých typů karet. Tyto jednotlivé dokumentace nejsou zdaleka tak obecné jako norma ISO/IEC 14443 [5], a proto jsem se jimi při návrhu a realizaci vůbec nezabýval. Současně jsou v naprosté většině už poměrně konkrétní a popisují funkcionalitu jednotlivých typů bezkontaktních čipových karet. Cílem mé práce je vytvořit obecný emulátor bezkontaktní karty typu A dle dříve zmíněné normy.

3.1 Průběžné testování během realizace

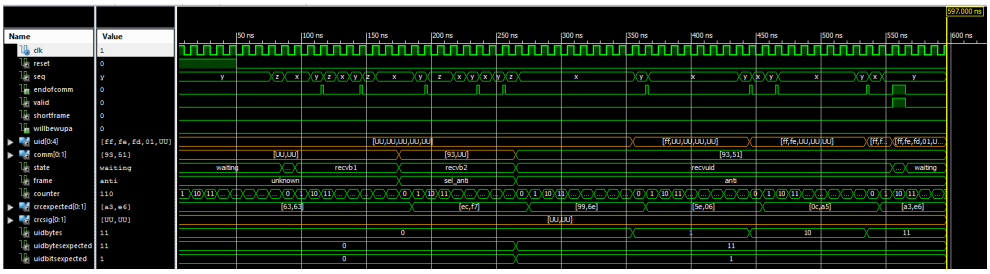
Pro testování během vývoje jsem využíval více způsobů. Všechny použité způsoby podrobněji popisuji v následujících podkapitolách. Obecně jsem se při realizaci řídil pravidlem testovat co nejčastěji co nejmenší bloky tak, abych v případě zanesení chyby mohl tuto chybu snadno nalézt a opravit.

3.1.1 Behaviorální simulace

Zaprvé jsem průběžně během psaní implementace testoval některé entity. Testování entit jsem prováděl pomocí behaviorální simulace, přičemž jsem do testbenchů zadal typické průběhy vstupních signálů a kontroloval výstupy ručně přímo v programu *iSim*. Program *iSim* je součástí instalace ISE Design Suite [22]. Aserce jsem pro kontrolu průběhů nepoužíval. Aserce je v testbenchích použitá pouze k ukončení simulace se zprávou *DONE*.

V případě entity SequenceDecoder je testbench poměrně složitý. Potřeboval jsem totiž zjistit, jak se entita chová na různé poměrně dlouhé streamy vstupních sekvencí. Z toho důvodu jsem program v jazyce C pro výpočet CRC

rozšířil o generování části testbenchce v mnou používaném formátu. Program potom podle zadaných vstupních dat a CRC z vlastního výpočtu vygeneruje text, který pouze zkopíruji do testbenche entity SequenceDecoder. Průběh behaviorální simulace entity SequenceDecoder lze vidět na obrázku 3.1.



Obrázek 3.1: Průběh behaviorální simulace entity SequenceDecoder

Současné verze entit již nemusí nutně reagovat na své testbenche jako při původním testování. Při složení všech komponent emulátoru dohromady jsem totiž musel ošetřit několik drobných chyb vyplynuvších až s větší komplexností celého systému. Vždy se jednalo o zpoždění či předstih některého řídícího signálu o jeden takt hodin, a to ať už o frekvenci 100 MHz nebo o frekvenci bitové rychlosti. Tyto drobné chyby pro mě bylo velmi složité odhalit již v průběhu návrhu, jelikož tato práce je moji první větší praktickou zkušeností s návrhem systému s více hodinovými doménami.

3.1.2 Zjednodušující verze entit

Další metodou použitou k testování je napsání zjednodušujících entit na vyšší úrovni abstrakce. Jako příklad mohu uvést entitu SimpleSeqChanger, která měla na vstupu pouze vstupní sekvenci (X,Y,Z) a na výstupu pak výstupní sekvenci (D,E,F). Jednoduchou bijekcí vstupních sekvencí na výstupní jsem simuloval velmi primitivní (a samozřejmě nesprávné) zpracování přijímaných signálů. Nicméně tento přístup mi umožnil již v rané fázi implementace otestovat přímo v hardwaru entity na nejnižší úrovni abstrakce – Decoder a Encoder. Na osciloskopu pak bylo dobře vidět, jak se hodnota na signálu `SIGOUT` přenáší na signál `SIGIN`.

3.1.3 Testování na osciloskopu

Při testování obecně pro mě byl ohromným přínosem osciloskop [8], který mi pomohl přesně se zorientovat v časových průbězích jednotlivých signálů. Především při práci s hardwarem, který aktivně pracuje pouze v elektromagnetickém poli čtečky, kterým je současně řízen, je tato možnost neocenitelná. Pro pohodlnější použití osciloskopu jsem k vybraným entitám přidal některé

výstupní signály čistě z toho důvodu, abych je pak mohl vyvést na výstupy desky Nexys3. Typickým příkladem vyvedených signálů byly například signály zachycující stavy automatů v emulátoru. Potom jsem pomocí zobrazení průběhu stavu automatu, hodinových signálů a vstupně–výstupních signálů přípravku PN532 poměrně snadno identifikoval místo a způsob, jakým se můj emulátor odchyluje od normy.

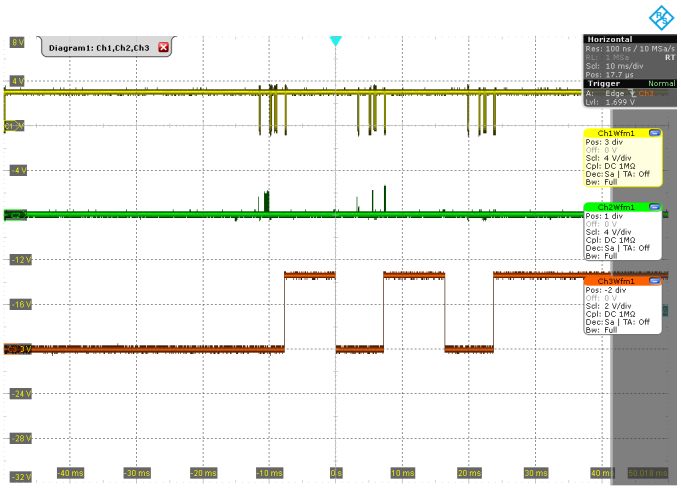
Nejzávažnější takto zjištěnou chybou během realizace bylo chybné zpracování příkazu WUPA. Při průběžném testování jsem podcenil fakt, že příkazy REQA a WUPA se liší v posledním datovém bitu. Dotčenou entitu SequenceDecoder jsem otestoval behaviorálně z příkazů používajících krátký rámec pouze na příkaz REQA. Poslední datový bit příkazu REQA je nulový, zatímco příkaz WUPA má poslední datový bit v úrovni logická jedna. Chyba se u příkazu REQA neprojevila, jelikož logická hodnota nula je v streamu bitů posílaném na výstup již po resetu přítomna. Opravu této chyby podrobně popisuje podkapitola „Oprava WUPA 2.4.17“.

3.2 Testování konečné implementace

Ve chvíli, kdy jsem implementaci považoval za dokončenou, jelikož na čtečku Cardman [19] emulátor úspěšně reagoval až do stavu ACTIVE – tedy úspěšného výběru karty příkazem SELECT – jsem chtěl emulátor otestovat vůči karetním systémům na ČVUT. Jako první se nabízela čtečka u dveří pro vstup do různých místností. Pomocí aplikace [25] v tabletu [20] jsem tedy přečetl UID karty oprávněné ke vstupu a zadal do konfiguračního souboru. Při pokusu o přečtení takto emulované karty aplikace v tabletu však zobrazila chybovou zprávu s informací o neočekávaném odebrání karty z pole čtečky. Rozhodl jsem se příčinu zjistit pomocí vyvedení signálů na sondy osciloskopu a pozorováním. Výsledek zjištění popisuje následující kapitola a reakci provedenou v realizaci pak jiná podkapitola 2.4.18.

3.2.1 Testování správného SAK

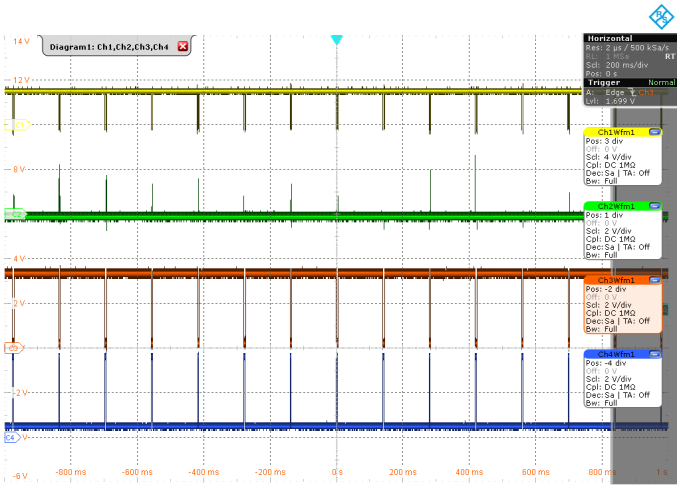
Po zjištění, že aplikace v tabletu má problém s přečtením emulované karty, jsem emulátor otestoval proti čtečce u dveří. Ta reagovala podle svícení LED diod, na které jsou vyvedeny stavy automatů, stejně jako čtečka v tabletu. I přes jednoznačné dosažení stavu ACTIVE emulované karty čtečka neúplně identifikovala kartu. To jsem poznal tak, že čtečka nepípla, jak to provádí při detekování karty ať už umožněním přístupu či nikoli. Po připojení emulátoru k oscilátoru jsem nastavil trigger tak, aby reagoval na opuštění stavu ACTIVE. Vyčetl jsem, že čtečka kartu po prvním dosažení stavu ACTIVE uspí příkazem HALT a následně probudí příkazem WUPA a vybere příkazem SELECT. Tato smyčka se ještě jednou zopakuje, nicméně napotřetí namísto příkazu HALT karta obdrží neznámý příkaz, na který reaguje resetem. Průběh je vidět na obrázku 3.2.



Obrázek 3.2: Průběh stavů při emulaci SAK = 0x00. CH1 = Vstupní signál; CH2 = CLKbit; CH3 = stav == ACTIVE*

Z průběhu signálu `SIGOUT` jsem vyčetl, že onen neznámý příkaz má hodnotu 0x3002 následovaný dle normy dvěma byty CRC. Při vyhledávání hodnoty příkazu jsem narazil na dokumentaci popisující chování karet Mifare Ultralight, kde tento příkaz slouží pro přečtení 4 po sobě jdoucích datových bloků od bloku číslo 2 dále [29] [30]. Ve správnosti této domněnky mě utvrdil také fakt, že aplikace v tabletu emulovanou kartu určila právě jako typ Mifare Ultralight. Hledal jsem tedy dále a v jiné dokumentaci [32] našel specifikaci bitů v odpovědi SAK, které jsou v normě ISO/IEC 14443 [5] určeny jako rezervované pro budoucí použití. Zkusil jsem na radu vedoucího tedy emulátoru nastavit také vlastní hodnotu odpovědí SAK a ATQA, přičemž jsem použil hodnoty vyčtené čtečkou z reálných karet. V případě odpovědi SAK je však nutné případně nulovat bit určující podporu protokolu dle čtvrté části normy.

Ukázalo se, že jakmile hodnotu SAK nastavím na 0x08, čtečka nabude domnění, že se jedná o Mifare Classic [33]. Tento fakt vyústí v to, že se čtečka pokusí přečíst datové bloky karty, načež můj emulátor nereaguje žádnou odpovědí. Čtečka tedy nabude dojmu, že přístupové klíče, které ona zná (zkouší klíče defaultní), nejsou správné a proto není schopna data přečíst. Nicméně kartu jako takovou úspěšně detekuje. V tomto případě vnitřní stav karty cyklí mezi stavy ACTIVE*, HALT, READY*, ACTIVE*,... podle obrázků 3.3 a 3.4.

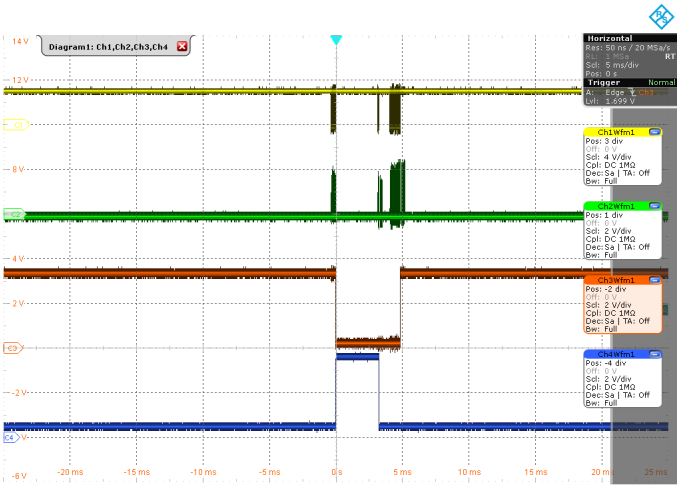


Obrázek 3.3: Průběh stavů při emulaci SAK = 0x08. CH1 = Vstupní signál; CH2 = CLKbit; CH3 = stav == ACTIVE*; CH4 = stav == HALT

3.2.2 Testování proti nasazeným systémům

Po úpravě odpovědi SAK dle předchozí kapitoly se mi daří úspěšně emulovat kartu nejen při použití čtečky Cardman [19], aplikace v tabletu [25] ale také při použití s přístupovými systémy na ČVUT či s platebními systémy v jeho menzách. Při emulování libovolného UID vždy testovaný systém detekuje kartu – reaguje pípnutím či textem na displeji – a v případě správného UID také otevře dveře případně zobrazí zůstatek na přednabytém účtu apod. Videoukázky této funkčnosti jsou k dispozici na přiloženém CD.

Chyby objevené testováním během realizace byly vždy úspěšně opraveny. Výsledný emulátor tedy prošel úspěšně všemi navrženými a současně zde popsanými testy včetně závěrečného praktického ověření proti systémům pracujícím s bezkontaktními kartami na ČVUT.



Obrázek 3.4: Průběh stavů při emulaci SAK = 0x08. CH1 = Vstupní signál; CH2 = CLKbit; CH3 = stav == ACTIVE*; CH4 = stav == HALT

Kapitola 4

Budoucí práce

Systém je v současnosti poměrně obtížně konfigurovatelný. Pro změnu údajů karet určených k emulaci je třeba upravit zdrojový kód v jazyce VHDL a znovu podstoupit celou cestu od syntézy zdrojového kódu syntézním nástrojem až po vygenerování konfiguračního bitstreamu pro FPGA. Jedním z možných řešení je využití možnosti komunikace po sériové lince, kterou deska Nexys3 již emuluje na jednom z micro USB portů. Vzhledem k velmi širokým možnostem hodnoty UID by byla konfigurace pomocí ovládacích prvků – vstupů – přítomných na desce nepříliš uživatelsky přívětivá. I když použití přepínačů pro zadávání hodnot jednotlivých bytů, tlačítek pro jejich přijímání resp. umisťování v rámci UID a informaci o průběhu zadávání zobrazenou na sedmisegmentovém displeji si také ještě dokáži představit.

Co se týče emulace samotné, je zde zajisté ukrytý potenciál. Bylo by vhodné implementovat také část protokolu popsanou ve čtvrté části normy ISO/IEC 14443 [5] a případně také navazující protokoly/příkazy některých vybraných typů karet, např. rozšířené Mifare DESFire EV1. Zajisté by bylo vhodné rozšířit emulátor také o možnost emulovat karty typu B, avšak tento typ nepodporuje nyní používaný přípravek PN532. Tento krok by tedy vyžadoval využití jiné součástky jako antény.

Další možnosti na rozšíření emulátoru spočívají v teoretické možnosti jeho využití při útoku na bezkontaktní karty. Mnou uvažovaný typ útoku se nazývá *relay attack* [34] a spočívá v tom, že komunikaci ve skutečnosti vzdálené čtečky a karty umožňují dvě spolu na dálku komunikující proxy zařízení umístěné jedno u čtečky a druhé u karty. Tyto proxy tak nahrazují jinak nutnou blízkost čtečky a karty. Skutečnost, že můj emulátor je na rozdíl od jiných obdobných řešení implementován téměř výhradně v hardwaru, může být teoreticky výhodou k získání odolnosti vůči některým ochranným opatřením proti *relay attackům*.

Závěr

Povedlo se do FPGA naimplementovat emulátor karty typu A dle protokolu v normě ISO/IEC 14443 až po její třetí část. Část protokolu sloužící především k zahájení komunikace dle proprietárního protokolu, která je popsána ve čtvrté části normy, nebyla implementována. Taktéž nebyla implementována emulace karet typu B. Hlavním důvodem k neimplementaci typu B byla nepodpora této komunikace ze strany přípravku PN532 Breakout board, který slouží pro emulátor v FPGA jako anténa pro příjem a vysílání dat. Kromě přípravku PN532 byla k implementaci použita deska Nexys3 s FPGA čipem XC6SLX16.

Emulátor umí emulovat obecnou část normy bez přihlédnutí k dalším dokumentacím jednotlivých typů karet. Také je možné emulovat kartu včetně dalších vlastních parametrů mimo samotné UID. Tato druhá možnost má za následek schopnost emulovat úspěšně jakýkoliv bezkontaktní tag ať už v podobě karty, čipu či jiné v bezkontaktních systémech provozovaných na ČVUT. Emulátor byl úspěšně otestován vůči čtečkám u turniketů, dveří do místností i transakčnímu systému nasazeném v menzách.

Údaje emulované karty (především UID) je nutné zadat přímo do zdrojového kódu implementace a z návrhu v jazyce VHDL postupnými kroky v návrhovém nástroji vygenerovat konfigurační bitstream. Do budoucna by bylo vhodné toto změnit buď implementací konfiguračního zadávání přímo pomocí prvků desky s FPGA čipem nebo spíše pomocí komunikace po sériové lince či obdobným způsobem. Manuál k použití je přílohou této práce.

Emulace bezkontaktní karty téměř výhradně hardwarovou cestou nabízí do budoucna nové potenciální možnosti v oblasti relay útoků na bezkontaktní transakční technologii.

Literatura

[1] Boon, L. S.: NFC (Near Field Communication), RFID. [online], [cit. 7. 5. 2016]. Dostupné z: http://www.siongboon.com/projects/2012-03-03_rfid/

[2] Westhues, J.: A Test Instrument for HF/LF RFID. [online], 2009, [cit. 7. 5. 2016]. Dostupné z: http://www.cq.cx/proxmark3.pl

[3] Kruse, N.: Simple NFC. [online], 2013, [cit. 7. 5. 2016]. Dostupné z: http://blog.nonan.net/2013/11/simple-nfc.html

[4] Kasper, T.: Chameleon Project. [online], [cit. 7. 5. 2016]. Dostupné z: https://github.com/emsec/ChameleonMini/wiki

[5] International Organization for Standardization: *ISO/IEC 14443 – Identification cards – Contactless integrated circuit cards – Proximity cards.* 2001.

[6] NXP Semiconductors: *UM0701-02 – PN532 User Manual.* 2007, virtual card mĂŁd od str. 89, [cit. 7. 5. 2016]. Dostupné z: http://www.nxp.com/documents/user_manual/141520.pdf

[7] Digilent: *Nexys3 Board Reference Manual.* [rev. 3. 4. 2013], [cit. 7. 5. 2016]. Dostupné z: http://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPNexys3/documentation/Nexys3_rm.pdf

[8] Rhode & Schwarz: *R&SÂ®RTO Digital Oscilloscope User Manual.* [cit. 7. 5. 2016]. Dostupné z: http://www.naic.edu/~phil/hardware/oscilloscopes/RTO_UserManual_en.pdf

[9] Xilinx: *Xilinx CORE Generator System.* [cit. 7. 5. 2016]. Dostupné z: http://www.xilinx.com/tools/coregen.htm

[10] Chhabra, T.; Chindaphorn, P.: Smart Cards. [online], 2004, santa Clara University, Department of Computer Engineering, [cit. 7. 5. 2016]. Dostupné z: http://www.cse.scu.edu/~jholliday/COEN150Sp04/projects/Smart%20Cards.doc

[11] Neznámý: Uses of Smart Cards. [online], 1997, massachusetts Institute of Technology, [cit. 7. 5. 2016]. Dostupné z: http://web.mit.edu/ecom/Spring1997/gr12/2USES.HTM

[12] Macich, J.: ČTÚ: Aktivních SIM karet je v ČR více neĺ obyvatel, ale počet již neroste. [online], 2013, channelWorld.cz, [cit. 7. 5. 2016]. Dostupné z: http://channelworld.cz/analyzy/ctu-aktivnich-sim-karet-je-v-cr-vice-nez-obyvatel-ale-pocet-jiz-neroste-8756

[13] Financninoviny.cz: Počet platebních karet v ČR loni stoupl o procento na 10,25 mil. [online], 2014, financninoviny.cz, [cit. 7. 5. 2016]. Dostupné z: http://www.finance.cz/zpravy/finance/412319-pocet-platebnich-karet-v-cr-loni-stoupl-o-procento-na-10-25-mil/

[14] International Organization for Standardization: *ISO/IEC 13239 – Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*. 2002.

[15] GlobalPlatform: Global Platform Value Proposition for Remote Post-Issuance Secure Access Modules (SAM) Management. [online], 2011, [cit. 7. 5. 2016]. Dostupné z: http://www.globalplatform.org/%5C/documents/globalplatform_remote_sam_white_paper_nov2011.pdf

[16] Shannon, C. E.: A Mathematical Theory of Communication. [online], 1948, [cit. 7. 5. 2016]. Dostupné z: http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf

[17] Xilinx: *Spartan-6 Family Overview*. Verze 2.0, [rev. 25. 10. 2011], [cit. 7. 5. 2016]. Dostupné z: http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

[18] Marshall, D.: Nyquist's Sampling Theorem. [online], 2001, [cit. 7. 5. 2016]. Dostupné z: https://www.cs.cf.ac.uk/Dave/Multimedia/node149.html

[19] HID Global: *5321 CL USB*. [cit. 7. 5. 2016]. Dostupné z: https://www.hidglobal.com/products/readers/omnikey/5321-cl

[20] ASUSTeK Computer: *Detailed Technical Datasheet of Google Nexus 7C ME370TG 32GB (Asus Tilapia)*. [cit. 7. 5. 2016]. Dostupné z: http://pdadb.net/index.php?m=specs&id=3968&view=1&c=google_nexus_7c_me370tg_32gb_asus_tilapia

[21] Neundorf, A.: CuteCom. [online], verze 0.21.0, [rev. 11. 5. 2009], [cit. 7. 5. 2016]. Dostupné z: http://cutecom.sourceforge.net/

[22] Xilinx: *ISE Design Suite*. [cit. 7. 5. 2016]. Dostupné z: http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm

[23] Digilent: *Digilent, Adept*. 2011, [cit. 7. 5. 2016]. Dostupné z: http://store.digilentinc.com/digilent-adept-2-download-only/

[24] NetBeans.org: *NetBeans IDE*. [cit. 7. 5. 2016]. Dostupné z: https://netbeans.org/features/index.html

[25] "NFC Research Lab": *NFC TagInfo*. Verze 1.12a, [rev. 25. 9. 2014], [cit. 7. 5. 2016]. Dostupné z: https://play.google.com/store/apps/details?id=at.mroland.android.apps.nfctaginfo&hl=cs

[26] nfc-tools.org: *Libnfc*. [cit. 7. 5. 2016]. Dostupné z: http://nfc-tools.org/index.php?title=Libnfc

[27] graphviz.org: *Graphviz*. [cit. 7. 5. 2016]. Dostupné z: http://www.graphviz.org/Home.php

[28] Microsoft: *Microsoft Visio 2013*. [cit. 7. 5. 2016]. Dostupné z: https://www.dreamspark.com/

[29] NXP Semiconductors: *AN1303 MIFARE Ultralight as Type 2 Tag*. Verze 1.5, [rev. 2. 10. 2012], [cit. 7. 5. 2016]. Dostupné z: http://www.nxp.com/documents/application_note/AN1303.pdf

[30] NFC forum: *Type 2 Tag Operation Specification*. Verze 1.5, [rev. 2. 10. 2012], [cit. 7. 5. 2016]. Dostupné z: http://apps4android.org/nfc-specifications/NFCForum-TS-Type-2-Tag_1.1.pdf

[31] Xilinx: *Spartan-3a Family Overview*. Verze 2.0, [rev. 19. 8. 2010], [cit. 7. 5. 2016]. Dostupné z: http://www.xilinx.com/support/documentation/data_sheets/ds529.pdf

[32] NXP Semiconductors: *AN10833 MIFARE Type Identification Procedure*. Verze 3.5, [rev. 27. 3. 2014], [cit. 7. 5. 2016]. Dostupné z: http://cache.nxp.com/documents/application_note/AN10833.pdf

[33] NXP Semiconductors: *MIFARE Classic*. [cit. 7. 5. 2016]. Dostupné z: http://www.nxp.com/products/identification-and-security/smart-card-ics/mifare-ics/mifare-classic:MC_41863

[34] Francis, L.; Hancke, G.; Mayes, K.; aj.: Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. [online], 2005, information Security Group, Smart Card Centre, [cit. 7. 5. 2016]. Dostupné z: https://eprint.iacr.org/2011/618.pdf

Příloha **A**

Uživatelská příručka

Emulátor karet zvládá emulovat bezkontaktní karty typu A dle normy ISO/IEC 14443 [5] vyjma části protokolu popsané ve čtvrté části normy. Celý systém emulátoru se skládá z přípravku PN532 Breakout board ve funkci antény, desky Nexys3 s FPGA čipem ve funkci bezkontaktního čipu a volitelně některé power banky jako snadno přenosného a dostupného zdroje.

Otevřeme projekt emulátoru v ISE Design Suite [22] a pro úpravy soubor confMux.vhd. V tomto souboru nalezneme 4 skupiny stejných konstant lišících se v názvu pouze v číslici na konci (0–3). Do těchto konstant vepíšeme požadované hodnoty (nakonfigurovat můžeme až 4 karty současně).

Ujistíme se, že jumpery na desce Nexys3 odpovídají tomuto nastavení (také na obrázku A.1):

- Jumper J8 je nasazen v levé polovině (M0).
- Jumper JP1 je nasazen na dolních dvou pinech (USB).
- Jumper JP8 je nasazen na horních dvou pinech (3V3).

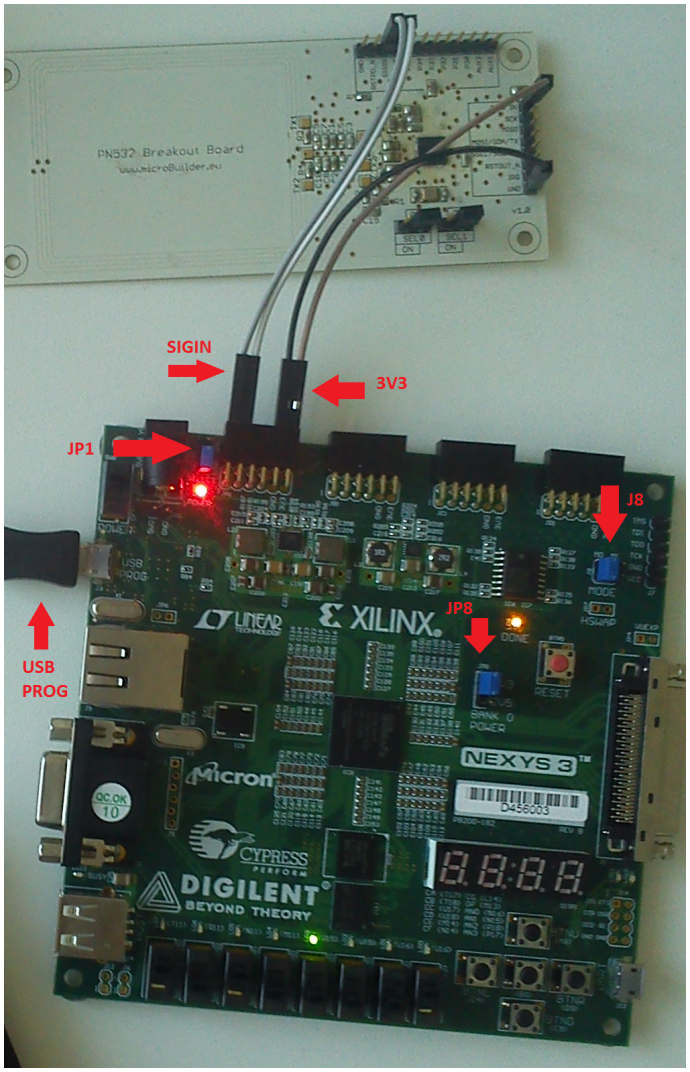
Pak se ujistíme, že přípravek PN532 je nakonfigurovaný do módu *virtual card*. Konfigurace se ukládá do nonvolatilní paměti – přípravek si konfiguraci pamatuje i po odpojení napájení. Pokud tomu tak není, navážeme s přípravkem spojení přes sériovou linku s parametry:

- Baudrate 115200
- Datových bitů 8
- Stop bitů 1
- Bez handshaku

Poté pošleme následující byty:

```
0x00 0x00 0xFF 0x05 0xFB 0xD4 0x14 0x02 0x00 0x00 0x16 0x00
```

Poté spojíme 4 drátky PN532 s deskou Nexys3 podle obrázku A.1. Tedy signál `SIGIN` přípravku PN532 napojíme na nejlevější horní pin nejlevějšího konektoru *Pmod* (*PmodA*) desky Nexys3. Signál `SIGOUT` napojíme hned vedle vpravo. Výstupy na pravém okraji konektoru *Pmod* označené zkraje 3V3 a GND napojíme na odpovídající piny na levém okraji přípravku PN532.



Obrázek A.1: Zapojení součástí emulátoru.

Poté desku Nexys3 pomocí levého micro USB konektoru připojíme k počítači a desku spustíme vypínačem hned u konektoru. Pomocí softwaru Adept v záložce Memory vybereme vygenerovaný konfigurační bitstream main.bit,

v okně na pravé straně pak zvolíme možnost SPI Flash a stiskneme tlačítko Program. Po dokončení programování můžeme v případě, že máme jiný zdroj (např. power banku) počítač odpojit a stejným konektorem připojit právě tento zdroj.

Emulátor je v tuto chvíli funkční a pomocí páru přepínačů 7 a 6 (nejlevější) volíme aktuálně používané konfigurační hodnoty (SW7 je významnější bit). Na LED diodách s čísly 0–2 (nejpravější) můžeme pozorovat aktuální stav karty. Nejvýznamnějšími stavy jsou IDLE – nesvítí nic, ACTIVE – svítí současně LED0 a LED1 – a ACTIVE* – svítí LED2.

Příloha B

Seznam použitých zkratek

ATQA Answer To Request, Typ A

BCC UID CLn check byte, počítaný jako XOR 4 předchozích bytů, Typ A

CT Cascade Tag, Typ A

CRC Cyclic Redundancy Check

FDT Frame Delay Time, Typ A

FPGA Field-Programmable Gate Array

IEC International Electrotechnical Commission

ISO International Organization for Standardization

IP Core Intellectual Property Core

HLTA Příkaz Halt, Typ A

LSB Least Significant Bit

MIFARE Mikron FARE Collection System

MSB Most Significant Bit

NVB Number of Valid Bits, Typ A

NXP NXP Semiconductors

Pmod Peripheral Module

REQA Příkaz Request, Typ A

SAK Select AcKnowledge, Typ A

SELECT Příkaz Select, Typ A

XOR Exclusive OR

UID Unique Identifier, Typ A

USB Universal Serial Bus

VHDL VHSIC Hardware Description Language

VHSIC Very High Speed Integrated Circuits

WUPA Příkaz Wake-UP, Typ A

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `main.bit` ..... výsledný bitstream
- `src`
  - `impl` ..... zdrojové kódy implementace včetně testbenche
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS
- `videa` ..... videa z testování emulátoru vůči systémům ČVUT