VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY

DEPARTMENT OF MICROELECTRONICS

VERIFIKACE DIGITÁLNÍHO OBVODU MICROCORE GNSS BASEBAND

VERIFICATION OF DIGITAL CIRCUIT MICROCORE GNSS BASEBAND

DIPLOMOVÁ PRÁCE

MASTER´S THESIS

AUTOR PRÁCE — Bc. ONDŘEJ PEROUTKA

AUTHOR

VEDOUCÍ PRÁCE — Ing. VOJTĚCH DVOŘÁK

SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Mikroelektronika**
Ústav mikroelektroniky

Student: ***Bc.*** Ondřej Peroutka ***ID:*** 164358
Ročník: 2 ***Akademický rok:*** 2017/2018

NÁZEV TÉMATU:

Verifikace digitálního obvodu Microcore GNSS Baseband

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce se seznamte s metodikou UVM pro verifikaci integrovaných obvodů. Navrhněte verifikační prostředí pro digitální obvod Microcore GNSS Baseband (dle potřeby na úrovni jednotky nebo na úrovni menších celků) a implementujte v jazyce SystemVerilog. V závěru práce uveďte výhody a nevýhody zvoleného řešení a diskutujte možná rozšíření. Při návrhu i realizaci uvažujte pokrytí testovaného kódu. Pro realizaci návrhu lze použít sadu skriptů a již existujících komponent v databázi společnosti Honeywell.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 22.5.2018

Vedoucí práce: Ing. Vojtěch Dvořák
Konzultant: Ing. Tomáš Šulc

doc. Ing. Lukáš Fujcik, Ph.D.
předseda oborové rady

UPOZORNĚNÍ:

Abstrakt:

Tématem této diplomové práce je verifikace Akviziční jednotky a Sledovací jednotky digitálního obvodu Microcore GNSS Baseband společnosti Honeywell. Teoretická část práce obsahuje stručný úvod o určování polohy pomocí satelitního signálu, princip činnosti verifikovaných jednotek a představení metodiky UVM. Praktická část práce obsahuje požadavky na testované jednotky, testové scénáře a procedury. Také je popsáno verifikační prostředí. Poslední částí je průběh verifikace a její výsledky.

Abstract:

The topic of the master´s thesis is to verify Acquisition Engine and Tracking Engine in the Microcore GNSS Baseband digital circuit from Honeywell. Theoretical part contains a brief introduction into the satellite position determination, basic principles of the verified blocks is given and UVM methodology is introduced. Practical part contains requirements, test cases and test procedures. The verification environment is also described. In the last part of the thesis is the verification process and it´s results.

Klíčová slova:

Funkční pokrytí, GNSS, SystemVerilog, UVM, verifikace

Keywords:

Functional coverage, GNSS, SystemVerilog, UVM, verification

Bibliografická citace

PEROUTKA, O. *Verifikace digitálního obvodu Microcore GNSS Baseband*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 42 s. Vedoucí diplomové práce Ing. Vojtěch Dvořák.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma **„Verifikace digitálního obvodu Microcore GNSS Baseband“** jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne .....

.....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Vojtěchu Dvořákovi za jeho pedagogickou pomoc. Dále děkuji Ing. Tomáši Šulcovi za jeho konzultace a připomínky ke zpracování této práce. Dále děkuji Ing. Marku Santovi za jeho cenné rady ohledně verifikace a metodiky UVM.

V Brně dne .....

.....
podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této diplomové práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Počátek satelitní navigace se datuje na začátek 60. let minulého století, kdy se vládní organizace USA začaly zajímat o vytvoření satelitního systému pro určení polohy. První takový systém byl spuštěn v roce 1964 pod názvem Transit. Tento systém však potřeboval k určení polohy čas v řádu desítek minut, který navíc také záležel na zeměpisné šířce. Tento systém byl dostatečný pouze pro určení polohy cílů s nízkou rychlostí pohybu, například lodě a podobně. Bylo tedy nutné vytvořit lepší navigační systém. Tím se později stal systém známý jako GPS – globální navigační systém. Tento systém v současnosti již splňuje požadavky na ideální satelitní navigační systém: globální pokrytí, kontinuální použití za jakéhokoliv počasí, přesnost a schopnost určit polohu i rychle se pohybujících cílů [1]. Původně byl satelitní navigační systém určen pouze pro armádu. Dnes je navigační systém dostupný i pro civilní účely.

Tato diplomová práce je zaměřena na digitální obvod Microcore GNSS Baseband společnosti Honeywell. Cílem diplomové práce je verifikace Akviziční jednotky a Sledovací jednotky v tomto digitálním obvodu. Práce je rozdělena 8 kapitol. První kapitola se věnuje základním principům určení polohy, architektuře satelitního systému a přehledu současných satelitních systémů. Druhá kapitola se věnuje principu funkce testovaných jednotek. Třetí kapitola obsahuje stručné představení metodiky UVM a typického verifikačnímu prostředí realizovaného pomocí této metodiky. Čtvrtá kapitola obsahuje požadavky na testované jednotky. V páté kapitole je sepsán přehled testovacích scénářů. Šestá kapitola obsahuje přehled testových procedur. Verifikační prostředí je popsáno v šesté kapitole. Výsledky verifikace jsou v osmé kapitole.

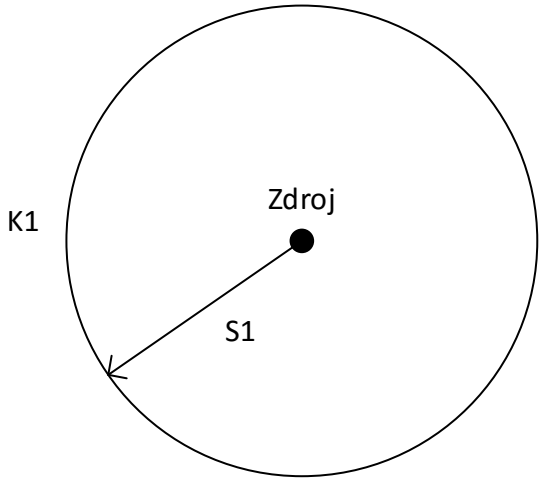
1 Globální navigační satelitní systém

Satelitní navigační systém používá pro určení pozice uživatele koncept TOA (Time of Arrival) [1]. Tento koncept je založen na měření času, který potřebuje signál k překonání vzdálenosti od zdroje tohoto signálu k přijímači. Tuto naměřenou hodnotu času poté stačí vynásobit rychlostí šíření signálu a tím získáme hodnotu vzdálenosti od zdroje signálu. Níže je uveden příklad pro určení polohy tímto způsobem ve dvojrozměrném systému.

Předpoklady jsou následující:

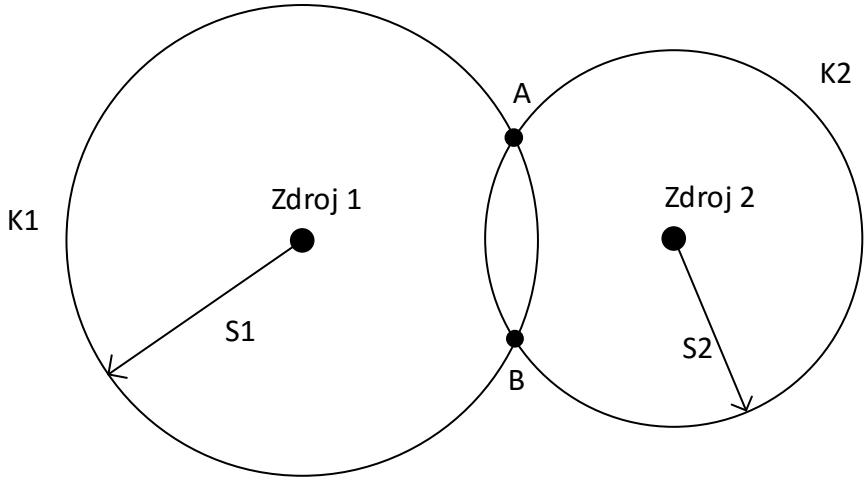
- je známa přibližná poloha vysílačů signálu,
- signál je vyslán v přesně stanovený čas ze všech zdrojů signálu,
- hodiny vysílačů i přijímače jsou synchronizovány.

Jakmile signál dorazí k přijímači, je vypočten uplynulý čas, který je vynásoben rychlostí šíření signálu. Tím se získá vzdálenost od zdroje signálu. Pokud je k dispozici pouze jeden zdroj signálu, tak tímto způsobem je možná poloha přijímače v jakémkoliv bodě na kružnici K1 s poloměrem vypočtené vzdálenosti S1 a středem této kružnice je zdroj signálu. Tato situace je zobrazena na Obr. 1.



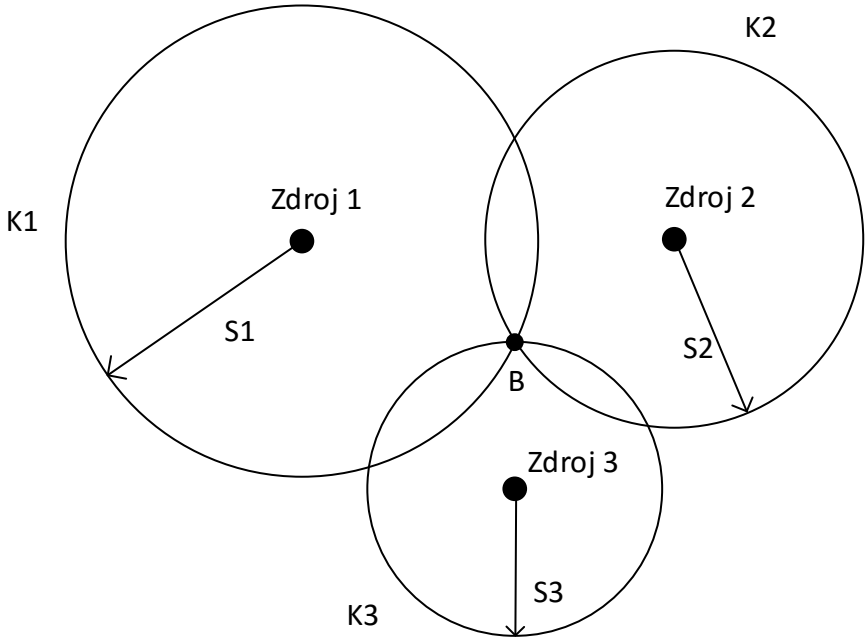
Obr. 1: Určení pozice s jedním zdrojem signálu

Se dvěma zdroji signálu bude pozice přijímače ve vzdálenosti S1 od zdroje signálu 1 a zároveň ve vzdálenosti S2 od zdroje signálu 2, tedy na průsečících A B kružnic K1 a K2. Tím se počet možných poloh přijímače značně redukuje. Tato situace je zobrazena na Obr. 2.



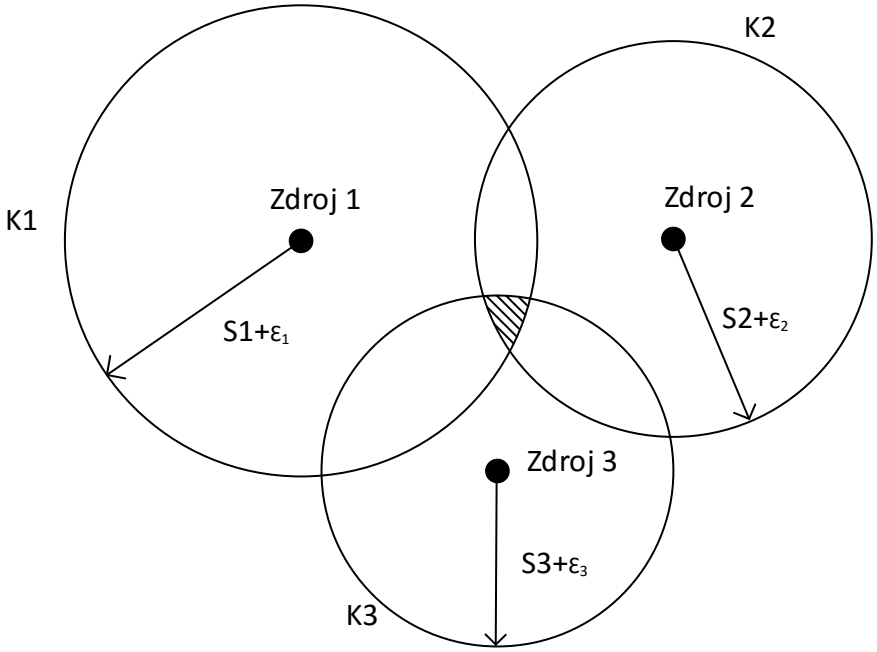
Obr. 2: Určení pozice se dvěma zdroji signálu

Přidáním dalšího zdroje signálu je již pozice přijímače jasně určena, jak je zobrazeno na Obr. 3.



Obr. 3: Určení pozice s třemi zdroji signálu

K přesnému určení polohy ve dvojrozměrném prostoru je tedy potřeba alespoň 3 zdrojů signálu. Tyto případy ovšem uvažují synchronizaci hodin všech prvků a ideální šíření signálu. V reálné situaci mají hodiny v přijímači odchylku od hodin v satelitech a šíření signálu je ovlivněno vlivy atmosféry. Vlivem těchto vzájemně nezávislých chyb je pozice přijímače ve šrafované oblasti zobrazené na Obr. 4.



Obr. 4: Nejistota při určení polohy se započtením chyb měření

V trojrozměrném systému se pozice určuje stejným způsobem, místo kružnic je však potřeba uvažovat koule. Ve skutečnosti by stačily pro určení polohy 2 zdroje pro určení pozice ve dvojrozměrném a 3 zdroje ve trojrozměrném systému. Jeden průsečík se totiž nachází na Zemi a druhý je daleko ve vesmíru nebo uprostřed Země [2].

Přesnost hodin v přijímačích se pohybuje v řádu jednotek mikrosekund. To dává chybu až 1500 metrů. Satelitní hodiny mají maximální odchylku 50 pikosekund, což dává chybu 1,5 centimetru. Proto se pro určení polohy pomocí satelitního systému používají alespoň 4 satelity a počítá se poloha i čas [2].

1.1 Architektura globálního navigačního satelitního systému

Satelitní systém se skládá ze tří hlavních segmentů: vesmírný segment, kontrolní segment a uživatelský segment.

Vesmírný segment se skládá ze satelitů na oběžných drahách v rozmezí výšek od 18 do 35 tisíc km nad Zemí. Počet satelitů a jejich rozmístění záleží na konkrétním systému a požadovaném pokrytí. Více v následující kapitole.

Kontrolní segment se skládá z kontrolních stanic určených k monitorování satelitů na oběžných drahách a případnou korekci jejich drah a korekci satelitních hodin k zachování přesnosti navigace.

Uživatelský segment jsou všechna zařízení přijímací satelitní signál k určení polohy nebo času.

1.2 Současné satelitní systémy

V současnosti existuje 7 satelitních systémů, ale jenom 3 jsou v provozu s globálním pokrytím. Ostatní systémy jsou pouze regionální nebo jejich podpora globálního pokrytí se teprve připravuje.

Globálními systémy jsou:

- systém GPS od USA,
- ruský systém GLONASS,
- evropský systém Galileo.

Čínský systém BeiDou bude mít globální pokrytí v roce 2020. Dále existují regionální navigační systémy, kterými jsou:

- indický systém IRNSS,
- japonský systém QZSS,
- francouzský systém DORIS.

V Tab. 1 je přehled parametrů systémů GPS, GLONASS, Galileo a BeiDou [2], [3].

Tab. 1: Přehled satelitních konstelací s globálním pokrytím

	GPS	**GLONASS**	**Galileo**	**BeiDou**
Země	USA	Rusko	EU	Čína
Kódování signálů	CDMA	FDMA	CDMA	CDMA
Počet satelitů	27 + 4 náhradní	24 + 3 náhradní	24 + 6 náhradní	35
Počet orbitálních rovin	6	3	3	3
Poloměr oběžné dráhy [km]	20200	19140	23222	21525 a 35787
Nosné frekvence signálů [MHz]	L1-1575,42 L2-1227,60 L5-1176,45	L1 1598,0625 - 1609,3125 L2 1242,9375 - 1251,6875	E1-1575,42 E5-1176,45 E6-1278,75	B1-1561,098 B2-1207,140 B3-1268 520

1.3 Struktura navigační zprávy a její zakódování

V této kapitole je rozebrána generace satelitního signálu a obsah vysílaných zpráv. Je uveden příklad pro civilní GPS signál vysílaný na frekvenci L1 = 1575,42 MHz.

Každý satelit odesílá pravidelně navigační zprávu s daným formátem. Navigační zpráva je složena z 5 podrámců o délce 300 bitů. Každý podrámec obsahuje 10 slov o délce 30 bitů. První dvě slova každého podrámce jsou telemetrická data a předávací slovo. Zbylá slova prvního podrámce obsahují číslo týdne GPS (modulo 1024 počtu uplynulých týdnů od 5. ledna 1980), přesnost a stav satelitu a korekce pro hodiny na satelitu. Druhý a třetí podrámec obsahuje parametry dráhy satelitu. Podrámec 4 obsahuje parametry dráhy a stav satelitů 25-32, speciální zprávy, konfigurační vlajky satelitu, ionosférická data a data koordinovaného světového času UTC (referenční atomové hodiny). Poslední podrámec obsahuje parametry dráhy a stav satelitů 1-24, referenční čas a číslo týdne, kdy byly nahrány

parametry drah satelitů. Podrámce 4 a 5 obsahují 25 stránek každý a během jedné navigační zprávy je v podrámci odeslána jedna stránka parametrů. Pro odeslání všech parametrů je tedy potřeba odeslat celkem 25 navigačních zpráv [1]. Rychlost odesílání dat je 50 bitů za sekundu, celkově je tedy potřeba 12,5 minuty vysílání. Formát navigační zprávy je na Obr. 5.

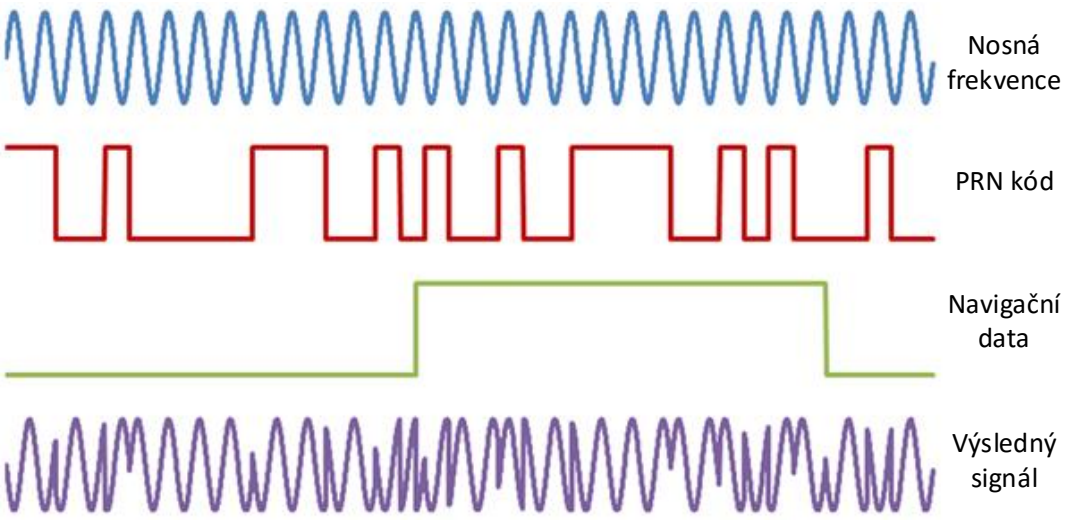
Podrámec 1	Telemetrie	Předávací slovo	GPS týden, přesnost a stav satelitu, korekce pro satelitní hodiny
Podrámec 2	Telemetrie	Předávací slovo	Parametry dráhy satelitu
Podrámec 3	Telemetrie	Předávací slovo	Parametry dráhy satelitu
Podrámec 4 (stránky 1-25)	Telemetrie	Předávací slovo	Parametry dráhy a stav satelitů 25-32, speciální zprávy, konfigurační vlajky, ionosférická data, referenční UTC čas
Podrámec 5 (stránky 1-25)	Telemetrie	Předávací slovo	Parametry a stav satelitů 1-24, čas a číslo týdne nahrání parametrů dráhy a stavu satelitů

Obr. 5: Formát navigační zprávy. Překresleno z [1].

V následující podkapitole je rozebráno, jak jsou data navigační zprávy zakódovány na nosnou frekvenci.

1.3.1 Generace satelitního signálu

Satelitní systémy používají pro modulaci dat techniku DSSS (direct sequence spread spectrum) a BOC (binary offset carrier) modulaci [1]. Modulace DSSS se používá u starších typů signálu jako je například GPS L1 C/A kód. U moderních satelitních signálů se používá BOC modulace, případně její modifikace. Příklad modulace DSSS je na Obr. 6. První signál na obrázku je sinusový průběh nosné frekvence. Poté následuje PRN kód, který nabývá hodnot 1 nebo -1 (v digitální podobě reprezentované logickou 1 a logickou 0). Třetí průběh jsou data navigační zprávy. Poslední průběh na obrázku je výsledný signál po DSSS modulaci.



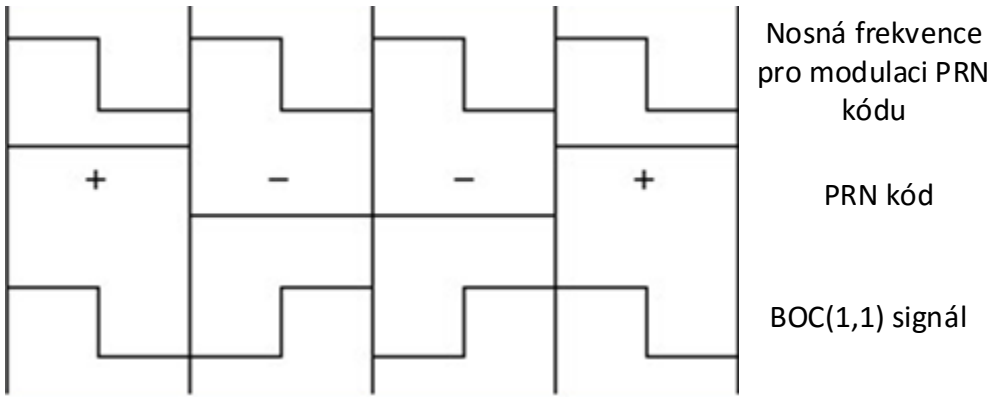
Obr. 6: Proces DSSS modulace. Převzato z [5].

DSSS modulace začíná vynásobením datového signálu s PRN kódem. Poté je tento výsledek namodulován na nosnou frekvenci. Když vynásobený signál změní hodnotu, změní se fáze nosné frekvence o 180º. Tímto se rozšíří šířka pásma datového signálu, daná poměrem frekvence rozšiřujícího PRN kódu a frekvencí dat, a snižují se tím nároky na výkon vysílače [1].

Pro GPS frekvenci L1 se používají dva PRN kódy, které jsou jedinečné pro každý satelit. První PRN kód se nazývá C/A (coarse acqusition) kód o délce 1023 bitů s frekvencí 1,023 Mbps. C/A kód je dostupný pro civilní účely. Druhý PRN kód se nazývá P (precise) kód a je určen pouze pro vojenské účely. Oba kódy jsou modulovány a vysílány na stejné nosné frekvenci, pouze jeden je posunutý o 90º oproti druhému. Toto je umožněno díky jejich autokorelační schopnosti, kdy k maximální korelaci dojde pouze s přesně frekvenčně i fázově sesouhlasenou kopií kódu [4].

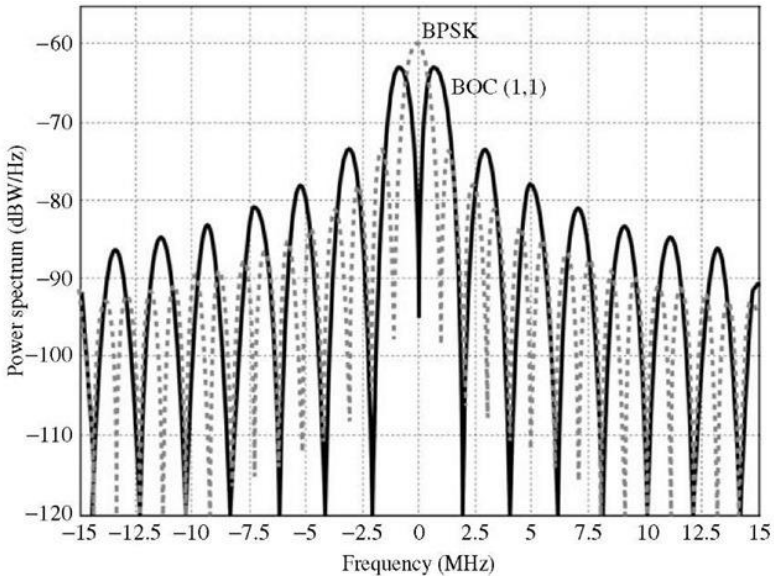
BOC modulace je rozšíření DSSS modulace. Před násobením PRN kódu a dat je PRN kód vynásoben pomocnou nosnou frekvencí. Poté následuje stejný proces jako u DSSS modulace.

Pomocná nosná frekvence PRN signálu je výsledek signum funkce sinusového nebo kosinusového průběhu. Signál po modulaci je značen jako BOC(*m*, *n*) signál, kde *m* je frekvence pomocné nosné frekvence PRN kódu a *n* je frekvence PRN kódu v jednotkách 1,023 MHz. Ukázka BOC(1,1) modulace je na Obr. 7.



Obr. 7: BOC(1,1) modulace. Převzato z [9].

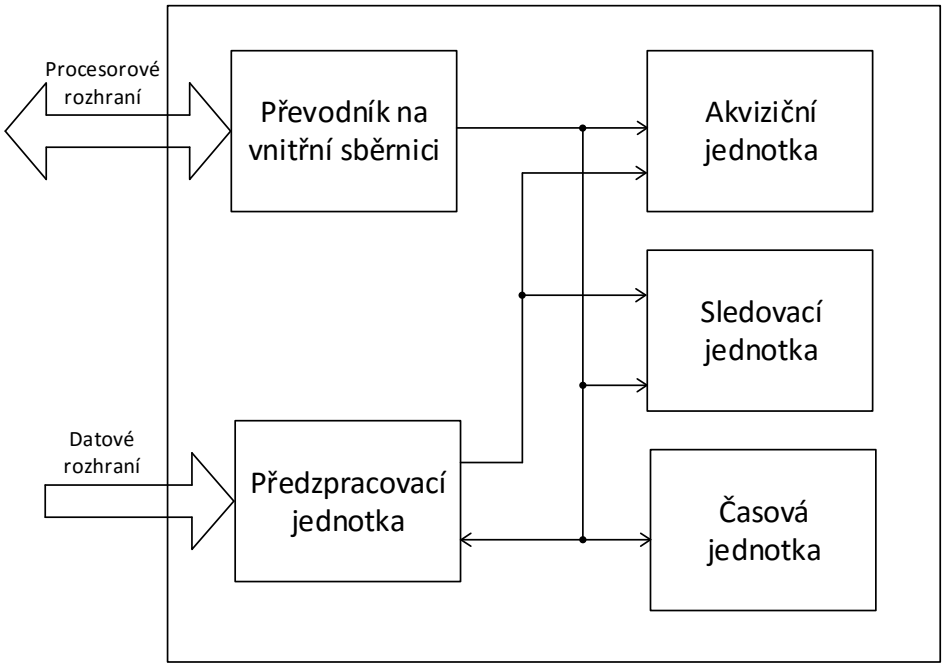
Výhodou BOC modulace je rozdělení spektra signálu symetricky kolem nosné frekvence. Ovšem je nutno dávat pozor na falešná maxima při korelaci. Příklad spektra BOC(1,1) a DSSS (BPSK) modulace je na Obr. 8.



Obr. 8: Spektrum signálu po BOC(1,1) a DSSS modulaci. Převzato z [10].

2 Představení testovaných bloků

V této kapitole je představen testovaný design. Dále je popsán princip funkce bloků, které budou verifikovány. Blokové schéma designu je na Obr. 9. Testovanými bloky v této práci jsou Akviziční a Sledovací jednotky.



Obr. 9: Blokové schéma testovaného obvodu

Design je ovládán procesorem, který zapisuje a čte registry designu přes Procesorové rozhraní. Příkazy z procesoru jsou převedeny na vnitřní sběrnici.

Vstupní data jdou přes Datové rozhraní do Předzpracovací jednotky. Ta se stará o převod dat do časové domény designu, jejich filtraci a bitovou redukci.

Z Předzpracovací jednotky putují data do Akviziční a Sledovací jednotky. Akviziční jednotka je určená k nalezení zadaného satelitního signálu. Princip nalezení satelitního signálu je popsán v kapitole 2.1.

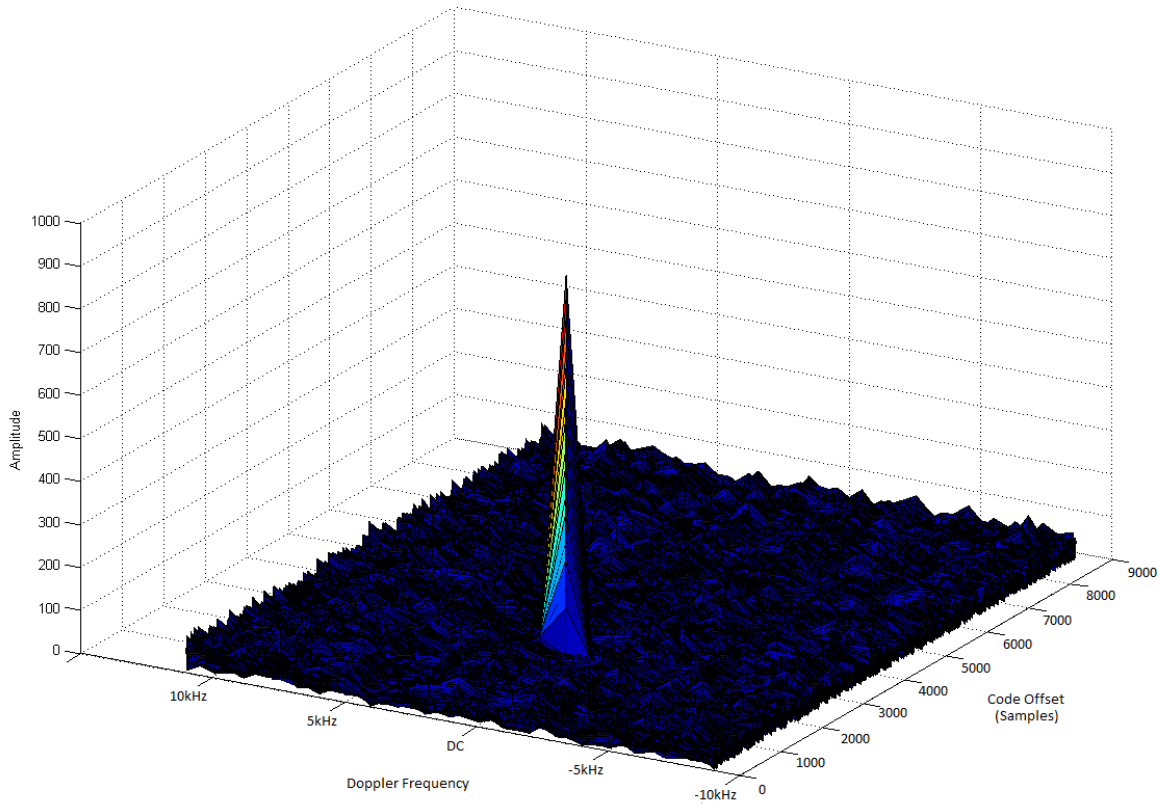
Sledovací jednotka má za úkol sledovat již nalezený satelitní signál z konkrétního satelitu. Princip sledování satelitního signálu je popsán v kapitole 2.2.

Časová jednotka se stará o synchronizaci mezi Akviziční a Sledovací jednotkou a získání časové informace z přijatého satelitního signálu.

2.1 Nalezení satelitního signálu

Každý satelit má svůj vlastní známý PRN kód a vysílá na známé frekvenci. Ovšem je potřeba uvažovat i Dopplerův jev, kdy přijímaný signál má jinou frekvenci, než s jakou vysílá satelit. Hledání satelitního signálu je tedy dvojrozměrné. Úkolem Akviziční jednotky je synchronizace PRN kódu satelitu s lokálně generovaným PRN kódem. To se provádí hledáním korelace přijatých dat s posunutým lokálním PRN kódem. PRN kódy jsou generovány tak, aby maximum korelace měli pouze s přesně synchronizovanou kopií kódu.

Příklad průběhu korelační amplitudy v závislosti na frekvenci přijatého signálu a posunutí lokálního PRN kódu je na Obr. 10.



Obr. 10: Velikost korelační amplitudy v závislosti na Dopplerově frekvenci a posunu PRN kódu. Převzato z [6].

Proces získání satelitního signálu je opačný proces ke generace signálu. Nejprve je nutné demodulovat data a PRN kód z nosné frekvence. To je v digitálních obvodech realizované pomocí číslicově kontrolovaného oscilátoru (NCO) nosné frekvence. Výstup z NCO je namapován pomocí odpovídajících trigonometrických funkcí. V případě použití NCO s 32 bity s 3bitovým výstupem, nastane celkem 8 možných hodnot fáze [1]. Hodnoty trigonometrických funkcí lze vypočítat z výrazu $exp(-j*\varphi(n))$, kde $\varphi(n)$ je fáze [7]. Hodnoty amplitud trigonometrických funkcí jsou uvedeny v Tab. 2.

Tab. 2: Namapované hodnoty trigonometrických funkcí (nejvyšší bit je znaménkový)

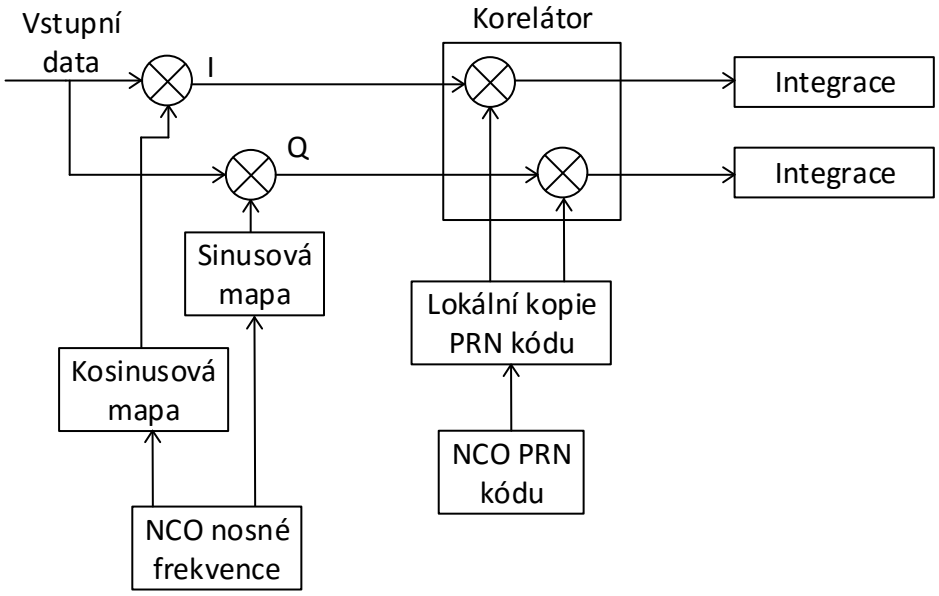
Fáze [°]	Výstup z NCO	Sinusová mapa	Kosinusová mapa
0	0b000	0b000	0b011
45	0b001	0b110	0b010
90	0b010	0b111	0b000
135	0b011	0b110	0b110
180	0b100	0b000	0b111
225	0b101	0b010	0b110
270	0b110	0b011	0b000
315	0b111	0b010	0b010

Demodulovaná data jsou získána vynásobením vstupních dat aktuální hodnotou z mapy trigonometrických funkcí. Reálná část *I* je získána násobením kosinovou hodnotou, imaginární část *Q* je získána násobením sinusovou hodnotou.

Další krokem je odstranění PRN kódu. To má za úkol korelátor. Korelátor provádí násobení reálné a imaginární části aktuálním bitem PRN kódu. Pokud má bit logickou hodnotu 1, tak se násobí +1. Pokud je PRN bit v logické 0, hodnota *I* a *Q* se násobí -1. Korelované hodnoty *I* a *Q* jsou poté integrovány. Digitální realizace je opět použití NCO pro frekvenci PRN kódu a generátoru lokální kopie PRN kódu. Pokud NCO přeteče, generátor dá na výstup další bit kódu. Tento proces se opakuje, dokud není vygenerována celá sekvence PRN kódu.

Další korelátory pracují s posunutým PRN kódem. Celkový počet korelátorů závisí na požadované přesnosti, která je daná poměrem počtu bitů PRN kódu a počtem korelátorů. Je potřeba mít na paměti, že signál se šíří rychlostí světla. Pokud je použit stejný počet korelátorů jako je délka PRN kódu, tak pro GPS L1 C/A kód, který má frekvenci 1,023Mbps, může být chyba určení polohy až 300 metrů.

Z konečných hodnot *I* a *Q* po integraci pro každý korelátor je vypočtena amplituda a porovnána pro nalezení maximální korelace. Blokové schéma principu Akviziční jednotky je na Obr. 11.

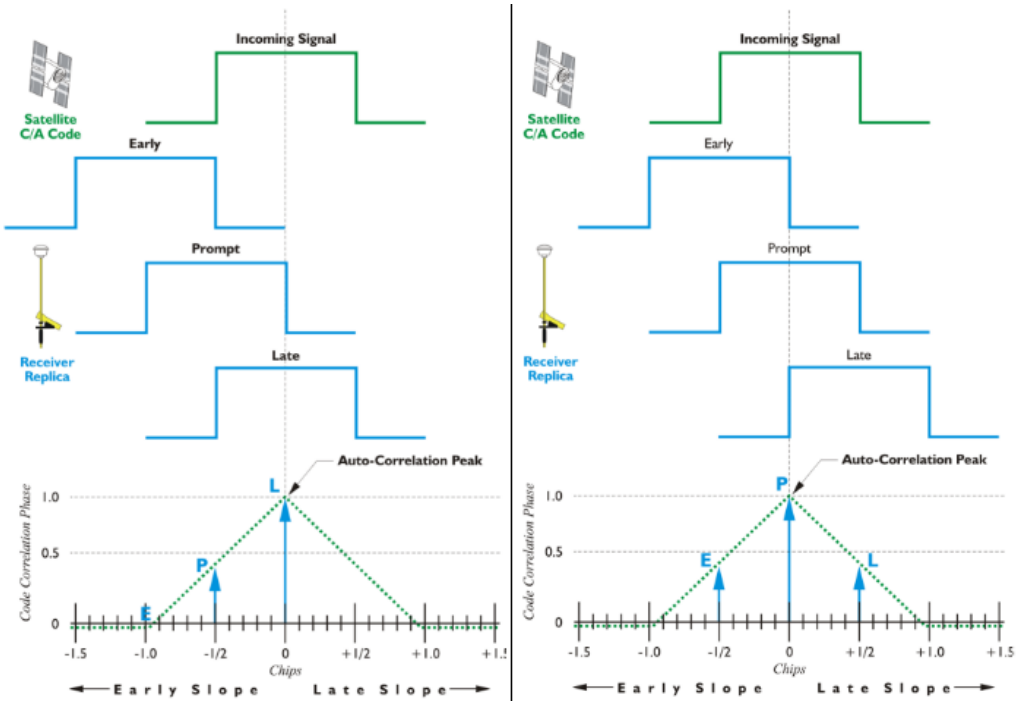


Obr. 11: Blokové schéma akvizičního procesu

2.2 Proces sledování satelitního signálu

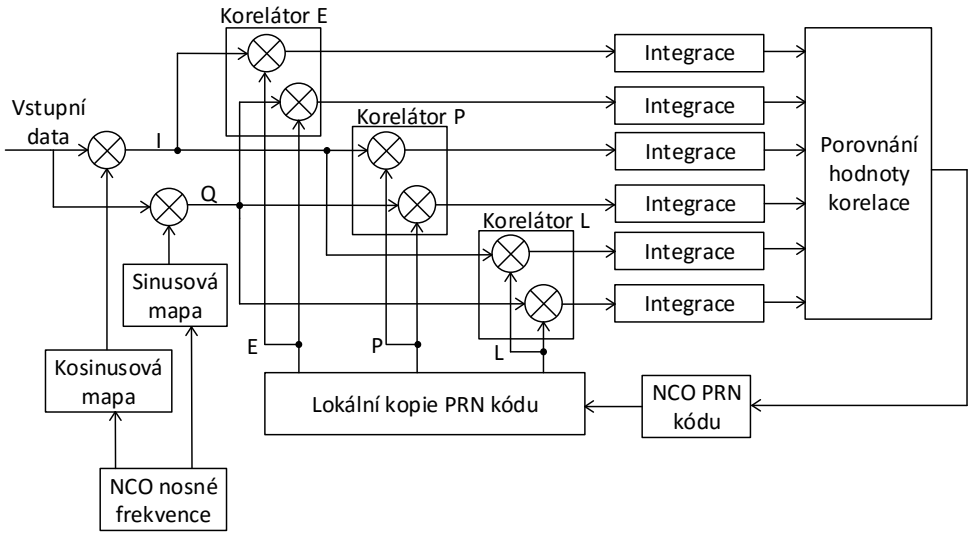
Po nalezení satelitního signálu Akviziční jednotkou přichází fáze sledování tohoto signálu Sledovací jednotkou. Jejím úkolem je udržet ideální synchronizaci přijatého a lokálního PRN kódu. Princip je totožný s procesem v Akviziční jednotce. Posun PRN kódu je již známý, proto stačí menší počet korelátorů ke sledování satelitního signálu. Pro sledování satelitů vysílající starší PRN kódy stačí 3 korelátory, ale pro novější je potřeba alespoň 5 korelátorů z důvodu falešných maxim způsobených BOC modulací a dalšího vysílaného PRN kódu (*pilot code* – PRN kód na který nejsou modulována data). Absence dat u PRN pilot kódu snižuje vliv šumu a zlepšuje přesnost sledování satelitního signálu [11].

Vstupní data jsou nejprve zbavena nosné frekvence vynásobením aktuální hodnotou sinusové a kosinusové mapy. Tím dostaneme reálnou část *I* a imaginární část *Q*. Dále je potřeba demodulovat data z PRN kódu. Opět vynásobením hodnoty *I* a *Q* hodnotou aktuálního PRN bitu. Jeden korelátor, označený E (early) má posunutý PRN kód napřed. Druhý korelátor, označený P (prompt) pracuje se synchronizovaným PRN kódem. Poslední korelátor, označený L (late), má PRN kód opožděný. Dále se provádí integrace a porovnává hodnota korelace, zda nedošlo k posunu přijatého PRN kódu. Pokud je maximum na jiném korelátoru než na P korelátoru, tak se posune lokální PRN kód tak, aby maximum bylo opět v P korelátoru. Ukázka této situace je na Obr. 12.



Obr. 12: Korelační fáze PRN kódu. Převzato z [8].

Na Obr. 13 je zobrazeno blokové schéma sledovacího procesu.



Obr. 13: Blokové schéma sledovacího procesu

3 Metodika UVM

Metodika UVM (Universal Verification Methodology) je standardizovaná metodika verifikace digitálních obvodů vytvořená organizací Accellera. Tato metodika vznikla spojením metodik OVM (Open Verification Methodology) a VMM (Verification Methodology Manual). Všechny zmíněné metodiky používají pro svou implementaci jazyk SystemVerilog [12][13].

SystemVerilog je programovací jazyk pro popis obvodu i jeho verifikaci. Umožňuje modelování digitálních obvodů na vyšší úrovni abstrakce pomocí objektově orientovaného programování díky kombinaci prvků jazyka VHDL a Verilog pro popis digitálních obvodů, prvků specializovaných jazyků pro verifikaci digitálních obvodů a prvků z jazyka C a C++ [14].

Hlavní cíl metodiky UVM je vytvoření verifikačního prostředí, které lze snadno použít pro různé projekty, aniž by se prováděly zásadní změny ve verifikačním prostředí. Tohoto cíle je dosaženo komunikací jednotlivých bloků verifikačního prostředí založeném na standardu TLM (transaction-level modeling) [12]. Jednotlivé bloky spolu komunikují předáváním transakcí přes komunikační rozhraní, jehož argumenty jsou objekty v transakci. Transakce je třída, která obsahuje informace potřebné pro modelování komunikace nižší úrovně. Například pro jednoduché modelování sběrnice bude transakce obsahovat:

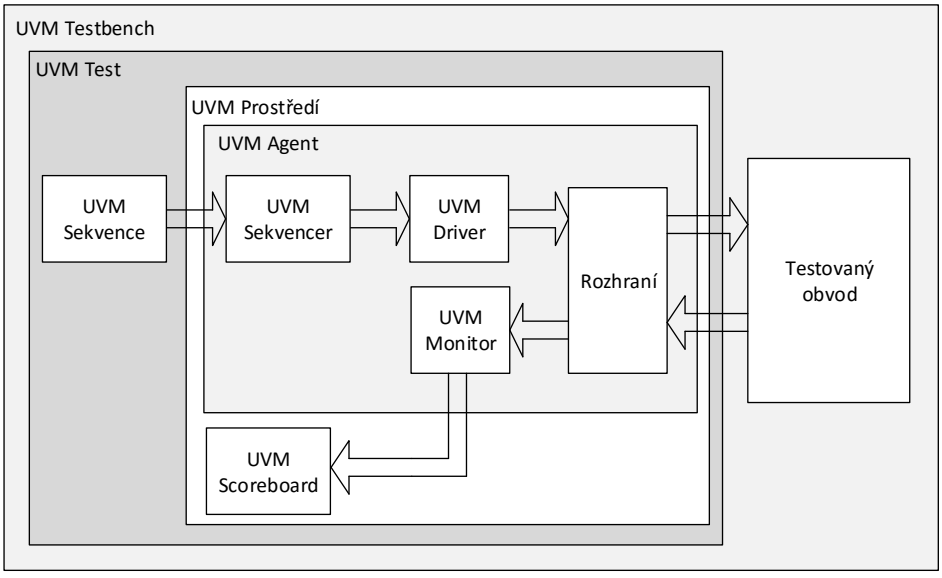
- data,
- adresu,
- typ operace (zápis nebo čtení).

Tyto hodnoty je možné v UVM generovat dvěma způsoby podle typu testu:

- přímými testy – hodnoty jsou přímo psány verifikačním inženýrem,
- testy s náhodnými čísly – pomocí algoritmu jsou vypočteny pseudonáhodné hodnoty.

3.1 Struktura verifikačního prostředí

Verifikační prostředí je vytvořeno z UVM tříd. Na Obr. 14 je ukázka verifikačního prostředí v metodice UVM.



Obr. 14: Ukázka verifikačního prostředí v metodice UVM. Upraveno z [13].

UVM Testbench

Na nejvyšší úrovni je třída UVM Testbench. Zde je instanciován testovaný obvod, třída UVM Test a konfiguruje jejich propojení.

UVM Test

UVM Test má za úkol instanciovat nejvyšší úroveň prostředí, konfigurovat prostředí pomocí konfigurační databáze nebo přepsáním pomocí UVM Factory (umožňuje použití upravených bloků pro určitý test bez zásahu do zdrojového kódu bloků) a aplikovat stimuly vyvoláním UVM Sekvencí skrz prostředí do testovaného obvodu. Typicky jeden základní test instancuje prostředí a ostatní testy rozšiřují tento základní test.

UVM Prostředí

UVM Prostředí je hierarchická komponenta sdružující vzájemně související komponenty jako například UVM Agent, UVM Scoreboard nebo jiná UVM Prostředí.

UVM Scoreboard

UVM Scoreboard, neboli výsledková tabulka, kontroluje chování testovaného obvodu porovnáváním očekávané transakce z referenčního modelu a pozorovaných transakcí z testovaného obvodu.

UVM Agent

UVM Agent je hierarchická komponenta sdružující komponenty pro stimulaci a kontrolu výstupů testovaného obvodu připojeného přes určité rozhraní. Typickými komponentami jsou například UVM Sekvencer, UVM Driver a UVM Monitor. UVM Agent má 2 módy, aktivní a pasivní. Aktivní Agent obsahuje výše zmíněné komponenty a je schopen generovat stimuly a monitorovat rozhraní testovaného obvodu. Pasivní Agent obsahuje pouze Monitor a je schopen jenom monitorovat výstupy z testovaného obvodu.

UVM Sekvencer

UVM Sekvencer kontroluje a řídí tok transakcí generovaných v UVM Sekvencích.

UVM Sekvence

UVM Sekvence obsahuje proces generace stimulu. Sekvence může trvat od doby trvání jedné transakce až po dobu celé simulace. UVM Sekvence může vyvolávat hierarchicky další Sekvence, ale každá Sekvence potřebuje být vázána na UVM Sekvencer.

UVM Driver

UVM Driver, neboli řadič, převádí jednotlivé obdržené transakce ze Sekvence na signálovou úroveň a aplikuje na rozhraní testovaného obvodu.

UVM Monitor

UVM Monitor sleduje rozhraní testovaného obvodu a převádí hodnoty ze signálové úrovně na transakční úroveň. Vygenerované transakce jsou odeslány do UVM Scoreboard k vyhodnocení.

3.2 UVM fáze

UVM fáze slouží jako sychronizační mechanismus při verifikačním procesu. Každá fáze má definované své funkce nebo procedury. UVM fázování je spuštěno zavoláním metody *run_test()*. Tato metoda je volána buď přímo s názvem testu jako parametrem nebo v příkazovém řádku pomocí argumentu UVM_TESTNAME=*název_testu*, který je poté vyhledán v UVM Factory. UVM fáze (phase) probíhají v následujícím pořadí:

Konstrukční fáze (Build phase)

V této fázi probíha konstrukce jednotlivých komponent v UVM Testbench. Jako jediná probíhá hierarchicky od nejvyššího komponentu k nižším. Ostatní faze probíhají hierarchicky od nejnižšího komponentu k nejvyššímu.

Propojovací fáze (Connect phase)

V této fázi jsou vytvořeny TLM propojení mezi jednotlivými komponenty.

Fáze konečných úprav (End of elaboration phase)

Po vytvoření a propojení komponentů v předchozích fázích zde probíhají případné finální úpravy struktury, konfigurace komponent nebo propojení. Po této fázi začíná simulační část.

Fáze počátku simulace (Start of simulation phase)

Tato fáze je určena k zobrazení topologie v UVM Testbench a konfigurace komponent.

Fáze aplikace stimulů (Run phase)

V této fázi začíná konzumace simulačního času. Je určena pro generaci stimulů pro testovaný obvod a jeho monitorování.

Extrakční fáze (Extract phase)

Tato fáze je určena pro kontrolní kompenty. Probíhá zde zpracování informací z UVM Scoreboard a monitorů funkčního pokrytí.

Kontrolní fáze (Check phase)

V této fázi je kontrolováno korektní chování testovaného obvodu, tedy zda během fáze aplikace stimulů nenastaly chyby.

Fáze hlášení výsledků (Report phase)

V této fázi jsou zobrazeny nebo zapsány do souboru výsledky simulace.

4 Požadavky na testovaný obvod

V této kapitole jsou sepsány požadavky na Akviziční a Sledovací jednotku, které budou během verifikačního procesu ověřovány. Některé z požadavků jsou zaměřeny na implementaci výpočetních algoritmů. Pokud byl k tomuto algoritmu nalezen veřejný zdroj, tak je algoritmus v požadavku uveden i s odkazem na zdroj. Jinak je požadavek uveden bez tohoto algoritmu.

Na každý požadavek bude vypracován jeden testovací scénář, ve kterém bude uveden postup ověření správné implementace požadavku. Postup uvedený v testovacích scénářích poté bude implementován v testových procedurách.

4.1 Požadavky na Akviziční jednotku

Požadavky na Akviziční jednotku jsou obsaženy v Tab. 3.

Tab. 3: Požadavky na Akviziční jednotku

Index požadavku	Znění požadavku
RQ_ACQ_1	Akviziční jednotka bude zpracovávat vstupní data, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1. • *Komentář: Enable bit* slouží k povolení/přerušení akvizičního procesu.
RQ_ACQ_2	Akviziční jednotka bude indikovat svůj stav pomocí *ACQUISITION_STATUS* registru.
RQ_ACQ_3	Akviziční jednotka bude zpracovávat data z konkrétního vstupního datového kanálu podle hodnoty *Channel Select* bitů *ACQUISITION_CONTROL* registru. • *Komentář:* Tento bit slouží k přepínání mezi frekvenčním pásmem L1 a L5.
RQ_ACQ_4	Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN kódu pomocí *CODE_NCO_RATE* registru. • *Komentář*: V registru *CODE_NCO_RATE* je uložena hodnota inkrementu pro NCO PRN kódu.
RQ_ACQ_5	Akviziční jednotka bude generovat lokální PRN kód z paměti pro PRN kód Akviziční jednotky.
RQ_ACQ_6	Akviziční jednotka bude generovat lokální PRN kód o délce X+1 bitů, kde X je hodnota *ACQ_PRN_LENGTH* registru. • *Komentář:* Satelitní signály používají PRN kódy o délkách 511, 1023, 4092 a 10230 bitů.
RQ_ACQ_7	Akviziční jednotka při operaci zápisu dat do registru *ACQUISITION_PRN_DATA* uloží tyto data do paměti PRN kódu Akviziční jednotky na adresu podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.
RQ_ACQ_8	Akviziční jednotka při operaci čtení z registru *ACQUISITION_PRN_DATA* vrátí hodnotu dat v paměti PRN kódu Akviziční jednotky z adresy podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

RQ_ACQ_9	Akviziční jednotka bude ignorovat operaci zápisu do PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_10	Akviziční jednotka bude ignorovat operaci čtení z PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_11	Akviziční jednotka bude modulovat lokální PRN kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_12	Akviziční jednotka bude umožňovat konfigurovatelné zpoždění lokálního PRN kódu mezi korelátory o X+1 vzorků, kde X je hodnota *DECIMATION_FACTOR* registru.
RQ_ACQ_13	Akviziční jednotka bude umožňovat konfiguraci hodnoty nosné frekvence pomocí *CARRIER_NCO_RATE_1-10* registrů. • *Komentář*: V registrech *CARRIER_NCO_RATE_1-10* jsou uloženy hodnoty inkrementů pro příslušné NCO nosné frekvence.
RQ_ACQ_14	Akviziční jednotka bude generovat jednu nosnou frekvenci pomocí jediného NCO, které bude inkrementovat o hodnotu v registru *CARRIER_NCO_RATE_*1, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0. • *Komentář*: V tomto módu jsou všechny korelátory zapojeny sériově do jedné linky.
RQ_ACQ_15	Akviziční jednotka použije bude generovat 10 nosných frekvencí pomocí 10 NCO, které bude inkrementovat o hodnotu v registrech *CARRIER_NCO_RATE_1-10*, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0. • *Komentář*: V tomto módu jsou korelátory zapojeny do 10 paralelních linek.
RQ_ACQ_16	Akviziční jednotka zpracuje v jednom integračním cyklu 1000*(X+1) vzorků, kde X je hodnota *COHERENT_INTEGRATION_TIME* registru.
RQ_ACQ_17	Akviziční jednotka provede X+1 integračních cyklů, kde X je hodnota *INTEGRATION_TIME* registru.
RQ_ACQ_18	Akviziční jednotka provede nekoherentní integraci, pokud je *Acqusition Type* bit *ACQUISITION_CONTROL* registru v logické 0, podle rovnice $S = \sum_{n=1}^{IntegrationTime} \lvert R_n \rvert^2,$ kde R_n je výsledek akumulace po integračním cyklu [15].
RQ_ACQ_19	Akviziční jednotka provede diferenciální integraci, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 1, podle rovnice $S = \sum_{n=2}^{IntegrationTime} \lvert R_n * R_{n-1}^* \rvert^2,$ kde R_n je výsledek akumulace po integračním cyklu [15].
RQ_ACQ_20	Akviziční jednotka uloží reálnou hodnotu *I*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_I* registrů, kde X je index skupiny 0-9.

RQ_ACQ_21	Akviziční jednotka uloží imaginární hodnotu *Q*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_Q* registrů, kde X je index skupiny 0-9.
RQ_ACQ_22	Akviziční jednotka uloží index korelátoru, který vygeneroval maximální korelační amplitudu, z každé skupiny do *OUTPUT_X_CORRELATOR_NUMBER* registrů, kde X je index skupiny 0-9.

4.2 Požadavky na Sledovací jednotku

Požadavky na Sledovací jednotku jsou v Tab. 4.

Tab. 4: Požadavky na Sledovací jednotku

Index požadavku	Znění požadavku
RQ_TRK_1	Sledovací jednotka bude zpracovávat vstupní data, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_2	Sledovací jednotka bude ve stavu *IDLE*, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 0.
RQ_TRK_3	Sledovací jednotka změní svůj stavu na *ARMED*, pokud *Enable* bit *TRACKING_CONTROL* registru změní hodnotu z logické 0 na logickou 1.
RQ_TRK_4	Sledovací jednotka změní svůj stavu na *TRACKING*, pokud *Enable* bit *TRACKING_CONTROL* registru je v logické 1 a signál *SAMPLE_TIMESTAMP* se rovná hodnotě v registru *TRACKING_START_TIMESTAMP*. • *Komentář*: Tímto se zajistí přesné sesouhlasení se satelitním signálem.
RQ_TRK_5	Sledovací jednotka bude zpracovávat data z konkrétního vstupního datového kanálu podle hodnoty *Channel Select* bitů *TRACKING_CONTROL* registru. • *Komentář:* Tento bit slouží k přepínání mezi frekvenčním pásmem L1 a L5.
RQ_TRK_6	Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN PILOT a PRN DATA kódu pomocí *CODE_NCO_RATE* registru. • *Komentář*: V registru *CODE_NCO_RATE* je uložena hodnota inkrementu pro NCO PRN kódů.
RQ_TRK_7	Sledovací jednotka bude generovat lokální PRN DATA kód z paměti pro PRN DATA kód Sledovací jednotky.
RQ_TRK_8	Sledovací jednotka bude generovat lokální PRN PILOT kód z paměti pro PRN PILOT kód Sledovací jednotky.
RQ_TRK_9	Sledovací jednotka bude generovat lokální PRN DATA kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru. • *Komentář:* Satelitní signály používají PRN kódy o délkách 511, 1023, 4092 a 10230 bitů.

RQ_TRK_10	Sledovací jednotka bude generovat lokální PRN PILOT kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru. • *Komentář:* Satelitní signály používají PRN kódy o délkách 511, 1023, 4092 a 10230 bitů.
RQ_TRK_11	Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_DATA* uloží tyto data do paměti pro PRN DATA kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
RQ_TRK_12	Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_DATA* vrátí hodnotu dat v paměti pro PRN DATA kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
RQ_TRK_13	Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_PILOT* uloží tyto data do paměti pro PRN PILOT kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
RQ_TRK_14	Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_PILOT* vrátí hodnotu dat v paměti pro PRN PILOT kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
RQ_TRK_15	Sledovací jednotka bude ignorovat operaci zápisu do PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_16	Sledovací jednotka bude ignorovat operaci čtení z PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_17	Sledovací jednotka bude ignorovat operaci zápisu do PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_18	Sledovací jednotka bude ignorovat operaci čtení z PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_19	Sledovací jednotka bude modulovat lokální PRN DATA kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_20	Sledovací jednotka bude modulovat lokální PRN PILOT kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_21	Sledovací jednotka bude integrovat X+1 bitů PRN DATA kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru. • *Komentář:* Většina satelitních signálů má integrační čas 1 milisekunda. Ovšem signál Galileo E1 má integrační čas 4 milisekundy.

RQ_TRK_22	Sledovací jednotka bude integrovat X+1 bitů PRN PILOT kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru. • *Komentář:* Většina satelitních signálů má integrační čas 1 milisekunda. Signál Galileo E1 má integrační čas 4 milisekundy.
RQ_TRK_23	Sledovací jednotka bude umožňovat po dokončení integrace posun fáze NCO pro PRN kódy o hodnotu v *CODE_NCO_PHASE_INCREMENT* registru. • *Komentář:* Toto je potřeba pro případné korekce, pokud dojde k desynchronizaci lokální kopie PRN kódu a sledovaného satelitního signálu.
RQ_TRK_24	Sledovací jednotka po dokončení integrace vymaže hodnotu v *CODE_NCO_PHASE_INCREMENT* registru. • *Komentář:* Tímto je zajištěno, že korekce budou aplikovány pouze jednou.
RQ_TRK_25	Sledovací jednotka bude umožňovat po dokončení integrace posun v PRN DATA a PRN PILOT kódech o hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru. • *Komentář:* Toto je potřeba pro případné korekce, pokud dojde k desynchronizaci lokální kopie PRN kódu a sledovaného satelitního signálu.
RQ_TRK_26	Sledovací jednotka po dokončení integrace vymaže hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru. • *Komentář:* Tímto je zajištěno, že korekce budou aplikovány pouze jednou.
RQ_TRK_27	Sledovací jednotka bude generovat jednu nosnou frekvenci pomocí NCO, které bude inkrementovat o hodnotu v registru *TRACKING_CARRIER_NCO_RATE*.
RQ_TRK_28	Sledovací jednotka umožní konfiguraci zpoždění každého z korelátorů o X+1 vzorků, kde X je hodnota *CORRELATOR_DELAY_1-6* registrů.
RQ_TRK_29	Sledovací jednotka bude ukládat počet dokončených integrací od začátku procesu sledování do *INTEGRATION_RESULT_COUNT* registru.
RQ_TRK_30	Sledovací jednotka bude ukládat výsledek reálné části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_I* registru, kde X je číslo korelátoru. • *Komentář:* PILOT korelátory pracují s PRN PILOT kódem.
RQ_TRK_31	Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_Q* registru, kde X je číslo korelátoru. • *Komentář:* PILOT korelátory pracují s PRN PILOT kódem.
RQ_TRK_32	Sledovací jednotka bude ukládat výsledek reálné části korelace z PILOT PROMPT korelátoru do *CORRELATION_RESULT_PILOT_PROMPT_I* registru. • *Komentář:* PILOT korelátor pracuje s PRN PILOT kódem.

RQ_TRK_33	Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT PROMPT korelátoru do registru *CORRELATION_RESULT_PILOT_PROMPT_Q*. • *Komentář:* PILOT korelátor pracuje s PRN PILOT kódem.
RQ_TRK_34	Sledovací jednotka bude ukládat výsledek reálné části korelace z DATA PROMPT korelátoru do *CORRELATION_RESULT_DATA_PROMPT_I* registru. • *Komentář:* DATA korelátor pracuje s PRN DATA kódem.
RQ_TRK_35	Sledovací jednotka bude ukládat výsledek imaginární části korelace z DATA PROMPT korelátoru do registru *CORRELATION_RESULT_DATA_PROMPT_Q*. • *Komentář:* DATA korelátor pracuje s PRN DATA kódem.
RQ_TRK_36	Sledovací jednotka uloží aktuální počet integrací od začátku sledovacího procesu do registru *PVT_INTEGRATION_COUNT*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_37	Sledovací jednotka uloží aktuální hodnotu ukazatele na bit PRN kódu do registru *PVT_PRIMARY_ADDRESS*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_38	Sledovací jednotka uloží aktuální hodnotu fáze PRN kódu do registru *PVT_CODE_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_39	Sledovací jednotka uloží aktuální počet přetečení NCO nosné frekvence do registru *PVT_CARRIER_CYCLES*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_40	Sledovací jednotka uloží aktuální hodnotu fáze nosné frekvence do registru *PVT_CARRIER_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.

5 Testovací scénaře

V této kapitole je v uveden přehled testovacích scénářů pro požadavky uvedené v předchozí kapitole. Testovací scénáře obsahují postup ověřování správné implementace příslušného požadavku. Každý požadavek má vlastní testovací scénář. Podrobnější popis průběhu testovacích scénářů je v příloze A. Přehled testových procedur, pomocí kterých jsou implementovány tyto testovací scénáře, je v kapitole 6.

Testovací scénáře pro požadavky na Akviziční jednotku jsou v Tab. 5.

Tab. 5: Testovací scénáře pro požadavky Akviziční jednotky

Požadavek	Testovací scénář	Popis
RQ_ACQ_1	tc_acq_enable	Tento scénář kontroluje, zda Akviziční jednotka zpracovává vstupní data, pouze pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_2	tc_acq_status	Tento scénář kontroluje, zda Akviziční jednotka správně indikuje svůj stav pomocí *Acquisition Status* bitů *ACQUISITION_CONTROL* registru.
RQ_ACQ_3	tc_acq_channel_select	Tento scénář kontroluje, zda Akviziční jednotka zpracovává data ze vstupního datového kanálu podle hodnoty *Channel Select* bitů *ACQUISITION_CONTROL* registru.
RQ_ACQ_4	tc_acq_code_nco_rate	Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfiguraci rychlosti generace lokálního PRN kódu pomocí *CODE_NCO_RATE* registru.
RQ_ACQ_5	tc_acq_prn_generation	Tento scénář kontroluje, zda Akviziční jednotka generuje lokální PRN kód z paměti pro PRN kód Akviziční jednotky.
RQ_ACQ_6	tc_acq_prn_length	Tento scénář kontroluje, zda Akviziční jednotka generuje lokální PRN kód o délce X+1 bitů, kde X je hodnota *ACQ_PRN_LENGTH* registru.
RQ_ACQ_7	tc_acq_prn_data_write	Tento scénář kontroluje, zda Akviziční jednotka při operaci zápisu dat do registru *ACQUISITION_PRN_DATA* uloží tyto data do paměti PRN kódu Akviziční jednotky na adresu podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.
RQ_ACQ_8	tc_acq_prn_data_read	Tento scénář kontroluje, zda Akviziční jednotka při operaci čtení z registru *ACQUISITION_PRN_DATA* vrátí hodnotu dat v paměti PRN kódu Akviziční jednotky z adresy podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.
RQ_ACQ_9	tc_acq_prn_ignore _data_write	Tento scénář kontroluje, zda Akviziční jednotka bude ignorovat operaci zápisu do PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

RQ_ACQ_10	tc_acq_prn_ignore _data_read	Tento scénář kontroluje, zda Akviziční jednotka bude ignorovat operaci čtení z PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_11	tc_acq_prn_boc	Tento scénář kontroluje, zda Akviziční jednotka moduluje lokální PRN kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *ACQUISITION_CONTROL* registru v logické 1.
RQ_ACQ_12	tc_acq_decimation _factor	Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfigurovatelné zpoždění lokálního PRN kódu mezi korelátory o X+1 vzorků, kde X je hodnota *DECIMATION_FACTOR* registru.
RQ_ACQ_13	tc_acq_carrier_nco_rates	Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfiguraci hodnoty nosné frekvence pomocí *CARRIER_NCO_RATE_1-10* registrů.
RQ_ACQ_14	tc_acq_single_carrier	Tento scénář kontroluje, zda Akviziční jednotka generuje jednu nosnou frekvenci pomocí jediného NCO, které bude inkrementovat o hodnotu v registru *CARRIER_NCO_RATE_*1, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.
RQ_ACQ_15	tc_acq_carriers	Tento scénář kontroluje, zda Akviziční jednotka použije generuje 10 nosných frekvencí pomocí 10 NCO, které bude inkrementovat o hodnotu v registrech *CARRIER_NCO_RATE_1-10*, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.
RQ_ACQ_16	tc_acq_coherent _integration_time	Tento scénář kontroluje, zda Akviziční jednotka zpracuje v jednom integračním cyklu 1000*(X+1) vzorků, kde X je hodnota *COHERENT_INTEGRATION_TIME* registru.
RQ_ACQ_17	tc_acq_integration_time	Tento scénář kontroluje, zda Akviziční jednotka provede X+1 integračních cyklů, kde X je hodnota *INTEGRATION_TIME* registru.
RQ_ACQ_18	tc_acq_noncoherent _integration	Tento scénář kontroluje, zda Akviziční jednotka provede nekoherentní integraci, pokud je *Acqusition Type* bit *ACQUISITION_CONTROL* registru v logické 0, podle rovnice $S = \sum_{n=1}^{IntegrationTime} \lvert R_n \rvert^2$, kde R_n je výsledek akumulace po integračním cyklu [15].
RQ_ACQ_19	tc_acq_differential _integration	Tento scénář kontroluje, zda Akviziční jednotka provede diferenciální integraci, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 1, podle rovnice $S = \sum_{n=2}^{IntegrationTime} \lvert R_n * R_{n-1}^* \rvert^2$, kde R_n je výsledek akumulace po integračním cyklu [15].

RQ_ACQ_20	tc_acq_inphase_results	Tento scénář kontroluje, zda Akviziční jednotka uloží reálnou hodnotu *I*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_I* registrů, kde X je index skupiny 0-9.
RQ_ACQ_21	tc_acq_quadrature _results	Tento scénář kontroluje, zda Akviziční jednotka uloží imaginární hodnotu *Q*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_Q* registrů, kde X je index skupiny 0-9.
RQ_ACQ_22	tc_acq_correlator _max_amplitude	Tento scénář kontroluje, zda Akviziční jednotka uloží index korelátoru, který vygeneroval maximální korelační amplitudu, z každé skupiny do *OUTPUT_X_CORRELATOR_NUMBER* registrů, kde X je index skupiny 0-9.

Testovací scénáře pro požadavky na Sledovací jednotku jsou v Tab. 6.

Tab. 6: Testovací scénáře pro požadavky Sledovací jednotky

Požadavek	**Testovací scénář**	**Popis**
RQ_TRK_1	tc_track_enable	Tento scénář kontroluje, zda Sledovací jednotka zpracovává vstupní data, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_2	tc_track_status_idle	Tento scénář kontroluje, zda Sledovací jednotka je ve stavu *IDLE*, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 0.
RQ_TRK_3	tc_track_status_armed	Tento scénář kontroluje, zda Sledovací jednotka změní svůj stavu na *ARMED*, pokud *Enable* bit *TRACKING_CONTROL* registru změní hodnotu z logické 0 na logickou 1.
RQ_TRK_4	tc_track_status_tracking	Tento scénář kontroluje, zda Sledovací jednotka změní svůj stavu na *TRACKING*, pokud *Enable* bit *TRACKING_CONTROL* registru je v logické 1 a signál *SAMPLE_TIMESTAMP* se rovná hodnotě v registru *TRACKING_START_TIMESTAMP*.
RQ_TRK_5	tc_track_channel_select	Tento scénář kontroluje, zda Sledovací jednotka bude zpracovávat data z konkrétního vstupního datového kanálu podle hodnoty *Channel Select* bitů *TRACKING_CONTROL* registru.
RQ_TRK_6	tc_track_code_nco_rate	Tento scénář kontroluje, zda Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN PILOT a PRN DATA kódu pomocí *CODE_NCO_RATE* registru.
RQ_TRK_7	tc_track_prn_data _generation	Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN DATA kód z paměti pro PRN DATA kód Sledovací jednotky.
RQ_TRK_8	tc_track_prn_pilot _generation	Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN PILOT kód z paměti pro PRN PILOT kód Sledovací jednotky.

RQ_TRK_9	tc_track_prn _data_length	Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN DATA kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.
RQ_TRK_10	tc_track_prn _pilot_length	Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN PILOT kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.
RQ_TRK_11	tc_track_prn_data_write	Tento scénář kontroluje, zda Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_DATA* uloží tyto data do paměti pro PRN DATA kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
RQ_TRK_12	tc_track_prn_data_read	Tento scénář kontroluje, zda Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_DATA* vrátí hodnotu dat v paměti pro PRN DATA kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
RQ_TRK_13	tc_track_prn_pilot_write	Tento scénář kontroluje, zda Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_PILOT* uloží tyto data do paměti pro PRN PILOT kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
RQ_TRK_14	tc_track_prn_pilot_read	Tento scénář kontroluje, zda Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_PILOT* vrátí hodnotu dat v paměti pro PRN PILOT kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
RQ_TRK_15	tc_track_prn_data _ignore_write	Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci zápisu do PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_16	tc_track_prn_data _ignore_read	Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci čtení z PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_17	tc_track_prn_pilot _ignore_write	Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci zápisu do PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_18	tc_track_prn_pilot _ignore_read	Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci čtení z PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

RQ_TRK_19	tc_track_prn_data_boc	Tento scénář kontroluje, zda Sledovací jednotka moduluje lokální PRN DATA kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_20	tc_track_prn_pilot_boc	Tento scénář kontroluje, zda Sledovací jednotka moduluje lokální PRN PILOT kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.
RQ_TRK_21	tc_track_prn_data _integration_length	Tento scénář kontroluje, zda Sledovací jednotka integruje X+1 bitů PRN DATA kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.
RQ_TRK_22	tc_track_prn_pilot _integration_length	Tento scénář kontroluje, zda Sledovací jednotka integruje X+1 bitů PRN PILOT kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.
RQ_TRK_23	tc_track_code_phase _increments	Tento scénář kontroluje, zda Sledovací jednotka umožňuje po dokončení integrace posun fáze NCO pro PRN kódy o hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.
RQ_TRK_24	tc_track_code_phase _increments_clear	Tento scénář kontroluje, zda Sledovací jednotka po dokončení integrace vymaže hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.
RQ_TRK_25	tc_track_prn_address _increments	Tento scénář kontroluje, zda Sledovací jednotka umožňuje po dokončení integrace posun v PRN DATA a PRN PILOT kódech o hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.
RQ_TRK_26	tc_track_prn_address _increments_clear	Tento scénář kontroluje, zda Sledovací jednotka po dokončení integrace vymaže hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.
RQ_TRK_27	tc_track_carrier _nco_rate	Tento scénář kontroluje, zda Sledovací jednotka generuje jednu nosnou frekvenci pomocí NCO, které bude inkrementovat o hodnotu v registru *TRACKING_CARRIER_NCO_RATE*.
RQ_TRK_28	tc_track_correlator _delay	Tento scénář kontroluje, zda Sledovací jednotka umožňuje konfiguraci zpoždění každého z korelátorů o X+1 vzorků, kde X je hodnota *CORRELATOR_DELAY_1-6* registrů.
RQ_TRK_29	tc_track_integration _count	Tento scénář kontroluje, zda Sledovací jednotka ukládá počet dokončených integrací od začátku procesu sledování do *INTEGRATION_RESULT_COUNT* registru.
RQ_TRK_30	tc_track_correlation _results_pilot_i	Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_I* registru, kde X je číslo korelátoru.

RQ_TRK_31	tc_track_correlation _results_pilot_q	Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek imaginární části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_Q* registru, kde X je číslo korelátoru.
RQ_TRK_32	tc_track_correlation _results_pilot_prompt_i	Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z PILOT PROMPT korelátoru do *CORRELATION_RESULT_PILOT_PROMPT_I* registru.
RQ_TRK_33	tc_track_correlation _results_pilot_prompt_q	Tento scénář kontroluje, zda Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT PROMPT korelátoru do registru *CORRELATION_RESULT_PILOT_PROMPT_Q*.
RQ_TRK_34	tc_track_correlation _results_data_prompt_i	Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z DATA PROMPT korelátoru do *CORRELATION_RESULT_DATA_PROMPT_I* registru.
RQ_TRK_35	tc_track_correlation _results_data_prompt_q	Tento scénář kontroluje, zda Sledovací jednotka bude ukládat výsledek imaginární části korelace z DATA PROMPT korelátoru do registru *CORRELATION_RESULT_DATA_PROMPT_Q*.
RQ_TRK_36	tc_track_pvt _integration_count	Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální počet integrací od začátku sledovacího procesu do registru *PVT_INTEGRATION_COUNT*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_37	tc_track_pvt_prn _primary_address	Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu ukazatele na bit PRN kódu do registru *PVT_PRIMARY_ADDRESS*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_38	tc_track_pvt _code_phase	Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu fáze PRN kódu do registru *PVT_CODE_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_39	tc_track_pvt _carrier_cycles	Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální počet přetečení NCO nosné frekvence do registru *PVT_CARRIER_CYCLES*, pokud signál *PVT_LATCH* přejde do logické 1.
RQ_TRK_40	tc_track_pvt _carrier_phase	Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu fáze nosné frekvence do registru *PVT_CARRIER_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.

6 Testové procedury

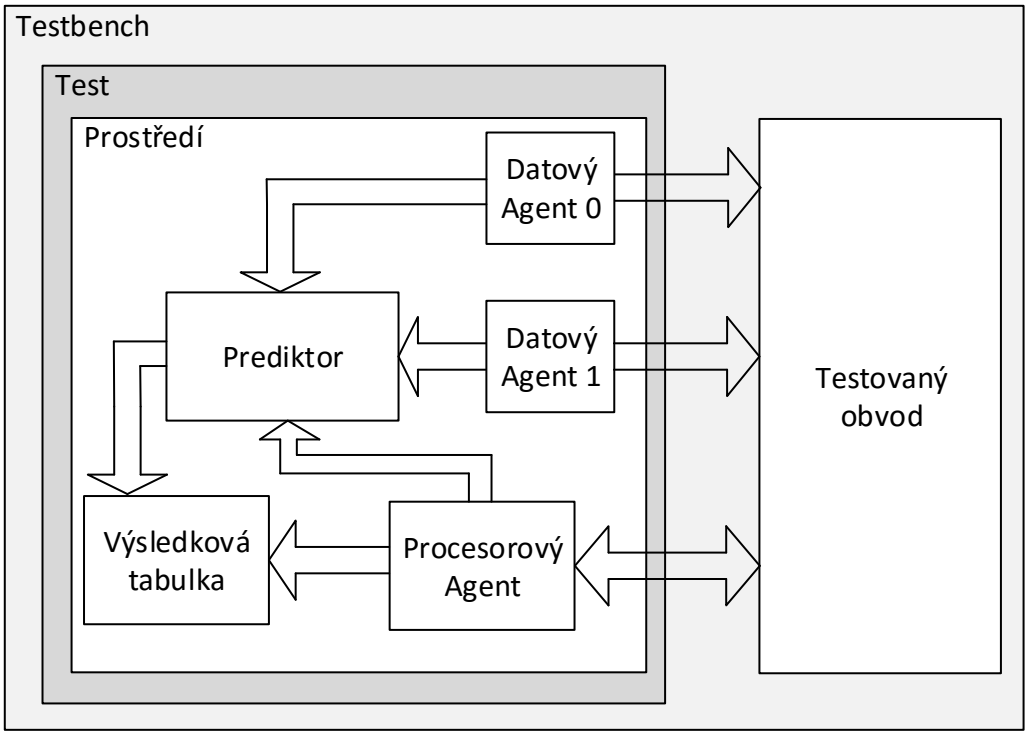
Tato kapitola obsahuje přehled testových procedur. V testových procedurách jsou implementovány postupy z testovacích scénářích v předchozí kapitole. Každý testovací scénář je implementován alespoň v jedné testové proceduře. Většina testových procedur přistupuje k testovanému obvodu stylem „black box“, tedy pouze k rozhraní testovaného obvodu. Výjimkou jsou pouze testové procedury *tp_acq_code_gen*, *tp_track_code_gen*, *tp_track_code_gen_no_boc* a *tp_track_code_gen_no_incr*, které monitorují výstupní signály generátoru PRN kódu uvnitř testovaných jednotek. Přehled testových procedur je v Tab. 7. Podrobný průběh jednotlivých testových procedur je v příloze B.

Tab. 7: Seznam testových procedur

Testová procedura	Popis
tp_acq_status	Tato testová procedura ověřuje správnou indikaci stavu Akviziční jednotky.
tp_acq_prn_memory	Tato testová procedura testuje požadavky na PRN paměť Akviziční jednotky.
tp_acq_prn_memory _no_single_read	Tato testová procedura testuje požadavky na PRN paměť Akviziční jednotky. Neprovádí čtení o délce 1 z PRN paměti.
tp_acq_code_gen	Tato testová procedura ověřuje požadavky na generaci a modulaci PRN kódu v Akviziční jednotce.
tp_acq_engine	Tato testová procedura testuje funkci Akviziční jednotky při hledání reálných satelitních signálů.
tp_acq_engine_directed	Tato testová procedura testuje funkci Akviziční jednotky při hledání reálných satelitních signálů. Generuje vstupní hodnoty nepokryté v testové proceduře *tp_acq_engine*.
tp_track_status	Tato testová procedura ověřuje přechody stavů ve Sledovací jednotce.
tp_track_prn_memory	Tato testová procedura testuje požadavky na PRN paměti Sledovací jednotky.
tp_track_prn_memory _no_single_read	Tato testová procedura testuje požadavky na PRN paměti Sledovací jednotky. Neprovádí čtení o délce 1 z PRN paměti.
tp_track_code_gen	Tato testová procedura ověřuje požadavky na generaci a modulaci PRN kódů ve Sledovací jednotce.
tp_track_code_gen_no_boc	Tato testová procedura ověřuje požadavky na generaci PRN kódů ve Sledovací jednotce.
tp_track_code_gen_no_incr	Tato testová procedura ověřuje požadavky na generaci a modulaci PRN kódů ve Sledovací jednotce. Neprovádí korekce fáze PRN kódů a ukazetle do pametí PRN kódů.
tp_track_engine	Tato testová procedura testuje výpočet správných integračních a PVT výsledků ve Sledovací jednotce při sledování reálných satelitních signálů.
tp_track_engine_no_boc	Tato testová procedura testuje výpočet správných integračních a PVT výsledků ve Sledovací jednotce při sledování reálných satelitních signálů. BOC modulace je neaktivní.
tp_track_engine_no_incr	Tato testová procedura testuje výpočet správných integračních a PVT výsledků ve Sledovací jednotce při sledování reálných satelitních signálů. Neprovádí korekce fáze PRN kódů a ukazetle do pametí PRN kódů.

7 Verifikační prostředí

Blokové schéma verifikačního prostředí je zobrazeno na Obr. 15. Zobrazuje pouze bloky, které byly vytvořeny nebo použity v této práci.



Obr. 15: Blokové schéma verifikačního prostředí

Nejvyšší jednotkou verifikačního prostředí je modul *gnss_tb_top* (na obrázku Testbench), ktyrý zajišťuje propojení testovaného obvodu a instance třídy *test_base* (na obrázku Test), která implementuje testovací prostředí.

Třída *test_base* je základní třídou pro testové procedury. Obsahuje prvky společné pro všechny testové procedury, jako například vytvoření třídy Prostředí a inicializaci testovaného obvodu a Prediktoru.

V třídě Prostředí jsou vytvořeny a propojeny třídy Výsledková tabulka, Prediktor, Procesorový Agent a Datový Agent.

Procesorový Agent slouží k simulaci komunikace testovaného obvodu s procesorem. Zapisovaná data do testovaného obvodu jsou také odeslána do Prediktoru. Při vyčítání jsou vyčtená data odeslána do Výsledkové tabulky.

Výsledková tabulka v průběhu simulace porovnává data vyčtená z testovaného obvodu Procesorovým Agentem s referenčními hodnotami z Predictoru.

Výše zmíněné bloky nebyly vytvořeny v této práci, proto nebudou více popisovány. Bloky vytvořené v rámci této práce jsou popsány v následujících kapitolách.

7.1 Datový Agent

Úkolem Datového Agenta je zápis dat satelitního signálu do testovaného obvodu. Na každý vstupní kanál je jeden Datový Agent. Obsahuje třídy Datový driver, Datový sekvencer, Datová sekvence, rozhraní Datové rozhraní, funkce *build_phase()*, *connect_phase()* a procedury *read_input_file()* a *send_transaction()*.

Ve funkci *build_phase()* je vytvořen Datový driver, Datový sekvencer, analyzační port (slouží k propojení a předávání transakcí mezi driverem a prediktorem) a je získáno Datové rozhraní z konfigurační databáze. Tato funkce je volána v UVM *build* fázi.

Funkce *connect_phase()* propojí rozhraní v Datovém Agentu s rozhraním v Datovém driveru, Datový driver s Datovým sekvencerem a analyzační port Agenta s portem v driveru. Tato funkce je volána v UVM *connect* fázi.

Procedura *read_input_file()* slouží k načtení 4bitových dat ze souboru. Procedura má 3 vstupní parametry. První vstupní parametr je cesta k vyčítanému souboru. Počet vyčtených dat je dán třetím parametrem. Jeho výchozí hodnota je nastavena do 0, kdy je vyčten celý soubor. Vyčtená data jsou převedena binární podoby. Poté je vytvořeno pole 12bitových náhodných hodnot, které jsou přepsány vyčtenými daty. Pozice nejméně významného bitu, který bude přepsán je v druhém vstupním parametru procedury. Následuje zavolání procedury *send_transaction()*.

Procedura *send_transaction()* má 2 vstupní parametry. Pole 12bitových dat a délku tohoto pole. Procedura vytvoří novou Datovou transakci a předá jí pole dat a jeho délku. Nakonec předá transakci Datovému sekvenceru, který transakci pomocí Datového driveru odešle přes Datové rozhraní do testovaného obvodu.

Zdrojový kód Datového Agenta je v souboru *data_ifc_agent.sv*.

7.1.1 Datový sekvencer

Datový sekvencer řídí komunikaci mezi sekvencemi a Datovým driverem. Je parametrizován na Datovou transakci. Obsahuje pouze funkci konstruktoru. Zdrojový kód Datového sekvenceru je v souboru *data_ifc_sequencer.sv*.

7.1.2 Datový driver

Datový driver odesílá Datové transakce přes Datové rozhraní do testovaného obvodu. Jeho parametrem je Datová transakce. Obsahuje funkci *build_phase()* a procedury *run_phase()*, *driver_default_values()* a *process_transaction()*.

Ve funkci *build_phase()* je vytvořen analyzační port a z konfigurační databáze je získáno zpoždění zápisu transakce do analyzačního portu. Zpoždění je nutné kvůli způsobu implementace v Prediktoru. Tato funkce je volána v UVM *build* fázi.

Procedura *driver_default_values()* nastaví na pinech Datového rozhraní výchozí hodnoty.

Procedura *run_phase()* v nekonečné smyčce vydává požadavek Datovému sekvenceru, zda je k dispozici další Datová transakce. Pokud je k dispozici, tak zavolá proceduru *process_transaction()* a transakci předá jako parametr. Po dokončení procedury oznámí

sekvenceru dokončení transakce a vznese další požadavek. Paralelně s tímto zapíše transakci do analyzačního portu pro Prediktor. Tato procedura je aktivní v UVM *run* fázi.

V proceduře *process_transaction()* je každý hodinový takt na piny Datového rozhraní vystavena hodnota dat v transakci.

Zdrojový kód Datového driveru je v souboru *data_ifc_driver.sv*.

7.1.3 Datová transakce

Pomocí Datové transakce je předáváno pole 12bitových dat a délka pole jako atributy transakce. Obsahuje 4 funkce *do_copy(), do_compare(), do_print()* a *convert2string()*.

Funkce *do_copy()* zkopíruje předané atributy do nově vytvořených atributů.

Funkce *do_compare()* porovná předané atributy s atributy transakce a vrátí výsledek porovnání.

Funkce *do_print()* vypíše atributy do konzole nebo předá třídě *uvm_printer*.

Funkce *convert2string()* vrátí atributy jako řetězec znaků.

Zdrojový kód Datové transakce je v souboru *data_ifc_transaction.sv*.

7.1.4 Datové rozhraní

Datové rozhraní propojuje prostředí s datovým vstupem testovaného obvodu. Obsahuje jeden 12bitový signál, na který jsou hodnoty vystaveny na sestupnou hranu hodinového signálu.

Zdrojový kód Datového rozhraní je v souboru *data_ifc_interface.sv*.

7.2 Prediktor

Úkolem Prediktoru je generovat referenční hodnoty. Obsahuje registrovou mapu a modely jednotlivých jednotek testovaného obvodu, které jsou vytvořeny a propojeny v nejvyšším úrovni Prediktoru. Transakce jsou přijímány analyzačními exporty, které jsou propojeny s analyzačními porty v agentech. Každý analyzační export má přiřazenu funkci, kde je nadefinováno zpracování transakce.

7.2.1 Prediktor Akviziční jednotky

Prediktor Akviziční jednotky počítá očekávané výsledky akvizičního procesu.

Po přijetí transakce z Datového Agenta zavolá Prediktor funkci *process_transaction()*, které předá délku transakce a pole 3bitových dat.

Funkce *process_transaction()* zavolá funkci *read_registers()*, která vrátí hodnotu *Enable* bitu Akviziční jednotky. Pokud je *Enable* bit v logické 0, tak je transakce ignorována. V opačném případě následuje volání funkcí *create_prn_vector(), carrier_wipeoff(), calculate_max_results()* a *write_results()*.

Funkce *read_registers(),* v případě *Enable* bitu v logické 1, načte hodnoty registrů Akviziční jednotky, které souvisí s akvizičním procesem, a PRN kód. PRN kód je poté převeden z hodnot 0 a 1 na ±1. Tím se zjednoduší pozdější korelace na jednoduché násobení.

Dále je vypočten počet potřebných datových vzorků pro vypočtení výsledků korelační amplitudy. Tato hodnota je porovnána s přijatou délkou transakce. Pokud je přijatá transakce menší, je simulace ukončena.

Ve funkci *create_prn_vector()* je z PRN kódu vytvořen vektor, který bude vstupem pro korelátory. Dále je zde provedena BOC (1, 1) modulace PRN vektoru, pokud je povolena, a jeho decimace.

Funkce *carrier_wipeoff()* provede demodulaci dat z nosné frekvence. Výsledkem je reálná a imaginární část.

Zpracování transakce je dokončeno ve funkci *calculate_max_results()*. Pro každý korelátor je provedena korelace PRN vektoru s reálnou a imaginární hodnotou dat. Korelované výsledky jsou poté akumulovány. Následuje bitová redukce a následuje nekoherentní nebo diferenciální integrace. Poté je vypočtena korelační amplituda. V případě nejvyšší hodnoty korelační amplitudy je uložena hodnota reálné a imaginární části spolu s indexem korelátoru. Proces od korelace po uložení hodnot generujících maximu korelační amplitudy je opakován podle hodnoty v *INTEGRATION_TIME* registru.

Výsledné hodnoty jsou ve funkci *write_results()* uloženy do registrů Akviziční jednotky.

Zdrojový kód Prediktoru Akviziční jednotky je v souboru *acq_engine_predictor.sv*.

7.2.2 Prediktor Sledovací jednotky

Prediktor Sledovací jednotky počítá očekávané integrační a PVT výsledky.

Po přijetí transakce z Datového Agenta je zavolána funkce *new_transaction()*, která zkopíruje obsah transakce do atributů Prediktoru Sledovací jednotky a vygeneruje událost přijetí nové transakce *new_tr_event*.

Procedura *run_phase()* čeká v nekonečné smyčce na událost *new_tr_event* a zavolá proceduru *process_transaction()*, které předá atributy s obsahem transakce. Tato funkce je aktivní v UVM *run* fázi.

V proceduře *process_transaction()* je transakce zpracována. Nejprve zavolá funkci *read_registers()*. Tato funkce má stejnou úlohu jako u Akviziční jednotky, tedy zkontrolovat hodnotu *Enable* bitu a vrátit jeho hodnotu. Pokud je v logické 1, načte hodnoty registrů Sledovací jednotky, které souvisí se sledovacím procesem, PRN DATA a PRN PILOT kódy, které převede z hodnot 0 a 1 na ±1, a zkontroluje, že délka transakce je dostatečná k uskutečnění alespoň jedné integrace.

Dále procedura *process_transaction()* pokračuje *for* cyklem, kde zpracuje data z transakce. Zpracování začíná až od vzorku podle hodnoty z *TRACKING_START_TIMESTAMP* registru. Nejprve probíhá část pro generaci PRN kódů a BOC (1, 1) modulace. Pokud je konec integrované části PRN kódů, tak jsou načteny korekce z registrů a tyto registry nastaveny do nuly. Vyčtené korekce pro PRN NCO a ukazatel na PRN bit v paměti jsou aplikovány. Vygenerované hodnoty PRN DATA a PRN PILOT bitu jsou zapsány do posuvného registru. Mezitím je provedena demodulace dat z nosné frekvence. Reálná a imaginární část je v korelátorech korelována s hodnotou v posuvném registru

na pozici, která je daná *CORRELATOR_DELAY_1-6* registry. Korelované hodnoty jsou poté akumulovány. Pokud je konec integrace, tak jsou akumulované hodnoty zapsány do registrů pro výsledky integrace. Dále se kontroluje, zda se mají zapsat PVT hodnoty, tedy počet integrací, ukazatel do PRN paměti, fáze PRN NCO, fáze NCO nosné frekvence a jeho počet přetečení. Po každém cyklu se čeká po dobu periody hodinového signálu v Datovém rozhraní, aby byly Sledovací jednotka v testovaném obvodu a Prediktor Sledovací jednotky přibližně synchronizovány.

Zdrojový kód Prediktoru Sledovací jednotky je v souboru *track_engine_predictor.sv*.

8 Průběh verifikace a její výsledky

Celý verifikační proces je proveden pomocí skriptů.

Nejprve je spuštěn skript na kompilaci zdrojových kódů s pokrytím kódu a optimalizací. Následuje skript, který spustí simulátor a pomocí příkazového řádku předá simulátoru parametry jako semínko pro generátor náhodných čísel, název testu a podobně.

Po spuštění simulátoru proběhne *initial* blok v modulu *gnss_tb_top*. Zde je nastaven jednotný formát času pro všechny komponenty, do konfigurační databáze jsou vloženy nadefinované časové konstanty a rozhraní a je nastaven globální časový limit pro simulaci. Následuje zavolání metody *run_test()*, čímž jsou spuštěny UVM fáze (viz kapitola 3.2).

Zde přebírá kontrolu nad simulací třída *test_base* a testová procedura. Třída *test_base* obsahuje prvky společné pro všechny testové procedury jako je vytvoření verifikačního prostředí, propojení s testovaným obvodem a podobně.

Na začátku fáze aplikování stimulů je "vznesena námitka" (*raise_objection*) proti ukončení fáze. Následuje spuštění hodinových signálů a reset testovaného obvodu. Poté je zavolána procedura *run_test()*. Tato procedura je definována jako *pure virtual*, což znamená, že jiné třídy (všechny testové procedury) dědící z třídy *test_base* musí mít tuto proceduru implementovanou. Průběh procedury *run_test()* v jednotlivých testových procedurách je popsán v příloze B.

Po proběhnutí procedury *run_test()* v testové proceduře je vypsán její výsledek, třída *test_base* "vezme námitku zpět" (*drop_objection*), fáze aplikování stimulů je ukončena a po ukončení zbytku UVM fází je ukončen simulátor.

Po dokončení všech požadovaných testových procedurách je spuštěn skript na analýzu výpisů výsledků jednotlivých testových procedur a do souboru *results_rtl.html* vypíše počet *UVM_ERROR* (chyby), *UVM_WARNINGS* (varování) a *UVM_INFO* (oznámení), případně *UVM_FATAL* (fatální chyby), pro každou spuštěnou testovou proceduru.

Nakonec je spuštěn skript na sloučení výsledků pokrytí kódu a funkčního pokrytí pro všechny proběhlé testové procedury.

Požadavky budou považovány za splněné, pokud bude splněno funkční pokrytí a příslušné testové procedury proběhnou bez chyb. Testová procedura je automaticky ukončena po 10 chybách *UVM_ERROR*.

8.1 Výsledky verifikace Akviziční jednotky

Seznam spuštěných testových procedur zaměřených na verifikaci Akviziční jednotky je v Tab. 8. Informace jsou také v souboru *results_rtl.html*.

Tab. 8: Status spuštěných testových procedur Akviziční jednotky

Testová procedura	Semínko	Počet chyb
tp_acq_engine	0-16	0
tp_acq_engine_directed	0	0
tp_acq_status	0	0
tp_acq_code_gen	0	0
tp_acq_prn_memory	0	10
tp_acq_prn_memory_no_single_read	0	0

V testové proceduře *tp_acq_prn_memory* testovaný obvod vracel při operaci čtení o délce 1 s *Enable* bitem v logické 0 rozdílné hodnoty od referenčních hodnot z Prediktoru. Tato operace způsobila 10 chyb a testová procedura byla automaticky ukončena. Procedura *tp_acq_prn_memory_no_single_read*, která tuto operaci neprovádí, proběhla bez chyb. Požadavek RQ_ACQ_8 tedy není splněn.

V Tab. 9 je seznam splněných a nesplněných požadavků Akviziční jednotky. První sloupec obsahuje verifikovaný požadavek. Ve druhém sloupci je uveden příslušný testovací scénář. Třetí sloupeček obsahuje všechny testové procedury, kde je implementován postup z testovacího scénáře. K ověření požadavku stačí provedení jedné z uvedených testových procedur bez chyb. Poslední sloupeček uvádí, zda byl požadavek při verifikaci splněn.

Tab. 9: Výsledek verifikace Akviziční jednotky

Požadavek	Testovací scénář	Testová procedura	Splněno
RQ_ACQ_1	tc_acq_enable	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_2	tc_acq_status	tp_acq_status	ANO
RQ_ACQ_3	tc_acq_channel_select	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_4	tc_acq_code_nco_rate	tp_acq_engine tp_acq_engine_directed tp_acq_code_gen	ANO
RQ_ACQ_5	tc_acq_prn_generation	tp_acq_engine tp_acq_engine_directed tp_acq_code_gen	ANO
RQ_ACQ_6	tc_acq_prn_length	tp_acq_engine tp_acq_engine_directed tp_acq_code_gen	ANO
RQ_ACQ_7	tc_acq_prn_data_write	tp_acq_prn_memory tp_acq_prn_memory_no_single_read	ANO
RQ_ACQ_8	tc_acq_prn_data_read	tp_acq_prn_memory tp_acq_prn_memory_no_single_read	NE

RQ_ACQ_9	tc_acq_prn_ignore _data_write	tp_acq_prn_memory tp_acq_prn_memory_no_single_read	ANO
RQ_ACQ_10	tc_acq_prn_ignore _data_read	tp_acq_prn_memory tp_acq_prn_memory_no_single_read	ANO
RQ_ACQ_11	tc_acq_prn_boc	tp_acq_engine tp_acq_engine_directed tp_acq_code_gen	ANO
RQ_ACQ_12	tc_acq_decimation_factor	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_13	tc_acq_carrier_nco_rates	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_14	tc_acq_single_carrier	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_15	tc_acq_carriers	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_16	tc_acq_coherent _integration_time	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_17	tc_acq_integration_time	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_18	tc_acq_noncoherent _integration	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_19	tc_acq_differential _integration	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_20	tc_acq_inphase_results	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_21	tc_acq_quadrature_results	tp_acq_engine tp_acq_engine_directed	ANO
RQ_ACQ_22	tc_acq_correlator _max_amplitude	tp_acq_engine tp_acq_engine_directed	ANO

Pokrytí kódu Akviziční jednotky je 96,54 %. Nepokryté části kódu buď není možné pokrýt nebo bude pokryto v jiných testových procedurách, které se zaměřují na Procesorové rozhraní (například přístup na nedefinovanou adresu) a nejsou součástí této práce. Z pohledu funkčního pokrytí bylo pokryto vše s výjimkou čtení o délce 1 z PRN paměti, při kterém jsou vyčítány jiné než referenční hodnoty. Hodnoty funkčního pokrytí jsou v souboru *fcover_report.txt*.

8.2 Výsledky verifikace Sledovací jednotky

Seznam spuštěných testových procedur zaměřených na verifikaci Sledovací jednotky je v Tab. 10. Informace jsou také v souboru *results_rtl.html*.

Tab. 10: Status spuštěných testových procedur ověřující Sledovací jednotku

Testová procedura	Semínko	Počet chyb
tp_track_engine	0	10
tp_track_engine_no_incr	0	10
tp_track_engine_no_boc	0	10
tp_track_status	0	0
tp_track_code_gen	0	10
tp_track_code_gen_no_boc	0	10
tp_track_code_gen_no_incr	0	0
tp_track_prn_memory	0	10
tp_track_prn_memory_no_single_read	0	0

Ze 4 hlavních procedur nebyly dokončeny bez chyb procedury *tp_track_engine*, *tp_track_code_gen* a *tp_track_prn_memory*. U těchto procedur testovaný obvod vracel neočekávané hodnoty. Proto byly přidány modifikace těchto testových procedur.

Z testové procedury *tp_track_prn_memory* bylo stejně jako u *tp_acq_prn_memory* odstraněno čtení z PRN pamětí Sledovací jednotky o délce 1. Takto upravená procedura *tp_track_prn_memory_no_single_read* již proběhla v pořádku.

U testové procedury *tp_track_code_gen* byly přidány 2 modifikové verze. První, *tp_track_code_gen_no_boc*, neprovádí BOC modulaci PRN kódů. Generátor PRN kódů ve Sledovací jednotce obsahuje chybu a při aplikaci korekcí fáze PRN kódů a ukazatele do PRN paměti občas generuje špatné hodnoty PRN kódů. Tato procedura byla automaticky ukončena po 10 chybných porovnání očekávaných a vyčtených hodnot. Druhá modifikovaná procedura, *tp_track_code_gen_no_incr*, nezapisuje korekce fáze PRN kódů a ukazatele do PRN paměti. Tato procedura již proběhla v pořádku. Testová procedura *tp_track_code_gen_no_incr* neobsahuje funkční pokrytí, proto byly z přepisu průběhu testové procedury ručně zkontrolovány generované vstupní hodnoty:

- BOC Enable bitu,
- rychlosti generace PRN kódů,
- délka PRN kódů,
- integrovaná délka PRN kódů.

Testová procedura *tp_track_track_engine* a obě její modifikace vracely nekorektní hodnoty a byly automaticky ukončené po 10 špatných porovnání hodnot. Ve Sledovací jednotce jsou přítomné i jiné chyby než zmíněná v předchozím odstavci, protože při vyčítání integračních výsledků obvod vrací neočekávané hodnoty i bez aplikace korekce fáze PRN kódů a ukazatele do PRN paměti.

Většina požadavků na Sledovací jednotku má přiřazeny testové procedury, z nichž žádná neproběhla bez chyb, proto jsou tyto požadavky považovány za nesplněné.

V Tab. 11 je seznam splněných a nesplněných požadavků Sledovací jednotky. První sloupec obsahuje verifikovaný požadavek. Ve druhém sloupci je uveden příslušný testovací scénář. Třetí sloupeček obsahuje všechny testové procedury, kde je implementován postup z testovacího scénáře. K ověření požadavku stačí provedení jedné z uvedených testových procedur bez chyb. Poslední sloupeček uvádí, zda byl požadavek při verifikaci splněn.

Tab. 11: Výsledky verifikace Sledovací jednotky

Požadavek	Testovací scénář	Testová procedura	Splněno
RQ_TRK_1	tc_track_enable	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr	NE
RQ_TRK_2	tc_track_status_idle	tp_track_status	ANO
RQ_TRK_3	tc_track_status_armed	tp_track_status	ANO
RQ_TRK_4	tc_track_status_tracking	tp_track_status	ANO
RQ_TRK_5	tc_track_channel_select	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr	NE
RQ_TRK_6	tc_track_code_nco_rate	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen tp_track_code_gen_no_boc tp_track_code_gen_no_incr	ANO
RQ_TRK_7	tc_track_prn_data _generation	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen tp_track_code_gen_no_boc tp_track_code_gen_no_incr	ANO
RQ_TRK_8	tc_track_prn_pilot _generation	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen tp_track_code_gen_no_boc tp_track_code_gen_no_incr	ANO
RQ_TRK_9	tc_track_prn_data_length	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen tp_track_code_gen_no_boc tp_track_code_gen_no_incr	ANO
RQ_TRK_10	tc_track_prn_pilot_length	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen tp_track_code_gen_no_boc tp_track_code_gen_no_incr	ANO

RQ_TRK_11	tc_track_prn_data_write	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_12	tc_track_prn_data_read	tp_track_prn_memory tp_track_prn_memory_no_single_read	NE
RQ_TRK_13	tc_track_prn_pilot_write	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_14	tc_track_prn_pilot_read	tp_track_prn_memory tp_track_prn_memory_no_single_read	NE
RQ_TRK_15	tc_track_prn_data _ingore_write	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_16	tc_track_prn_data _ignore_read	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_17	tc_track_prn_pilot _ingore_write	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_18	tc_track_prn_pilot _ignore_read	tp_track_prn_memory tp_track_prn_memory_no_single_read	ANO
RQ_TRK_19	tc_track_prn_data_boc	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen	ANO
RQ_TRK_20	tc_track_prn_pilot_boc	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr tp_track_code_gen	ANO
RQ_TRK_21	tc_track_prn_data _integration_length	tp_track_engine tp_track_engine_no_boc tp_track_engine_no_incr	NE

Pokrytí kódu celé Sledovací jednotky je 66,87 %. U některých testových procedur byl z důvodů doby trvání simulace ověřován pouze první sledovací kanál. Pokrytí kódu prvního kanálu Sledovací jednotky je 91,62 %. Hodnoty funkčního pokrytí jsou v souboru *fcover_report.txt*.

9 Závěr

Cílem této práce byla verifikace Akviziční jednotky a Sledovací jednotky v digitálním obvodu Microcore GNSS Baseband. V prvním kapitole byl rozebrán princip globálních navigačních satelitních systémů. Ve druhé kapitole byl popsán princip činnosti testovaných jednotek. Třetí kapitola se věnovala úvodu do metodiky UVM. Ve čtvrté kapitole byly sepsány požadavky na testované jednotky. V páté kapitole byl v testovacích scénářích popsán postup k ověření požadavků. Šestá kapitola se věnovala přehledu testových procedur. V sedmé kapitole bylo popsáno verifikační prostředí pro verifikaci jednotek. Výsledky verifikace a přehled splněných a nesplněných požadavků jsou obsaženy v osmé kapitole.

Během verifikace bylo přistupováno pouze k rozhraním testovaného obvodu. Jedinou výjimkou byl generátor PRN kódu v obou jednotkách, kde se monitorovaly přímo výstupy generátoru. Výhodou tohoto přístupu je ověření všech požadavků v menším počtu testových procedur. Zároveň se ověřuje i správná synchronizace vnitřních bloků jednotek. Nevýhodami jsou obtížná detekce zdroje chyb. Dále jediná chyba způsobí nesplnění více požadavků, což je případ Sledovací jednotky.

Na Akviziční jednotku bylo sepsáno 22 požadavků. Splněno jich bylo celkem 21. Požadavek RQ_ACQ_8 nebyl splněn, protože obvod vracel jiné než referenční hodnoty. Pokrytí kódu v Akviziční jednotce dosáhlo hodnoty 96,54 %. Pro splněné požadavky bylo splněno i funkční pokrytí.

Na Sledovací jednotku bylo sepsáno 40 požadavků. Pro testové procedury, které proběhly bez chyb (*tp_track_status*, *tp_track_prn_memory_no_single_read* a *tp_track_code_gen_no_incr*) bylo zkontrolováno funkční pokrytí. Testová procedura *tp_track_code_gen_no_incr* neobsahovala skupinu funkčního pokrytí, proto byly z přepisu průběhu testové procedury generované vstupní hodnoty zkontrolovány ručně. Navíc v testové proceduře *tp_acq_code_gen* jsou stejné proměnné se stejně omezenými rozsahy možných hodnot a pro generátor náhodných hodnot bylo použito stejné semínko, takže funkční pokrytí musí být také splněno. Požadavků bylo splněno celkem 16. Zbývajících 24 požadavků nebylo splněno, protože Sledovací jednotka stále obsahovala chybu v generátoru PRN kódů při aplikací korekcí a také chybu u integrační délky PRN kódů. Tyto chyby zásadně ovlivňují integrační výsledky, pomocí kterých je ověřována většina požadavků. Ani jedna testová procedura určená k ověření požadavků přes integrační nebo PVT výsledky neproběhla bez chyb, a proto byly tyto požadavky považovány za nesplněné. Pokrytí kódu celé Sledovací jednotky dosáhlo hodnoty 66,87 %, pokud je počítáno se všemi kanály Sledovací jednotky. Z důvodu času potřebného k simulaci všech kanálů u některých testových procedur byl ověřován pouze první kanál. První kanál Sledovací jednotky měl pokrytí kódu 91,62 %.

Použité zdroje

[1] Kaplan, Elliott D., Hegarty, Christopher J. *Understanding GPS: Principles and Applications – 2nd edition*, Artech House, 2006, ISBN 1-58053-894-0

[2] An Introduction to GNSS, *NovAtel [online]*. [cit. 2017-11-29]. Dostupné z WWW: <https://www.novatel.com/an-introduction-to-gnss/>.

[3] What is Galileo, *European Space Agency* [online]. [cit. 2017-11-29]. Dostupné z WWW: <http://www.esa.int/Our_Activities/Navigation/Galileo/Building_Galileo>.

[4] Brierley-Green, Andrew, *Global Navigation Satellite System Fundamentals and Recent Advances in Receiver Design*, Maxim Integrated Inc. [online]. 2017 [cit. 2017-11-30]. Dostupné z WWW: <https://www.ieee.li/pdf/viewgraphs/gnss_fundamentals.pdf>

[5] Subinara, J. Sanz, Zornoza, JM. Juan, Hernandez-Pajares, M. *GNSS signal* [online]. Spain: University of Catalunia, 2011 [cit. 2017-11-30]. Dostupné z WWW: <http://www.navipedia.net/index.php/GNSS_signal>.

[6] Acquisition, [online]. [cit. 2017-12-05]. Dostupné z WWW: <https://jaybehaveco.files.wordpress.com/2013/10/acquisition.png>.

[7] Valkama, M. *Complex-Valued Signals and Systems: Basic Principles and Applications to Radio Communications and Radio Signal Processing* [online]. Finland: Tampere University of Technology, Department of Communications Engineering. 2011 [cit. 2017-12-5]. Dostupné z WWW: <http://www.cs.tut.fi/kurssit/TLT-9707/presentations/Complex-Signals-and-Radios-short-2pp.pdf>.

[8] Sickle, J. V. *GPS and GNSS for Geospatial Professionals* [online]. USA: Pennsylvania State University, Department of Geography. [cit. 2017-12-6]. Dostupné z WWW: <https://www.e-education.psu.edu/geog862/node/1785>.

[9] BOC(1,1) Signal Generation, [online]. [cit. 2017-12-7]. Dostupné z WWW: <http://mycoordinates.org/wp-content/uploads/2010/08/1a.jpg>.

[10] Power Spectrum of BPSK and BOC(1,1), [online]. [cit. 2017-12-8]. Dostupné z WWW: <http://what-when-how.com/wp-content/uploads/2012/02/tmpE199_thumb2.jpg>.

[11] Lohan, S. *Spread Spectrum techniques* [online]. Finland: Tampere University of Technology, Department of Communications Engineering. 2011 [cit. 2017-12-8]. Dostupné z WWW: <http://www.cs.tut.fi/kurssit/TLT-5606/Lec11_TLT5606_S_28Apr2011.pdf>.

[12] Universal Verification Methodology (UVM) 1.2 User´s Guide, *Accellera Systems Initiative* [online]. 2015 [cit. 2017-12-12]. Dostupné z WWW: <http://www.accellera.org/images//downloads/standards/uvm/uvm_users_guide_1.2.pdf>.

[13] The Universal Verification Methodology, *Doulos* [online]. [cit. 2017-12-12]. Dostupné z WWW: <https://www.doulos.com/knowhow/sysverilog/uvm/>.

[14] What is SystemVerilog?, *Doulos* [online]. [cit. 2017-12-13]. Dostupné z WWW: <https://www.doulos.com/knowhow/sysverilog/whatissv/>.

[15] Fabio Dovis and Tung Hai Ta (2012). High Sensitivity Techniques for GNSS Signal Acquisition, Global Navigation Satellite Systems: Signal, Theory and Applications, prof. Shuanggen Jin (Ed.), ISBN: 978-953-307-843-4, InTech, Dostupné z WWW: <http://www.intechopen.com/books/global-navigation-satellite-systems-signaltheory-and-applications/high-sensitivity-techniques-for-gnss-signal-acquisition>.

Seznam příloh

A Testovací scénáře

V této kapitole jsou podrobněji rozepsány testovací scénáře ke každému požadavku.

A.1 TC_ACQ_ENABLE

Popis

Tento scénář kontroluje, zda Akviziční jednotka zpracovává vstupní data, pouze pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Enable* bitu a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude zpracovávat vstupní data, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.2 TC_ACQ_STATUS

Popis

Tento scénář kontroluje, zda Akviziční jednotka správně indikuje svůj stav pomocí *ACQUISITION_STATUS* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou přečteny *Acquisition Status* bity v *ACQUISITION_CONTROL* registru po každém z následujících bodů:
    - inicializace obvodu,
    - akviziční proces je aktivní (*Enable* bit v logické 1),
    - akviziční proces přerušen (*Enable* bit z logické 1 do logické 0),
    - dokončení akvizičního procesu.

Očekávané výsledky

Akviziční jednotka bude správně indikuje svůj stav pomocí *ACQUISITION_STATUS* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Vstupní i výstupní hodnoty budou kontrolovány v testové proceduře.

A.3 TC_ACQ_CHANNEL_SELECT

Popis

Tento scénář kontroluje, zda Akviziční jednotka zpracovává data ze vstupního datového kanálu podle hodnoty *Channel Select* bitů *ACQUISITION_CONTROL* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Channel Select* bitů a ostatních registrů nastavující parametry akvizice.
3) Do každého vstupního kanálu budou posílána jiná data.
4) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude zpracovávat data ze vstupního datového kanálu podle hodnoty *Channel Select* bitů *ACQUISITION_CONTROL* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.4 TC_ACQ_CODE_NCO_RATE

Popis

Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfiguraci rychlosti generace lokálního PRN kódu pomocí *CODE_NCO_RATE* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *CODE_NCO_RATE* registru.
3) V závislosti na testové proceduře implementující tento testovací scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Do ostatních registrů nastavující parametry akvizice budou zapsány náhodné hodnoty. Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN kódu pomocí *CODE_NCO_RATE* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.5 TC_ACQ_PRN_GENERATION

Popis

Tento scénář kontroluje, zda Akviziční jednotka generuje lokální PRN kód z paměti pro PRN kód Akviziční jednotky.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN paměti Akviziční jednotky bude zapsán reálný PRN kód vygenerovaný pomocí interního generátoru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry akvizice. Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude generovat lokální PRN kód z paměti pro PRN kód Akviziční jednotky.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.6 TC_ACQ_PRN_LENGTH

Popis

Tento scénář kontroluje, zda Akviziční jednotka generuje lokální PRN kód o délce X+1 bitů, kde X je hodnota *ACQ_PRN_LENGTH* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN paměti Akviziční jednotky bude zapsán reálný PRN kód vygenerovaný pomocí interního generátoru a jeho délka zmenšená o 1 bude zapsána do *ACQ_PRN_LENGTH* registru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry akvizice. Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude generovat lokální PRN kód o délce X+1 bitů, kde X je hodnota *ACQ_PRN_LENGTH* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.7 TC_ACQ_PRN_DATA_WRITE

Popis

Tento scénář kontroluje, zda Akviziční jednotka při operaci zápisu dat do registru *ACQUISITION_PRN_DATA* uloží tyto data do paměti PRN kódu Akviziční jednotky na adresu podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do registru *ACQUISITION_PRN_DATA* budou zapisována náhodná data.
3) Po každém zápisu bude vyčtena hodnota v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.
4) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Akviziční jednotka při operaci zápisu dat do registru *ACQUISITION_PRN_DATA* uloží tyto data do paměti PRN kódu Akviziční jednotky na adresu podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.8 TC_ACQ_PRN_DATA_READ

Popis

Tento scénář kontroluje, zda Akviziční jednotka při operaci čtení z registru *ACQUISITION_PRN_DATA* vrátí hodnotu dat v paměti PRN kódu Akviziční jednotky z adresy podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do paměti PRN kódu Akviziční jednotky budou zapsána náhodná data.
3) Poté bude vyčítán *ACQUISITION_PRN_DATA* registr.
4) Po každém čtení bude vyčtena hodnota v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Očekávané výsledky

Akviziční jednotka při operaci čtení z registru *ACQUISITION_PRN_DATA* vrátí hodnotu dat v paměti PRN kódu Akviziční jednotky z adresy podle hodnoty v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.9 TC_ACQ_PRN_IGNORE_DATA_WRITE

Popis

Tento scénář kontroluje, zda Akviziční jednotka bude ignorovat operaci zápisu do PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *ACQUISITION_CONTROL* registru bude nastaven do logické 1.
3) Do registru *ACQUISITION_PRN_DATA* budou zapisována náhodná data.
4) Po každém zápisu bude vyčtena hodnota v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.
5) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Akviziční jednotka bude ignorovat operaci zápisu do PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.10 TC_ACQ_PRN_IGNORE_DATA_READ

Popis

Tento scénář kontroluje, zda Akviziční jednotka bude ignorovat operaci čtení z PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *ACQUISITION_CONTROL* registru bude nastaven do logické 1.
3) Do paměti PRN kódu Akviziční jednotky budou zapsána náhodná data.
4) Poté bude vyčítán *ACQUISITION_PRN_DATA* registr.
5) Po každém čtení bude vyčtena hodnota v registru *ACQUISITION_PRN_DATA_ADDRESS_OFFSET*.

Očekávané výsledky

Akviziční jednotka bude ignorovat operaci čtení z PRN paměti pro Akviziční jednotku, pokud je *Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.11 TC_ACQ_PRN_BOC

Popis

Tento scénář kontroluje, zda Akviziční jednotka moduluje lokální PRN kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *BOC Modulation Enable* bitu.

3) V závislosti na testové proceduře implementující tento testovací scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Do ostatních registrů nastavující parametry akvizice budou zapsány náhodné hodnoty. Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude modulovat lokální PRN kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *ACQUISITION_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.12 TC_ACQ_DECIMATION_FACTOR

Popis

Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfigurovatelné zpoždění lokálního PRN kódu mezi korelátory o X+1 vzorků, kde X je hodnota *DECIMATION_FACTOR* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *DECIMATION_FACTOR* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude umožňovat konfigurovatelné zpoždění lokálního PRN kódu mezi korelátory o X+1 vzorků, kde X je hodnota *DECIMATION_FACTOR* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.13 TC_ACQ_CARRIER_NCO_RATES

Popis

Tento scénář kontroluje, zda Akviziční jednotka umožňuje konfiguraci hodnoty nosné frekvence pomocí *CARRIER_NCO_RATE_1-10* registrů.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *CARRIER_NCO_RATE_1-10* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude umožňovat konfiguraci hodnoty nosné frekvence pomocí *CARRIER_NCO_RATE_1-10* registrů.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.14 TC_ACQ_SINGLE_CARRIER

Popis

Tento scénář kontroluje, zda Akviziční jednotka generuje jednu nosnou frekvenci pomocí jediného NCO, které bude inkrementovat o hodnotu v registru *CARRIER_NCO_RATE_*1, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Acqusition Mode* bitu, *CARRIER_NCO_RATE_1-10* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude generovat jednu nosnou frekvenci pomocí jediného NCO, které bude inkrementovat o hodnotu v registru *CARRIER_NCO_RATE_*1, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.15 TC_ACQ_CARRIERS

Popis

Tento scénář kontroluje, zda Akviziční jednotka použije generuje 10 nosných frekvencí pomocí 10 NCO, které bude inkrementovat o hodnotu v registrech *CARRIER_NCO_RATE_1-10*, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Acqusition Mode* bitu, *CARRIER_NCO_RATE_1-10* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka použije bude generovat 10 nosných frekvencí pomocí 10 NCO, které bude inkrementovat o hodnotu v registrech *CARRIER_NCO_RATE_1-10*, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 0.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.16 TC_ACQ_COHERENT_INTEGRATION_TIME

Popis

Tento scénář kontroluje, zda Akviziční jednotka zpracuje v jednom integračním cyklu 1000*(X+1) vzorků, kde X je hodnota *COHERENT_INTEGRATION_TIME* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *COHERENT_INTEGRATION_TIME* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka zpracuje v jednom integračním cyklu 1000*(X+1) vzorků, kde X je hodnota *COHERENT_INTEGRATION_TIME* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.17 TC_ACQ_INTEGRATION_TIME

Popis

Tento scénář kontroluje, zda Akviziční jednotka provede X+1 integračních cyklů, kde X je hodnota *INTEGRATION_TIME* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *INTEGRATION_TIME* a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka provede X+1 integračních cyklů, kde X je hodnota *INTEGRATION_TIME* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.18 TC_ACQ_NONCOHERENT_INTEGRATION

Popis

Tento scénář kontroluje, zda Akviziční jednotka provede nekoherentní integraci, pokud je *Acqusition Type* bit *ACQUISITION_CONTROL* registru v logické 0, podle rovnice $S = \sum_{n=1}^{IntegrationTime} |R_n|^2$, kde R_n je výsledek akumulace po integračním cyklu [15].

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Acqusition Type* bitu a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka provede nekoherentní integraci, pokud je *Acqusition Type* bit *ACQUISITION_CONTROL* registru v logické 0, podle rovnice $S = \sum_{n=1}^{IntegrationTime} |R_n|^2$, kde R_n je výsledek akumulace po integračním cyklu.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.19 TC_ACQ_DIFFERENTIAL_INTEGRATION

Popis

Tento scénář kontroluje, zda Akviziční jednotka provede diferenciální integraci, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 1, podle

rovnice $S = \sum_{n=2}^{IntegrationTime} |R_n * R_{n-1}^*|^2$, kde R_n je výsledek akumulace po integračním cyklu [15].

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Acqusition Type* bitu a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka provede diferenciální integraci, pokud je *Acqusition Mode* bit *ACQUISITION_CONTROL* registru v logické 1, podle rovnice $S = \sum_{n=2}^{IntegrationTime} |R_n * R_{n-1}^*|^2$, kde R_n je výsledek akumulace po integračním cyklu.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.20 TC_ACQ_INPHASE_RESULTS

Popis

Tento scénář kontroluje, zda Akviziční jednotka uloží reálnou hodnotu *I*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_I* registrů, kde X je index skupiny 0-9.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky reálné části korelační amplitudy z *OUTPUT_1-10_AMPLITUDE_I* registrů.

Očekávané výsledky

Akviziční jednotka uloží reálnou hodnotu *I*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_I* registrů, kde X je index skupiny 0-9.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.21 TC_ACQ_QUADRATURE_RESULTS

Popis

Tento scénář kontroluje, zda Akviziční jednotka uloží imaginární hodnotu *Q*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_Q* registrů, kde X je index skupiny 0-9.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky reálné části korelační amplitudy z *OUTPUT_1-10_AMPLITUDE_Q* registrů.

Očekávané výsledky

Akviziční jednotka uloží imaginární hodnotu *Q*, která vygenerovala maximální korelační amplitudu, z každé skupiny korelátorů do *OUTPUT_X_AMPLITUDE_Q* registrů, kde X je index skupiny 0-9.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.22 TC_ACQ_CORRELATOR_MAX_AMPLITUDE

Popis

Tento scénář kontroluje, zda Akviziční jednotka uloží index korelátoru, který vygeneroval maximální korelační amplitudu, z každé skupiny do *OUTPUT_X_CORRELATOR_NUMBER* registrů, kde X je index skupiny 0-9.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky reálné části korelační amplitudy z *OUTPUT_1-10_CORRELATOR_NUMBER* registrů.

Očekávané výsledky

Akviziční jednotka uloží index korelátoru, který vygeneroval maximální korelační amplitudu, z každé skupiny do *OUTPUT_X_CORRELATOR_NUMBER* registrů, kde X je index skupiny 0-9.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.23 TC_TRACK_ENABLE

Popis

Tento scénář kontroluje, zda Sledovací jednotka zpracovává vstupní data, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Enable* bitu a ostatních registrů nastavující parametry akvizice.
3) Po dokončení akvizice budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude zpracovávat vstupní data, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.24 TC_TRACK_STATUS_IDLE

Popis

Tento scénář kontroluje, zda Sledovací jednotka je ve stavu *IDLE*, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 0.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Sledovací jednotka bude uvedena do různých stavů.
3) Po každé změně *Enable* bitu do logické 0 budou vyčteny *Status* bity v *TRACKING_CONTROL* registru.

Očekávané výsledky

Sledovací jednotka bude ve stavu *IDLE*, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 0.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Vstupní i výstupní hodnoty budou kontrolovány v testové proceduře.

A.25 TC_TRACK_STATUS_ARMED

Popis

Tento scénář kontroluje, zda Sledovací jednotka změní svůj stavu na *ARMED*, pokud *Enable* bit *TRACKING_CONTROL* registru změní hodnotu z logické 0 na logickou 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Sledovací jednotka bude uvedena do různých stavů.
3) Po každé změně *Enable* bitu z logické 0 na logickou 1 budou vyčteny *Status* bity v *TRACKING_CONTROL* registru.

Očekávané výsledky

Sledovací jednotka změní svůj stavu na *ARMED*, pokud *Enable* bit *TRACKING_CONTROL* registru změní hodnotu z logické 0 na logickou 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Vstupní i výstupní hodnoty budou kontrolovány v testové proceduře.

A.26 TC_TRACK_STATUS_TRACKING

Popis

Tento scénář kontroluje, zda Sledovací jednotka změní svůj stavu na *TRACKING*, pokud *Enable* bit *TRACKING_CONTROL* registru je v logické 1 a signál *SAMPLE_TIMESTAMP* se rovná hodnotě v registru *TRACKING_START_TIMESTAMP*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Sledovací jednotka bude uvedena do stavu *ARMED*.
3) Budou vyčteny *Status* bity v *TRACKING_CONTROL* registru po splnění každé z následujících podmínek:
    - *SAMPLE_TIMESTAMP* < *TRACKING_START_TIMESTAMP*,
    - *SAMPLE_TIMESTAMP* = *TRACKING_START_TIMESTAMP*,
    - *SAMPLE_TIMESTAMP* > *TRACKING_START_TIMESTAMP*.

Očekávané výsledky

Sledovací jednotka změní svůj stavu na *TRACKING*, pokud *Enable* bit *TRACKING_CONTROL* registru je v logické 1 a signál *SAMPLE_TIMESTAMP* se rovná hodnotě v registru *TRACKING_START_TIMESTAMP*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Vstupní i výstupní hodnoty budou kontrolovány v testové proceduře.

A.27 TC_TRACK_CHANNEL_SELECT

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude zpracovávat data z konkrétního vstupního datového kanálu podle hodnoty *Channel Select* bitů *TRACKING_CONTROL* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *Channel Select* bitů a ostatních registrů nastavující parametry integrace.
3) Do každého vstupního kanálu budou posílána jiná data.
4) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude zpracovávat data z konkrétního vstupního datového kanálu podle hodnoty *Channel Select* bitů *TRACKING_CONTROL* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.28 TC_TRACK_CODE_NCO_RATE

Popis

Tento scénář kontroluje, zda Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN PILOT a PRN DATA kódu pomocí *CODE_NCO_RATE* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *CODE_NCO_RATE* registru.

3) V závislosti na testové proceduře implementující tento testovací scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Do ostatních registrů nastavující parametry integrace budou zapsány náhodné hodnoty. Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Akviziční jednotka bude umožňovat konfiguraci rychlosti generace lokálního PRN PILOT a PRN DATA kódu pomocí *CODE_NCO_RATE* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.29 TC_TRACK_PRN_DATA_GENERATION

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN DATA kód z paměti pro PRN DATA kód Sledovací jednotky.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN DATA paměti Sledovací jednotky bude zapsán reálný PRN DATA kód vygenerovaný pomocí interního generátoru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace. Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude generovat lokální PRN DATA kód z paměti pro PRN DATA kód Sledovací jednotky.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.30 TC_TRACK_PRN_PILOT_GENERATION

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN PILOT kód z paměti pro PRN PILOT kód Sledovací jednotky.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN DATA paměti Sledovací jednotky bude zapsán reálný PRN DATA kód vygenerovaný pomocí interního generátoru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace. Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude generovat lokální PRN PILOT kód z paměti pro PRN PILOT kód Sledovací jednotky.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.31 TC_TRACK_PRN_DATA_LENGTH

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN DATA kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN DATA paměti Sledovací jednotky bude zapsán reálný PRN DATA kód vygenerovaný pomocí interního generátoru a jeho délka zmenšená o 1 bude zapsána do *TRACKING_PRN_LENGTH* registru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace. Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude generovat lokální PRN DATA kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.32 TC_TRACK_PRN_PILOT_LENGTH

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude generovat lokální PRN PILOT kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN PILOT paměti Sledovací jednotky bude zapsán reálný PRN PILOT kód vygenerovaný pomocí interního generátoru a jeho délka zmenšená o 1 bude zapsána do *TRACKING_PRN_LENGTH* registru.
3) V závislosti na testové proceduře implementující tento scénář bude proveden jeden z následujících bodů:
    a. Budou monitorovány přímo výstupy PRN generátoru.
    b. Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace. Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude generovat lokální PRN DATA kód o délce X+1 bitů, kde X je hodnota *TRACKING_PRN_LENGTH* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.33 TC_TRACK_PRN_DATA_WRITE

Popis

Tento scénář kontroluje, zda Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_DATA* uloží tyto data do paměti pro PRN DATA kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do registru *TRACKING_PRN_DATA* budou zapisována náhodná data.
3) Po každém zápisu bude vyčtena hodnota v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
4) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_DATA* uloží tyto data do paměti pro PRN DATA kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.34 TC_TRACK_PRN_DATA_READ

Popis

Tento scénář kontroluje, zda Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_DATA* vrátí hodnotu dat v paměti pro PRN DATA kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do paměti PRN DATA kódu Sledovací jednotky budou zapsána náhodná data.
3) Poté bude vyčítán *TRACKING_PRN_DATA* registr.
4) Po každém čtení bude vyčtena hodnota v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Očekávané výsledky

Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_DATA* vrátí hodnotu dat v paměti pro PRN DATA kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.35 TC_TRACK_PRN_PILOT_WRITE

Popis

Tento scénář kontroluje, zda Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_PILOT* uloží tyto data do paměti pro PRN PILOT kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do registru *TRACKING_PRN_PILOT* budou zapisována náhodná data.
3) Po každém zápisu bude vyčtena hodnota v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
4) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Sledovací jednotka při operaci zápisu dat do registru *TRACKING_PRN_PILOT* uloží tyto data do paměti pro PRN PILOT kód Sledovací jednotky na adresu podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.36 TC_TRACK_PRN_PILOT_READ

Popis

Tento scénář kontroluje, zda Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_PILOT* vrátí hodnotu dat v paměti pro PRN PILOT kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do paměti PRN PILOT kódu Sledovací jednotky budou zapsána náhodná data.
3) Poté bude vyčítán *TRACKING_PRN_PILOT* registr.
4) Po každém čtení bude vyčtena hodnota v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Očekávané výsledky

Sledovací jednotka při operaci čtení z registru *TRACKING_PRN_PILOT* vrátí hodnotu dat v paměti pro PRN PILOT kód Sledovací jednotky z adresy podle hodnoty v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.37 TC_TRACK_PRN_DATA_IGNORE_WRITE

Popis

Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci zápisu do PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *TRACKING_CONTROL* registru bude nastaven do logické 1.
3) Do registru *TRACKING_PRN_DATA* budou zapisována náhodná data.
4) Po každém zápisu bude vyčtena hodnota v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.
5) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Sledovací jednotka bude ignorovat operaci zápisu do PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.38 TC_TRACK_PRN_DATA_IGNORE_READ

Popis

Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci čtení z PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *TRACKING_CONTROL* registru bude nastaven do logické 1.
3) Do paměti PRN DATA kódu Sledovací jednotky budou zapsána náhodná data.
4) Poté bude vyčítán *TRACKING_PRN_DATA* registr.
5) Po každém čtení bude vyčtena hodnota v registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*.

Očekávané výsledky

Sledovací jednotka bude ignorovat operaci čtení z PRN DATA paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.39 TC_TRACK_PRN_PILOT_IGNORE_WRITE

Popis

Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci zápisu do PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *TRACKING_CONTROL* registru bude nastaven do logické 1.
3) Do registru *TRACKING_PRN_PILOT* budou zapisována náhodná data.
4) Po každém zápisu bude vyčtena hodnota v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
5) Následně budou zapsaná data vyčtena.

Očekávané výsledky

Sledovací jednotka bude ignorovat operaci zápisu do PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.40 TC_TRACK_PRN_PILOT_IGNORE_READ

Popis

Tento scénář kontroluje, zda Sledovací jednotka ignoruje operaci čtení z PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) *Enable* bit *TRACKING_CONTROL* registru bude nastaven do logické 1.
3) Do paměti PRN PILOT kódu Sledovací jednotky budou zapsána náhodná data.
4) Poté bude vyčítán *TRACKING_PRN_PILOT* registr.

5) Po každém čtení bude vyčtena hodnota v registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

Očekávané výsledky

Sledovací jednotka bude ignorovat operaci čtení z PRN PILOT paměti pro Sledovací jednotku, pokud je *Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne přímou testovou procedurou. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.41 TC_TRACK_PRN_DATA_BOC

Popis

Tento scénář kontroluje, zda Sledovací jednotka moduluje lokální PRN DATA kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *BOC Modulation Enable* bitu a ostatních registrů nastavující parametry integrace.
3) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude modulovat lokální PRN DATA kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.42 TC_TRACK_PRN_PILOT_BOC

Popis

Tento scénář kontroluje, zda Sledovací jednotka moduluje lokální PRN PILOT kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.

2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *BOC Modulation Enable* bitu a ostatních registrů nastavující parametry integrace.
3) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude modulovat lokální PRN PILOT kód pomocí BOC(1,1) modulace, pokud je *BOC Modulation Enable* bit *TRACKING_CONTROL* registru v logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Hodnoty přímo z PRN generátoru budou kontrovány přímo v testové proceduře. Výsledky korelační amplitudy budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.43 TC_TRACK_PRN_DATA_INTEGRATION_LENGTH

Popis

Tento scénář kontroluje, zda Sledovací jednotka integruje X+1 bitů PRN DATA kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN DATA paměti Sledovací jednotky bude zapsán reálný PRN DATA kód vygenerovaný pomocí interního generátoru.
3) Pro PRN DATA kód signálu Galileo E1 bude do *PRN_INTEGRATION_LENGTH* registru zapsána hodnota 1022, pro ostatní signály bude zapsána hodnota délky PRN DATA kódu zmenšená o 1.
4) Dále budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
5) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude integrovat X+1 bitů PRN DATA kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.44 TC_TRACK_PRN_PILOT_INTEGRATION_LENGTH

Popis

Tento scénář kontroluje, zda Sledovací jednotka integruje X+1 bitů PRN PILOT kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Do PRN PILOT paměti Sledovací jednotky bude zapsán reálný PRN PILOT kód vygenerovaný pomocí interního generátoru.
3) Pro PRN PILOT kód signálu Galileo E1 bude do *PRN_INTEGRATION_LENGTH* registru zapsána hodnota 1022, pro ostatní signály bude zapsána hodnota délky PRN PILOT kódu zmenšená o 1.
4) Dále budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
5) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude integrovat X+1 bitů PRN PILOT kódu, kde X je hodnota v *PRN_INTEGRATION_LENGTH* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.45 TC_TRACK_CODE_PHASE_INCREMENTS

Popis

Tento scénář kontroluje, zda Sledovací jednotka umožňuje po dokončení integrace posun fáze NCO pro PRN kódy o hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení první integrace a každé následující budou vygenerovány a zapsány náhodné hodnoty do *CODE_NCO_PHASE_INCREMENT* registru.
4) Po dokončení každé integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude umožňovat po dokončení integrace posun fáze NCO pro PRN kódy o hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.46 TC_TRACK_CODE_PHASE_INCREMENTS_CLEAR

Popis

Tento scénář kontroluje, zda Sledovací jednotka po dokončení integrace vymaže hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení každé integrace před zapsáním nových hodnot bude vyčten *CODE_NCO_PHASE_INCREMENT* registr.

Očekávané výsledky

Sledovací jednotka po dokončení integrace vymaže hodnotu v *CODE_NCO_PHASE_INCREMENT* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.47 TC_TRACK_PRN_ADDRESS_INCREMENTS

Popis

Tento scénář kontroluje, zda Sledovací jednotka umožňuje po dokončení integrace posun v PRN DATA a PRN PILOT kódech o hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení první integrace a každé následující budou vygenerovány a zapsány náhodné hodnoty do *PRIMARY_CODE_MEMORY_INCREMENT* registru.
4) Po dokončení každé integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude umožňovat po dokončení integrace posun v PRN DATA a PRN PILOT kódech o hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.48 TC_TRACK_PRN_ADDRESS_INCREMENTS_CLEAR

Popis

Tento scénář kontroluje, zda Sledovací jednotka po dokončení integrace vymaže hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení každé integrace před zapsáním nových hodnot bude vyčten *PRIMARY_CODE_MEMORY_INCREMENT* registr.

Očekávané výsledky

Sledovací jednotka po dokončení integrace vymaže hodnotu v *PRIMARY_CODE_MEMORY_INCREMENT* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.49 TC_TRACK_CARRIER_NCO_RATE

Popis

Tento scénář kontroluje, zda Sledovací jednotka generuje jednu nosnou frekvenci pomocí NCO, které bude inkrementovat o hodnotu v registru *TRACKING_CARRIER_NCO_RATE*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *TRACKING_CARRIER_NCO_RATE* a ostatních registrů nastavující parametry integrace.
3) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka bude generovat jednu nosnou frekvenci pomocí NCO, které bude inkrementovat o hodnotu v registru *TRACKING_CARRIER_NCO_RATE*.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.50 TC_TRACK_CORRELATOR_DELAY

Popis

Tento scénář kontroluje, zda Sledovací jednotka umožňuje konfiguraci zpoždění každého z korelátorů o X+1 vzorků, kde X je hodnota *CORRELATOR_DELAY_1-6* registrů.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty *CORRELATOR_DELAY_1-6* a ostatních registrů nastavující parametry integrace.
3) Po dokončení integrace budou vyčteny výsledky korelační amplitudy.

Očekávané výsledky

Sledovací jednotka umožní konfiguraci zpoždění každého z korelátorů o X+1 vzorků, kde X je hodnota *CORRELATOR_DELAY_1-6* registrů.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.51 TC_TRACK_INTEGRATION_COUNT

Popis

Tento scénář kontroluje, zda Sledovací jednotka ukládá počet dokončených integrací od začátku procesu sledování do *INTEGRATION_RESULT_COUNT* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace bude vyčten *INTEGRATION_RESULT_COUNT* registr.

Očekávané výsledky

Sledovací jednotka bude ukládat počet dokončených integrací od začátku procesu sledování do *INTEGRATION_RESULT_COUNT* registru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.52 TC_TRACK_CORRELATION_RESULTS_PILOT_I

Popis

Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_I* registru, kde X je číslo korelátoru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace budou vyčteny *CORRELATION_RESULT_PILOT_1-6_I* registry.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek reálné části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_I* registru, kde X je číslo korelátoru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.53 TC_TRACK_CORRELATION_RESULTS_PILOT_Q

Popis

Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek imaginární části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_Q* registru, kde X je číslo korelátoru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.

3) Po dokončení integrace budou vyčteny *CORRELATION_RESULT_PILOT_1-6_Q* registry.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT korelátorů 1-6 do příslušného *CORRELATION_RESULT_PILOT_X_Q* registru, kde X je číslo korelátoru.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.54 TC_TRACK_CORRELATION_RESULTS_PILOT_PROMPT_I

Popis

Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z PILOT PROMPT korelátoru do *CORRELATION_RESULT_PILOT_PROMPT_I* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace bude vyčten *CORRELATION_RESULT_PILOT_PROMPT_I* registr.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek reálné části korelace z PILOT PROMPT korelátoru do *CORRELATION_RESULT_PILOT_PROMPT_I* registru**.**

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.55 TC_TRACK_CORRELATION_RESULTS_PILOT_PROMPT_Q

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT PROMPT korelátoru do registru *CORRELATION_RESULT_PILOT_PROMPT_Q*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.

2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace bude vyčten *CORRELATION_RESULT_PILOT_PROMPT_Q* registr.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek imaginární části korelace z PILOT PROMPT korelátoru do registru *CORRELATION_RESULT_PILOT_PROMPT_Q*.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.56 TC_TRACK_CORRELATION_RESULTS_DATA_PROMPT_I

Popis

Tento scénář kontroluje, zda Sledovací jednotka ukládá výsledek reálné části korelace z DATA PROMPT korelátoru do *CORRELATION_RESULT_DATA_PROMPT_I* registru.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace bude vyčten *CORRELATION_RESULT_DATA_PROMPT_I* registr.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek reálné části korelace z DATA PROMPT korelátoru do *CORRELATION_RESULT_DATA_PROMPT_I* registru**.**

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru

A.57 TC_TRACK_CORRELATION_RESULTS_DATA_PROMPT_Q

Popis

Tento scénář kontroluje, zda Sledovací jednotka bude ukládat výsledek imaginární části korelace z DATA PROMPT korelátoru do registru *CORRELATION_RESULT_DATA_PROMPT_Q*.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po dokončení integrace bude vyčten *CORRELATION_RESULT_DATA_PROMPT_Q* registr.

Očekávané výsledky

Sledovací jednotka bude ukládat výsledek imaginární části korelace z DATA PROMPT korelátoru do registru *CORRELATION_RESULT_DATA_PROMPT_Q*.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.58 TC_TRACK_PVT_INTEGRATION_COUNT

Popis

Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální počet integrací od začátku sledovacího procesu do registru *PVT_INTEGRATION_COUNT,* pokud signál *PVT_LATCH* přejde do logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po každém přechodu signálu *PVT_LATCH* do logické 1 bude vyčten *PVT_INTEGRATION_COUNT* registr.

Očekávané výsledky

Sledovací jednotka uloží aktuální počet integrací od začátku sledovacího procesu do registru *PVT_INTEGRATION_COUNT,* pokud signál *PVT_LATCH* přejde do logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.59 TC_TRACK_PVT_PRN_PRIMARY_ADDRESS

Popis

Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu ukazatele na bit PRN kódu do registru *PVT_PRIMARY_ADDRESS,* pokud signál *PVT_LATCH* přejde do logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po každém přechodu signálu *PVT_LATCH* do logické 1 bude vyčten *PVT_PRIMARY_ADDRESS* registr.

Očekávané výsledky

Sledovací jednotka uloží aktuální hodnotu ukazatele na bit PRN kódu do registru *PVT_PRIMARY_ADDRESS*, pokud signál *PVT_LATCH* přejde do logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.60 TC_TRACK_PVT_CODE_PHASE

Popis

Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu fáze PRN kódu do registru *PVT_CODE_PHASE,* pokud signál *PVT_LATCH* přejde do logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po každém přechodu signálu *PVT_LATCH* do logické 1 bude vyčten *PVT_CODE_PHASE* registr.

Očekávané výsledky

Sledovací jednotka uloží aktuální hodnotu fáze PRN kódu do registru *PVT_CODE_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.61 TC_TRACK_PVT_CARRIER_CYCLES

Popis

Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální počet přetečení NCO nosné frekvence do registru *PVT_CARRIER_CYCLES,* pokud signál *PVT_LATCH* přejde do logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po každém přechodu signálu *PVT_LATCH* do logické 1 bude vyčten *PVT_CARRIER_CYCLES* registr.

Očekávané výsledky

Sledovací jednotka uloží aktuální počet přetečení NCO nosné frekvence do registru *PVT_CARRIER_CYCLES,* pokud signál *PVT_LATCH* přejde do logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

A.62 TC_TRACK_PVT_CARRIER_PHASE

Popis

Tento scénář kontroluje, zda Sledovací jednotka uloží aktuální hodnotu fáze nosné frekvence do registru *PVT_CARRIER_PHASE,* pokud signál *PVT_LATCH* přejde do logické 1.

Průběh

1) Microcore GNSS Baseband obvod bude uveden do stavu, kdy bude schopen přijímat příkazy z procesoru.
2) Budou vygenerovány a do obvodu zapsány náhodné hodnoty registrů nastavující parametry integrace.
3) Po každém přechodu signálu *PVT_LATCH* do logické 1 bude vyčten *PVT_CARRIER_PHASE* registr.

Očekávané výsledky

Sledovací jednotka uloží aktuální hodnotu fáze nosné frekvence do registru *PVT_CARRIER_PHASE*, pokud signál *PVT_LATCH* přejde do logické 1.

Generace vstupů, kontrola výstupů

Ověření proběhne testovou procedurou s náhodnými čísly. Generace vstupních hodnot bude kontrolována funkčním pokrytím. Výsledné hodnoty budou kontrolovány Výsledkovou tabulkou, kde se hodnoty porovnají s hodnotami z Prediktoru.

B Testové procedury

Tato kapitola obsahuje popis testových procedur a v nich obsažených skupin funkčního pokrytí. Většina testových procedur přistupuje k textovanému obvodu stylem „black box“, tedy pouze k jeho vnějším signálům. Výjimkou jsou pouze testové procedury *tp_acq_code_gen*, *tp_track_code_gen*, *tp_track_code_gen_no_boc* a *tp_track_code_gen_no_incr*, které monitorují výstupní signály generátoru PRN kódu.

B.1 Tp_acq_status

Tato testová procedura ověřuje správnou indikaci stavu Akviziční jednotky. Jedná se o přímý test.

Testová procedura si ve funkci *build_phase()* načte z konfigurační databáze hodnoty periody hodinových signálů pro obě rozhraní, zpoždění odeslání datové transakce a zpoždění pro povolení akvizičního procesu potřebné pro stejné synchronizování uvnitř Akviziční jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Dále tato testová procedura obsahuje funkci *compare()*, která přijímá dvě 16bitové hodnoty jako parametry, očekávaný a vyčtený stav, tyto hodnoty porovná a vypíše výsledek porovnání. Tato funkce je volaná při každé kontrole *ACQUISITION_STATUS* registru.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Zkontrolování hodnoty *ACQUISITION_STATUS* registru.
2) Zapsání náhodného PRN kódu.
3) Zapsání definovaných hodnoty registrů potřebné pro akviziční proces.
4) Povolení akvizičního procesu (*Enable* bit do logické 1) a vyčkání po dobu 10 mikrosekund.
5) Zkontrolování hodnoty *ACQUISITION_STATUS* registru a vyčkání po dobu 20 mikrosekund.
6) Přerušení akvizičního procesu (*Enable* bit do logické 0) a vyčkání po dobu 1 mikrosekunda.
7) Zkontrolování hodnoty *ACQUISITION_STATUS* registru.
8) *Fork....join_none* blok:
    a. povolení akvizičního procesu,
    b. zápis transakce se vstupními daty přes Datové rozhraní.
9) Výpočet počtu datových vzorků v transakci.
10) *Fork....join* blok:
    a. zkontrolování hodnoty *ACQUISITION_STATUS* registru po odeslání jedné třetiny, respektive dvou třetin transakce,
    b. vyčkání přibližně po dobu odesílání transakce.
11) Vyčkání na zpracování výsledků z jednotlivých korelátorů.
12) Zkontrolování hodnoty *ACQUISITION_STATUS* registru.
13) Body 8-12 jsou znovu opakovány.

Zdrojový kód této testové procedury je v souboru *tp_acq_status.sv*.

B.2 Tp_acq_prn_memory

Tato testová procedura ověřuje přístupy do PRN paměti Akviziční jednotky. Obsahuje funkce *start_of_simulation_phase()*, *report_phase()*, procedury *process_transaction()*, *run_test()* a skupinu funkčního pokrytí *acq_prn_memory_cg*.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* na po ukončení testové procedury vypíše úroveň funkčního pokrytí v *acq_prn_memory_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *process_transaction()* je volána z procedury *run_test()* při každé operaci zápisu nebo čtení. Obsahuje 4 vstupní parametry:

- typ transakce,
- adresu,
- délku transakce,
- data pro transakci zápisu.

Po zavolání je dekódován typ transakce a zavolána příslušná funkce v Procesorovém Agentovi. Funkci pro zápis předá adresu a pole dat k zapsání. Funkci pro čtení předá adresu, délku transakce a lokální proměnou, do které budou uložena vyčtená data. Na konci procedury vzorkuje *acq_prn_memory_cg*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Operace zápisu do PRN paměti o délce 1 (sekvenční přístup do celé paměti):
   a. Zápis do registru *ACQUISITION_PRN_DATA*,
   b. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
2) Vynulování registru *ACQUISITION_PRN_DATA_OFFSET*.
3) Operace čtení z PRN paměti o délce 1 (sekvenční přístup do celé paměti):
   a. Čtení z registru *ACQUISITION_PRN_DATA*,
   b. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
4) Operace zápisu do PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registru *ACQUISITION_PRN_DATA_OFFSET*,
   b. Zápis náhodných hodnot do registru *ACQUISITION_PRN_DATA*,
   c. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
5) Operace čtení z PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registru *ACQUISITION_PRN_DATA_OFFSET*,
   b. Čtení náhodné délky z registru *ACQUISITION_PRN_DATA*,
   c. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
6) Zápis do registrů *COHERENT_INTEGRATION_TIME*, *INTEGRATION_TIME*, *CODE_NCO_RATE* a *ACQUISITION_PRN_LENGTH*.
7) Povolení akvizičního procesu (*Enable* bit do logické 1).
8) Opakování bodů 1-5.

Zdrojový kód této testové procedury je v souboru *tp_acq_prn_memory.sv*.

B.2.1 Acq_prn_mem_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- typ transakce,
- adresa,
- *Enable* bit,
- délka transakce,
- kombinace výše zmíněných.

B.3 Tp_acq_prn_memory_no_single_read

Tato testová procedura ověřuje přístupy do PRN paměti Akviziční jednotky. Obsahuje funkce *start_of_simulation_phase()*, *report_phase()*, procedury *process_transaction()*, *run_test()* a skupinu funkčního pokrytí *acq_prn_memory_cg*.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* na po ukončení testové procedury vypíše úroveň funkčního pokrytí v *acq_prn_memory_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *process_transaction()* je volána z procedury *run_test()* při každé operaci zápisu nebo čtení. Obsahuje 4 vstupní parametry:

- typ transakce,
- adresu,
- délku transakce,
- data pro transakci zápisu.

Po zavolání je dekódován typ transakce a zavolána příslušná funkce v Procesorovém Agentovi. Funkci pro zápis předá adresu a pole dat k zapsání. Funkci pro čtení předá adresu, délku transakce a lokální proměnou, do které budou uložena vyčtená data. Na konci procedury vzorkuje *acq_prn_memory_cg*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Operace zápisu do PRN paměti o délce 1 (sekvenční přístup do celé paměti):
   a. Zápis do registru *ACQUISITION_PRN_DATA*,
   b. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
2) Vynulování registru *ACQUISITION_PRN_DATA_OFFSET*.
3) Operace zápisu do PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registru *ACQUISITION_PRN_DATA_OFFSET*,
   b. Zápis náhodných hodnot do registru *ACQUISITION_PRN_DATA*,
   c. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.
4) Operace čtení z PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registru *ACQUISITION_PRN_DATA_OFFSET*,

b. Čtení náhodné délky z registru *ACQUISITION_PRN_DATA*,
c. Čtení z registru *ACQUISITION_PRN_DATA_OFFSET*.

5) Zápis do registrů *COHERENT_INTEGRATION_TIME, INTEGRATION_TIME, CODE_NCO_RATE* a *ACQUISITION_PRN_LENGTH.*
6) Povolení akvizičního procesu (*Enable* bit do logické 1).
7) Opakování bodů 1-4.

Zdrojový kód této testové procedury je v souboru *tp_acq_prn_memory_no_single_read.sv*.

B.3.1 Acq_prn_mem_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- typ transakce,
- adresa,
- *Enable* bit,
- délka transakce,
- kombinace výše zmíněných.

B.4 Tp_acq_code_gen

Tato testová procedura ověřuje generování PRN kódu. Monitoruje výstupy generátoru PRN kódu v Akviziční jednotce přes rozhraní připojené do obvodu pomocí *bind* příkazu. Obsahuje funkce *build_phase(), start_of_simulation_phase(), report_phase(),* procedury *generate_prn_chipstream(), monitor_prn_ifc(), run_test()* a skupinu funkčního pokrytí *acq_prn_code_cg*.

Ve funkci *build_phase()* je získáno PRN rozhraní a hodnota periody hodinového signálu v Procesorovém rozhraní. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *acq_prn_code_cg*. Tato funkce je volána v UVM *report* fázi.

Očekávaná sekvence z PRN generátoru je vygenerována v proceduře *generate_prn_chipstream()*. Procedura má jako vstupní parametr délku očekávané sekvence. Vygenerovaná sekvence je výstupním parametrem.

Monitorování výstupu PRN generátoru a jeho porovnávání s očekávanou sekvencí probíhá v proceduře *monitor_prn_ifc()*. Vstupními parametry procedury jsou očekávaná sekvence hodnot a délka sekvence. Po zkontrolování celé sekvence vrátí procedura počet správných a počet chybných porovnání.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Vygenerování a vypsání náhodných hodnot *BOC_enable* bitu, rychlosti generace PRN kódu (hodnota pro *CODE_NCO_RATE* registr), délky PRN kódu a PRN kód. PRN kód není vypisován.

2) Vypočtení délky očekávané sekvence, aby bylo zkontrolováno alespoň 5 úplných cyklů PRN kódu.
3) Vygenerování očekávané sekvence výstupu PRN generátoru.
4) Zapsání hodnot do registrů *ACQ_PRN_LENGTH, CODE_NCO_RATE, COHERENT_INTEGRATION_TIME, INTEGRATION_TIME* a zapsání PRN kódu do PRN paměti Akviziční jednotky.
5) *fork....join* blok:
    a. povolení akvizičního procesu,
    b. zavolání procedury *monitor_prn_ifc()*.
6) Vzorkování *acq_prn_code_cg*.
7) Ukončení akvizičního procesu.
8) Vynulování registru *ACQUISITION_PRN_DATA_OFFSET*.
9) Vypsání výsledků.
10) Body 1-9 jsou opakovány podle hodnoty proměnné *runs* v proceduře.

Zdrojový kód této testové procedury je v souboru *tp_acq_code_gen.sv*.

B.4.1 Acq_prn_code_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- *BOC enable* bit,
- rychlost generace PRN kódu,
- délka PRN kódu.

B.5 Tp_acq_engine

Tato testová procedura se zaměřuje na celý akviziční proces. Ověřuje, zda pro různé hodnoty registrů ovlivňující akviziční proces jsou vypočteny správné výsledky akvizičního procesu. Jsou generovány kombinace vstupních hodnot pro nalezení reálných satelitních signálů. Obsahuje funkce *build_phase(), report_test_param(), report_phase()*, procedury *read_input_prn_file(), send_prn_code_length(), send_prn_code(), set_registers(), run_test()* a skupinu funkčního pokrytí *acq_engine_cg*.

Ve funkci *build_phase()* jsou z konfigurační databáze získány hodnoty zpoždění odeslání datové transakce a zpoždění pro povolení akvizičního procesu potřebné pro stejné synchronizování uvnitř Akviziční jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Funkce *report_test_param()* vypíše parametry akvizičního procesu.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *acq_engine_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *read_input_prn_file()* vyčte PRN kód ze souboru a zformátuje kód do 16bitových slov. Procedura má 2 výstupní parametry: PRN kód a jeho délku.

Procedura *send_prn_code_length()* zapíše délku PRN kódu do registru *ACQ_PRN_LENGTH*.

Procedura *send_prn_code()* zapíše PRN kódu do paměti PRN kódu Akviziční jednotky.

Procedura *set_registers()* zapíše náhodné do registrů *CODE_NCO_RATE, CODE_NCO_RATE_1-10, COHERENT_INTEGRATION_TIME, INTEGRATION_TIME* a *DECIMATION_FACTOR.*

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Generace náhodných hodnot.
2) Načtení PRN kódu ze vstupního souboru.
3) Vypsání parametrů akvizice.
4) Zapsání délky PRN kódu do registru *ACQ_PRN_LENGTH*.
5) Zapsání PRN kódu do paměti PRN kódu Akviziční jednotky.
6) Zapsání náhodných hodnot do registrů *CODE_NCO_RATE, CODE_NCO_RATE_1-10, COHERENT_INTEGRATION_TIME, INTEGRATION_TIME* a *DECIMATION FACTOR.*
7) *fork....join* blok:
    a. zápis do registru *ACQUISITION_CONTROL* a povolení akvizičního procesu,
    b. odeslání dat do vstupního kanálu 1,
    c. odeslání dat do vstupního kanálu 2.
8) Vyčkání na dokončení integrace.
9) Vyčtení integračních výsledků z registrů *OUTPUT_0-9_AMPLITUDE_I, OUTPUT_0-9_AMPLITUDE_Q, OUTPUT_0-9_CORRELATOR_NUMBER.*
10) Vzorkování *acq_engine_cg*.

Zdrojový kód této testové procedury je v souboru *tp_acq_engine.sv*.

B.5.1 Acq_engine_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- *Enable* bit,
- *BOC enable* bit,
- rychlost generace PRN kódu,
- *Channel select* bity,
- délka PRN kódu,
- *Acquisition mode* bit,
- *Acquisition type* bit,
- hodnotu *Decimation factor*,
- kombinace *BOC enable, Acquisition mode* a *Acquisition type*.

B.6 Tp_acq_engine_directed

Tato testová procedura byla přidána k pokrytí kombinací vstupních hodnot, které nejsou vygenerovány v proceduře *tp_acq_engine*. Ověřuje, zda pro různé hodnoty registrů ovlivňující akviziční proces jsou vypočteny správné výsledky akvizičního procesu. Jsou generovány kombinace vstupních hodnot pro nalezení reálných satelitních signálů. Obsahuje funkce *build_phase(), report_test_param(), report_phase(),* procedury *read_input_prn_file(), send_prn_code_length(), send_prn_code(), set_registers(), run_test()* a skupinu funkčního pokrytí *acq_engine_cg*.

Ve funkci *build_phase()* jsou z konfigurační databáze získány hodnoty zpoždění odeslání datové transakce a zpoždění pro povolení akvizičního procesu potřebné pro stejné synchronizování uvnitř Akviziční jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Funkce *report_test_param()* vypíše parametry akvizičního procesu.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *acq_engine_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *read_input_prn_file()* vyčte PRN kód ze souboru a zformátuje kód do 16bitových slov. Procedura má 2 výstupní parametry: PRN kód a jeho délku.

Procedura *send_prn_code_length()* zapíše délku PRN kódu do registru *ACQ_PRN_LENGTH.*

Procedura *send_prn_code()* zapíše PRN kódu do paměti PRN kódu Akviziční jednotky.

Procedura *set_registers()* zapíše náhodné do registrů *CODE_NCO_RATE, CODE_NCO_RATE_1-10, COHERENT_INTEGRATION_TIME, INTEGRATION_TIME* a *DECIMATION_FACTOR.*

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Generace náhodných hodnot. Pro *BOC Modulation Enable* bit je generována hodnota 0, pro *Acquisition Mode* bit hodnota 1 a pro *Acquisition Type* hodnota 0.
2) Načtení PRN kódu ze vstupního souboru.
3) Vypsání parametrů akvizice.
4) Zapsání délky PRN kódu do registru *ACQ_PRN_LENGTH*.
5) Zapsání PRN kódu do paměti PRN kódu Akviziční jednotky.
6) Zapsání náhodných hodnot do registrů *CODE_NCO_RATE, CODE_NCO_RATE_1-10, COHERENT_INTEGRATION_TIME, INTEGRATION_TIME* a *DECIMATION FACTOR.*
7) *fork...join* blok:
    a. zápis do registru *ACQUISITION_CONTROL* a povolení akvizičního procesu,
    b. odeslání dat do vstupního kanálu 1,
    c. odeslání dat do vstupního kanálu 2.
8) Vyčkání na dokončení integrace.
9) Vyčtení integračních výsledků z registrů *OUTPUT_0-9_AMPLITUDE_I, OUTPUT_0-9_AMPLITUDE_Q, OUTPUT_0-9_CORRELATOR_NUMBER.*
10) Vzorkování *acq_engine_cg*.

Zdrojový kód této testové procedury je v souboru *tp_acq_engine_directed.sv*.

B.6.1 Acq_engine_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- *Enable* bit,
- *BOC enable* bit,

- rychlost generace PRN kódu,
- *Channel select* bity,
- délka PRN kódu,
- *Acquisition mode* bit,
- *Acquisition type* bit,
- hodnotu *Decimation factor*,
- kombinace *BOC enable, Acquisition mode* a *Acquisition type*.

B.7 Tp_track_status

Tato testová procedura ověřuje správnou indikaci stavu Sledovací jednotky. Jedná se o přímý test.

Dále tato testová procedura obsahuje funkci *compare()*, která přijímá dvě 2bitové hodnoty jako parametry, očekávaný a vyčtený stav, tyto hodnoty porovná a vypíše výsledek porovnání. Tato funkce je volaná po každém čtení *TRACKING_CONTROL* registru.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
2) Zapsání náhodné hodnoty do *TRACKING_START_TIMESTAMP* registru.
3) Povolení sledovacího procesu (*Enable* bit do logické 1).
4) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
5) Přerušení sledovacího procesu (*Enable* bit do logické 0).
6) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
7) Povolení sledovacího procesu (*Enable* bit do logické 1).
8) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
9) Povolení *Timestamp* čítače v Časovací jednotce.
10) Periodické vyčítání hodnoty *Timestamp* čítače v Časovací jednotce, dokud není splněna podmínka
    SAMPLE_TIMESTAMP => TRACKING_START_TIMESTAMP.
11) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru a vyčkání po dobu 10 mikrosekund.
12) Přerušení sledovacího procesu (*Enable* bit do logické 0).
13) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru a vyčkání po dobu 10 mikrosekund.
14) Povolení sledovacího procesu (*Enable* bit do logické 1) a vyčkání po dobu 10 mikrosekund.
15) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
16) Přerušení sledovacího procesu (*Enable* bit do logické 0).
17) Zkontrolování hodnoty *Status* bitů v *TRACKING_CONTROL* registru.
18) Vypnutí *Timestamp* čítače v Časovací jednotce.
19) Body 1-18 jsou opakovány pro každý kanál.

Zdrojový kód této testové procedury je v souboru *tp_track_status.sv*.

B.8 Tp_track_prn_memory

Tato testová procedura ověřuje přístupy do PRN DATA a PRN PILOT paměti Sledovací jednotky. Z důvodu délky simulace je kontrolován pouze 1 sledovací kanál. Obsahuje funkce *start_of_simulation_phase(), report_phase(),* procedury *process_transaction(), run_test()* a skupinu funkčního pokrytí *track_prn_memory_cg*.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* na po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_prn_memory_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *process_transaction()* je volána z procedury *run_test()* při každé operaci zápisu nebo čtení. Obsahuje 4 vstupní parametry:

- typ transakce,
- adresu,
- délku transakce,
- data pro transakci zápisu.

Po zavolání je dekódován typ transakce a zavolána příslušná funkce v Procesorovém Agentovi. Funkci pro zápis předá adresu a pole dat k zapsání. Funkci pro čtení předá adresu, délku transakce a lokální proměnou, do které budou uložena vyčtená data. Na konci procedury vzorkuje *track_prn_memory_cg*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Operace zápisu do PRN pamětí o délce 1 (sekvenční přístup do celé paměti):
    c. Zápis do registru *TRACKING_PRN_DATA*,
    d. Čtení z registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*,
    e. Zápis do registru *TRACKING_PRN_PILOT*,
    f. Čtení z registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
2) Vynulování registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
3) Operace čtení z PRN paměti o délce 1 (sekvenční přístup do celé paměti):
    a. Čtení z registru *TRACKING_PRN_DATA*,
    b. Čtení z registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*,
    c. Čtení z registru *TRACKING_PRN_PILOT*,
    d. Čtení z registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
4) Operace zápisu do PRN paměti o délce větší než 1 (opakováno 100x):
    a. Zápis náhodné hodnoty do registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*,
    b. Zápis náhodných hodnot do registrů *TRACKING_PRN_DATA* a *TRACKING_PRN_PILOT*,
    c. Čtení z registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
5) Operace čtení z PRN paměti o délce větší než 1 (opakováno 100x):

a. Zápis náhodné hodnoty do registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*,
b. Čtení náhodné délky z registrů *TRACKING_PRN_DATA* a *TRACKING_PRN_PILOT*,
c. Čtení z registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

6) Povolení sledovacího procesu (*Enable* bit do logické 1).
7) Opakování bodů 1-5.

Zdrojový kód této testové procedury je v souboru *tp_track_prn_memory.sv*.

B.8.1 Track_prn_mem_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- typ transakce,
- adresa,
- *Enable* bit,
- délka transakce,
- kombinace výše zmíněných.

B.9 Tp_track_prn_memory_no_single_read

Tato testová procedura ověřuje přístupy do PRN DATA a PRN PILOT paměti Sledovací jednotky. Z důvodu délky simulace je kontrolován pouze 1 sledovací kanál. Obsahuje funkce *start_of_simulation_phase()*, *report_phase()*, procedury *process_transaction()*, *run_test()* a skupinu funkčního pokrytí *track_prn_memory_cg*.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* na po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_prn_memory_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *process_transaction()* je volána z procedury *run_test()* při každé operaci zápisu nebo čtení. Obsahuje 4 vstupní parametry:

- typ transakce,
- adresu,
- délku transakce,
- data pro transakci zápisu.

Po zavolání je dekódován typ transakce a zavolána příslušná funkce v Procesorovém Agentovi. Funkci pro zápis předá adresu a pole dat k zapsání. Funkci pro čtení předá adresu, délku transakce a lokální proměnou, do které budou uložena vyčtená data. Na konci procedury vzorkuje *track_prn_memory_cg*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Operace zápisu do PRN pamětí o délce 1 (sekvenční přístup do celé paměti):

a. Zápis do registru *TRACKING_PRN_DATA*,
b. Čtení z registru *TRACKING_PRN_DATA_ADDRESS_OFFSET*,
c. Zápis do registru *TRACKING_PRN_PILOT*,
d. Čtení z registru *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.

2) Vynulování registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
3) Operace zápisu do PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*,
   b. Zápis náhodných hodnot do registrů *TRACKING_PRN_DATA* a *TRACKING_PRN_PILOT*,
   c. Čtení z registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
4) Operace čtení z PRN paměti o délce větší než 1 (opakováno 100x):
   a. Zápis náhodné hodnoty do registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*,
   b. Čtení náhodné délky z registrů *TRACKING_PRN_DATA* a *TRACKING_PRN_PILOT*,
   c. Čtení z registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
5) Povolení sledovacího procesu (*Enable* bit do logické 1).
6) Opakování bodů 1-4.

Zdrojový kód této testové procedury je v souboru *tp_track_prn_memory_no_single_read.sv*.

B.9.1 Track_prn_mem_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- typ transakce,
- adresa,
- *Enable* bit,
- délka transakce,
- kombinace výše zmíněných.

B.10 Tp_track_code_gen

Tato testová procedura ověřuje generování PRN DATA a PRN PILOT kódů. Monitoruje výstupy generátoru PRN kódu ve Sledovací jednotce přes rozhraní připojené do obvodu pomocí *bind* příkazu. Obsahuje funkce *build_phase()*, *start_of_simulation_phase()*, *report_phase()*, procedury *run_test()* a skupinu funkčního pokrytí *track_prn_code_cg*.

Ve funkci *build_phase()* je získáno PRN rozhraní a hodnota periody hodinového signálu v Procesorovém rozhraní. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_prn_code_cg*. Tato funkce je volána v UVM *report* fázi.

Generace očekávaných PRN DATA a PRN PILOT kódů, monitorování výstupů PRN generátoru a porovnání s očekávanými hodnotami probíhá v testové proceduře *run_test()*. Průběh této procedury je následující:

1) Vygenerování a vypsání náhodných hodnot *BOC_enable* bitu, rychlosti generace PRN kódu (hodnota pro *CODE_NCO_RATE* registr), délky PRN kódu a PRN DATA a PRN PILOT kódy. PRN kódy nejsou vypisovány.
2) Vypočtení délky očekávané sekvence, aby bylo zkontrolováno alespoň 5 úplných cyklů PRN kódů.
3) Zapsání hodnot do registrů *TRACKING_PRN_LENGTH, PRN_INTEGRATION_LENGTH, CODE_NCO_RATE, TRACKING_START_TIMESTAMP* a zapsání PRN kódů do PRN pamětí Sledovací jednotky.
4) Povolení sledovacího procesu (*Enable* bit do logické 1).
5) *fork....join* blok:
    a. povolení *Timestamp* čítače v Časovací jednotce,
    b. *while* smyčka periodicky generující a zapisující náhodné hodnoty korekcí do *CODE_NCO_PHASE_INCREMENT* a *PRIMARY_CODE_MEMORY_INCREMENT* registrů, pokud je monitorování aktivní,
    c. while smyčka monitorující výstupy PRN generátoru:
        i. generace očekávaných hodnot na výstupech PRN generátoru,
        ii. sejmutí hodnot na výstupů PRN generátoru,
        iii. porovnání sejmutých a očekávaných hodnot,
        iv. inkrementace čítače správných porovnání nebo chybných porovnání a výpis výsledku porovnání,
6) Vzorkování *track_prn_code_cg*.
7) Přerušení sledovacího procesu (*Enable* bit do logické 0).
8) Vypnutí *Timestamp* čítače v Časovací jednotce.
9) Ukončení akvizičního procesu.
10) Vynulování registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
11) Vypsání výsledků.
12) Body 1-11 jsou opakovány podle hodnoty proměnné *runs* v proceduře.

Zdrojový kód této testové procedury je v souboru *tp_track_code_gen.sv*.

B.10.1 Track_prn_code_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- *BOC enable* bit,
- rychlost generace PRN kódů,
- délka PRN kódů,
- integrovaná délka PRN kódů.

B.11 Tp_track_code_gen_no_boc

Tato testová procedura ověřuje generování PRN DATA a PRN PILOT kódů. Monitoruje výstupy generátoru PRN kódu ve Sledovací jednotce přes rozhraní připojené do obvodu pomocí *bind* příkazu. V této proceduře není povolena BOC modulace PRN kódů. Obsahuje funkce *build_phase(), start_of_simulation_phase(), report_phase(),* procedury *run_test().*

Ve funkci *build_phase()* je získáno PRN rozhraní a hodnota periody hodinového signálu v Procesorovém rozhraní. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Generace očekávaných PRN DATA a PRN PILOT kódů, monitorování výstupů PRN generátoru a porovnání s očekávanými hodnotami probíhá v testové proceduře *run_test()*. Průběh této procedury je následující:

1) Vygenerování a vypsání náhodných hodnot rychlosti generace PRN kódu (hodnota pro *CODE_NCO_RATE* registr), délky PRN kódu a PRN DATA a PRN PILOT kódy. PRN kódy nejsou vypisovány. Pro *BOC_enable* bit je generována hodnota 0.
2) Vypočtení délky očekávané sekvence, aby bylo zkontrolováno alespoň 5 úplných cyklů PRN kódů.
3) Zapsání hodnot do registrů *TRACKING_PRN_LENGTH, PRN_INTEGRATION_LENGTH, CODE_NCO_RATE, TRACKING_START_TIMESTAMP* a zapsání PRN kódů do PRN pamětí Sledovací jednotky.
4) Povolení sledovacího procesu (*Enable* bit do logické 1).
5) *fork....join* blok:
    a. povolení *Timestamp* čítače v Časovací jednotce,
    b. *while* smyčka periodicky generující a zapisující náhodné hodnoty korekcí do *CODE_NCO_PHASE_INCREMENT* a *PRIMARY_CODE_MEMORY_INCREMENT* registrů, pokud je monitorování aktivní,
    c. while smyčka monitorující výstupy PRN generátoru:
    d. generace očekávaných hodnot na výstupech PRN generátoru,
    e. sejmutí hodnot na výstupů PRN generátoru,
    f. porovnání sejmutých a očekávaných hodnot,

g. inkrementace čítače správných porovnání nebo chybných porovnání a výpis výsledku porovnání,

6) Přerušení sledovacího procesu (Enable bit do logické 0).
7) Vypnutí Timestamp čítače v Časovací jednotce.
8) Ukončení akvizičního procesu.
9) Vynulování registrů TRACKING_PRN_DATA_ADDRESS_OFFSET a TRACKING_PRN_PILOT_ADDRESS_OFFSET.
10) Vypsání výsledků.
11) Body 1-10 jsou opakovány podle hodnoty proměnné runs v proceduře.

Zdrojový kód této testové procedury je v souboru *tp_track_code_gen_no_boc.sv*.

B.12 Tp_track_code_gen_no_incr

Tato testová procedura ověřuje generování PRN DATA a PRN PILOT kódů. Monitoruje výstupy generátoru PRN kódu ve Sledovací jednotce přes rozhraní připojené do obvodu pomocí *bind* příkazu. V této proceduře nejsou generovány a zapisovány korekce fáze PRN kódů a ukazatele do PRN paměti. Obsahuje funkce *build_phase(), start_of_simulation_phase(), report_phase(),* procedury *run_test().*

Ve funkci *build_phase()* je získáno PRN rozhraní a hodnota periody hodinového signálu v Procesorovém rozhraní. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Generace očekávaných PRN DATA a PRN PILOT kódů, monitorování výstupů PRN generátoru a porovnání s očekávanými hodnotami probíhá v testové proceduře *run_test()*. Průběh této procedury je následující:

1) Vygenerování a vypsání náhodných hodnot *BOC_enable* bitu, rychlosti generace PRN kódu (hodnota pro *CODE_NCO_RATE* registr), délky PRN kódu a PRN DATA a PRN PILOT kódy. PRN kódy nejsou vypisovány.
2) Vypočtení délky očekávané sekvence, aby bylo zkontrolováno alespoň 5 úplných cyklů PRN kódů.
3) Zapsání hodnot do registrů *TRACKING_PRN_LENGTH, PRN_INTEGRATION_LENGTH, CODE_NCO_RATE, TRACKING_START_TIMESTAMP* a zapsání PRN kódů do PRN pamětí Sledovací jednotky.
4) Povolení sledovacího procesu (*Enable* bit do logické 1).
5) *fork....join* blok:
    a. povolení *Timestamp* čítače v Časovací jednotce,
    b. while smyčka monitorující výstupy PRN generátoru:
        i. generace očekávaných hodnot na výstupech PRN generátoru,
        ii. sejmutí hodnot na výstupů PRN generátoru,
        iii. porovnání sejmutých a očekávaných hodnot,
        iv. inkrementace čítače správných porovnání nebo chybných porovnání a výpis výsledku porovnání,

6) Přerušení sledovacího procesu (*Enable* bit do logické 0).
7) Vypnutí *Timestamp* čítače v Časovací jednotce.
8) Ukončení akvizičního procesu.
9) Vynulování registrů *TRACKING_PRN_DATA_ADDRESS_OFFSET* a *TRACKING_PRN_PILOT_ADDRESS_OFFSET*.
10) Vypsání výsledků.
11) Body 1-10 jsou opakovány podle hodnoty proměnné *runs* v proceduře.

Zdrojový kód této testové procedury je v souboru *tp_track_code_gen_no_incr.sv*.

B.13 Tp_track_engine

Tato testová procedura se zaměřuje na celý sledovací proces. Ověřuje, zda pro různé hodnoty registrů ovliňující sledovací proces jsou vypočteny správné integrační a PVT výsledky. Jsou generovány kombinace vstupních hodnot pro sledování reálných satelitních signálů. Obsahuje funkce *build_phase(),start_of_simulation_phase(), report_test_param(), report_phase()*, procedury *read_input_prn_file(), send_prn_code_length(), send_prn_code(), set_registers(), run_test()* a skupiny funkčního pokrytí *track_engine_cg, track_corr_dly_cg, track_incr_cg*.

Ve funkci *build_phase()* jsou z konfigurační databáze získány hodnoty zpoždění odeslání datové transakce a zpoždění pro povolení sledovacího procesu potřebné pro stejné synchronizování uvnitř Sledovací jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_test_param()* vypíše parametry sledovacího procesu.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_engine_cg, track_corr_dly_cg a track_incr_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *read_input_prn_file()* vyčte PRN kód ze souboru a zformátuje kód do 16bitových slov. Procedura má 1 vstupní, typ PRN kódu (PRN DATA nebo PRN PILOT), a 2 výstupní parametry, PRN kód a jeho délku.

Procedura *send_prn_code_length()* zapíše délku PRN kódů do registru *TRACKING_PRN_LENGTH*.

Procedura *send_prn_code()* zapíše PRN kód do příslušné paměti PRN kódu Sledovací jednotky.

Procedura *set_registers()* zapíše náhodné do registrů *CODE_NCO_RATE, TRACKING_START_TIMESTAMP, TRACKING_CARRIER_NCO_RATE, TRACKING_CONTROL* a *CORRELATOR_1-6_DELAY*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Vygenerování náhodných hodnot.
2) Vygenerování a zapsání náhodné hodnoty do *TIME_PVT_COUNTER_LENGTH* registru.
3) Načtení PRN DATA a PRN PILOT kódů ze vstupních souborů.
4) Vypsání parametrů integrace.
5) Zapsání délky PRN kódů do registru *TRACKING_PRN_LENGTH*.
6) Zapsání PRN kódů do příslušné PRN paměti Sledovací jednotky.
7) Zapsání délky PRN kódů k integraci do registru *PRN_INTEGRATION_LENGTH*.
8) Zapsání náhodných hodnot do registrů *CODE_NCO_RATE, TRACKING_START_TIMESTAMP, TRACKING_CARRIER_NCO_RATE, CORRELATOR_1-6_DELAY* a *TRACKING_CONTROL*.
9) Vzorkování *track_engine_cg* a *track_corr_dly_cg*.
10) Vypočtení délky transakce pro alespoň pro 10 integračních a 10 PVT cyklů.
11) *fork....join_none* blok:
    a. povolení *Timestamp* čítače v Časovací jednotce,
    b. odeslání transakce do vstupního kanálu 1,
    c. odeslání transakce do druhého vstupního kanálu.
11) Vyčkání po dobu 2 mikrosekund.
12) *fork....join* blok:
    a. *repeat*(délka transakce) smyčka počítající počet odeslaných dat a nastavující kontrolní vlajky k vyčtení integrační a PVT výsledky, po každém cyklu se čeká po dobu periody hodinového signálu Datového rozhraní,
    b. *while* smyčka provádějící vyčtení integračních a PVT výsledků, po vyčtení integračních výsledků zapíše náhodné hodnoty korekcí do *CODE_NCO_PHASE_INCREMENT* a *PRIMARY_CODE_MEMORY_INCREMENT* registrů a probíhá vzorkování *track_incr_cg*.

Zdrojový kód této testové procedury je v souboru *tp_track_engine.sv*.

B.13.1 Track_engine_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- *Enable* bit,
- *BOC enable* bit,
- rychlost generace PRN kódu,
- *Channel select* bity,
- délka PRN kódů,
- integrovaná délka PRN kódů,
- odstup zpoždění mezi korelátory.

B.13.2 Track_corr_dly_cg

Tato skupina pokrytí kontroluje generované hodnoty pro zpoždění korelátorů.

B.13.3 Track_incr_cg

Tato skupina pokrytí kontroluje generované hodnoty korekcí pro NCO PRN kódů a ukazatele v PRN pamětech.

B.14 Tp_track_engine_no_boc

Tato testová procedura se zaměřuje na celý sledovací proces. Ověřuje, zda pro různé hodnoty registrů ovliňující sledovací proces jsou vypočteny správné integrační a PVT výsledky. Jsou generovány kombinace vstupních hodnot pro sledování reálných satelitních signálů. V této proceduře není povolena BOC modulace PRN kódů. Obsahuje funkce *build_phase(),start_of_simulation_phase(), report_test_param(), report_phase(),* procedury *read_input_prn_file(), send_prn_code_length(), send_prn_code(), set_registers(), run_test()* a skupiny funkčního pokrytí *track_engine_cg*, *track_corr_dly_cg*, *track_incr_cg*.

Ve funkci *build_phase()* jsou z konfigurační databáze získány hodnoty zpoždění odeslání datové transakce a zpoždění pro povolení sledovacího procesu potřebné pro stejné synchronizování uvnitř Sledovací jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_test_param()* vypíše parametry sledovacího procesu.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_engine_cg, track_corr_dly_cg a track_incr_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *read_input_prn_file()* vyčte PRN kód ze souboru a zformátuje kód do 16bitových slov. Procedura má 1 vstupní, typ PRN kódu (PRN DATA nebo PRN PILOT), a 2 výstupní parametry, PRN kód a jeho délku.

Procedura *send_prn_code_length()* zapíše délku PRN kódů do registru *TRACKING_PRN_LENGTH*.

Procedura *send_prn_code()* zapíše PRN kód do příslušné paměti PRN kódu Sledovací jednotky.

Procedura *set_registers()* zapíše náhodné do registrů *CODE_NCO_RATE, TRACKING_START_TIMESTAMP, TRACKING_CARRIER_NCO_RATE, TRACKING_CONTROL* a *CORRELATOR_1-6_DELAY*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Vygenerování náhodných hodnot. Pro *BOC Enable* bit je generována hodnota 0.
2) Vygenerování a zapsání náhodné hodnoty do *TIME_PVT_COUNTER_LENGTH* registru.
3) Načtení PRN DATA a PRN PILOT kódů ze vstupních souborů.
4) Vypsání parametrů integrace.
5) Zapsání délky PRN kódů do registru *TRACKING_PRN_LENGTH*.

6) Zapsání PRN kódů do příslušné PRN paměti Sledovací jednotky.
7) Zapsání délky PRN kódů k integraci do registru *PRN_INTEGRATION_LENGTH*.
8) Zapsání náhodných hodnot do registrů *CODE_NCO_RATE*, *TRACKING_START_TIMESTAMP*, *TRACKING_CARRIER_NCO_RATE*, *CORRELATOR_1-6_DELAY* a *TRACKING_CONTROL*.
9) Vzorkování *track_engine_cg* a *track_corr_dly_cg*.
10) Vypočtení délky transakce pro alespoň pro 10 integračních a 10 PVT cyklů.
11) *fork....join_none* blok:
    1) povolení *Timestamp* čítače v Časovací jednotce,
    2) odeslání transakce do vstupního kanálu 1,
    3) odeslání transakce do druhého vstupního kanálu.
12) Vyčkání po dobu 2 mikrosekund.
13) *fork....join* blok:
    a. *repeat*(délka transakce) smyčka počítající počet odeslaných dat a nastavující kontrolní vlajky k vyčtení integrační a PVT výsledky, po každém cyklu se čeká po dobu periody hodinového signálu Datového rozhraní,
    b. *while* smyčka provádějící vyčtení integračních a PVT výsledků, po vyčtení integračních výsledků zapíše náhodné hodnoty korekcí do *CODE_NCO_PHASE_INCREMENT* a *PRIMARY_CODE_MEMORY_INCREMENT* registrů a probíhá vzorkování *track_incr_cg*.

Zdrojový kód této testové procedury je v souboru *tp_track_engine_no_boc.sv*.

B.14.2 Track_engine_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- Enable bit,
- BOC enable bit,
- rychlost generace PRN kódu,
- Channel select bity,
- délka PRN kódů,
- integrovaná délka PRN kódů,
- odstup zpoždění mezi korelátory.

B.14.3 Track_corr_dly_cg

Tato skupina pokrytí kontroluje generované hodnoty pro zpoždění korelátorů.

B.14.4 Track_incr_cg

Tato skupina pokrytí kontroluje generované hodnoty korekcí pro NCO PRN kódů a ukazatele v PRN pamětech.

B.15 Tp_track_engine_no_incr

Tato testová procedura se zaměřuje na celý sledovací proces. Ověřuje, zda pro různé hodnoty registrů ovliňující sledovací proces jsou vypočteny správné integrační a PVT výsledky. Jsou generovány kombinace vstupních hodnot pro sledování reálných satelitních signálů. V této proceduře nejsou generovány a zapisovány korekce fáze PRN kódů a ukazatele do PRN paměti. Obsahuje funkce *build_phase(),start_of_simulation_phase(), report_test_param(), report_phase(),* procedury *read_input_prn_file(), send_prn_code_length(), send_prn_code(), set_registers(), run_test()* a skupiny funkčního pokrytí *track_engine_cg, track_corr_dly_cg*.

Ve funkci *build_phase()* jsou z konfigurační databáze získány hodnoty zpoždění odeslání datové transakce a zpoždění pro povolení sledovacího procesu potřebné pro stejné synchronizování uvnitř Sledovací jednotky a jejího prediktoru. Tato funkce je volána v UVM *build* fázi.

Ve funkci *start_of_simulation_phase()* je nastavena nižší úroveň pro vypisování zpráv z registrové mapy a Procesorového Agenta, aby výpis testu byl přehlednějším. Tato funkce je volána v UVM *start_of_simulation* fázi.

Funkce *report_test_param()* vypíše parametry sledovacího procesu.

Funkce *report_phase()* po ukončení testové procedury vypíše úroveň funkčního pokrytí v *track_engine_cg, track_corr_dly_cg a track_incr_cg*. Tato funkce je volána v UVM *report* fázi.

Procedura *read_input_prn_file()* vyčte PRN kód ze souboru a zformátuje kód do 16bitových slov. Procedura má 1 vstupní, typ PRN kódu (PRN DATA nebo PRN PILOT), a 2 výstupní parametry, PRN kód a jeho délku.

Procedura *send_prn_code_length()* zapíše délku PRN kódů do registru *TRACKING_PRN_LENGTH*.

Procedura *send_prn_code()* zapíše PRN kód do příslušné paměti PRN kódu Sledovací jednotky.

Procedura *set_registers()* zapíše náhodné do registrů *CODE_NCO_RATE, TRACKING_START_TIMESTAMP, TRACKING_CARRIER_NCO_RATE, TRACKING_CONTROL* a *CORRELATOR_1-6_DELAY*.

Průběh testové procedury je nadefinován v proceduře *run_test()*:

1) Vygenerování náhodných hodnot.
2) Vygenerování a zapsání náhodné hodnoty do *TIME_PVT_COUNTER_LENGTH* registru.
3) Načtení PRN DATA a PRN PILOT kódů ze vstupních souborů.
4) Vypsání parametrů integrace.
5) Zapsání délky PRN kódů do registru *TRACKING_PRN_LENGTH*.
6) Zapsání PRN kódů do příslušné PRN paměti Sledovací jednotky.
7) Zapsání délky PRN kódů k integraci do registru *PRN_INTEGRATION_LENGTH*.

8) Zapsání náhodných hodnot do registrů *CODE_NCO_RATE, TRACKING_START_TIMESTAMP, TRACKING_CARRIER_NCO_RATE, CORRELATOR_1-6_DELAY* a *TRACKING_CONTROL*.
9) Vzorkování *track_engine_cg* a *track_corr_dly_cg*.
10) Vypočtení délky transakce pro alespoň pro 10 integračních a 10 PVT cyklů.
11) *fork....join_none* blok:
    a. povolení *Timestamp* čítače v Časovací jednotce,
    b. odeslání transakce do vstupního kanálu 1,
    c. odeslání transakce do druhého vstupního kanálu.
12) Vyčkání po dobu 2 mikrosekund.
13) *fork....join* blok:
    a. *repeat*(délka transakce) smyčka počítající počet odeslaných dat a nastavující kontrolní vlajky k vyčtení integrační a PVT výsledky, po každém cyklu se čeká po dobu periody hodinového signálu Datového rozhraní,
    b. *while* smyčka provádějící vyčtení integračních a PVT výsledků.

Zdrojový kód této testové procedury je v souboru *tp_track_engine_no_incr.sv*.

B.15.2 Track_engine_cg

Tato skupina pokrytí kontroluje generované hodnoty pro:

- Enable bit,
- BOC enable bit,
- rychlost generace PRN kódu,
- Channel select bity,
- délka PRN kódů,
- integrovaná délka PRN kódů,
- odstup zpoždění mezi korelátory.

B.15.3 Track_corr_dly_cg

Tato skupina pokrytí kontroluje generované hodnoty pro zpoždění korelátorů.