Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Távközlési és Médiainformatikai Tanszék

Fendler Tamás

FPGA-val megvalósított, GPS alapú időpecsételés hálózatmonitorozáshoz

TDK dolgozat

KONZULENS: Dr. Varga Pál

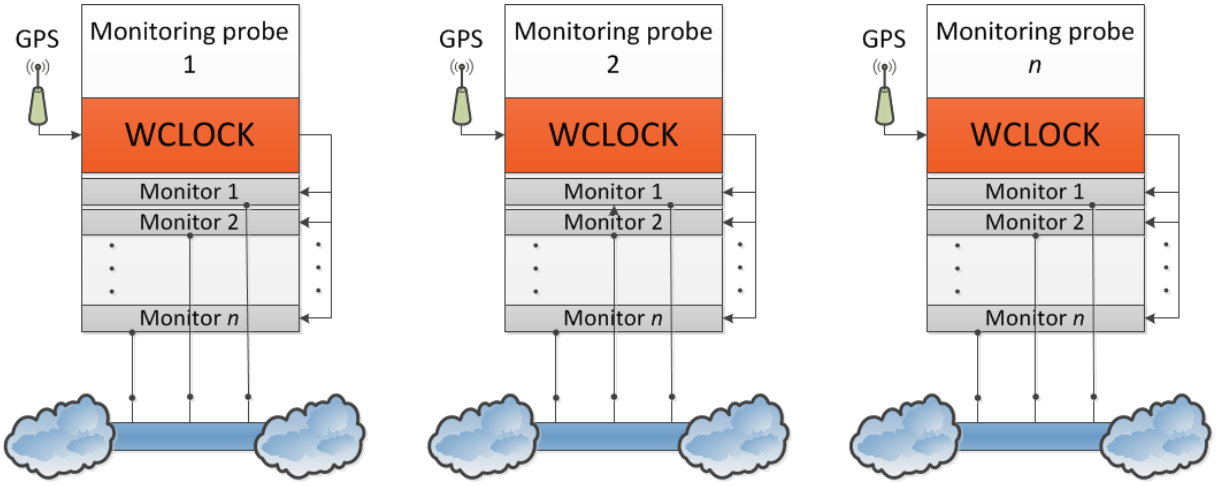
2011

Tartalom

1 Bevezetés

Hálózatok monitorozása során gyakran elosztott adatbázisok adatait szükséges összevetni ahhoz, hogy válaszolni lehessen számos fontos kérdésre. Ez az igény két esetben merülhet fel. Az egyik eset az, amikor földrajzilag egymástól távol levő alhálózatok forgalmának esetleges korrelációját szeretnénk vizsgálni. A másik eset az, amikor olyan nagy a hálózaton megjelenő adatsebesség és forgalommennyiség, hogy azt egy monitorozó egység képtelen feldolgozni. Ilyenkor hardveres szétosztás és előfeldolgozás után több egység létrehozza a saját adatbázisát, melyek együttesen reprezentálják a hálózati forgalmat.

Ezen adatbázisok adatainak feldolgozásakor kritikus a rekord-elemekben található csomagszintű információk valós sorrendjének megőrzése. Ezt a monitorozás során időpecsételéssel biztosíthatjuk, ami a különböző eszközöktől származó bejegyzések összefűzését teszi lehetővé a valós sorrend megtartásával. Ehhez elengedhetetlen, hogy minden monitorozó egység egymással szinkronban levő időpecséttel dolgozzon, földrajzi elhelyezkedéstől függetlenül. A dolgozatban egy FPGA alapú célhardvert mutatok be (WCLOCK bővítőkártya), mely időforrásként GPS-t használva képes kellő pontosságú időinformációt szolgáltatni az elosztott monitorozó rendszer egyes elemeinek. A GPS biztosítja az egész földön elérhető egyezményes rendszeridőt, a hardveres feldolgozás és időmérés pedig megfelelő időpecsételést tesz lehetővé akár 10 Gbps Ethernet hálózatok esetén is. A kártya elhelyezkedését egy elosztott monitorozó architektúrában az 1. ábra mutatja.



1. ábra: A monitorozó egységek WCLOCK kártyával kibővítve

Az egyes egységek (az ábrán „Monitoring probe” felirattal jelöltek) általános célú számítógépek, melyekben egy vagy több, monitorozást támogató bővítőkártya (Monitor *n*) működik. Ezekhez kapcsolódik a WCLOCK kártya, azaz monitorozó gépenként egy ilyen eszköz szükséges ahhoz, hogy az elosztott rendszer összegzett adatbázisában garantálható legyen a sorrendhelyesség.

Az FPGA-n futó firmware teljes egészében a saját munkám eredménye, és a bővítőkártya tervezésekor is döntő szerepem volt az architektúra kialakításában. A monitorozó egységek, melyekkel együtt kell működnie a WCLOCK kártyának, az AITIA International Zrt. termékei. Ezek szintén FPGA alapú, PCI-Express csatolóval rendelkező kártyák, melyek hagyományos PC-ben működve végzik 1 vagy 10 Gbps sebességű Ethernet hálózatok monitorozását [1]. Ahhoz, hogy ezek az eszközök képesek legyenek fogadni az időinformációt a WCLOCK kártyától, a már működő, ipari környezetben használt firmware-ek módosítására volt szükség. Ez szintén az én feladatom volt.

2 Szakmai háttér

2.1 Ethernet

Az Ethernet jelenleg a legelterjedtebb kábeles megoldás számítógépek hálózatba kapcsolására, de egyre nagyobb teret hódít a szolgáltatók hozzáférési- és maghálózataiban is. A végfelhasználók leggyakrabban a 100 Mbps és 1 Gbps sebességű változatot használják, azonban a városi hálózatokban ez a sebesség kevés, a szolgáltatók egyre gyorsabb és gyorsabb adatátvitelt szeretnének, hogy minél több szolgáltatást tudjanak egyidejűleg biztosítani. Erre az igényre reagálva az IEEE-nél 1999-re kidolgozták a 802.3z [2] és 802.3ab [3] szabványt, melyek az 1 Gbps sebességű Ethernetet specifikálják optikai és réz kábelek használatával. 2002-ben jelentkeztek a következő, 10 Gbps sebességet leíró szabvánnyal, a 802.3ae jelűvel [4], majd 2006-ban csavart rézkábelen is erre a szintre emelték az elérhető sebességet a 802.3an szabványban[5]. 2007-ben megkezdték a további fejlesztést, párhuzamosan kidolgozták a 40 és a 100 Gbps sebességet rögzítő szabványt, melyet 2010-ben hagytak jóvá és a 802.3ba jelölést kapta [6]. Az IEEE úgy ítélte meg, hogy mindkét verziónak megvan a létjogosultsága [7]. A szerverek forgalmát egyesítő hálózatoknak a 40 Gbps sebességű megoldás jelenti a legjobb kompromisszumot a teljesítmény és a költségek között, a 100 Gbps pedig a szolgáltatók hozzáférési és maghálózataiban elégítheti ki a nagy sávszélesség-igényt.

Az IEEE mindig úgy növelte az Ethernet sebességét, hogy az alapvető keretszerkezethez nem nyúlt. A minimális keretméret 64 bájt, az fejléc nélküli adatmező minimális hossza 46 bájt, maximális hossza 1500 bájt[1]. Ennél kevesebb adatot tartalmazó kereteket is 64 bájt elküldésével lehet továbbítani, a nem használt biteket 0-ra állítva. A változatlan keretfelépítésnek köszönhetően alapvetően mindegy, melyik 802.3 Ethernet szabvánnyal van dolgunk, az 1. táblázat mutatta szerkezettel találkozunk.

1. táblázat: Egy szabványos Ethernet csomag felépítése

Méret	Mező neve
12 bájt	***Keretek közti szünet (Inter Frame Gap)***
7 bájt	***Előtag (Preamble)***
1 bájt	***Keret kezdet (Start Frame Delimiter)***
6 bájt	Szolgáltatói célcím (Provider-MAC Destination Address)
6 bájt	Szolgáltatói forráscím (Provider-MAC Source Address)
6 bájt	**Célcím (MAC Destination Address)**
6 bájt	**Forráscím (MAC Source Address)**
4 bájt	VLAN címke (Virtual LAN tag)
4 bájt	S/P VLAN címke (Service/Provider VLAN tag)
2 bájt	**Hossz/típus (Length/Type)**
46 – 1500 bájt	**Adatmező (Data Field)**
4 bájt	**Ellenőrző összeg (Frame Check Sequence)**

[1] A 802.3 szabvány ide számolja a Q-tagged kereteket (802.1q), ami a 4 bájtos VLAN címkét definiálja, ebben az esetben az adatmező maximális méretét 1504 bájtban szabja meg (ennek ellenére a VLAN címke az Ethernet fejlécben, a forráscím és a hossz mező között helyezkedik el). Egy másik kerettípus a kiterjesztett keret, melyet a 802.3as kiegészítés definiál és a maximális kerethosszt 2000 bájtban határozza meg.

A táblázatban félkövér betűvel szedett mezők minden 802.3-nak megfelelő Ethernet keretben kötelezők, a hosszuk a megadott értékek között változhat. A normál betűvel írt mezőket különböző szabványok (pl. 802.1ad: Q-in-Q és 802.1ah: MAC-in-MAC) definiálják, használatuk elsősorban szolgáltatói hálózatban jellemző. A félkövér, dőlt betűvel jelölt első három mező (IFG, Preamble, SFD) nem tartalmaz érdemi adatot, de mindhármat kötelező elküldeni minden Ethernet keret előtt, így a későbbiekben az időzítések számításánál ezek is ugyanolyan fontosak lesznek, mint a tényleges adatot tartalmazó mezők.

A szabvány az előtaggal és a keret kezdetét jelző bájttal kiegészített keretet csomagnak nevezi. Minél rövidebbek a monitorozott hálózatban a keretek, annál gyakrabban érkezhetnek újabbak, így az időzítéseket úgy kell kezelni, hogy a lehető legkisebb méretű keretekből álló folyam esetén is teljesüljenek a követelmények. Ezért a továbbiakban nem foglalkozom a szabvány különböző kiegészítéseivel, melyek hosszabb kereteket is megengednek, hiszen minél hosszabbak a keretek, annál „könnyebb” dolga van az időpecsételési módszernek[2].

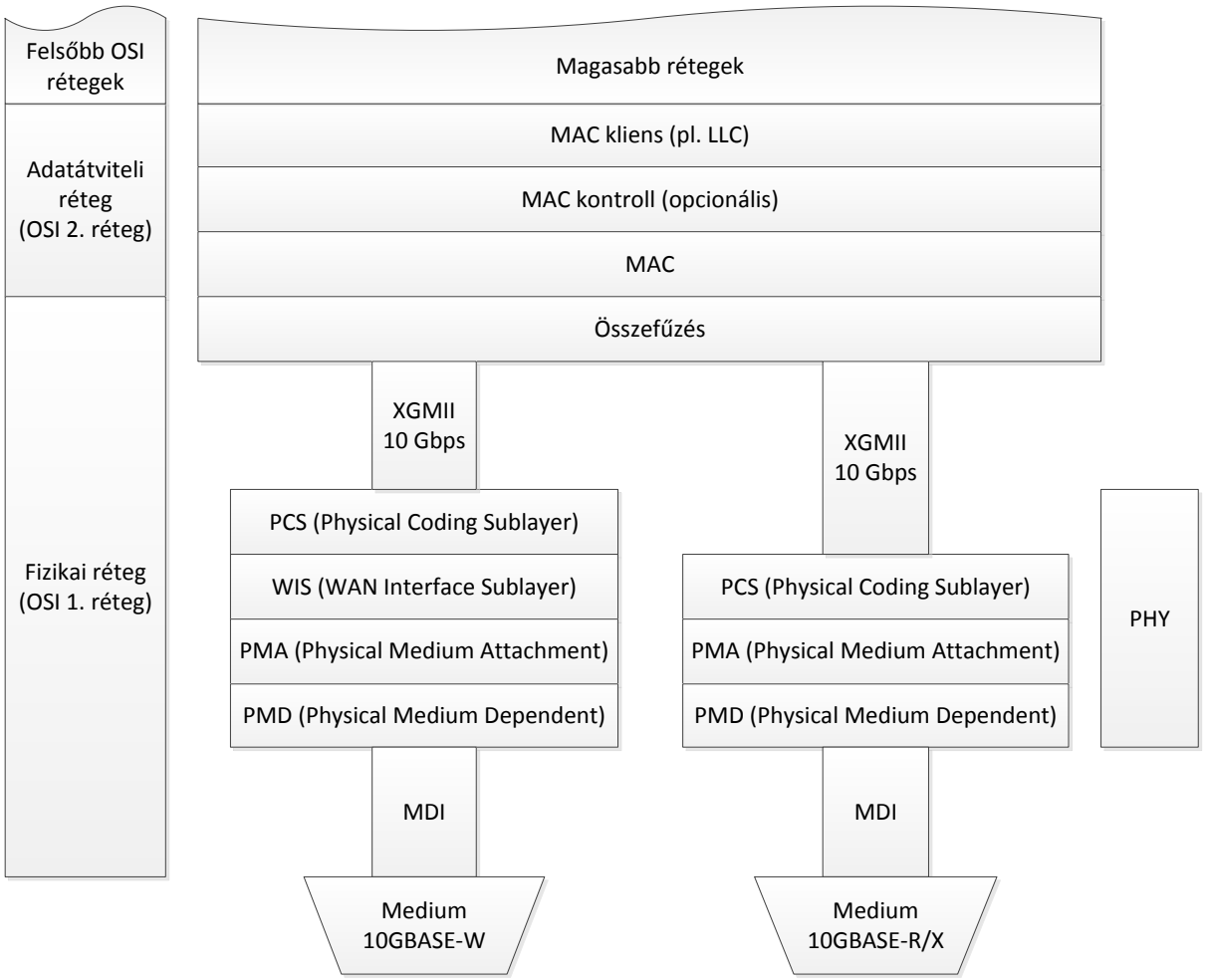
2.1.1 10 Gigabit Ethernet – 802.3ae és 802.3an

A 2002-ben véglegesített 802.3ae szabvány definiálja a 10 Gbps sebességű Ethernetet (10GBASE) optikai szálon. Alapjában véve megtartotta a klasszikus Ethernet tulajdonságokat (pl. csomagok szerkezete), de néhány dologban szükséges volt a változtatás. Az egyik legfőbb újítás, hogy 2002-ben csak optikai szálra definiálták a 10 Gbps Ethernetet, de ezt 2006-ban kiegészítette a 802.3an réz vezetőre. A másik fontos változás, hogy a 10GBASE csak full-duplex üzemmódot támogat. Ebből következik, hogy innentől kezdve nincs szükség a CSMA/CD funkcióra, mely ekkora sebességen már komoly problémákat jelentene [8]. Minden más lényeges szempontból illeszkedik a korábbi szabványokhoz, felépítését az 2. ábra mutatja.

Az adatátviteli MAC réteg egy média független (XGMII) és Ethernet PHY funkción keresztül csatlakozik az adott médiához (optikai szálhoz). Ez a PHY réteg több alrétegből áll, ezek a fizikai médium függő (PMD), a fizikai médium csatolmány (PMA) és a fizikai kódolás (PCS). A WAN (Wide Area Network) hálózathoz készült változatban szerepel még egy WAN interfész alréteg (WIS) a PMA és a PCS között. A 802.3an szabvány ezt annyiban módosítja, hogy a PMD alréteg helyett Auto-Negotiation (AN) réteg szerepel.

A szabvány a csomagok alapvető szerkezetén nem változtatott, azaz az 1. táblázat szerinti felépítés érvényben marad 10 Gbps sebességen is egy mező kivételével. A keretek közti szünet (IFG), mely a 100 Mbps Ethernetig 12 bájt hosszú volt, már az 1 Gbps szabványban is lerövidült 8 bájtra, a 802.3ae azonban már 5 bájtot is megenged. Ez ugyan nem tartozik a csomagok szerkezetéhez, de befolyásolja az időzítéseket, így a monitorozásnál szerepet fog játszani a későbbiekben.

[2] A keretek tárolása persze annál nehezebb feladat, minél nagyobb a méretük, hiszen ekkor ritkábban van ideje „pihenni” az eszköznek és időegység alatt több az értékes adat.



2. ábra: A 802.3ae architektúra [4]

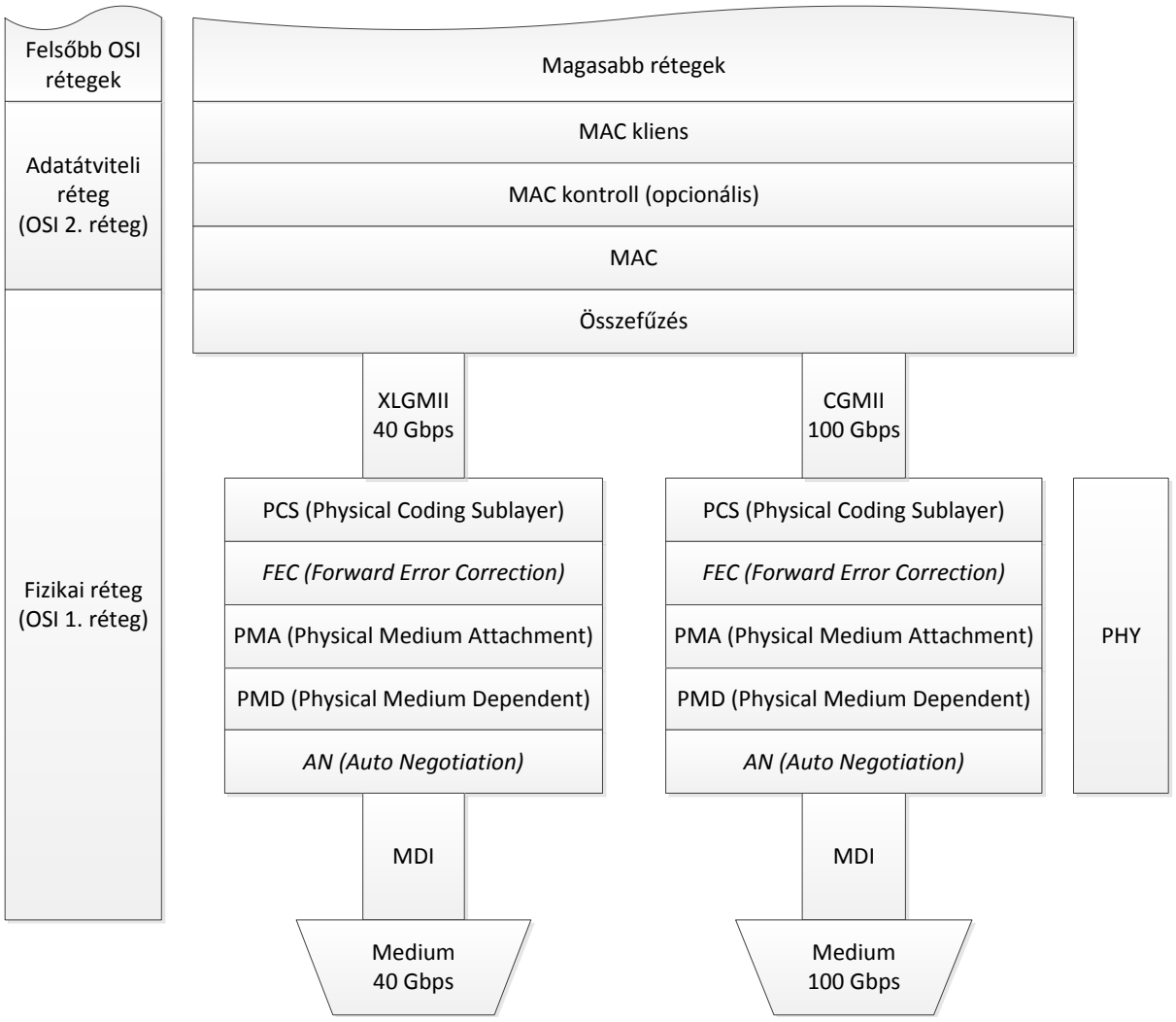
2.1.2 40 és 100 Gigabit Ethernet – 802.3ba

A 40 és 100 Gbps Ethernet kidolgozása 2008 januárjában kezdődött és a 802.3ba számú kiegészítéssel zárult 2010. június 17-én. A kezdeti célkitűzések a következők voltak [9]:

- csak full-duplex átvitel támogatása,
- a 802.3/Ethernet keret formátum megőrzése,
- a legkisebb és legnagyobb keretméret megőrzése,
- 10^{-12}-nél jobb bithiba-arány a MAC/ fizikai réteg interfészen,
- az optikai szállító hálózat megfelelő támogatása,
- 40 és 100 gigabit/másodperc MAC adatsebesség támogatása,
- fizikai szintű specifikáció, mely támogatja a 40 Gbps sebességet:
    - legalább 10 km távolságon egymódusú optikai szálon (SMF)
    - legalább 100 m távolságon OM3 többmódusú optikai szálon (MMF)
    - legalább 7 m távolságon, rézkábelen
    - legalább 1 m távolságon, alaplapi vezetéken
- fizikai szintű specifikáció, mely támogatja a 100 Gbps sebességet:
    - legalább 40 km távolságon egymódusú optikai szálon (SMF)
    - legalább 100 m távolságon OM3 többmódusú optikai szálon (MMF)

- legalább 7 m távolságon, rézkábelen

A szabvány egy egyszerű architektúrát ad meg, mely illeszkedik mind a 40, mind a 100 Gbps Ethernethez és bármely fizikai rétegbeli specifikációhoz. Ezt a szerkezetet a 3. ábra mutatja.



3. ábra: IEEE 802.3ba architektúra [6]

A felépítés nagyon hasonló ahhoz, amit a 802.3ae bemutatott. A MAC réteg a média független (XLGMII vagy CGMII) és Ethernet PHY rétegeken keresztül csatlakozik az optikai vagy réz szálhoz. Utóbbi esetén két extra alréteg jelenik meg, amit dőlt betűvel jelöltem az ábrán, a hibajavító (FER) és az auto-tárgyaló (AN) alréteg.

2.1.3 Ethernet alapú hálózatok monitorozása

Bármilyen hálózatról van szó, alapvetően kétféle eljárást különböztetünk meg: az *aktív* és a *passzív* monitorozást. Aktív monitorozásról beszélünk, ha a hálózathoz csatlakozunk egy eszközzel, bizonyos csomagokat vagy hálózati folyamokat *küldünk* el és ezt valahol a hálózat egy pontján lévő másik eszközzel *vesszük*, regisztráljuk. Ez vagy azt jelenti, hogy a hálózat normális üzeme alatt extra forgalmat generálunk (és ezzel „feleslegesen” terheljük azt), vagy egy használaton kívüli hálózathoz kizárólagosan a teszteszközöknek biztosítunk hozzáférést. Mindkét változat kedvezőtlen pl. egy

internetszolgáltató esetében, ugyanakkor a hálózat teljesítményét mutató paraméterek (pl. throughput, latency, frame loss rate, stb.) javarészt csak aktív monitorozással mérhetőek.

A passzív monitorozás azt jelenti, hogy egy adott hálózat forgalmát annak üzemi *működése közben* egy eszköz folyamatosan *veszi* és regisztrálja. Ez sokkal kedvezőbb szolgáltatói szempontból, hiszen nem terheli feleslegesen a hálózatot és az adott szakasz forgalomból való kivonását sem igényli. Emellett a passzív monitorozás feladata alapvetően más, mint az aktívé: a fő cél ebben az esetben a valós forgalom (vagy annak valamilyen mérőszámainak) regisztrálása későbbi elemzés céljából. Látható, hogy a két módszernek mások az igényei és más célt szolgálnak, de mindkettőre megvan az igény a különböző felhasználási területeken, ezért mindkettőnek van létjogosultsága, kiegészítik egymást. Mindkettő összeállítás tartalmaz vevőegységeket, amikkel szemben alapvetően ugyanazt a két fő követelményt támaszthatjuk:

- *veszteségmentes* monitorozás csomagvesztés nélkül,
- a vett csomagok (vagy azok paramétereinek) *sorrendhelyes* tárolása.

Az első követelmény magától értetődő, hiszen a cél éppen az, hogy megfigyeljük a hálózat forgalmát. A másodikat azért fontos kiemelni, mert ha az egyes kereteket nem az eredeti sorrendjükben tárolnánk, akkor a későbbi vizsgálatok során pl. előfordulhat, hogy egy kérést előbb látunk, mint az arra érkezett választ. Ez lehetetlenné tenné pl. a felhasználók viselkedésének, szokásainak vizsgálatát így a sorrendhelyes tárolás legalább annyira fontos követelmény, mint a veszteségmentes detektálás

Mindkettő monitorozási eljárás esetében gyakori, hogy egy időben több eszköz regisztrálja a forgalmat és adatbázisukat egy későbbi időpontban összefűzik. Ekkor nem elég, ha az eszközök külön-külön sorrendhelyesen tárolják a kereteket, meg kell oldani azt is, hogy az egyesített adatbázisban látható sorrend szintén megfeleljen a valóságnak. Ezt teszi lehetővé a vett keretek *időpecsételése*. Ez egyszerűen annyit jelent, hogy mindegyik vevő a vétel pillanatában egy *egyezményes* időnek megfelelő időpecsétet csatol a kerethez, így összefűzéskor a különböző helyen regisztrált csomagokat az egyesített adatsorban az időpecsétjük szerint sorba lehet rendezni. Így már nem csak az egyéni, hanem az összefűzött adatbázisban látható sorrend is helyes lesz.

Ezeknek megfelelően olyan időpecsételési módszert kellett találnom, mely a mai, tipikusan 1 vagy 10 Gbps sebességű szolgáltatói Ethernet hálózatokban is biztosítani tudja ezeket a követelményeit. Ehhez először azt kell rögzíteni, hogy milyen felbontású időpecsétre van szükség ekkora sebesség esetén, azaz mennyi az a legrövidebb idő, ami eltelik két csomag kezdete között. Az 1. táblázat alapján ezt könnyen ki lehet számítani: 64 bájt a minimális keretméret, 1 bájt a keret kezdetét jelző bájt, 7 bájt az előtag és (a 2.1.1 fejezet alapján) legalább 5 bájt a keretek közti szünet. Így a következőt kapjuk:

$$minPacketSize = 5_{IFG} + 7_{Preamble} + 1_{SFD} + 64_{frame} = 77\ byte$$

Ez 10 Gbps sebesség esetén időben a következőt jelenti:

$$t_{min} = 77 \times 8 \times 10^{-10} = 61{,}6\ ns$$

Tehát legrosszabb esetben is ennyi idő telik el két keret kezdete között, így az időpecsételési módszernek is legalább ilyen felbontásúnak kell lennie. Azt is biztosítani kell, hogy a különböző

helyen működő vevők órái ne térjenek el egymástól ennél jobban, mert abban az esetben már előfordulhat olyan csomag, ami rossz sorrendben szerepel az összefűzött adatbázisban.

2.2 GPS alapú órajel

A GPS (Global Positioning System) pontos, folyamatos, az egész világon elérhető információt szolgáltat a megfelelő vevőberendezéssel rendelkező felhasználóknak pozíciójukról. Emellett sok egyéb adat is kinyerhető a rendszer üzeneteiből, pl. az UTC (Universal Time, Coordinated) alapú GPS idő. A rendszer felépítése három nagyobb részre bontható: űr szegmens, irányító szegmens és felhasználói szegmens. Az űr szegmens 24 műholdból áll, ezek négyesével egy-egy föld körüli pályán keringenek. A földi irányító szegmens folyamatosan figyeli a műholdak állapotát és szükség esetén frissíti a navigációs üzenetek tartalmát. Felhasználói szegmensnek nevezzük a felhasználók, illetve vevőkészülékeik összességét. Ezek az eszközök a műholdak szempontjából passzívnak tekinthetők (csak vevők), így a rendszer tetszőleges nagyságú felhasználói kört ki tud szolgálni.

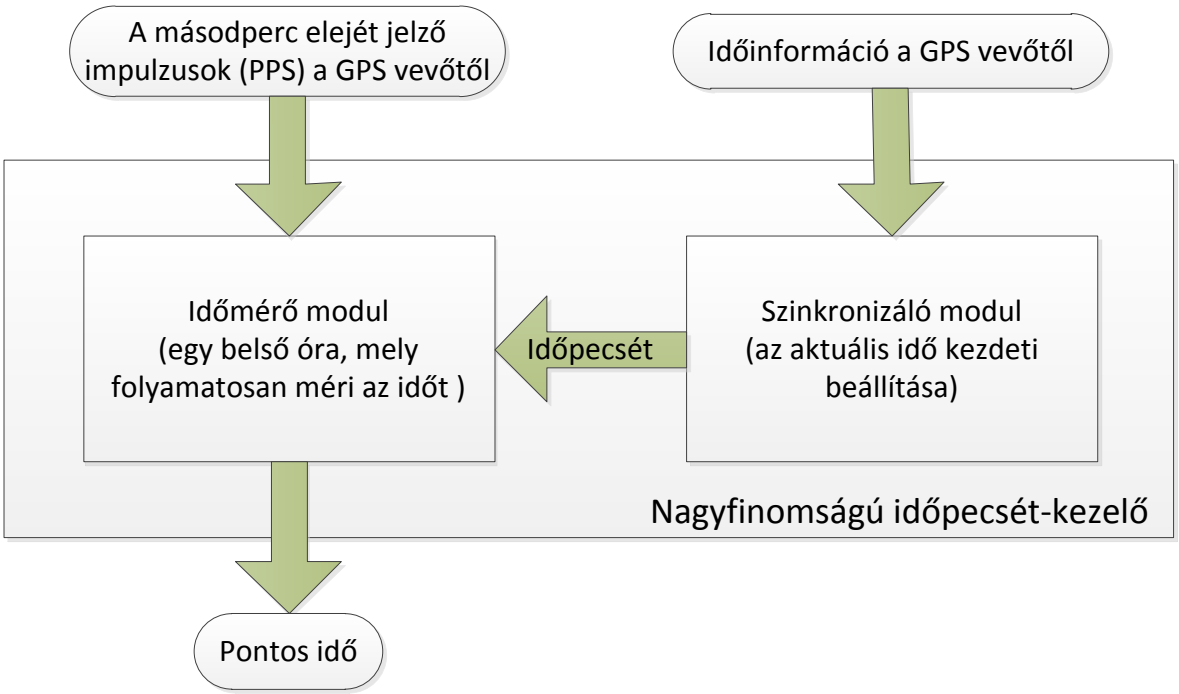
A pozíció meghatározása az egyirányú jelterjedési idő mérésén alapszik. Mindegyik műholdon található egy nagypontosságú atomóra, mely szinkronban van a GPS idővel. A műholdak 50 bit/s sebességgel 1500 bites kereteket küldenek, azaz egy keret 30 mp ideig tart, és minden keret kezdete a műhold atomórája alapján az adott perc elején/közepén kezdődik. Az átvitelhez csak két frekvenciát használnak (L1 – 1575,42 MHz és L2 – 1227,6 MHz) kódosztásos többszörös hozzáféréssel. A vevő az üzenetekből tudja meg a küldő műhold pontos helyzetét és az adás kezdetének pontos idejét, így a vétel időpontja alapján ki tudja számítani a műhold-felhasználó távolságot. Amennyiben a vevőben is van egy pontos óra, ami szinkronban jár a GPS idővel, három különböző műhold üzenete alapján meghatározható a felhasználó helyzete. Azonban a vevőkben található olcsó oszcillátor általában nem pontos, nincs szinkronizálva semmilyen egyezményes időhöz, így szükség van még egy mérésre egy negyedik műhold jele alapján. A négy mérés eredményéből meghatározható a vevő helyzete és a belső óra csúszása a GPS időhöz képest [10].

A GPS rendszer ideje szinkronban van az UTC idővel, azonban a szökőmásodperceket nem adják hozzá, az időskála 1980. január 6. óta folyamatos, megszakítástól és ugrástól mentes. Az eltérés tehát pontosan egy másodperc egész számú többszöröse, amit a vevők kompenzálnak a GPS üzenetek alapján, így ez nem befolyásolja az UTC-hez viszonyított pontosságot. A felhasználó pozíciójának meghatározása a minél pontosabb időinformáción alapszik, ezért a rendszer igen jól használható időzítési feladatok esetén is, mint referencia. Egymástól földrajzilag távol levő egységek olcsó és kicsi vevőkkel ellátva szinkronizált idővel rendelkezhetnek, így a GPS megfelelhet elosztott monitorozó rendszerek időforrásaként. A legtöbb vevő rendelkezik egy PPS (Pulse Per Second) kimenettel is, melyen egy rövid impulzus jelzi minden másodperc kezdetét, segítve ezzel az időzítési feladatokat. A kontroll szegmensnek úgy kell felügyelnie a GPS időt, hogy az legfeljebb 1 µs eltérést mutasson az UTC időhöz képest, azonban az általában elérhető pontosság 20-500 ns között mozog, mely nagyságrendileg megegyezik a 2.1.3 részben számított értékkel [11]. Ehhez persze szükséges egy, a műholdakra tisztán rálátó külső antenna. Számos nemzetközi cég használja a GPS jelét időforrásnak hálózati feladatokra.

- Endace
  A hardveres hálózatmonitorozásban piacvezető cég három lehetséges pontos időforrást javasol: GPS, CDMA és PTP (IEEE 1588). A legjobbnak a GPS technológiát mondja, állítása szerint +/- 15 ns pontosság érhető el a bolygó bármely pontján. A monitorozó egység órájának beállítására a vevő nagy pontosságú 1 Pulse-Per-Second jelét használja [12].
- CES.NET
  A csehországi egyetemek alkotta CESNET egyik projektje az időszinkron pontosságát vizsgálja nagy hálózatok esetén. Elsődleges időforrásnak GPS vevőt használnak, az 1 PPS jelhez szinkronizált idő pontosságát ns nagyságrendűnek mondják [13].
- Trimble
  A cég vezető a fejlett helymeghatározó megoldások piacán, egyik termékük egy kifejezetten időzítési célokra készült antenna. Állításuk szerint a szinkronhoz használt kimeneti impulzus 15 ns-on belül van a GPS rendszer idejéhez képest [14].

2.3 Az időpecsét

A 2.2 fejezet alapján a GPS kellő pontosságú időinformációt biztosít, azonban ez nem áll folyamatosan rendelkezésre. A műholdak csak fél másodpercenként küldenek üzeneteket, az időzítési célokra kialakított PPS kimeneten pedig csak másodpercenként jelenik meg egy impulzus. A 2.1.3 alapján az időpecsétnek legalább néhányszor 10 ns pontosságúnak kell lennie, tehát mindenképpen szükség van egy belső időmérő funkcióra a monitorozó eszközön belül, mely folyamatosan beállítja magát pl. a PPS impulzusok alapján. A pontos időpecsét-kezelés tehát két részfeladatra bontható. Az egyik egy belső óra működtetése, a másik a szinkronizálás. Előbbi annyit takar, hogy egy szinkronizálás után a hardver képességeihez mérten a lehető legpontosabban mérje az időt és a mindenkori aktuális időpecsétet megjelenítse a kimeneten. A szinkronizálást végrehajtó modul legfőbb feladata, hogy induláskor valamilyen forrástól lekérje a pontos időt (amikor az rendelkezésre áll) és ezt átadja az időmérő modulnak, ami ettől az időponttól méri majd az időt. A rendszer felépítését az 4. ábra mutatja.



4. ábra: Az időpecsét-kezelő modul rendszerterve

2.3.1 Felbontás, pontosság, precizitás, stabilitás

Az eszközön belüli időmérés az eszköz órajelén alapszik, melynek forrása tipikusan egy kristály oszcillátor. A belső óra gyakorlatilag egy számláló, mely minden órajelre növeli az értékét és a névleges frekvenciából számítható, hogy egy órajel mennyi időnek felel meg. Ahhoz, hogy ezt az órát felhasználhassuk, legalább néhányszor 10 ns felbontásúnak kell lennie. 10 ns esetén ez frekvenciában 1/10 ns = 100 MHz-t jelent, ilyen frekvenciájú oszcillátorok kereskedelmi forgalomban kaphatók. A kérdés, hogy megfelelő minőségű-e egy ilyen óra, azaz ha egyszer beállítja a pontos időt, képes-e azt tartani? Ennek az időmérésnek a minőségét csak a kvarc tulajdonságai befolyásolják: a felbontás (resolution), a pontosság (accuracy), a precizitás (precision) és a stabilitás (stability). Oszcillátorok esetén a következőt jelentik ezek a fogalmak:

- Felbontás (resolution)

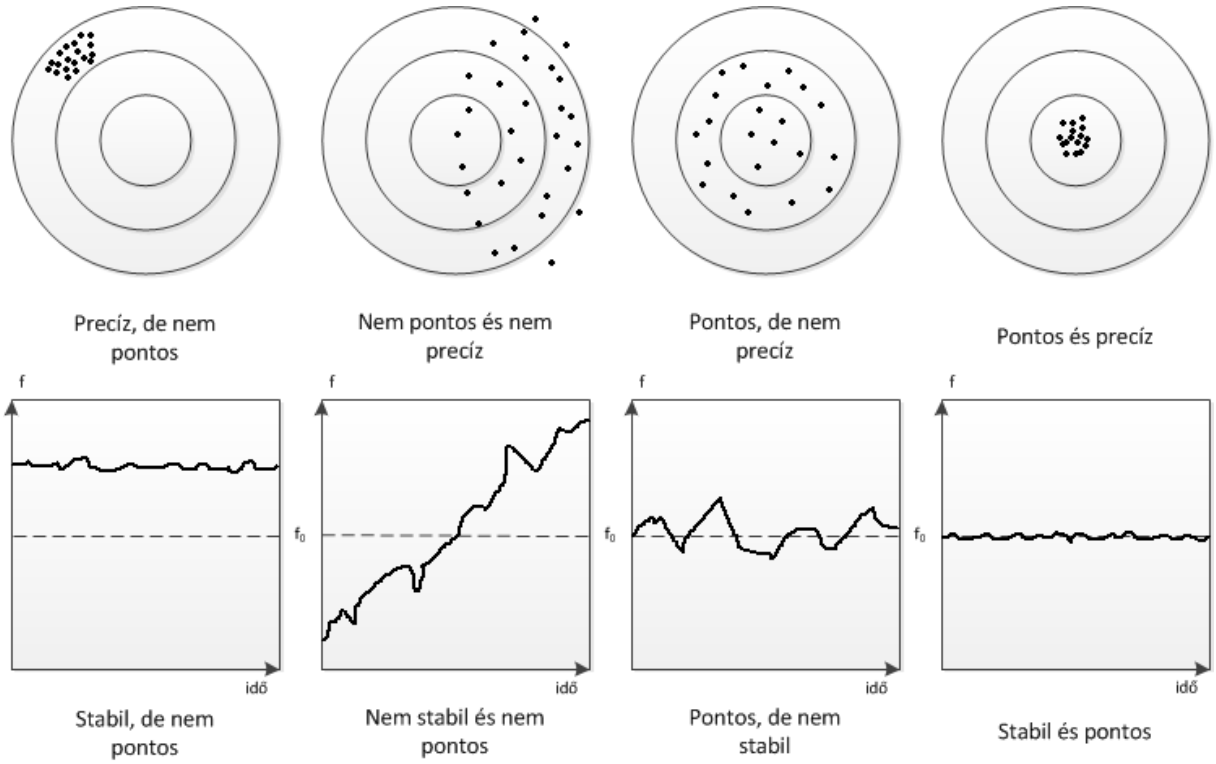
  A felbontás azt jelenti, hogy mekkora az a legkisebb idő, amit mérni képes az óra. Ez a névleges frekvenciából számolható: $t_{min} = 1/f_n$
- Pontosság (accuracy)

  Pontosnak mondunk egy mérést, ha a mért érték megfelel a valóságnak. Esetünkben pontos az óránk, ha a váltások valóban t_{min} időkülönbséggel követik egymást
- Precizitás (precision)

  Precíz az óránk, ha az ütések között eltelt időértékek szórása kicsi, azaz minden ütés között ugyanannyi idő telik el
- Stabilitás (stability)

  Ha az órának időben nem változnak a paraméterei, stabilnak mondjuk.

A fogalmak értelmezésében segít az 5. ábra.



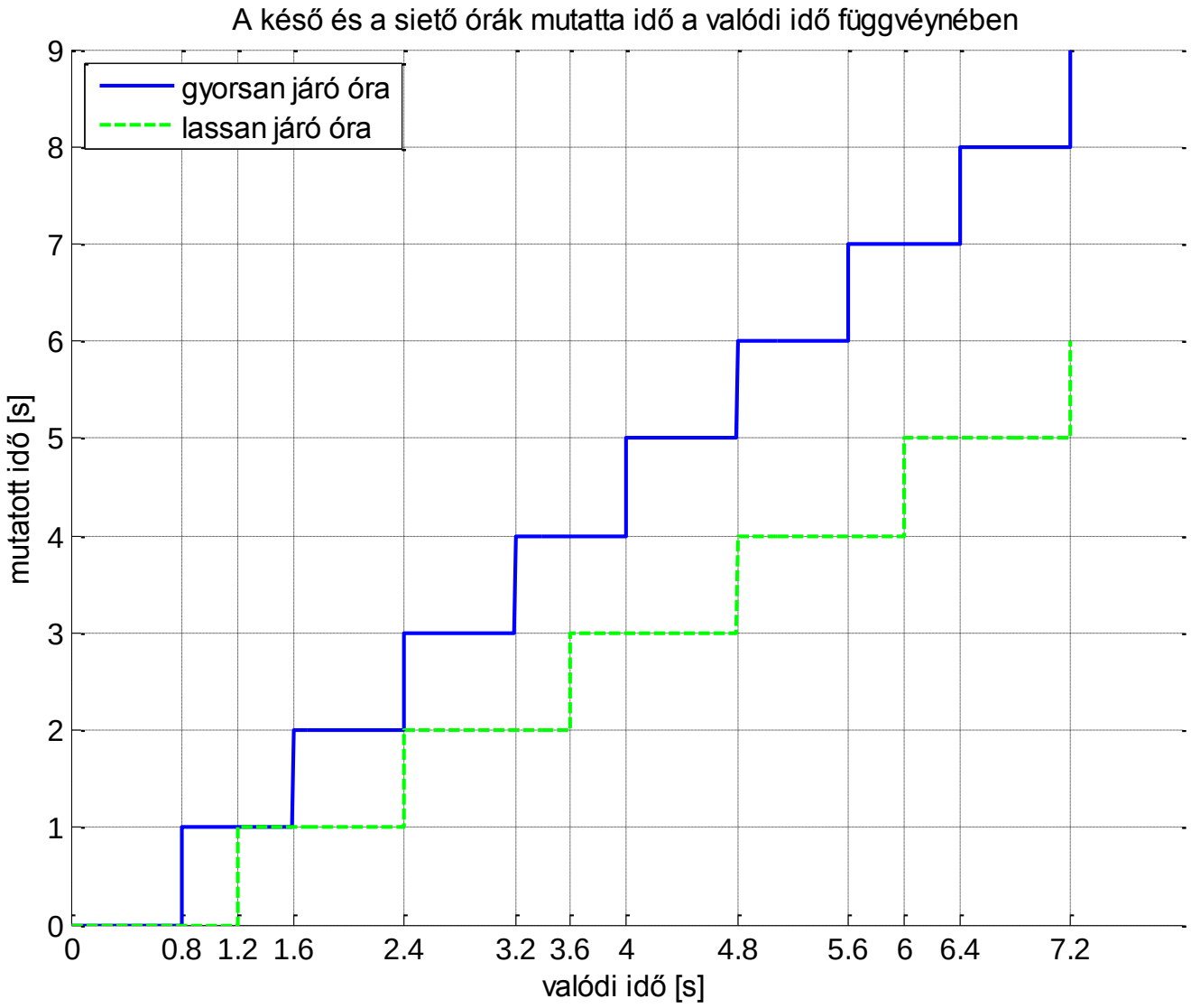
5. ábra: A pontosság, precizitás és stabilitás ábrázolása [15]

Egy kristály oszcillátor esetében a felbontás a névleges frekvenciából adódik, ha ez elég nagy, akkor a szükséges felbontás rendelkezésre áll. A pontosságot a gyártó egy tartományon belül garantálja, egy tipikus érték 100 MHz névleges frekvencia esetén a +/-50 ppm, azaz a valós frekvencia a névelegestől ennyivel térhet el. Ez az érték adott környezeti körülmények között állandó, tehát precíz, de pl. hőmérsékletváltozásra a valós frekvencia is megváltozhat. A kvarcok stabilitása folyamatos működés mellett is igen jó, de a környezeti jellemzők nagymértékben és gyorsan változhatnak az időben, így ezen keresztül a precizitás befolyásolja a stabilitást is [16].

Ezek alapján nem garantálható, hogy a belső óra valóban a névleges frekvenciából adódó periódusidőt méri. A valós periódusideje pedig változhat a környezeti hatásokra reagálva. Ha figyelembe vesszük, hogy különböző eszközök különböző helyeken monitoroznak, ahol különböző pl. a hőmérséklet, akkor könnyen előfordulhat, hogy a két eszköz órája folyamatosan elcsúszik nem csak a pontos időhöz, hanem egymáshoz képest is. Meg kell tehát vizsgálni, hogy a kereskedelmi forgalomban kapható kristály oszcillátorok paraméterei mellett ez mekkora problémát okozhat, és hogyan lehet azt kivédeni. Ehhez a következő fejezetben áttekintem, hogyan lehet két, kismértékben eltérő frekvenciájú órát összehasonlítani.

2.3.2 Órák összehasonlítása szemléletes példákon keresztül

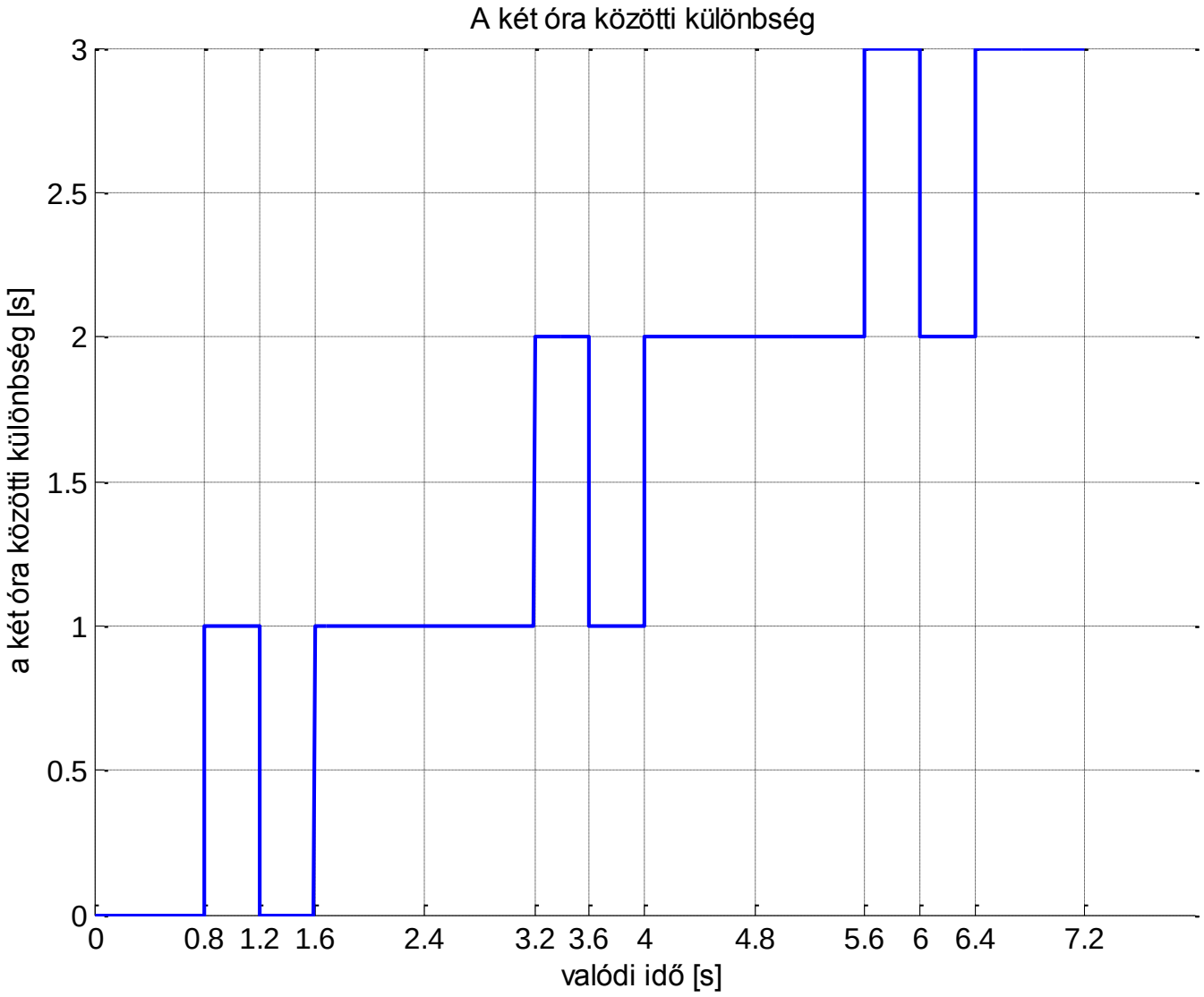
Az egyszerűbb számolás és ábrázolás érdekében legyen most a névleges periódusidő 1 s, a késő órának 0,8 s, a sietőnek 1,2 s a periódusideje. Tegyük fel, hogy ezeknek valamikor egy pillanatban következik a felfutó éle. Ezt vegyük az időtengely 0 pontjának, ettől kezdve vizsgáljuk a periódusidők kezdetét. Ebben segít a 6. ábra.



6. ábra: A késő és a sietős óra ideje

Azt látjuk, hogy a két órának először 2,4 s-nál van egy időben felfutó éle, azaz ekkor a sietős óra „megelőzi" a lassú órát. Ezután már mindegy mikor hasonlítjuk össze a kettőt, a sietős *egy ütéssel legalább előbb jár* a lassúnál. Eddig a pontig lehet találni olyan intervallumokat, amikor van eltérés a két óra között és olyanokat, amikor nincs. A [0;0,8] intervallumon mindkettő a kezdő értéket, 0 s-ot fog mutatni. A [0,8;1,2] tartományban a sietős órán már 1 s lesz az idő, míg a lassún 0 s. Ez 1 s különbség, pedig a két óra periódusideje között a különbség 1,2 – 0,8 = 0,4 s. Mivel a névleges periódusidő – így az órák felbontása is – 1 s, a két óra mutatta idő különbsége is csak 1 s egész számú többszöröse lehet. [1,2;1,6] tartományon újra ugyanazt, 1 s-ot fog mutatni mindkét óra, tehát a különbség eltűnik. 1,6 s-nál aztán másodszorra is lép egyet a sietős óra, ezzel újra 1 s különbséget produkálva a két óra között. [1,6;2,4] tartományon ez a különbség marad fenn, amikor 2,4 s-nál mindkettő óra vált, azaz a sietős órán 3 s, a késő órán 2 s lesz látható. [2,4;3,2]-n ezek az értékek maradnak érvényben, majd 3,2-nél újra a sietős óra vált, ezzel már 2 s különbséget produkálva, mely fenn is marad a [3,2;3,6] intervallumon. 3,6 s-nál a késő óra behoz 1 s-ot (a különbség újra 1 s), de 4

s-nál a siető óra újra vált, a különbség újra 2 s lesz. 4,8 s-nál pedig ismét egyszerre váltanak, a különbség 2 s marad. A két idő közti különbség alakulását a 7. ábra mutatja.



7. ábra: A két óra közötti különbség

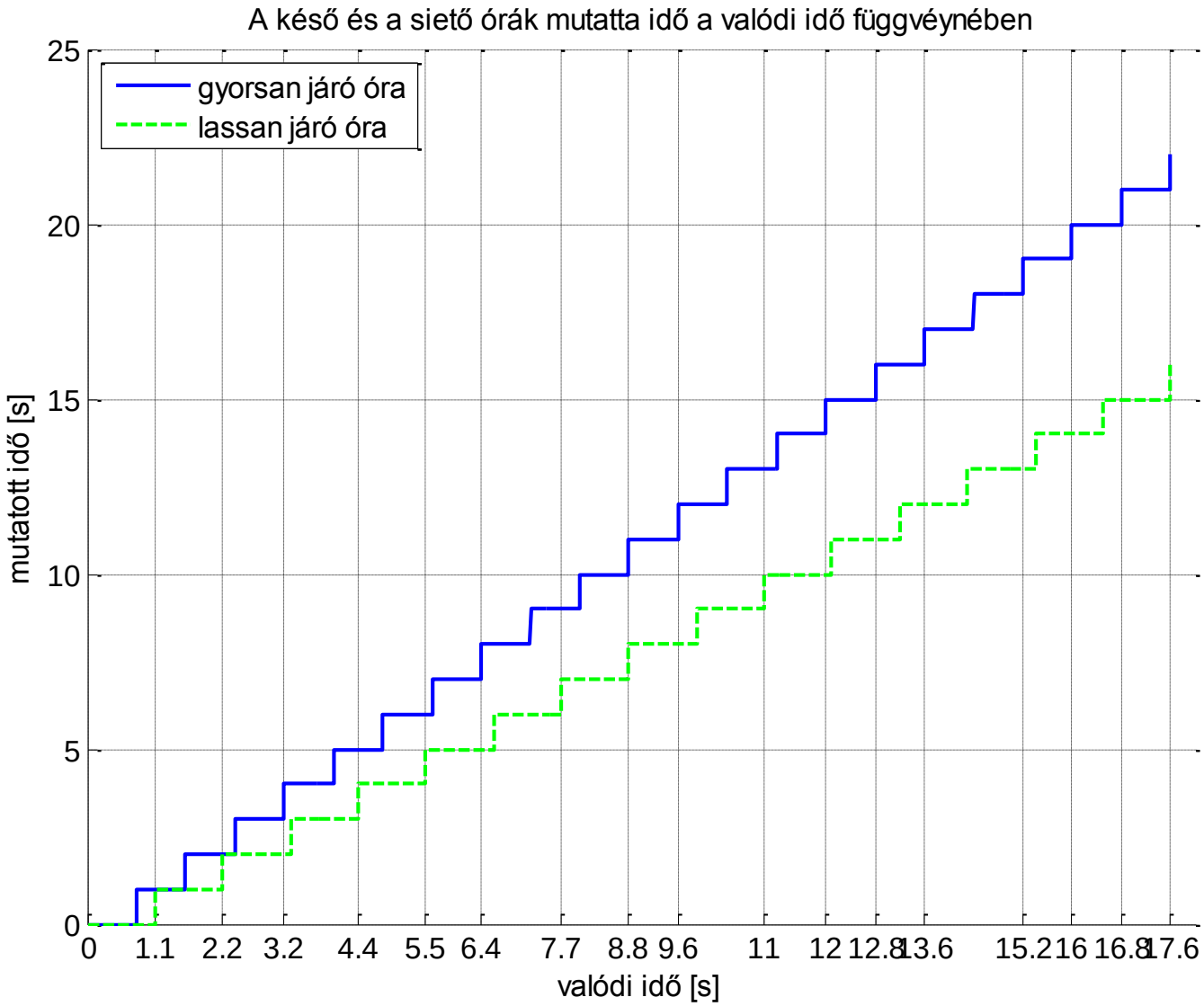
Jól látható a különbség ingadozása, azonban észrevehető egy fajta periodikusság. A forma 2,4 s után ismétlődik, csak eggyel eltolva felfelé a függőleges tengely mentén. Ez a 2,4 s persze nem véletlen, a 6. ábra grafikonja azt mutatja, hogy 2,4-nél és többszöröseinél van egyszerre a két óra felfutó éle. Ezt az értékét ki is lehet számítani, ehhez tekintsük a két órát számtani sorozatnak:

A sietős óra ideje: $a_{n+1} = n \times 0{,}8$ A késő óra ideje: $a_{k+1} = k \times 1{,}2$

Annak kiszámításához, hogy mikor lesz egyszerre a két órának felfutó éle, egyenlővé kell tenni a két sorozatot Ha van olyan (n, k) pár, amire egyenlő a két oldal, akkor a gyors és a lassú óra n. illetve k. ütése ugyanabban az időpillanatban történik. Így a következőt kapjuk:

$n \times 0{,}8 = k \times 1{,}2$ amiből $n/k = 1{,}2/0{,}8 = 3/2$.

Az egyenletben n és k a két óra eltelt periódusainak számát jelöli, ezért mindkettő csak egész szám lehet. Ez azt jelenti, hogy a gyorsabban járó órának 3, a lassabban járónak 2 órajel-periódusonként ugyanakkor lesz felfutó éle, mint a másiknak. Ha a lassan járó óra periódusidejét 1,1 s-ra csökkentem, a 8. ábra mutatja az órák mutatta idő alakulását.



8. ábra: A 0,8 s és az 1,1 s periódusidejű órák ideje

Az előbbi gondolatmenet szerint a közös felfutó élek előfordulása itt is periodikus. Ha n a gyorsabb, k a lassabb óra periódusainak számát jelöli, akkor:

$$n/k = 1{,}1/0{,}8 = 11/8$$

Időben ez $11 \times 0{,}8 = 8 \times 1{,}1 = 8{,}8\ s$, azaz 8,8 s után van az első közös felfutó él. A két óra idejének grafikonja is ezt támasztja alá.

A fenti két példa alapján azt mondhatjuk, hogy ha két különböző periódusidejű órának létezik egyszerre felfutó éle, akkor ez adott időközönként bekövetkezik. A következőkben az alábbi jelöléseket fogom használni:

- T_k a lassabb óra periódusideje,
- T_n a gyorsabb óra periódusideje,
- T_0 a néveleges periódusidő,
- k a lassabb óra periódusainak a száma,
- n a gyorsabb óra periódusainak a száma.

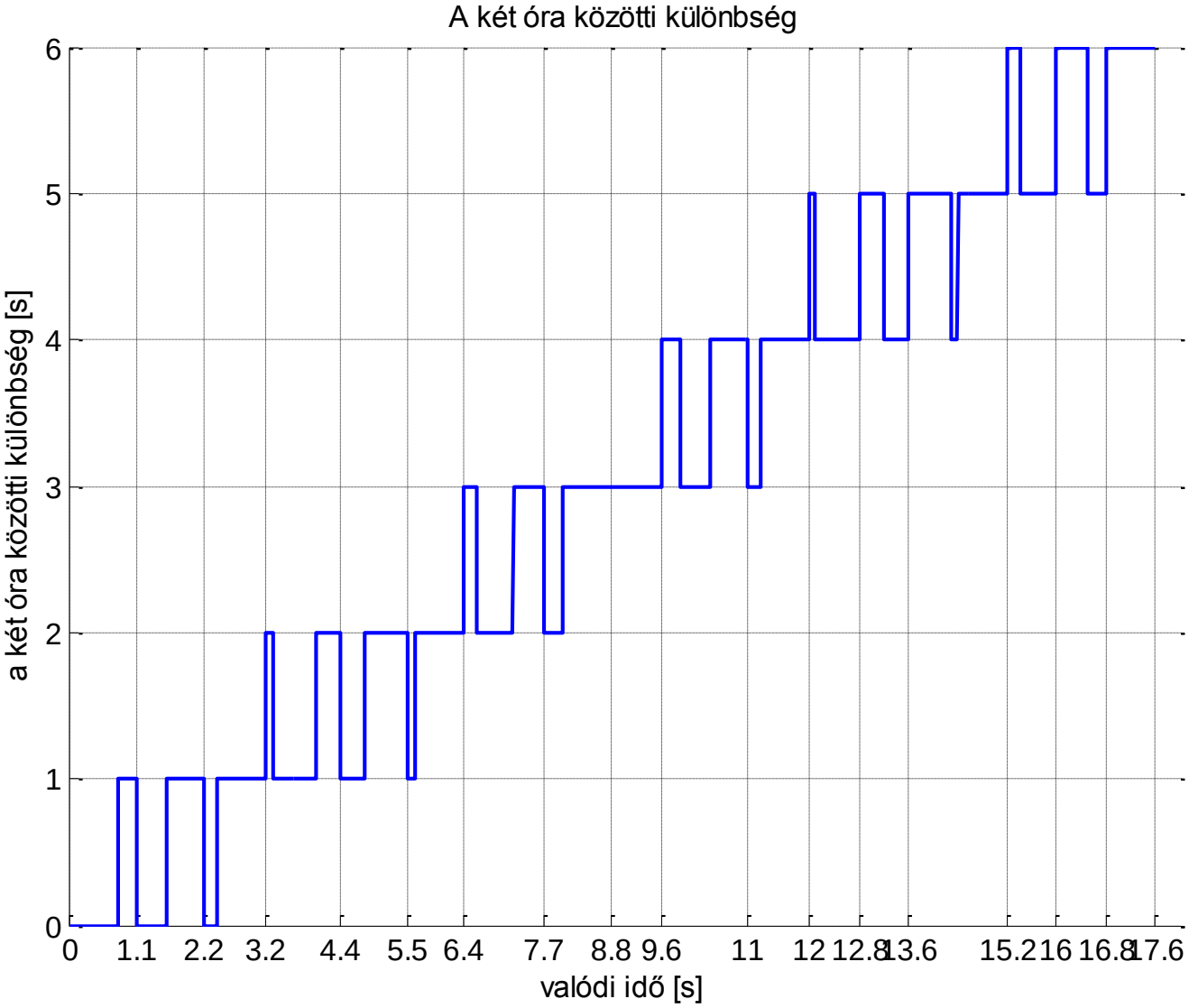
A közös felfutó élek ideje a két periódusidőből számítható:

$$n/k = T_k/T_n$$

ahol n és k pozitív egész számok. Ennek az egyenletnek (racionális T_k és T_n esetén[3]) végtelen sok (n, k) pár megoldása van, ezek közül a legkisebb legyen (n_1, k_1), a következő (n_2, k_2), és így tovább. Az első közös felfutó él idejét az $n_1 \times T_n = k_1 \times T_k$ érték adja meg, a második idejét az $n_2 \times T_n = k_2 \times T_k$, az i. idejét az $n_i \times T_n = k_i \times T_k$. Ez alatt a két óra különbsége ugrásokkal ugyan, de lineárisan nő. A különbség persze az órák felbontásából adódóan korlátozott, a T_0 névleges periódusidő többszöröse.

A 0,8 s és az 1,1 s periódusidejű órák közti hibát a 9. ábra mutatja. Ha összehasonlítjuk ezt a 7. ábra grafikonjával, akkor azt látjuk, hogy míg az előbbinél mindig eggyel nőtt a lehetséges hiba a közös felfutó élekkel, az utóbbinál az első közös felfutó él előtt (8,8 s) három kisebb szakaszra bontható a hiba alakulása. Ezeken a tartományokon belül a különbség vagy egy bizonyos Δ vagy Δ+1. Ha ezeknek a kis szakaszoknak a kezdetét és a végét ki lehet számítani a periódusidőkből, akkor két óra összehasonlításakor biztosat tudunk mondani egy adott időpontban a különbségükről (Δ vagy Δ+1).

[3] T_k és T_n egy-egy oszcillátornak a valós periódusideje, melyet méréssel határozhatunk meg. A mérőműszernek (pl. oszcilloszkóp) van egy adott felbontása, tehát a mért értékek véges számú tizedes jegyig ábrázolt számok, amik mindig racionálisak. Ez a feltétel tehát a valóságban teljesül.



9. ábra: A 0,8 s és az 1,1 s periódusidejű órák különbsége

Legyen a 8. ábra és a 9. ábra két kitüntetett időpontja a 3,2 s és a 6,4 s. Az első jelzett időpontnál van a siető órának a 4. felfutó éle. A késő órának viszont eddig még csak 2 felfutó éle volt, azaz a két óra között most először már 2 s a különbség. Eddig ez a hiba 0 vagy 1 s volt, innen kezdve 1 vagy 2 s. A következő ilyen jelölt időpont 6,4 s, ekkor lesz először a két óra közti különbség 3 s. A 3,2 s és a 6,4 s közötti időtartamról tehát elmondhatjuk, hogy vagy 1 és 2 s között van az órák különbsége. Látható, hogy a különbségek alakulása ilyen intervallumokra bontható. Legyenek ezeknek a tartományoknak a szélei $t_1, t_2, .., t_i$, az ezekhez tartozó periódusok száma pedig $n_1, n_2, .., n_i$ és $k_1, k_2, .., k_i$ (a gyors és a lassú óra periódusainak száma). Egy adott időpontban az órák mutatta idő $n_i \times T_n$ és $k_i \times T_k$ Az i. időpontban az teljesül, hogy a lassú óra i-vel kevesebbet ütött, mint a sietős, de időben mégis többet mutat:

$$(n_i - i) \times T_k \geq n_i \times T_n$$

Ezt n_i-re rendezve a következőt kapjuk:

$$n_i \geq i \times \frac{T_k}{T_k - T_n}$$

A betűk jelentése:

- T_k a lassabb óra periódusideje,
- T_n a gyorsabb óra periódusideje,
- T_0 a névleges periódusidő,
- n_i a siető óra periódusainak száma, ami után a két óra különbsége biztosan nagyobb, vagy egyenlő, mint $i \times T_0$

Ha megvan a legkisebb lehetséges n_i és n_{i+1} érték, abból számítható t_i és t_{i+1}. Ekkor biztosan állíthatjuk, hogy a két óra közötti különbség t_i és t_{i+1} időintervallumban $i \times T_0$ vagy $(i+1) \times T_0$.

Az eddigi számítások abból a feltételezésből indultak ki, hogy a két órának valamikor létezik egyszerre felfutó éle és onnan indulnak a mérések. Ez segített eljutni a fenti kifejezéshez, mellyel meghatározhatóak két óra hibájának lehetséges értékei egy adott időpontban. Ha azonban a két órának nincs egyszerre felfutó éle a mérés kezdetén, akkor a kifejezés helytelen eredményt ad, tehát valahogy bele kell venni a két óra felfutó éle közti különbséget. Legyen a referencia pont a siető óra felfutó éle, ekkor a lassú óra felfutó éle ehhez képest $\pm d$ idővel késik vagy siet. Az előbbi egyenlet tehát ezzel módosul:

$$(n_i - i) \times T_k + d \geq n_i \times T_n$$

amiből n_i-re a következőt kapjuk:

$$n_i \geq \frac{i \times T_k - d}{T_k - T_n}$$

Ezzel számításba vettük, hogy a két órajel nem feltétlenül van szinkronban a mérés kezdetén. A d értéke a $\Delta d = \left[-{}^{T_k}/_2 ; +{}^{T_k}/_2\right]$ tartományon belül változhat, így $d = 0 \pm {}^{T_k}/_2$, ezzel a kifejezés egy $n_i(d)$ kimeneti függvényre módosult, aminek érzékenysége:

$$c = \frac{\partial n_i}{\partial d} = \frac{1}{T_k - T_n}$$

A d negatív előjelét elhagytam az értékkészlet alapján. A kimeneti függvény megváltozása a d megváltozására:

$$\Delta n_i = c \times \Delta d = \frac{{}^{T_k}/_2}{T_k - T_n}$$

A kifejezés relatív hibája ezután:

$$\frac{\Delta n_i}{n_i} = \frac{T_k/_2}{T_k - T_n} \times \frac{T_k - T_n}{i \times T_k} = \frac{1}{2 \times i}$$

Látható, hogy a relatív hiba annál kisebb, minél nagyobb az i változó értéke, azaz minél nagyobb időkülönbségre használjuk a kifejezést. Ezek alapján jogos a d csúszás nélküli összefüggés használata előzetes számításokra, amennyiben szem előtt tartjuk, hogy a kapott n_i értékek (a két óra különbségétől függően) kis mértékben eltérhetnek a valóságban mérhető eredményektől.

Ezeket az összefüggéseket a GPS alapú időmérés módszerének kialakításához (3.2.1 fejezet) használtam fel a gyakorlati megvalósítás során.

3 A megvalósítandó feladat – a WCLOCK bővítőkártya

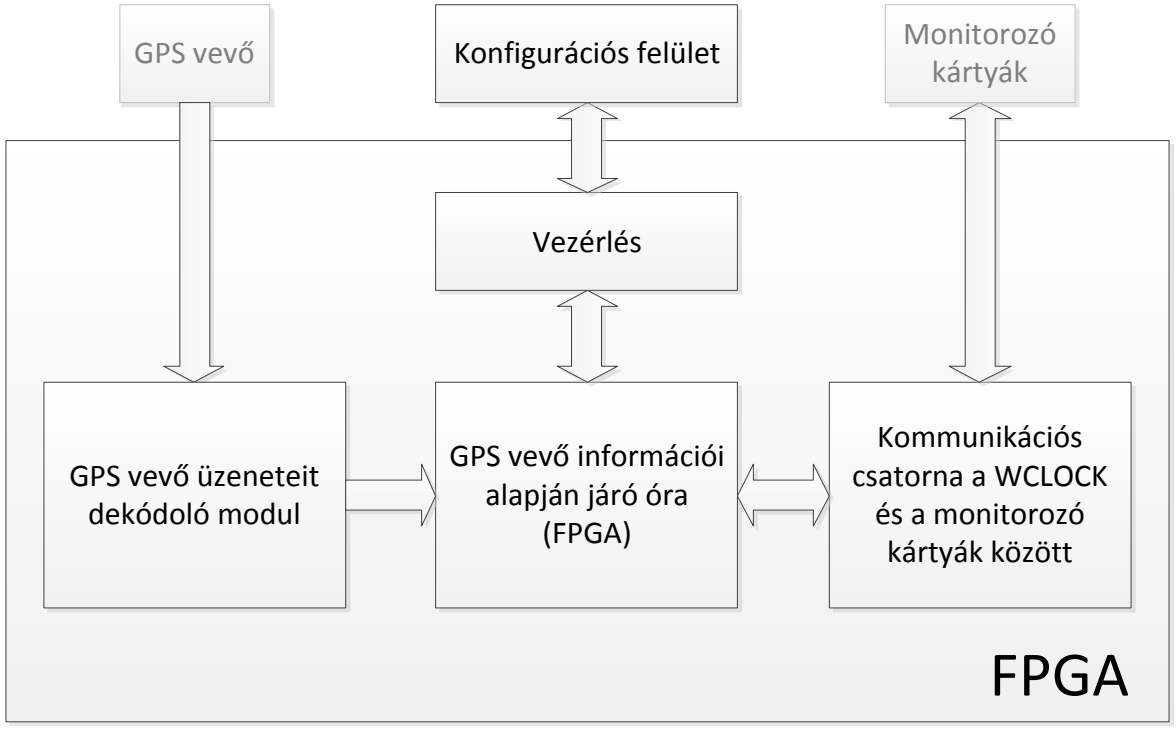
A pontos időinformációt a monitorozó egységek számára egy célhardver, a WCLOCK bővítőkártya fogja biztosítani, melynek központi része egy FPGA. A következőkben áttekintem, hogy milyen egyéb hardverelemek szükségesek a működéshez, valamint az FPGA-nak milyen fő feladatai lesznek a működés során.

3.1 A hardver – a működéshez szükséges alegységek

A feladat tehát a 2.3 alapján két fő részre bontható: az időpecsét szinkronizálása a GPS időhöz, valamint két PPS impulzus között egy belső óra működtetése. Ez az óra szolgáltat pontos időt a monitorozó egységeknek akkor is, amikor éppen nem áll rendelkezésre GPS jel. Ez utóbbi feladatot teljes egészében az FPGA látja el, azonban kapcsolatot kell teremteni közte és a GPS vevő között. Szükség van még valamilyen kommunikációs csatornára a monitorozó kártyák és a WCLOCK között, valamint meg kell oldani a WCLOCK külső vezérlését, konfigurációját. A főbb funkciók tehát:

- a GPS vevő üzeneteit dekódoló modul (GPS - belső óra kapcsolata),
- a GPS vevő információi alapján járó óra,
- kommunikáció a WCLOCK és a monitorozó kártyák között,
- konfigurációs felület.

Ezen alegységek kapcsolatát a 10. ábra mutatja. A halványabb színnel ábrázolt GPS vevő és monitorozó kártyák csak az érthetőség kedvéért vannak feltüntetve. Ezeken kívül mindegyik blokk egy-egy funkciót jelöl, amiknek a tervezése és az implementálása az én feladatom volt.



10. ábra: A WCLOCK működéséhez szükséges funkciók, és azok kapcsolatai egymással

3.1.1 A GPS modul üzeneteinek felhasználása

A legtöbb GPS modul UART vonalon (adatkapcsolati réteg) kommunikál a külvilággal SiRF bináris protokollt [17] vagy NMEA-t [18] használva. Mindkét esetben egyszerűen ki lehet nyerni FPGA-n belül a PPS jel vételének idejét. A fizikai réteg leggyakrabban RS-232, esetleg RS-422 hosszabb kábelszakaszok esetén, így szintillesztés után direkt az FPGA-ba vezethető a modul jele.

Ez a két szabvány két különböző feszültségszint-rendszert ír le, ráadásul az RS-232 sokszor TTL jelszintet használ, jellemzően beágyazott rendszerekbe szánt GPS vevőknél fordul elő ilyen eset. A WCLOCK lehetséges felhasználási helyei között könnyen elképzelhető olyan, ahol nincs szabad rálátás a műholdakra (pl. egy szerverterem), ezért kültéri antennát feltételezve fel kell készíteni a kártyát a szélesebb feszültségtartományok FPGA-hoz való továbbítására. A kültéri antennák többsége (figyelembe véve az esetleges hosszú kábelszakaszt, hozzávezetést) RS-422 szabványt használ, azonban mégis praktikus RS-232 szabványra felkészíteni a kártyát, hiszen így egyszerű debug interfész alakítható ki, melyre pl. HyperTerminal programmal is csatlakozhatunk. A piacon számos olcsó RS-422/RS-232 átalakító kapható, így ezzel megoldható az ilyen szabványt használó vevők csatlakoztatása is.

3.1.2 Egy ipari igény: NTP

Számos helyzet elképzelhető, amikor nincs lehetőség GPS antenna elhelyezésére. Elég pl. egy ablaktalan pincehelységre gondolni, ahol nem hosszútávra telepített mérőberendezés telepítéséről van szó, hanem egy bemutatóról, próbamérésről, stb. Ilyenkor nem gazdaságos egy kültéri antenna jelét eljuttatni a helységbe, viszont időpecsétre így is szükség van. A GPS-nél jóval pontatlanabb, de elterjedten használt ipari gyakorlat erre egy NTP szerver idejének felhasználása. Korábbi munkám során már kifejlesztettem egy FPGA-n futó NTP klienst, így ezt csak át kellett emelnem a WCLOCK moduljai közé. Mivel ennek működését bemutattam a 2010-es TDK során [19], ezért ezután csak hivatkozni fogok rá. Az NTP-hez szükség van egy hálózati interfészre, ami az FPGA-hoz csatlakozik, ez egy fontos követelmény a konkrét hardverrel szemben.

Figyelembe véve az ipari környezetet, a WCLOCK kártyának Ethernet porttal kell rendelkeznie ahhoz, hogy NTP kliensként működjön. A szolgáltatók szervertermeiben, gerinchálózathoz való csatlakozási pontjaiban sok esetben csak 1 vagy 10 Gbps Ethernet interfészek érhetőek el. Utóbbinak jelentősen nagyobb az erőforrásigénye, így 1 Gbps sebességű Ethernet interfészt kapott a kártya.

3.1.3 Kapcsolat a monitorozó kártyákkal

A kártyán belül rendelkezésre álló időpecsétnek valahogy át kell jutnia a monitorozó kártyához. Erre számos megoldás adódik, azonban néhány fontos követelménynek teljesülnie kell:

- az időpecsét átvitele mindig azonos ideig tartson,
- a monitorozó kártya mindig ugyanakkora idő alatt dolgozza fel az időpecsétet,
- valamint mindig rendelkezzen érvényes időpecséttel.

Az AITIA monitorozó kártyái PCI-Express csatolóval rendelkező, FPGA alapú interfész kártyák [20][21], ezért az egyik lehetőség, hogy maga a WCLOCK is kommunikáljon valahogy a monitorozásra használt PC-vel, így szoftveresen át lehetne adni az időinformációt. Ezzel a megoldással a jelentősebb problémák a hosszú ideig tartó feldolgozás, az előre nem látható késleltetés és a monitorozás miatt amúgy is terhelt PCI-Express csatlakozás további terhelése a monitorozó kártya felé.

Egy másik lehetőség az időpecsét hálózaton való továbbítása. Az NTP miatt a WCLOCK rendelkezik hálózati interfésszel, így akár a monitorozott alhálózaton, akár egy erre a célra kialakított hálózaton lehetőség lenne pl. szabványos NTP keretekben továbbítani az időpecsétet. Ebben az esetben az FPGA alapú csomaggenerálás és feldolgozás biztosítaná a konstans késleltetést, gyakorlatilag egy hardveres Stratum 1 NTP szerverként üzemelne a WCLOCK. Sajnos számos esetben a monitorozott hálózat forgalmába nem avatkozhatunk bele (passzív monitorozás), így mindenképpen dedikált portot kellene hozzárendelni a monitorozó kártyán az időpecsét fogadásához. Egy ilyen kártyán általában 2 vagy 4 hálózati interfész van, ezek közül igen nagy pazarlás lenne kizárólag az időpecsét vételére használni egyet.

A hálózati interfészen és a PCI-Express csatolón kívül egyetlen további csatlakozási lehetőség található meg az AITIA interfész kártyák mindegyikén, ez az ún. *Feature Connector*, mely egy 40 pines BERG típusú csatlakozó. Ez közvetlen összeköttetésben van a kártyákon található FPGA-val, így biztosítható az állandó késleltetés az átvitel és a feldolgozás során, és nem vesz el erőforrást más feladatoktól. Problémát jelenthet azonban az előbb felsorolt követelmények közül az utolsó, a monitorozó kártyán belüli időpecsét folyamatos érvényessége. A 2.1.3 alapján 61,6 ns időközönként érkezhetnek keretek, tehát legalább ilyen gyakran kellene frissíteni az időpecsétet. Vegyük pl. az NTP formátumát, ahol az időpecsét 64 bites (ez egyébként elterjedt minden hasonló felbontással rendelkező időpecsét esetén). A szükséges adatátviteli sebesség a WCLOCK és a monitorozó kártya között ez alapján:

$$64\ bit/61{,}6\ ns\ \approx 1{,}039\ Gbps$$

Ekkora sebességet ezen az összeköttetésen keresztül nem lehet stabilan biztosítani, azaz nem fordulhat a monitorozó a WCLOCK kártyához időpecsétért minden egyes keret beérkezésekor. Ez azt eredményezi, hogy a monitorozó kártyákon futó firmwaret két funkcióval szükséges kiegészíteni:

1) a WCLOCK kártyával való kommunikációt megvalósító modullal,
2) az eddig használt belső óra helyett egy, a WCLOCK időpecsétjét felhasználó órával

Az 1) modult mindenképpen szükséges integrálni a meglévő firmwarebe, hiszen a monitorozó kártyának kommunikálnia kell a WCLOCK kártyával. A 2) módosítás pedig nem más, mint a WCLOCK-ba tervezett óra átültetése a monitorozókba, tehát ez sem jelent különösebb nehézséget. Erőforrás szempontjából sem nagy terhelés a monitorozóknak, hiszen eddig is működött bennük egy belső óra, ennek a cseréjéről van csak szó.

Azzal, hogy nem szükséges folyamatosan küldözgetni az időpecsétet, ez az összeköttetés alkalmassá válik egyéb szervizüzenetek továbbítására és a konfiguráció átvitelére is. Nem kell tehát újabb kommunikációs port a WCLOCK kártyára, annak beállítását és külső vezérlését elláthatja maga a monitorozó kártya. E nélkül szükség lenne pl. egy PCI vagy PCI-Express interfészre a WCLOCK kártyán, ami jelentősen drágítaná az eszközt és bonyolultabbá tenné a firmwaret (főleg PCI-E esetén). Így ezeket el lehet kerülni amellett, hogy drivert sem kell írni a kártyához, annak semmilyen számítógépes kapcsolata nem lesz.

3.1.4 Tápfeszültség, fizikai kialakítás

A WCLOCK kártyát alapvetően PC-ben történő elhelyezésre terveztem monitorozó PCI-Express interfészkártyák mellé. Így annak ellenére, hogy a működéséhez nem szükséges ilyen interfész, az elhelyezés és tápellátás szempontjából praktikus PCI vagy PCI-Express kártyaként kialakítani az

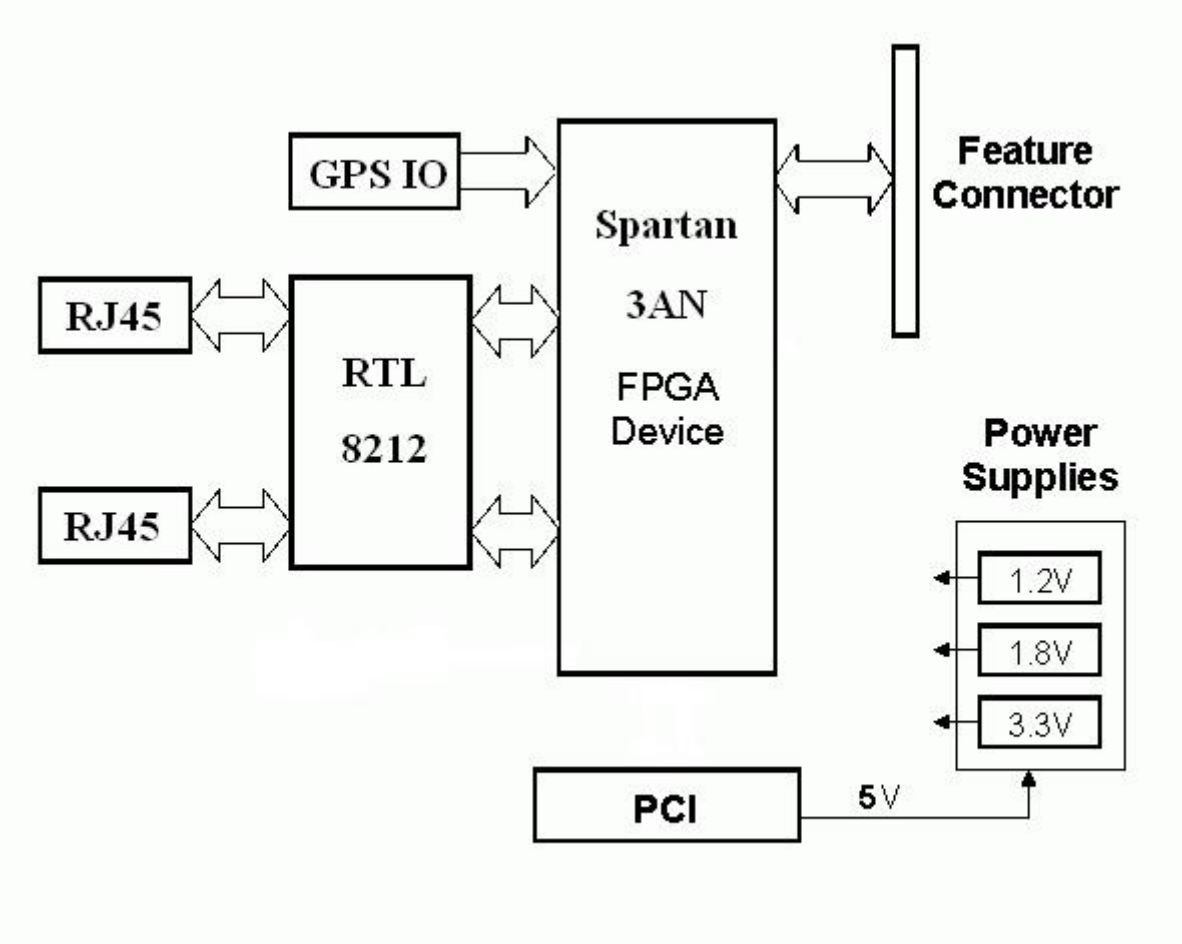
eszközt. Meghajtó áramkör nem szükséges az FPGA mellé, így nem drágítja a kártyát, csak a tápfeszültséget kapja innen. Mivel a monitorozók PCI-Express kártyák és a hozzájuk használt számítógépekben általában csak kettő ilyen interfész van, célszerű egy PCI portot elhasználni a tápfeszültség biztosítására.

3.1.5 Összefoglalás, blokkdiagram

A WCLOCK feladatainak áttekintése után fel lehet sorolni a konkrét hardverrel szemben támasztott követelményeket:

- illesztés RS-232 vonalon a GPS modulhoz
- 1G Ethernet interfész az NTP funkcióhoz
- 40 pines BERG típusú csatlakozó a monitorozó kártyákkal való kommunikációhoz
- az összes feladatot ellátni képes FPGA
- PCI csatlakozó és a szükséges tápfeszültségeket előállító alkatrészek

Ezek alapján felrajzolható a WCLOCK kártya blokkvázlata (11. ábra).



11. ábra: A WCLOCK kártya blokkvázlata

Ennek kialakításában jelentős szerepem volt, a konkrét NYÁK tervet Horváth György tanszéki mérnök kollégám készítette el.

3.2 A firmware – az FPGA főbb feladatai

3.2.1 Pontos idő biztosítása GPS használatával

A következőkben a GPS és egy általános oszcillátor korlátait figyelembe véve megvizsgálom, hogy milyen módszerek képzelhetők el a mindig kellően pontos időinformáció biztosítására. A számítások alapja egy 100 MHz-es oszcillátor +/- 50 ppm frekvencia stabilitással, ilyen paraméterű kristályok széles körben elérhetők a piacon (pl. EuroQuartz 3EQHM57-ET-100.000R-C1.5 [22]). A helyes módszer kiválasztásához felhasználom a 2.3.2 fejezet eredményeit.

Egyszeri szinkronizálás

Ebben az esetben a szinkronizáló modul *egyszer* olvas be pontos időpecsétet az időforrástól, azt átadja az időmérő modulnak, ami innentől kezdve jár, a hardver oszcillátorának megfelelő felbontású és pontosságú időinformációt szolgáltatva a kimenetén. Ha megvizsgáljuk az oszcillátor esetleges hibáját, láthatjuk, hogy ez az eljárás nem biztosít ns pontosságot. A példaként használt oszcillátor esetében a gyártó azt garantálja, hogy a valós frekvencia eltérése a névelegestől nem lesz nagyobb, mint:

$$50 \times (10^8/10^6) = 5000\ Hz$$

Ebből következően 99,995 ≤ f ≤ 100,005 MHz a lehetséges kimeneti frekvencia. A névleges értékből adódó periódusidő 10^{-8} s = 10 ns, azaz egy órajelre a belső óránkat ennyivel állítjuk előre. Ha a valós frekvencia a megengedett intervallum alján van, a lassú óra periódusideje:

$$T_k = 1/(99{,}995 \times 10^6) \approx 10{,}000500025\ ns$$ [4]

Ez azt jelenti, hogy egy órajel alatt 0,500025 ps-ot késik az óra. A névleges frekvencia alapján 10^8 órajel után fog egy mp-et állítani az órán, de ebben az esetben ekkorra 10^8 x 10,000500025 ns = 1,0000500025 s telt el valójában, azaz az óra 1 mp alatt 50,0025 µs-ot késett, ami közel *ezerszerese* a csomagok lehetséges érkezési gyakoriságának (61,6 ns). Ha az oszcillátor valós frekvenciája az intervallum felső szélén van, akkor hasonló gondolatmenettel 1 mp alatt az óra 49,9975 µs-ot siet, azaz a siető óra peiódusideje:

$$T_n = 1/(100{,}005 \times 10^6) \approx 9{,}999500025\ ns$$

Ha van két teljesen egyforma monitorozó eszköz ugyanolyan oszcillátorral, lehetséges, hogy az egyik 100,005 MHz-en, a másik 99,995 MHz-en jár. Tegyük fel, hogy egyszerre szinkronizál a két eszköz, egyszerre kezdenek el monitorozni és valamikor a két eszköz adatbázisa összefűzésre kerül. Az előző fejezet alapján ki lehet számolni, mennyi idő kell ahhoz, hogy az összefűzéskor felborulhasson a csomagok közti sorrend. Ehhez az kell, hogy a két óra közti különbség legalább 61,6 ns legyen, ami a néveleges periódusidő 6,16-szorosa. A két óra csak a periódusidő egész számú többszörösével térhet el egymástól, így azt kell kiszámolni, hogy a különbség mikor lesz minimum 70 ns, azaz 7-szerese a periódusidőnek:

$$n_i \geq i \times \frac{T_k}{T_k - T_n} = 7 \times \frac{10{,}000500025}{10{,}000500025 - 9{,}999500025} \approx 70003{,}85$$

[4] Ez kerekített érték, femtoszekundum (10^{-15}) nagyságrendig ábrázolva

Azaz ha a szinkronizáció után eltelik 70004 periódusa a siető órának, már biztosan akkora lesz a különbség a két monitorozó órája között, hogy összefésüléskor sérülhet a csomagok sorrendhelyessége. Ez időben kifejezve:

$$t = n_i \times T_n = 70004 \times 9{,}999500025 \approx 700{,}00499\ \mu s$$

Azt mondhatjuk tehát, hogy nem elegendő egyszer szinkronizálni a monitorozó egységek óráit, hiszen nem kell hozzá egy ms sem[5], hogy a közös adatbázisban két csomag más sorrendben szerepeljen, mint ahogy a valóságban megjelentek a hálózaton. Ez persze két olyan oszcillátor esetén igaz, amik a megengedett mértékben eltérnek egyik illetve másik irányban a névleges frekvenciától, ami ugyan valószínűtlen, de elméletileg lehetséges. Az olcsó és egyszerű órajelforrásoknak a pontossága nem elég jó ahhoz, hogy hálózatmonitorozás közben egyedül rájuk bízzuk az időpecsétek előállítását.

Periodikus szinkron

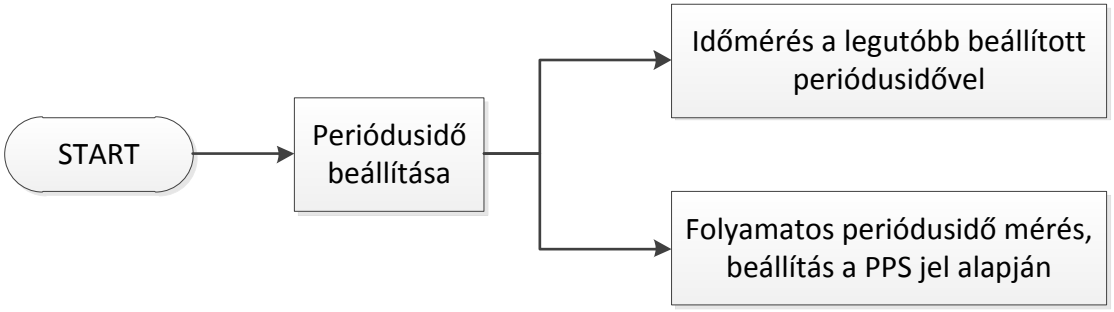
Láthattuk, hogy az egyszeri szinkron nem biztosít kellő pontosságot, az oszcillátorok pontatlanságát valahogy ki kell küszöbölni. Ezt legkönnyebben úgy tehetjük meg, ha a működés során periodikusan kérünk pontos időt a precíz forrásunktól és ez alapján mindig beállítjuk az óránkat. Ennek gyakoriságát leginkább két dolog szabja meg: az egyik a mi pontosság igényünk, a másik az időforrásunknak az a tulajdonsága, hogy milyen gyakran frissíti az időpecsétet a kimenetén. Hiába veszünk mintát tízszer egy másodperc alatt, ha a forrás csak másodpercenként frissíti a kimenetén az időt. Nézzük meg, hogy milyen gyakran kellene szinkronizálni a belső óránkat egy pontos forráshoz, ha az oszcillátorok legnagyobb csúszását feltételezzük. Az előző rész alapján kb. $700\ \mu s$ kell ahhoz, hogy két ilyen óra már biztosan eltérjen annyit, hogy egy minimális méretű csomag rossz sorrendben szerepelhessen a közös adatsorban. Ez azt jelenti, hogy az időforrásnak legalább ilyen időközönként frissíteni kellene az időinformációt a kimenetén, ami a GPS esetén sajnos közel sem teljesül.

Ha félretesszük az 1 PPS-ből adódó korlátot és csak a belső órák frissítésére koncentrálunk egy másik probléma is felmerülhet: az új időpecsét a saját időmnél korábbi, azaz vissza kell állítani az órát. Ebben az esetben megtörténhet, hogy egy időben később beérkezett csomag az adatbázisba egy előtte beérkezett csomag elé kerül, ami miatt nem lesz sorrendhelyes az adatbázis már egy monitorozó eszköz esetén sem. Ennek a jelenségnek az oka (az időforrás pontatlanságától eltekintve), hogy az oszcillátor frekvenciája nagyobb a névlegesnél, így gyorsabban jár az órám, mint kellene. Ezt a problémát persze ki lehet védeni azzal az igen egyszerű módszerrel, hogy az órámat maximum arra az időpontra állítom vissza, amikor a legutóbbi keret rögzítésre került. Így nem borulhat fel a rögzített keretek sorrendje, viszont nem lesz pontos az órám. Nagy hálózati forgalom esetén az a helyzet is előállhat, hogy soha nem tudom visszaállítani az órát működés közben a pontos időre, így az idővel egyre többet fog sietni. Az egyéni adatbázis ugyan szekvencia helyes marad, azonban teljesen valótlan időpecsétek kerülnek a csomagokra, ami összefűzéskor ismét valótlan sorrendet eredményezhet. Ezzel a problémával foglalkozni kell a belső időmérő modul megvalósításakor.

[5] Az itt használt kifejezés feltételezi, hogy a vizsgálat kezdetekor a két órajelnek egyszerre van a felfutó éle. A 2.3.2 fejezet alapján ez a feltételezés időben legfeljebb $\pm\left\lceil 70003{,}85 \times \frac{1}{2\times 7}\right\rceil \times 9{,}999500025 \approx 98{,}01\ \mu s$ eltérést jelenthet. Ez itt nem jelent számottevő változást, nem befolyásolhatja azt a következtetést, hogy a módszer nem megfelelő.

Periodikus szinkron, kompenzációval

Az oszcillátorok pontatlanok, így az egyszeri szinkronizáció nem elegendő, a periodikus szinkron pedig nem lehet elég gyakori az időforrás korlátai miatt. Az utóbbi állítást úgy is meg lehet fogalmazni, hogy az oszcillátorok nagy pontatlansága miatt túl gyakran lenne szükség friss időpecsétre. A fő problémát tehát a belső időmérő eszköz *pontatlansága* okozza, ezt kellene valahogy kompenzálni. A 2.3.1 fejezet alapján egy oszcillátor frekvenciája lehet pontatlan, de adott körülmények között (pl. hőmérséklet, nyomás) igen precízen tartja az adott értéket. Ezt kihasználva járható útnak tűnik az, hogy minden egyes eszköz oszcillátorának lemérjük a valós periódusidejét és az eszközön belüli időmérő egység minden órajelre ezzel az értékkel növelje a belső óra idejét. Ezt azonban nem tehetjük meg egyszeri (pl. oszcilloszkópos) méréssel, hiszen a környezeti változások hatására változik a valós frekvencia is. Ezt a mérést tehát rendszeresen, az eszközön belül kell elvégezni, és mindig az aktuális periódusidőt használva kell növelni a belső óra értékét (12. ábra).



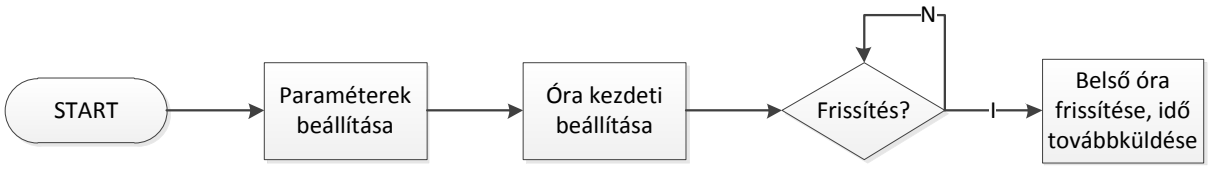
12. ábra: időmérés és kompenzáció a PPS jel alapján

Legyen a forrás egy GPS vevő 1 PPS kimenete, azaz másodpercenként kap a vevő egy impulzust, mely jelzi a következő másodperc kezdetét. A belső óra csúszása legyen az eddigiek alapján maximális, azaz 1 mp alatt késsen 50,0025 µs-ot. A működés kezdődjön egy impulzusra a PPS kimenetről, ekkor indul el az óra. Eleinte feltételezi, hogy 100 MHz a valós frekvencia, így egy órajelciklusra 10 ns-al növeli a belső időt. A következő impulzust 1 mp múlva kapja a vevő, de ő azt látja, hogy a saját órája csak 0,99995 s-ot mutat. Ez azt is jelenti, hogy nála 1 mp alatt nem 10^8, hanem csak 99,995 x 10^6 számú órajelciklus zajlott, így azt kell kiszámolnia, hogy valójában mennyi az órajel periódusideje. Elosztva az 1 mp-et az eltelt órajelciklusok számával megkapja a valós periódusidőt és innentől kezdve azzal növeli a belső óra értékét minden periódusra. Ha ezt minden PPS impulzusra kiszámolja és frissíti, akkor folyamatosan kompenzálja az oszcillátor pontatlanságát, még ha az változik is a környezeti hatások miatt.

Egy olyan időmérő modult kell tehát megvalósítani, ami képes beállítani magát a GPS ideje alapján, a PPS impulzusok hatására másodpercenként korrigálja az idejét (valahogy kiküszöbölve a visszaállítás problémáját) és folyamatosan változtatja az időmérés alapját a PPS jel segítségével. Mind a WCLOCK kártyán, mind a monitorozó kártyán ugyanannak az időmérőnek kell működnie. A monitorozóban lévő óra a kezdeti értéket a WCLOCK-tól kapja meg (ami pedig a GPS-től), onnantól viszont ugyanúgy járhat a PPS jel alapján. Ezért fontos szempont lesz a megvalósítás során, hogy a PPS jelet azonnal továbbítsa a WCLCOCK a monitorozó felé egy dedikált csatornán, hogy a két kártyán működő óra ugyanúgy járhasson.

3.2.2 A vezérlő modul

Ennek a modulnak a feladata az egész WCLOCK funkció vezérlése, egyszerűsített folyamatábráját a 13. ábra mutatja.



13. ábra: A vezérlő modul egyszerűsített folyamatábrája

Amikor a monitor kártyától a megfelelő információt megkapja a WCLOCK, a vezérlő beállítja a szükséges paramétereket. Ilyenek pl. az NTP-hez szükséges IP címek, a frissítés gyakorisága, a kimeneti időpecsét formátuma, stb. Ezután a belső óra kezdeti beálltása történik a GPS vagy az NTP szerver alapján attól függően, hogy mit állított be a monitorozó kártya. Miután ez megtörtént, a WCLOCK rendelkezik érvényes időpecséttel, melyet kérésre bármikor el tud küldeni a monitorozó kártyának. GPS esetén minden PPS jelre beállítja az órát a 3.2.1 részben leírtak alapján, NTP esetén a beállított időközönként fordul az NTP szerverhez pontos időért.

Emellett ennek a modulnak a feladata a folyamatos felügyelet, azaz bármilyen, a monitorozó kártyától érkező kérésre válaszolnia kell. Ilyenek lehetnek pl. az állapot lekérdezések, paraméter lekérdezések, stb.

3.2.3 A GPS üzeneteket feldolgozó modul

Ez fogadja a szintillesztőn át érkező UART kereteket és illeszti őket össze NMEA/SiRF üzenetekké. Ezután kinyeri a szükséges információkat az üzenetből (a PPS jel pontos idejét), majd továbbadja azt a belső órának. A modul hierarchikus felépítésű, azaz ha NMEA és SiRF között szeretnénk váltani, csak ezt a réteget megvalósító komponenst kell lecserélni, az UART vevő mindkét esetben dekódolja a 8 bites kereteket. Az üzenetekből kinyert időinformációt valamilyen gyakorlatban használt időpecsét formátumba kell átalakítani, ami alapján járhat a belső óra, és amit továbbíthat a monitorozó kártya felé.

3.2.4 Kapcsolat a monitorozó kártyával

A kapcsolatot két modul valósítja meg, az egyik egy szabványos USRT összeköttetést biztosít, a másik egy egyéni, 96 bites keretezést.

USRT modul

Az összeköttetéssel kapcsolatos elsődleges elvárás a hibatűrés és megbízhatóság. A sebesség nem elsődleges, hiszen a monitorozó kártyákban is működniük kell kompenzált időmérő moduloknak, így nem elsődleges az időpecsét minél gyorsabb átvitele. Az USRT kétirányú, full duplex átvitelt tesz lehetővé, az órajel miatt egyszerű és megbízható detektálást nyújt. A vonal konstanst késleltetést visz az időpecsét átvitelébe (ami így kompenzálható), és csak három vezeték szükséges hozzá (adat be, adat ki, órajel). Ezzel nem foglalja le teljesen a 40 pines feature connectort, így teret hagy későbbi fejlesztéseknek is.

Keretező modul

Kialakítottam egy egyszerű keretszerkezetet, melynek a felépítését a 2. táblázat mutatja.

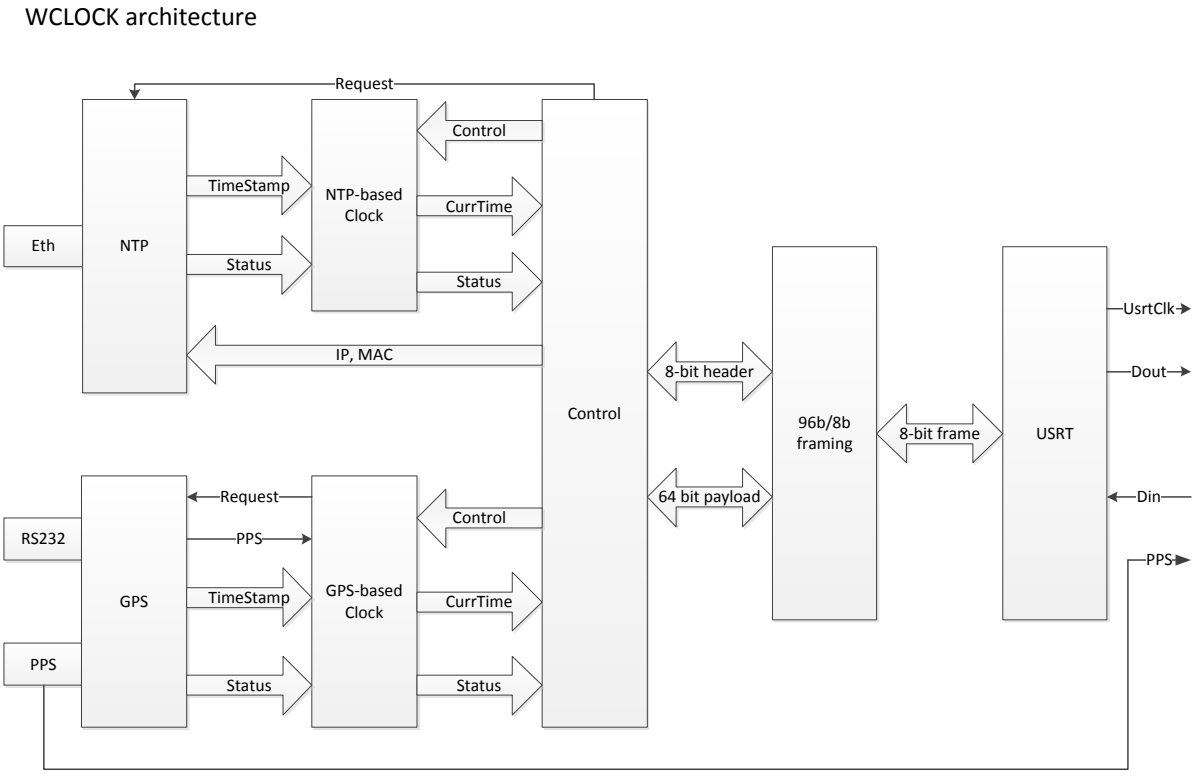
Magic Code	Type	Chk	Payload
16 bit	8 bit	8 bit	64 bit

2. táblázat: A WCLOCK keretfelépítése

Minden keret egy 16 bites fix kóddal kezdődik, ez jelzi a vevőnek a keret elejét. Utána jön a keret típusának jelölése, pl. ha a WCLOCK küld a monitorozó kártyának, akkor ez lehet időpecsét vagy státusz, ha a monitorozó küldd a WCLOCK kártyának, akkor ez lehet paraméter beállítás vagy bármilyen lekérdezés. ezután következik egy 8 bites ellenőrző összeg, majd a 64 bites adatmező. Ennek a hosszát az elterjedt időpecsét formátumok hossza határozta meg.

3.2.5 A modulok összekapcsolása

Az eddigiek alapján felrajzolható az egyes részfeladatokat ellátó modulok funkcionális elrendezése és összekapcsolása az FPGA-n belül (14. ábra).



14. ábra: Az egyes részfeladatokat ellátó modulok összekapcsolása az FPGA-n belül

4 A megvalósítás

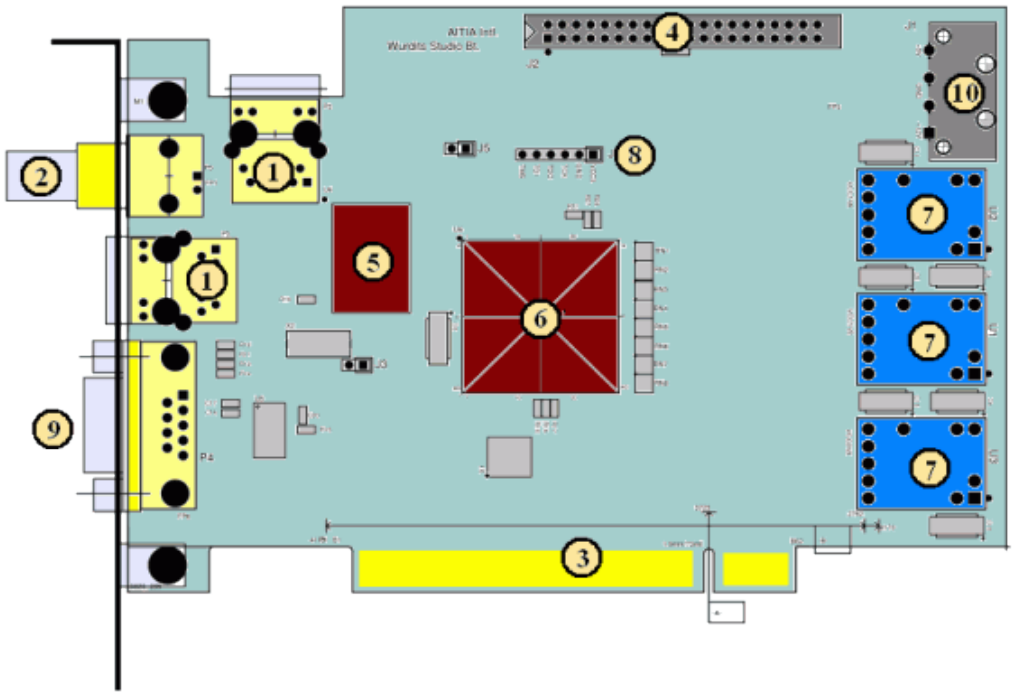
A részfeladatok áttekintése után a konkrét hardver és a megvalósított komponensek bemutatása következik. Mindegyik modult VHDL nyelven írtam, a fejlesztéshez a Xilinx ISE Design Suite 13 verzióját használtam.

Az általam megtervezett és megírt funkciók:

- GPS vevő üzeneteit dekódoló komponens,
- a helyi oszcillátor hibáját kompenzáló időmérő,
- a vezérlő modul,
- a 96 bites üzeneteket előállító keretező,
- az USRT kommunikációt megvalósító komponens,
- NTP kliens (korábbi munka eredménye).

4.1 A WCLOCK kártya felépítése

A kártya tervezése nem az én feladatom volt, így csak röviden áttekintem a számomra fontos alkatrészeket. A kártyát a 15. ábra mutatja



15. ábra: A WCLOCK kártya főbb alkatrészei

1. 10/100/1000 Mbps RJ45 csatlakozó
2. Koaxiális csatlakozó a PPS jelnek
3. PCI csatlakozó a tápfeszültséghez
4. 40 pines Berg típusú csatlakozó
5. Realtek RTL8212 Ethernet interfész IC
6. Xilinx Spartan 3 típusú FPGA
7. A szükséges tápfeszültségeket előállító tápegységek

8. JTAG csatlakozó az FPGA programozásához
9. RS-232 interfész a GPS vevő üzeneteinek vételére és PC-hez való csatlakozásra
10. PC tápcsatlakozó arra az esetre, ha nem volna szabad PCI interfész

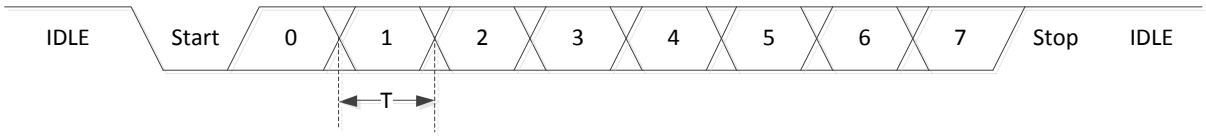
A kártyán az elsődleges órajelforrás egy 125 MHz +/- 50 ppm oszcillátor [23], ez a frekvencia szükséges a Realtek IC GMII interfész meghajtásához. Mivel nem tartozik szorosan a dolgozat tárgyához az NTP modul hálózati kapcsolata, a továbbiakban erre nem térek ki. Ami ebből az időméréssel kapcsolatban fontos, az a kvarc frekvenciája (125 MHz) és stabilitása (+/-50 ppm).

A választott FPGA típusa XC3S1400AN-FGG676-4, mely a Spartan 3 széria beépített flash memóriával ellátott változata. A Spartan 3 elegendő erőforrással rendelkezik a feladat ellátásához, a beépített memóriának köszönhetően pedig nem szükséges minden indításkor újraprogramozni. A fejlesztés során a JTAG csatlakozón keresztül lehet programozni gyári Xilinx programozókábellel, az iMPACT program használatával.

4.2 Illesztés a GPS-hez

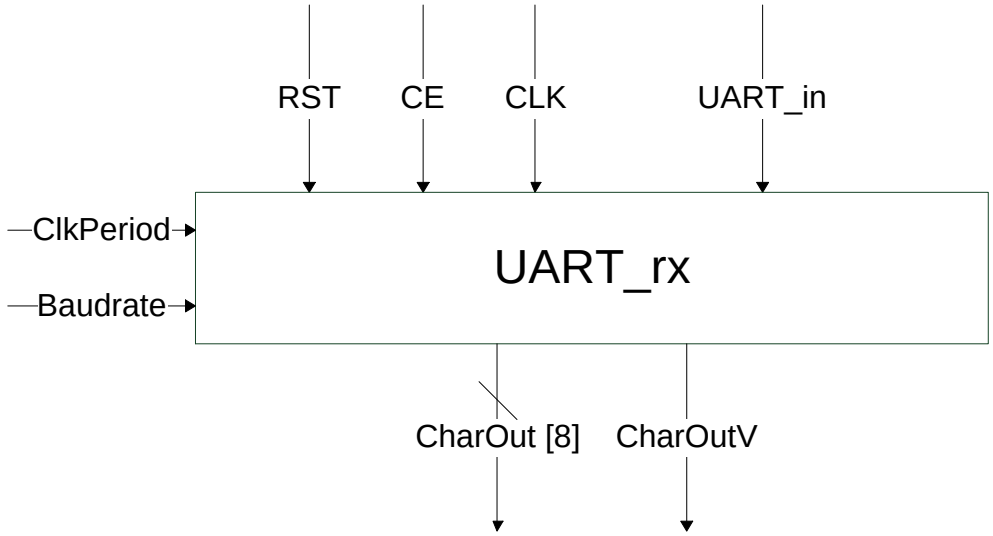
4.2.1 UART vevő

A GPS vevőtől szabványos 8 bites UART keretek érkeznek, egy ilyet mutat a 16. ábra.



16. ábra: Egy UART keret

Az átvitel aszinkron, azaz nincs órajel. A keret kezdetét a magas logikai szinten lévő vonal alacsony szintre esése jelzi (start bit). Ez után az ún. baud rate alapján tudja a vevő, hogy mikor kell mintát venni a bejövő vonalról. Ha pl. 4800 bps a baud rate, akkor T = 1/4800 ≈ 208 µs időközönként érkezik új bit a bemeneten. Ez alapján, ahogy leesett a bemeneti jelszint magasról alacsonyra, a vevő vár a fent említett időtartam másfélszereséig, így az első adat bit közepénél tart az adó (időben). Ekkor stabil a jel, ilyenkor lehet mintavételezni az első bitet, majd ezután már csak egy időtartamnyit kell várnia, így minden bitet középen mintavételez. A 8. bit után következik egy magas értékű stop bit és IDLE állapotban marad az adó. A 8. bit és a stop bit közé beékelődhet egy paritás bit, mely magas értékű, ha a 8 bites adatban páros számú egyes található, de ezt nem használják az NMEA üzeneteket küldő GPS modulok. A vevő ki- és bemeneteit a 17. ábra mutatja.



17. ábra: Az UART vevő ki- és bemenetei

Bemenetek

- **CLK**: Rendszer órajel
- **CE**: Engedélyező jel
- **RST**: Reset
- **UART_in:** A bejövő UART vonal a GPS vevőtől

Kimenetek

- **CharOut[8]:** A GPS vevőtől érkezett karakter
- **CharOutV:** Magas értéke jelzi, ha érvényes karakter van a kimeneten

Paraméterek

- **ClkPeriod:** A rendszer órajel periódusideje ns-ban
- **Baudrate:** Az UART átvitel sebessége (GPS modul függő)

A vevőt egy egyszerű állapotgép valósítja meg, mely alapállapotban folyamatosan mintavételezi és eltárolja az UART_in bemenet aktuális értékét. Ha azt tapasztalja, hogy magasról alacsony értékűre vált a bemenet (azaz lefutó éle van), akkor vételi módba kapcsol. A ClkPeriod (rendszer órajel periódusideje) és Baudrate (GPS modul függő) alapján megvárja az első adatbit időablakának közepét, mintát vesz a bemenetről és eltárolja azt. Ezt ismétli a 8. adatbitig, majd stop bit után alapállapotba tér vissza és a kimenetén megjeleníti a beérkezett 8 bites vektort a CharOutV magas értéke mellett.

4.2.2 NMEA dekóder

Az NMEA szabvány egy egyszerű ASCII karaktereken alapuló, soros egyirányú protokoll, melyben a „beszélő” „mondatokat” küld akár több „hallgatónak” egyszerre. Minden üzenet egy dollár ($) karakterrel kezdődik, a következő öt azonosítja a küldőt és az üzenet típusát (kettő és három karakter). Az ez után érkező adatmezőket vessző karakterek választják el egymástól egészen az utolsó mezőig. Ha a küldő ellenőrző összeget is küld, akkor az utolsó adatmezőt szorzásjel követi (tehát ezt már nem zárja le egy vessző), majd az ez utáni két karakter egy két számjegyes hexadecimális számot

reprezentál, ez az ellenőrző összeg. Ezt a kezdő dollár és a lezáró szorzás karakter közötti összes karakter kizáró vagy kapcsolatával lehet előállítani. Az üzenetet a CR (carriage return) és LF (line feed) karakterek zárják le.

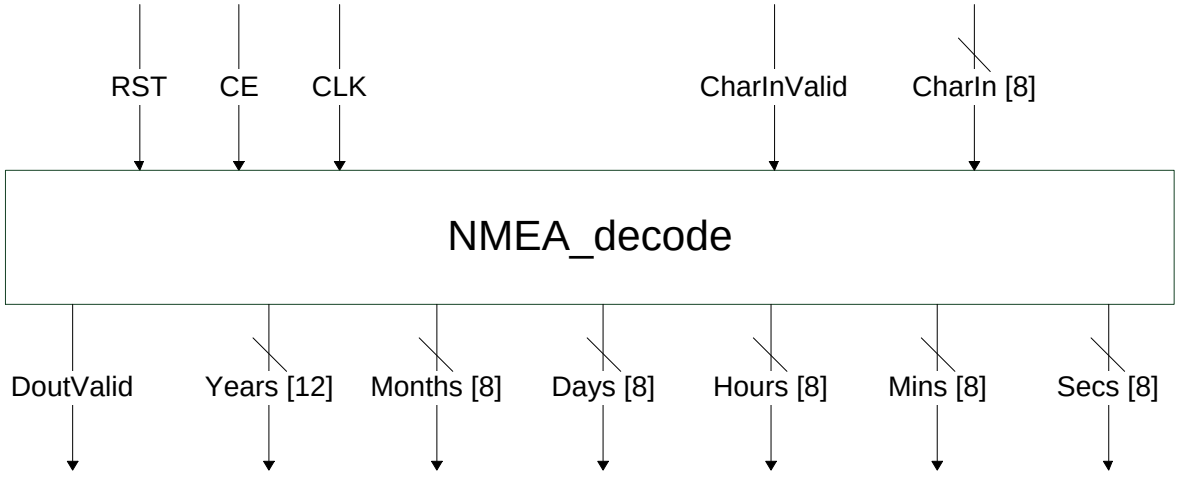
A dekóder az UART vevőtől kapja a karakterek 8 bites ASCII kódjait. Alapállapotban minden karaktert ellenőriz, a dollár karakter (0x24) jelzi egy üzenet kezdetét. Az ez után következő öt karakter értéke mondja meg, mit tartalmaz az üzenet, itt döntheti el a vevő, hogy van-e benne számára érdekes adat. A WCLOCK szempontjából az időinformáció lényeges, ezt tartalmazza pl. a minden vevőmodul által biztosított GPRMC üzenet, ahol a GP az adót GPS eszközként azonosítja, az RMC pedig a Recommended Minimum Specific GNSS Data rövidítése. Egy példát és az egyes mezők magyarázatait a 3. táblázat mutatja.

$$\$GPRMC, 161229.487, A, 3723.2475, N, 12158.3416, W, 0.13{,}309.62{,}120598, , *10$$

Mező neve	Példa	Egység	Magyarázat
Üzenet azonosító	GPRMC		RMC fejléc
UTC idő	161229.487		hhmmss.sss
Állapot	A		A = érvényes adat, B = nincs érvényes adat
Szélességi fok	3723.2475		ddmm.mmmm
É/D jelölés	N		N = észak, S = dél
Hosszúsági fok	12158.3416		dddmm.mmmm
K/NY jelölés	W		E = kelet, W = nyugat
Sebesség a felszínhez képest	0.13	csomó	
Valódi haladási irány	309.62	fok	
Dátum	120598		ddmmyy
Mágneses elhajlás		fok	
Ellenőrző összeg	*10		
<CR><LF>			Üzenet vége

3. táblázat: Egy GPRMC üzenet mezői

Látható, hogy ebből az üzenetből a teljes időinformáció kinyerhető. A fejléc vétele után számolja a vett karaktereket, a fontos mezőket (UTC idő, Dátum) eltárolja. Ezek alapján pontosan meg lehet határozni a következő PPS időpontját, hiszen az a következő egész másodperc elején lesz. A modul az ASCII karakterekből számértékeket állít elő, ez egyszerű kivonással megtehető, egy szám értéke az ASCII kódja – 48. Az egyes számjegyeket fel kell szorozni a helyi értékükkel, majd összeadással előállítja az év, hónap, nap, óra, perc, másodperc értékét. Tehát pl. a 18 órát a 49 és az 56 ASCII kód jelenti, ezekből 48-at kivonva megkapja az 1-et és a 8-at, az 1-et felszorozva 10-el és összeadva 8-al előállította a 18-at. A szorzást az FPGA-ban dedikált szorzó áramkörök támogatják. A dekóder ki és bemeneteit a 18. ábra mutatja.



18. ábra: Az NMEA dekódoló ki- és bemenetei

Bemenetek

- **CLK:** Rendszer órajel
- **CE:** Engedélyező jel
- **RST:** Reset
- **CharIn[8]:** Az UART vevőtől érkező karakter
- **CharInValid:** Magas értéke jelzi, ha érvényes karakter van a bemeneten

Kimenetek

- **Years[12]:** Az évnek megfelelő számérték (pl. 2011 = 0b011111011011)
- **Months[8]:** A hónapnak megfelelő számérték
- **Days[8]:** A napoknak megfelelő számérték
- **Hours[8]:** Az óráknak megfelelő számérték
- **Mins[8]:** A perceknek megfelelő számérték
- **Secs[8]:** A másodperceknek megfelelő számérték
- **DoutValid:** Magas értéke jelzi az érvényes kimenetet

Mikor előállította az egyes időértékeket, a DoutValid magas értéke jelzi, hogy azok megjelentek a megfelelő kimeneti lábakon. Itt elegendő csak egész másodpercig átadni az időt, mert a cél a következő PPS jel pontos idejének meghatározása. Ez az üzenet időinformációja alapján úgy tehető meg, hogy a másodperc tört részét elhanyagoljuk, az egész részhez pedig hozzáadunk egyet. A komponens ennek megfelelően állítja be a **Secs** kimenet értékét.

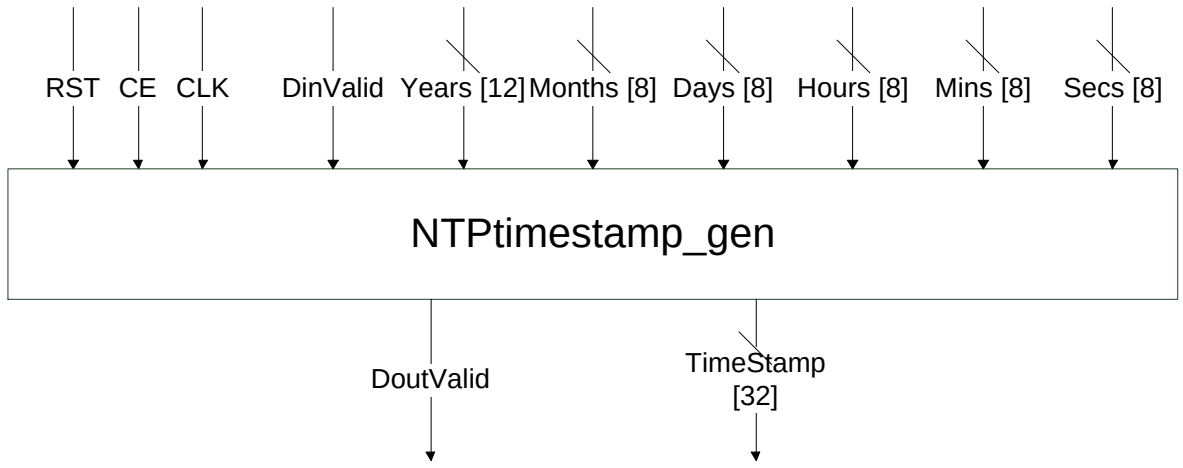
4.2.3 Időpecsét előállító modul

Az NMEA dekóder számértékekben ábrázolja a GPS vevőtől kapott időinformációt, ezt még át kell alakítani valamilyen gyakorlatban használt időpecsét formátumra, amire teljesülnek a következők:

- kellően nagy a felbontása (minimum ns nagyságrend),
- alkalmas arra, hogy a kompenzált időmérés alapjául szolgáljon,
- ipari gyakorlatban használt, lehetőleg a hálózatmonitorozás területén is.

Az NTP alapú óra praktikusan NTP formátumot használ, mely egy 64 bites időbélyeg és 1900. január 1. 0:00-tól méri az időt. A felső 32 biten az eltelt másodpercek számát jeleníti meg, az alsó 32 biten pedig a maradék másodperc tört részét, így $2^{-32} \approx 233$ ps felbontást biztosít. Ez bőven elegendő a GPS alapú időpecsét ábrázolására és 10 Gbps Ethernet hálózatok monitorozásához is kellően nagy a felbontás. Az NMEA üzenetből nyert információt mindenképpen át kell alakítani valamilyen gyakorlatban használt időpecsétre, praktikus tehát ezt is NTP formátumba konvertálni, így a rendszeren belül egységes az időábrázolás.

Először azt kell kiszámítani, hogy az aktuális dátum és 1900 között mennyi szökőév volt. Ehhez ki kell vonni az évet ábrázoló számból 1900-at, az eredményt pedig el kell osztani néggyel. Ez utóbbi kettes számrendszerben kettő helyi értékkel való eltolás lefelé, így FPGA-n belül egyszerűen elvégezhető (az eredmény értelmezési tartománya az egész számok, így a maradék elvesztése nem probléma). Ez alapján számítható az aktuális évig eltelt másodpercek száma 1900. január 1. óta. A hónapot reprezentáló szám átalakítható másodpercre, csak arra kell figyelni, hogy az aktuális év szökőév-e (ekkor eggyel több nap telt el, ha az aktuális hónap március, vagy későbbi). A napok, órák, percek másodperccé alakítása után csak össze kell adni az értékeket és megkapjuk az NTP időpecsét felső 32 bitjét. Az alsó 32 bit nulla, hiszen az időpecsét a következő PPS idejét jelöli, mely viszont pontosan jelzi egy másodperc kezdetét, tehát a tört rész itt mindenképpen nulla. Az időpecsétet előállító modul ki- és bemeneteit a 19. ábra mutatja.



19. ábra: Az NTP formátumú időpecsétet előállító modul

Bemenetek

- **CLK:** Rendszer órajel
- **CE:** Engedélyező jel
- **RST:** Reset
- **Years[12]:** Az évnek megfelelő számérték (pl. 2011 = 0b011111011011)
- **Months[8]:** A hónapnak megfelelő számérték
- **Days[8]:** A napoknak megfelelő számérték

- **Hours[8]:** Az óráknak megfelelő számérték
- **Mins[8]:** A perceknek megfelelő számérték
- **Secs[8]:** A másodperceknek megfelelő számérték
- **DinValid:** Magas értéke jelzi az érvényes bemenetet

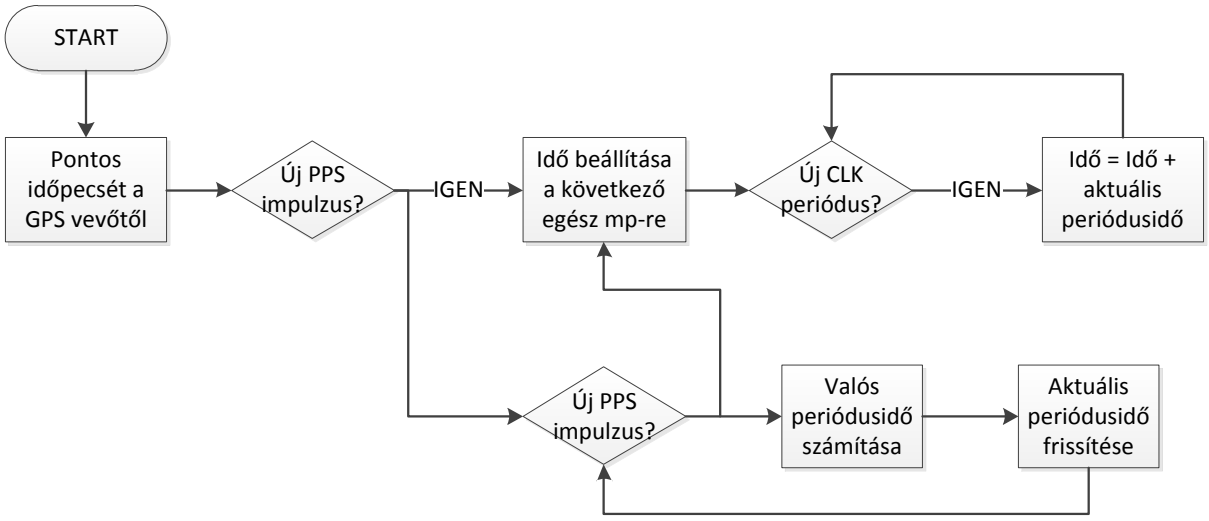
Kimenetek

- **TimeStamp[32]:** A következő PPS ideje NTP formátumban (alsó 32 bit fixen 0)
- **DoutValid:** Magas értéke jelzi az érvényes kimenetet

Ha rendelkezésre áll az időpecsét, megjeleníti azt a 32 bites kimeneten a DoutValid jel magas értéke mellett. Az időmérő modul tudja, hogy ez a következő PPS jel időpontja lesz, és ennek megfelelően állítja be magát.

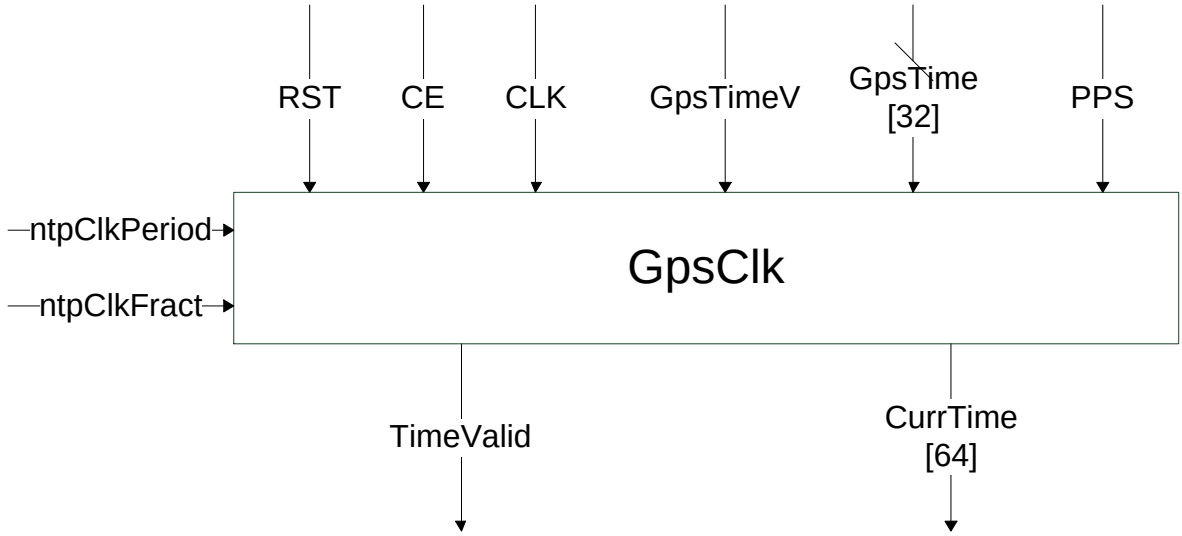
4.3 Kompenzált időmérés

A működés elve a 3.2.1 részben olvasható, periodikus szinkron kompenzációval. A lényege, hogy szinkron esetén összehasonlítja a bejövő pontos időt a sajátjával, a hiba és a legutóbbi szinkron ideje alapján számol egy valós periódusidőt és azzal mér tovább. GPS esetén az órát egyszer szükséges beállítani az NMEA dekóder és az NTP időpecsét konverter által biztosított dátumra és pontos időre, onnantól kezdve a szinkront a PPS jel biztosítja. Ennél ki lehet használni, hogy előre tudjuk, mikor kellene megérkeznie, azaz minden pillanatban tudjuk, mekkora lenne az óra hibája, ha most érkezne a PPS impulzus. Ezt persze utólag is ki lehetne számítani, de így időt nyerünk, hamarabb rendelkezésre áll majd az aktuális periódusidő. Az időmérés folyamatát a 20. ábra mutatja.



20. ábra: A kompenzált időmérés működése

Miután a GPS vevőtől beérkezett az időinformációt hordozó keret, azt NTP formátumú időpecsétté alakítja a konverter és átadja az időmérőnek. Ez után az első PPS impulzusra elindul az óra és jár az általa ismert periódusidő alapján. A következő PPS jel vételekor rendelkezésre áll az óra hibája 1 mp alatt valamint hogy mennyit „ütött” az óra ez alatt az idő alatt, azaz a valós periódusidő hányszor fér bele egy másodpercbe. Ha a hibát elosztjuk ezzel a számmal azt kapjuk meg, hogy mennyivel kell módosítani az aktuális periódusidőt ahhoz, hogy az megfeleljen a valóságnak. A modul ki- és bemeneteit a 21. ábra mutatja.



21. ábra: Az időmérő modul ki- és bemenetei

Bemenetek

- **CLK:** Rendszer órajel
- **CE:** Engedélyező jel
- **RST:** Reset
- **GpsTime[32]:** A következő PPS jel időpontja NTP formátumban
- **GpsTimeV:** Magas értéke jelzi az érvényes időpecsétet a bemeneten
- **PPS:** Impulzus minden másodperc elején, közvetlenül a GPS vevőtől

Kimenetek

- **CurrTime[64]:** A pontos idő NTP formátumban
- **TimeValid:** Magas értéke jelzi, ha már érvényes időpecsét van a kimeneten

4.3.1 Számábrázolási kérdések

A kompenzált időmérés elve egyszerű, azonban a hardverese megvalósítás felvet néhány problémát, az egyik ilyen a számábrázolás. A rendszer órajele 125 MHz, ekkor a periódusidő ennek a reciproka, 8 ns. Ez NTP formátumban:

$$ntpClkPeriod = 8 \times 10^{-9} \times 2^{32} = 34{,}359738368$$

Az NTP időpecsét formátum azonban nem lebegőpontos, hanem egész, ezért be kell vezetni a számítások során egy *ntpClkFract* segédváltozót, amelyben nyomon követjük a tört rész változását. Ez azt jelenti, hogy minden órajelre hozzá kell adni a periódusidő egész részét az időpecséthez, valamint hozzá kell adni a segédváltozóhoz a tört rész értékét. Ha a segédváltozó túlcsordul, akkor a következő órajelre nem a periódusidő egész részét, hanem annál eggyel többet kell hozzáadni az időpecséthez. Az *ntpClkFract* változót túlcsordulás esetén nem nullázni kell, hanem arra beállítani, amennyivel túlléptük a számábrázolás adta korlátot.

Az *ntpClkFract* bevezetése egy másik okból is kényelmes: egy 125 MHz +/- 50 ppm kvarc periódusideje NTP formátumban kb. 34,358020467 és 34,361456441 lehet, jól láthatóan az NTP-nek (és más, elterjedt időpecsét formátumoknak) nem elég nagy a felbontása, hogy egy ilyen kvarc

hibáját ábrázolni lehessen, ez a hiba ebben a segédváltozóban fog megjelenni. A megvalósított időmérő modulban ez egy 32 bites vektor, így a periódusidő tört részét és a kvarc hibáját is 32 biten ábrázolom, ezzel tulajdonképpen 96 bites időpecsétet tartok karban, aminek az alsó 64 bitje a másodperc tört részét ábrázolja. Így az oszcillátor hibáját $2^{-64} \approx 5{,}4\text{x}10^{-20}$ s felbontással kompenzálhatom, ami felesleges GPS esetében, ahol másodpercenként rendelkezésre áll a szinkron PPS jel, mert ennyi idő alatt ennél nagyságrendekkel nagyobb hiba sem okozna problémát. Azonban NTP üzemmódban, ahol pl. 15 percenként kérünk friss időbélyeget a szervertől, már van értelme ennek az ábrázolásnak.

NTP alapú felhasználásnál annyi a különbség, hogy a PPS jel helyett teljes időpecsét áll rendelkezésre, cserébe jóval ritkábban (ez a lekérdezések gyakoriságától függ, pl. 15 percenként). GPS esetén ki lehet használni, hogy amikor a PPS beérkezik, akkor kellene egész másodpercet mutatnia az órának, így folyamatosan meg tudjuk mondani, mekkora lenne a hiba, ha most érkezne a PPS impulzus. Ez NTP esetében nem lehetséges, itt ki kell számítani először a hibát (egy kivonás és egy szorzás, hogy az időpecsétek különbsége és az *ntpClkFract* dimenziója megegyezzen), és csak utána lehet ebből a valós periódusidőt meghatározni.

4.3.2 FPGA-n belüli aritmetikai műveletek

A valós periódusidő kiszámításához szükség van egy osztás műveletre, melyre több megoldás kínálkozik FPGA-n belül, de mindegyik kompromisszummal jár:

- DSP használata osztásra.
  - Előnye: gyors, nagy operandusokat is képes kezelni.
  - Hátránya: csak drága FPGA-ban van, a WCLCOK esetében ez felesleges lett volna.
- Softcore processzor használata az osztásra.
  - Előnye: egyszerű.
  - Hátránya: lassú, egyébként ki nem használt erőforrás lenne a beágyazott vezérlő.
- Osztó algoritmus (pl. Radix-2) implementálása.
  - Előnye: szintén egyszerű (elérhető, mint Xilinx IP core)
  - Hátránya: kb. 16 bitnél kisebb operandusok esetén hatékony, relatíve nagy az erőforrás igénye (kb. 6000 flip-flop, ami a használt FPGA flip-flopjainak harmada).

A fenti lehetőségek közül egyik sem volt számomra megfelelő, így egy igen egyszerű, viszont lassú megoldást választottam: egy regiszterhez minden órajelre hozzáadtam az osztó értékét addig, amíg a regiszter nagyobb nem lett az osztandóval. A hányados értéke az összeadások számával egyezik meg. Ez a módszer nyilván nem túl hatékony és csak akkor engedhető meg, ha nincs szükség azonnal a hányados értékére és/vagy biztos, hogy a hányados mindig kicsi lesz, így az összeadások számára tudunk felső korlátot mondani. Ez esetben éppen erről van szó: egy 125 MHz +/- 50 ppm kvarc két szélső periódusideje közötti legnagyobb különbség kb. $8\text{x}10^{-13}$ s, ami a *clkNtpFract* dimenziójában $8\text{x}10^{-13}\text{x}2^{-64} \approx 14757395$. Azaz ha egy PPS előtt az oszcillátor periódusideje az egyik, majd közvetlenül utána a másik szélsőértéket venné fel (ami igen szélsőséges, majdnem lehetetlen eset), akkor is 14757395 órajel-periódus alatt ki lehetne számolni a valós periódusidőt. Ez időben az előbb említett kvarc esetén 14757395x8 ≈ 118 ms-ot jelent, tehát a két PPS között eltelt idő kb. 10%-a alatt nem a legfrissebb mérésnek megfelelő periódusidővel működne az óra. Ez egy kis mérési hibát visz a rendszerbe, a következő számításnál „túlkompenzál”, de a valós életben előforduló különbségek két PPS között nagyságrendekkel kisebbek.

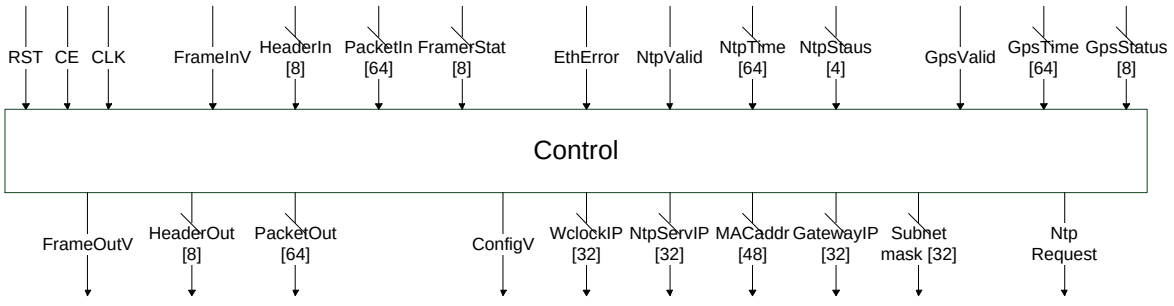
4.3.3 Az időpecsét ugrása

Az időmérő modul képes minden PPS után az éppen aktuális, valós periódus idő alapján működni, azonban az időt minden PPS jelre kerek egész másodpercre kellene állítani, hiszen ezt jelzi a PPS. Ez nyilván ugrásokat eredményez az időpecsétben, ami problémát okozhat. Két eset lehetséges, vagy siet az óra, vagy késik. Ha késik, akkor az időt előre kell állítani, ami a monitorozás során nem jelent nagy problémát (ésszerű keretek között persze), hiszen a hangsúly a sorrendhelyességen van. Ha siet az óránk, akkor a PPS jelre visszafelé kellene állítani, ami a felhasználáskor problémát jelenthet. Ha ugyanis épp a szinkron előtt érkezik be egy keret, majd egy következő a szinkron után azonnal, akkor előfordulhat, hogy a később beérkező keret korábbi időpecsétet kap, mint ami valójában előbb érkezett be.

Ezt a problémát legegyszerűbben úgy lehet kiküszöbölni, hogy amennyiben siet az óránk, a monitorozáshoz használt időpecsét mellett fenntartunk egy másikat, amit ilyen esetben visszaállítunk, ez lesz tehát a valódi pontos idő. A felhasználásra szánt időt a szinkron után nem a periódusidőnek megfelelő értékkel növeljük minden órajelre, hanem annál kevesebbel (akár 1-el). Ezzel biztosítjuk, hogy az ez idő alatt időpecsételt csomagok sorrendje is megfelel a valóságnak az adatbázisban, hiszen az időpecsét folyamatosan növekszik. Az ideiglenes időpecsétet persze az eddigiek szerint állítjuk, a kimért valós periódusidő alapján. Kezdetben ez korábbi időpontot fog mutatni, mint a monitorozáshoz használt, de mivel azt csak 1-el növeljük órajelenként, ezt meg a periódusidőnek megfelelő értékkel, az ideiglenes időpecsét valamikor utoléri (megelőzi) a monitorozáshoz használt időbélyeget. Innentől kezdve újra a korábban leírt módon növelhetem a monitorozáshoz használt időt, ami már a valós időnek megfelelő értéket mutatja.

4.4 A funkció vezérlése

A vezérlő komponens teremt kapcsolatot a monitorozó kártya és az időmérő modul között. Fogadja a beérkezett csomagokat, a fejlécük alapján eldönti, hogyan folytassa a feldolgozást. Akár GPS, akár NTP az időforrás, a vezérlő küldi tovább a monitorozó kártya felé az időpecsétet szinkron esetén. Ki- és bemeneteit a 22. ábra mutatja.



22. ábra: A vezérlő modul ki- és bemenetei

Bemenetek

- **CLK:** Rendszer órajel
- **CE:** Engedélyező jel
- **RST:** Reset

- **FrameInV:** Érvényes keret a monitorozó kártyától
- **HeaderIn[8]:** A bejövő üzenet fejléce
- **PacketIn[64]:** A bejövő üzenete törzse
- **FrameStat[8]:** A keretező és azon keresztül az USRT modul állapota

- **EthError:** Ethernet link hiba
- **NtpValid:** Érvényes az NTP alapú óra ideje
- **NtpTime[64]:** Az NTP alapú óra ideje
- **NtpStatus[8]:** Az NTP alapú óra állapota

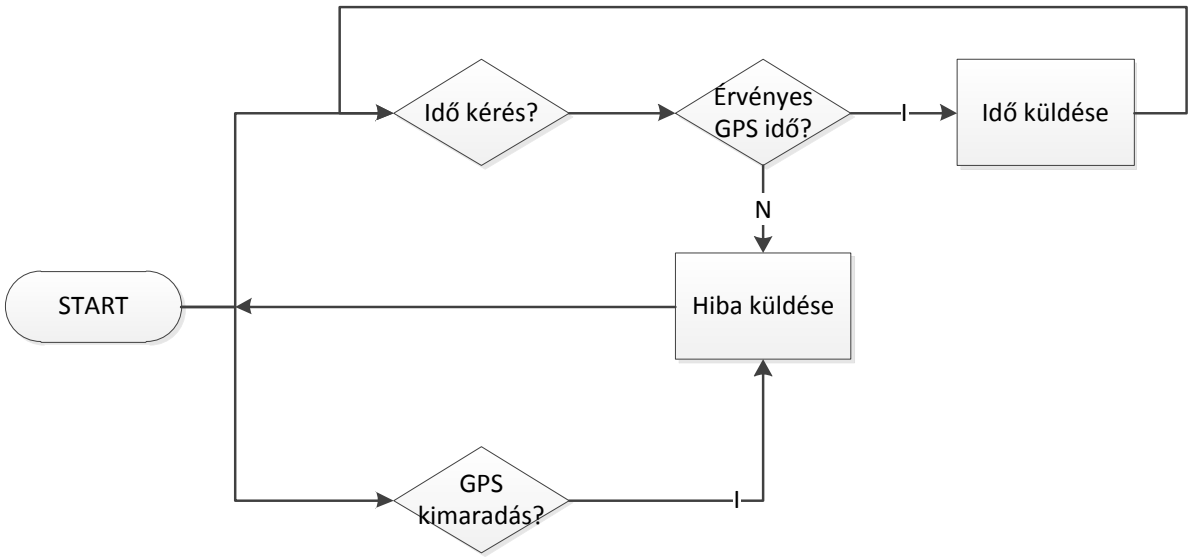
- **GpsValid:** Érvényes a GPS alapú óra ideje
- **GpsTime[64]:** A GPS alapú óra ideje
- **GpsStatus[8]:** A GPS alapú óra állapota

Kimenetek

- **FrameOutV:** Érvényes keret a monitorozó kártyának
- **HeaderOut[8]:** A kimenő üzenet fejléce
- **PacketOut[64]:** A kimenő üzenete törzse

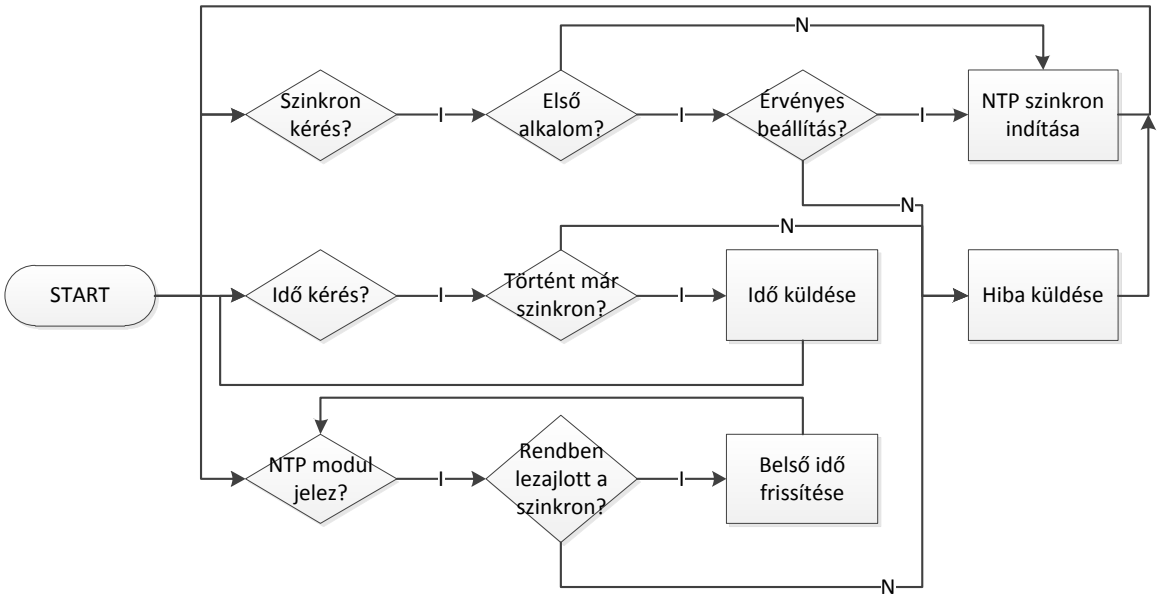
- **ConfigV:** Érvényes konfiguráció az NTP modulnak
- **WclockIP[32]:** A WCLOCK IP címe
- **NtpServIP[32]:** Az NTP szerver IP címe
- **MACaddr[48]:** A WCLOCK MAC címe
- **GatewayIP[32]:** Az adott hálózati átjáró IP címe
- **Subnetmask[32]:** Az adott hálózati maszk
- **NtpRequest:** Magas értékére az NTP komponens szinkront kezdeményez

Alapállapotban várja, hogy érkezzen üzenet a monitorozó felől. GPS üzemmódban már ekkor működik a belső időpecsételő, ha csatlakoztatva van a GPS modul. Ha a monitorozó időpecsétet kér, akkor válaszüzenetben elküldi azt feltéve, hogy van érvényes időinformáció. Ha nincs, akkor ezt is jelzi egy üzenetben a monitorozónak. Ebben a módban nincs más dolga a vezérlőnek, mint figyelni a GPS állapotát, kimaradás esetén jelezni és bármilyen bejövő üzenetre válaszolni. Ezt a folyamatot mutatja a 23. ábra.



23. ábra: A vezérlés folyamata GPS esetén

NTP módban kicsit több dologra kell figyelnie a vezérlőnek. Ha a monitorozó NTP időpecsétet kér, előbb el kell küldenie a szükséges paramétereket (pl. IP címek, NTP lekérdezések gyakorisága, stb.), melyet a vezérlő ad tovább az NTP modulnak. Ha ez nem történt meg, de időpecsétet kér, akkor válaszüzenetben jelzi a hiányt a vezérlő. Miután megkapta a paramétereket és lezajlott az NTP szinkron, elküldi azt a monitorozónak. Ez után a beállítástól függően adott időközönként kérdezi le az NTP szerver idejét és kérésre bármikor elküldi a pontos időt. Ha hibát észlel az NTP modul felől (pl. Ethernet link hiba, ARP hiba, stb.), azt jelzi a monitorozónak. A folyamatot a 24. ábra mutatja.



24. ábra: A vezérlés folyamata NTP esetén

4.5 Illesztés monitorozó eszközökhöz

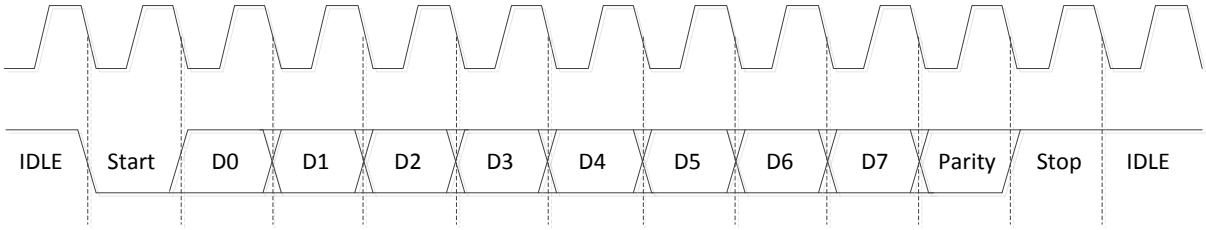
A monitorozó kártyákhoz való illesztés két szintből, az USRT fizikai réteget megvalósító modulból és az adatkapcsolati réteget megvalósító keretező modulból áll. Mindkét réteg kétirányú kommunikációt valósít meg, adó és vevő funkciót látnak el párhuzamosan (full-duplex). A WCLOCK kártyában működő és a monitorozó kártyákba integrált keretező és USRT modul szinte teljesen azonos, az esetleges különbségekre külön kitérek. Az összeköttetéshez hozzá tartozik még egy dedikált vonal, amelyen a WCLOCK késedelem nélkül továbbítja a PPS jelet a monitorozó kártyának, hogy az abban lévő időmérő ugyanúgy működhessen, mint a WCLOCK kártyán lévő.

4.5.1 USRT modul

Az USRT szinkron soros full-duplex átvitelt tesz lehetővé, azaz három vezeték szükséges hozzá: adat be, adat ki és órajel. Amikor két eszköz kommunikál egymással, az egyik generálja az órajelet, amihez szinkronizált az adatátvitel. Ez jelenti az egyetlen jelentős különbséget a WCLOCK kártyán és a monitorozón belül működő USRT modulok között: az egyik generálja az órajelet, ott kimenet ez a láb, a másik számára pedig bemenet. Mivel a monitorozó egyetlen kapcsolata a WCLOCK kártyával ez a vonal, az órajel előállítását a WCLOCK-ra bíztam. Ha a monitorozó nem veszi az órajelet, vagy nem a beállított paramétereknek megfelelőt tapasztal, az hibára utal és jelezheti a felhasználó felé.

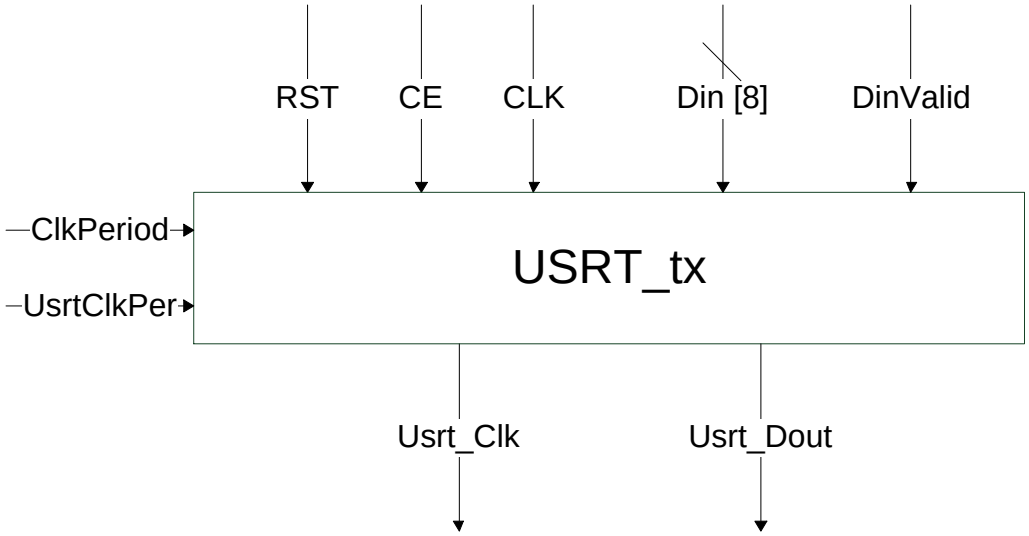
Adó rész

Egy USRT adás időzítését a 25. ábra mutatja.



25. ábra: Egy USRT adás jelei

A felső hullámforma az órajel, az alsó pedig a soros adatvonal. Látható, hogy az adó folyamatosan magasban tartja a vonal jelszintjét (IDLE), majd az adás megkezdését ennek lehúzásával jelzi az órajel lefutó élére (start bit). Ezután teszi ki a kimenetre a 8 adatbitet, mindegyiket lefutó élre. A 8. bit után jön a paritás bit, majd a fixen magas értékű stop bit. Végül visszatér alap állapotba. Az adó ki- és bemeneteit a 26. ábra mutatja.

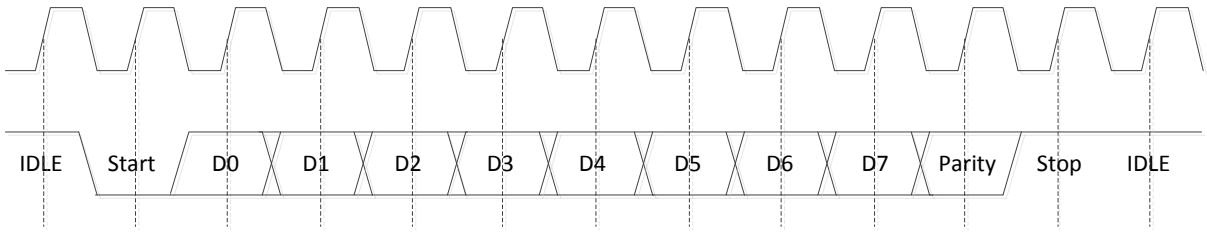


26. ábra: Az USRT adó ki- és bemenetei

A keretező a 8 bites Din bemenetre adja a küldendő adatot a DinValid magas értéke mellett, ekkor kezdődi el az adást a fent leírtak szerint. A rendszer órajel (UsrtClk) és a kívánt USRT órajel (UsrtClkPer) periódusideje alapján generálja az órajelet a vevőnek. A monitorozó kártyákba integrált adó annyiban különbözik ettől, hogy számára ez az órajel bemenet, ott tehát a WCLOCK-hoz szinkronizálva történik az adás. Ennek megfelelően ott szerepel még egy kimeneti láb, a UsrtClkError, mely jelzi a monitorozó kártyának, ha nem kap a WCLOCK felől megfelelő órajelet.

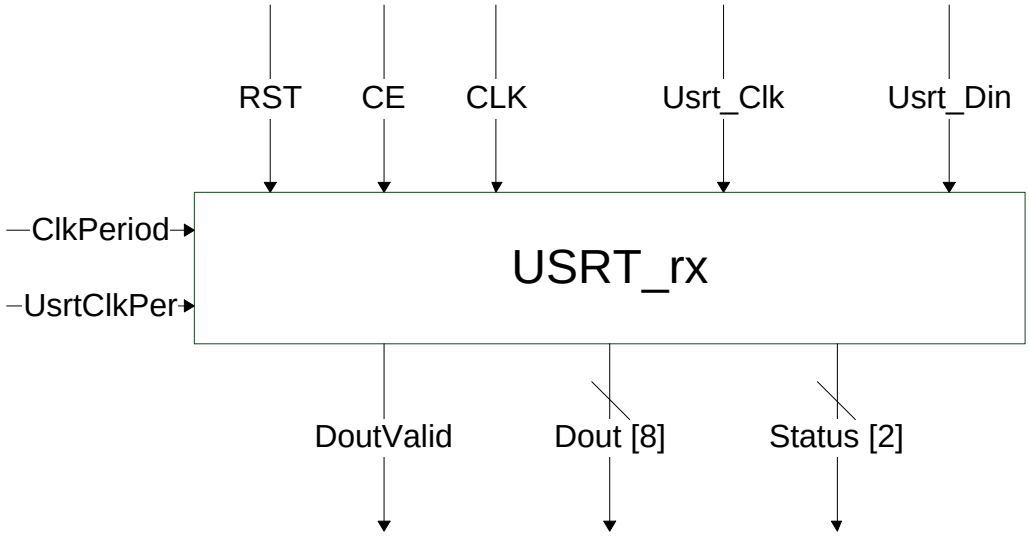
Vevő rész

Egy USRT vonal vételének időzítését a 27. ábra mutatja.



27. ábra: Egy USRT vétel időzítése

Az adó az órajel lefutó élére teszi ki a kimenetre az adatot, a vevő a felfutó élre mintavételez. Jól látszik az ábrán, hogy mindig az adott bit időablakának közepén történik a mintavételezés, így a stabil jelszinteknek köszönhetően megbízható ez az összeköttetés. Alapállapotban minden felfutó órajelre mintát vesz a bementről. Ha a bemenet értéke alacsony, akkor az start bit, a következő felfutó órajelre vett minta lesz az első adatbit. A 8. adatbit után következik a paritás bit, majd a stop bit, végül visszatér alapállapotba. A vevő ki- és bemeneteit a 28. ábra mutatja.



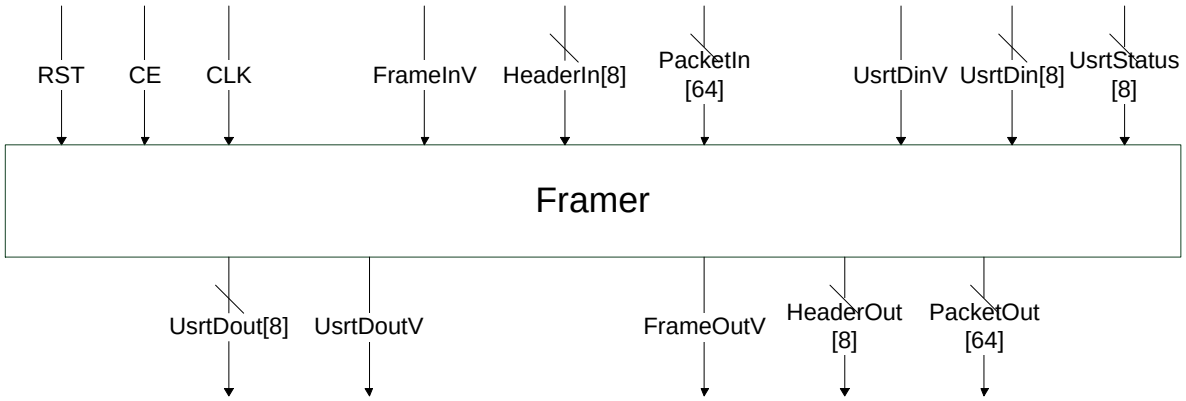
28. ábra: Az USRT vevő ki- és bemenetei

Ha beérkezett egy teljes 8 bites keret és rendben találja a paritást is, a keret megjelenik a Dout kimeneten a DoutValid magas értéke mellett, a Status kimeneten pedig jelzi, hogy vett egy érvényes keretet. Ha hibás a paritás bit, vagy nem érkezik a bemenetre megfelelő órajel (ezt a ClkPeriod és UsrtClkPeriod paraméterek alapján tudja ellenőrizni), akkor a DoutValid magas értéke mellett a Status vektor megfelelő beállításával jelzi ezt a WCLOCK vezérlő komponensének / a monitorozó kártyának.

A paritás bitet kétféleképpen lehetséges értelmezni. Alapesetben akkor magas értékű, ha a bitek közül az egyesek száma páros, egy másik értelmezés szerint ekkor alacsony értékű. Számos olyan USRT eszköz létezik, melyeknél választhatunk e között a két, az ún. *Odd Parity* és *Even Parity* üzemmód között, így én is így készítettem el az adó és a vevő modult. Fordítási időben megadható, hogy melyik módot használják a működés során. Természetesen arra figyelni kell, hogy két eszköz ugyanúgy értelmezze a paritást, különben nem fog működni köztük a kommunikáció.

4.5.2 Keretező modul

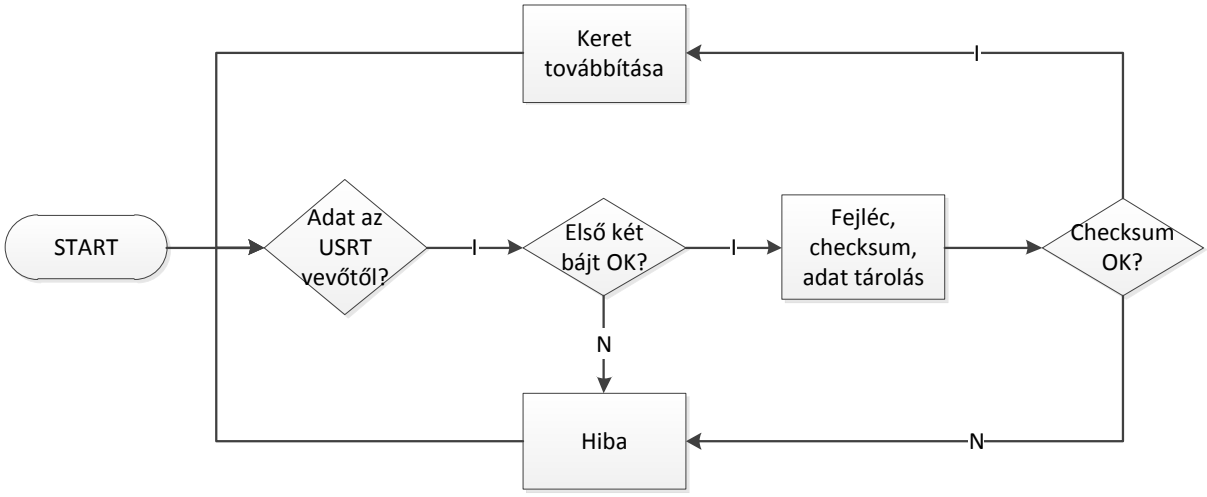
A WCLOCK és a monitorozó kártya kereteken keresztül kommunikál egymással, melynek fejlécét és tartalmát adás esetén megkapja a keretező modul és ebből állítja össze a keretet. Ezután bájtonként átadja a tartalmat az USRT modulnak. A vétel ehhez hasonlóan történik, csak fordított irányban. Az USRT modul jelez a keretezőnek, az elveszi bájtosával az adatot és a kimenetén megjeleníti a fejlécet és a tartalmat. A ki- és bemeneteket a 29. ábra mutatja.



29. ábra: A keretező modul ki- és bemenetei

Vevő rész

Alapállapotban várja, hogy az USRT modultól érkezzen adat. Az első két bájtnak meg kell egyeznie egy rögzített konstanssal (*Magic Code*), ebből tudja, hogy egy keret elején tart. Ezután eltárolja a fejlécet, aztán az ellenőrző összeget, végül magát a 64 bites adatot. Eközben folyamatosan összeadja az eddig vett bájtokat (kivétel az ellenőrző összeget), így az utolsó vett bájt után ellenőrizni tudja a vett keret helyességét. Ha hibás az összeg, azt jelzi a vezérlőnek, ha nem, akkor kiadja a kimenetén. A folyamatot a 30. ábra mutatja.



30. ábra: A keretező vevő folyamata

Adó rész

Alapállapotban várja, hogy a vezérlő küldjön fejlécet és adatot. Ha eltárolta ezeket, kiszámolja az ellenőrző összeget (a *Magic Code* értékét is figyelembe véve), összeállítja a keretszerkezetet és elküldi bájtosával az USRT modulnak.

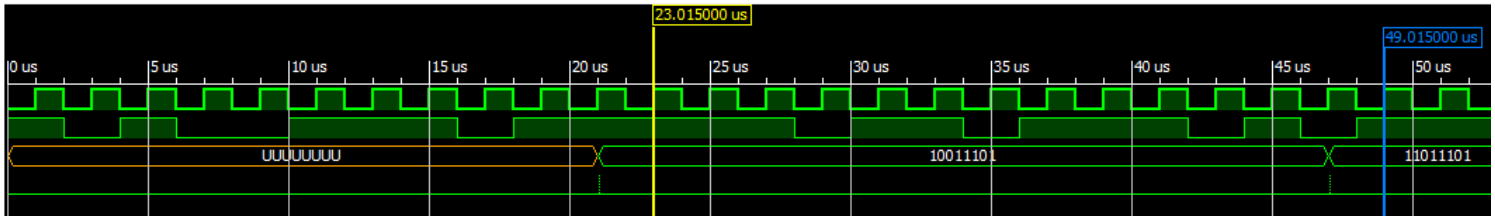
5 Tesztelés

A tesztelés során először az egyes részfeladatokat ellátó komponensek működését ellenőriztem, ahol lehet, szimulációt is felhasználtam. Ezután a teljes, rendszer szintű tesztelés következik. A szimulációhoz a Xilinx ISE beépítetett Isim szimulátorát használtam.

5.1 Kapcsolódó komponensek

5.1.1 USRT

A szimuláció abból állt, hogy az adónak két bájtot kellett elküldenie a vevő felé, aminek ezt a vétel után meg kellett jelenítenie a kimenetén. A két bájt értéke sorrendben 0b10011101 és 0b11011101 volt, a paritás bit akkor volt magas, ha az egyesek száma páratlan volt. A szimuláció eredményét a 31. ábra mutatja.

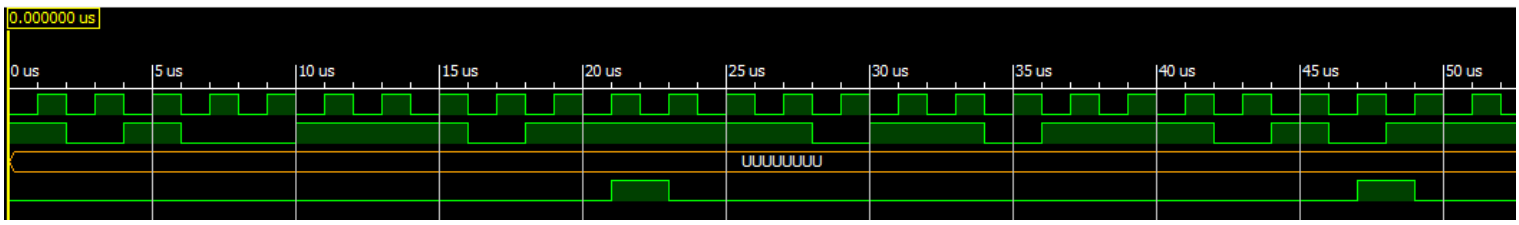


31. ábra: Az USRT adó és vevő szimulációs tesztje

A legfelső hullámforma az USRT csatorna órajele (ezt az adó biztosítja), alatta az adó kimenete látható. Ez alatt a vevő kimenete, legalul pedig a vevő DataVaild impulzusa található, mely az érvényes kimenetet jelzi egy rendszerórajel-periódus idejéig.

Látható, hogy a szimuláció elején az adó magas értéken tartja a kimenetét. Amikor ezt az órajel lefutó élére lehúzza alacsony szintre, megkezdődik az adás. A következő 8 lefutó élre kiteszi a 8 adatbitet, végigkövetve azt tapasztaljuk, hogy az adó az első bájtnak megfelelően változtatta a kimenetét. Ezután következett a paritás bit, mely a beállításnak megfelelően magas, hiszen páratlan számú egyest küldött az adó. Végül következett a stop bit, melynek vételekor (felfutó órajelnél) a vevő megjelenítette a kimenetén a vett bájtot egy DataValid impulzus mellett (legalsó jel). A második bájt elküldésénél ugyancsak helyes működés látható, különbség csak a paritás bitnél jelentkezik, hiszen a bemeneti vektorban most páros számú egyes van.

A következő szimulációnál megváltoztattam a vevő paritás értelmezését, így az különbözött az adóétól. A két bemeneti vektor megegyezett az eddig használt értékekkel. Az eredményt a 32. ábra mutatja.

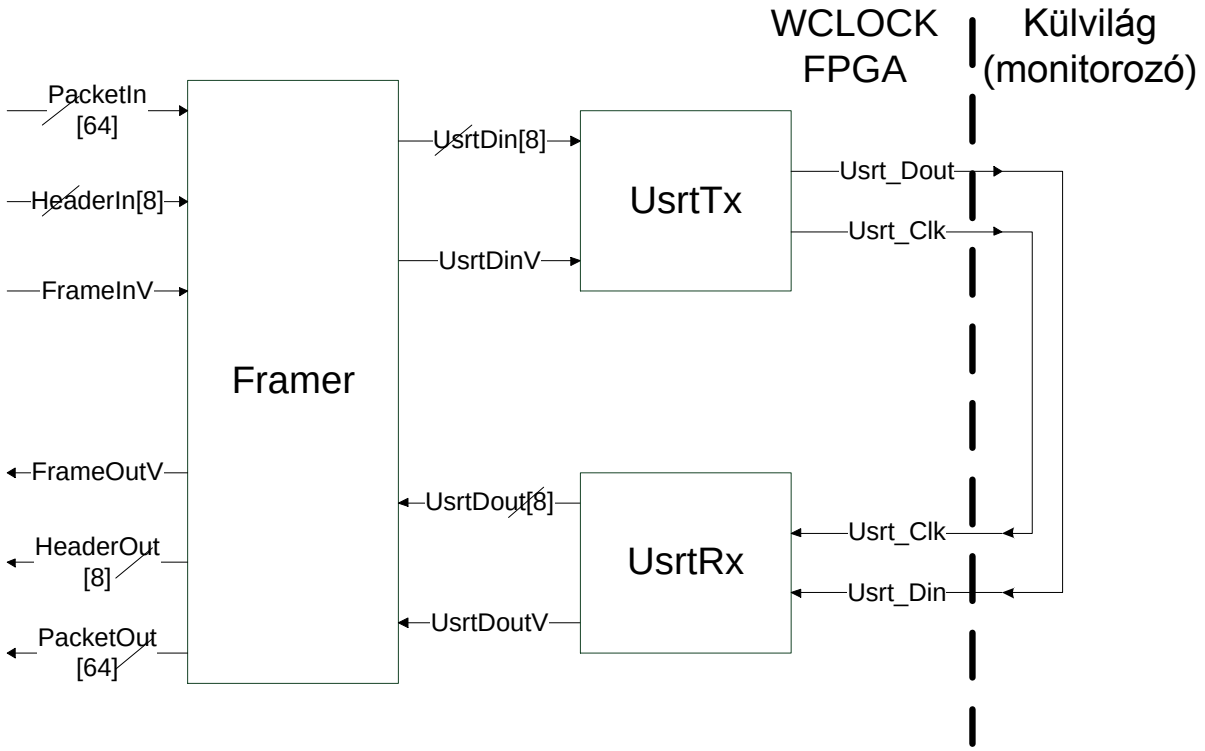


32. ábra: Az USRT adó és vevő szimulációs tesztje különböző paritás értelmezés esetén

A felső három hullámforma ugyanazt mutatja, mint az előző szimulációnál, a legalsó most viszont nem a vevő DataValid kimenete, hanem az USRT error lába. Látható, hogy az adó az első bájt küldése után ugyan azt a paritás bitet küldte el, mint az előbb (20 µs-nál), de a vevő most fordítva értelmezi ezt, így hibát jelez az error lábon.

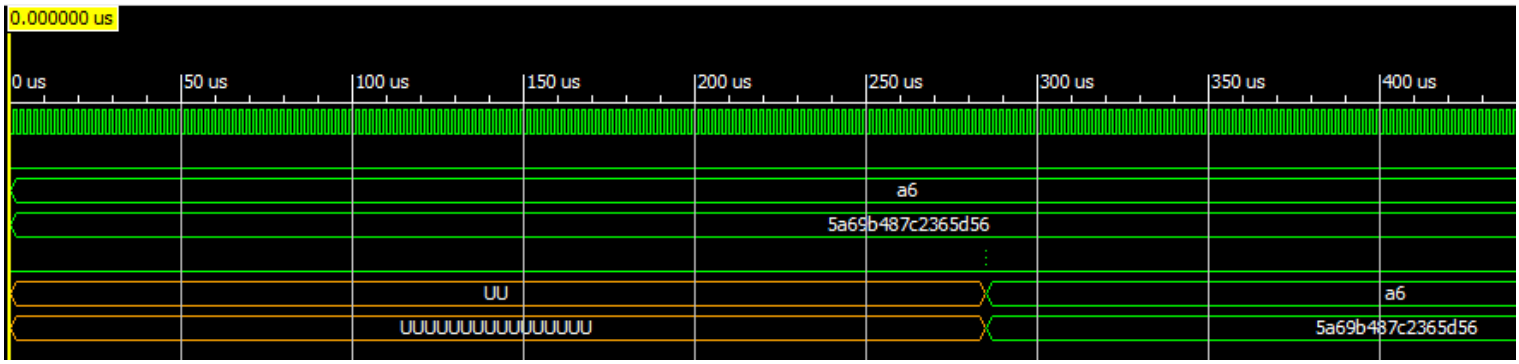
5.1.2 Keretező

Miután megbizonyosodtam az USRT adó és vevő helyes működéséről, a keretező tesztelése következett. A szimulációs környezetet a 33. ábra mutatja.



33. ábra: A keretező szimulációs elrendezése

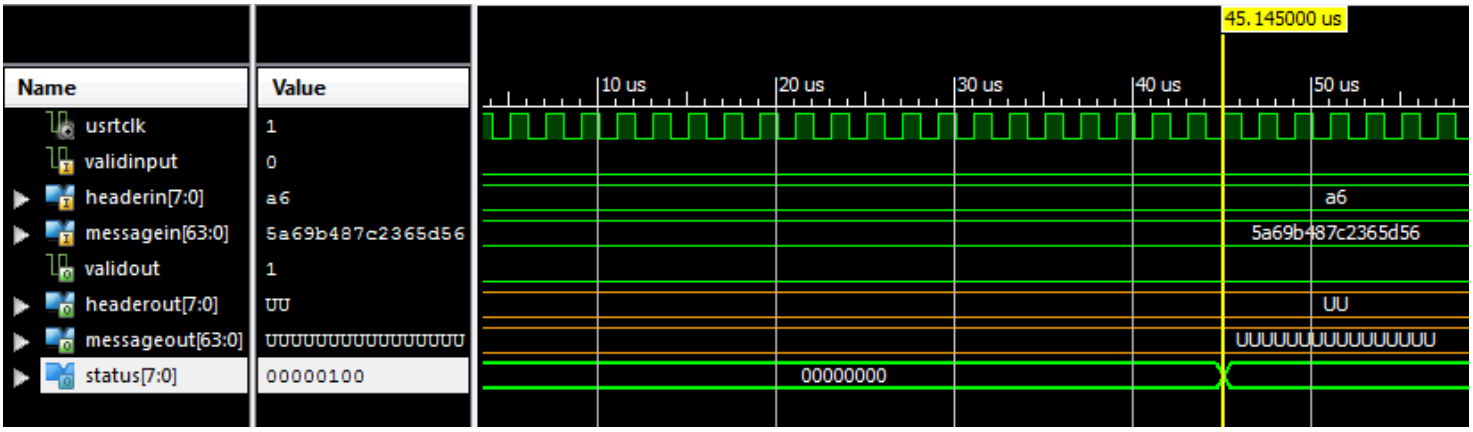
A bemenetre a FrameInV magas értéke mellett beállítottam egy fejlécet (HeaderIn) és egy csomagot (PacketIn), majd figyeltem a FrameOutV magas értéke mellett megjelenő fejlécet és csomagot. Azt vártam, hogy ugyanazt az adatot kapom vissza, mint amit megadtam. A keretezőt rákötöttem az USRT adó és vevő modulokra, azokat pedig összekötöttem. Ez az összeköttetés a normális működés során nem található meg, ha a WCLOCK felől nézzük, ide kapcsolódik a monitorozó kártya USRT adó és vevő modulja. Az eredményt a 34. ábra mutatja.



34. ábra: A keretező szimulációs tesztje

Legfelül az USRT órajele látható, alatta sorrendben a FrameInV, HeaderIn és PacketIn, végül a FrameOutV, HeaderOut és PacketOut. Látható, hogy miután az USRT adó átküldte az egész keretet, az megjelenik a keretező kimenetén egy FrameOutV impulzus mellett. A keretező tehát megfelelően működik nem csak önmagában, hanem az USRT modulokkal együttműködve.

A következő szimuláció során szándékosan *Magic code* hibát építettem be a keretező adó részébe, az eredményt a 35. ábra mutatja.



35. ábra: A keretező szimulációja *Magic code* hiba esetén

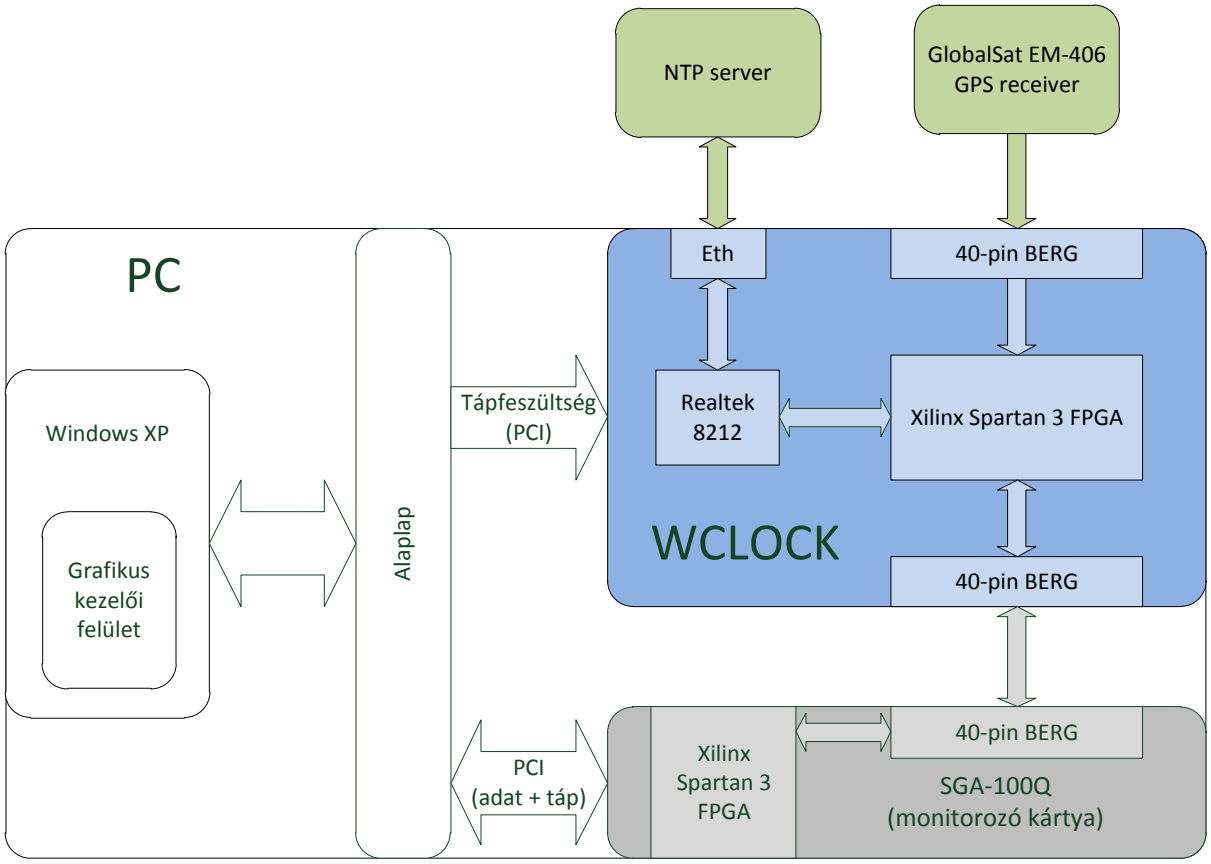
Az látható, hogy 45 µs után (ez az alkalmazott USRT beállítások mellett 2 bájt elküldéséhez szükséges idő) megjelenik egy ValidOut impulzus, ezzel egy időben a *status* vektor értéke nullától különböző. Ez az érték egy, a terven belül rögzített állandó, melyet akkor állít be a keretező, ha olyan keretet kap, aminek az eleje nem a *Magic code*. Ugyanezt teszi, ha hibás ez ellenőrző összeg, csak akkor más értékre állítja be ezt a kimenetet. A vezérlőnek tehát figyeli a ValidOut kimenet mellett ezt a kimenetet is, ha ez nullától különböző, akkor valamilyen hiba történt a vétel során.

5.2 Rendszer szintű tesztelés

Miután megbizonyosodtam a szimulációval ellenőrizhető modulok helyes működéséről, összeállítottam egy tesztelrendezést, melynek a következőket kellett biztosítania:

- szükség volt egy, a monitorozó kártya szerepét betöltő eszközre, amivel a WCLOCK együttműködik,
- ezzel az eszközzel valamilyen magasabb szinten kellett kommunikálnom, mint felhasználó, hogy rögzíthessem a WCLOCK által küldött információkat,
- csatlakoztatnom kellett a WCLOCK kártyához egy GPS vevőt és egy NTP szervert, hogy a funkcionális tesztet is elvégezhessem.

A tesztelrendezést, mely ezeket az igényeket kielégíti, a 36. ábra mutatja.



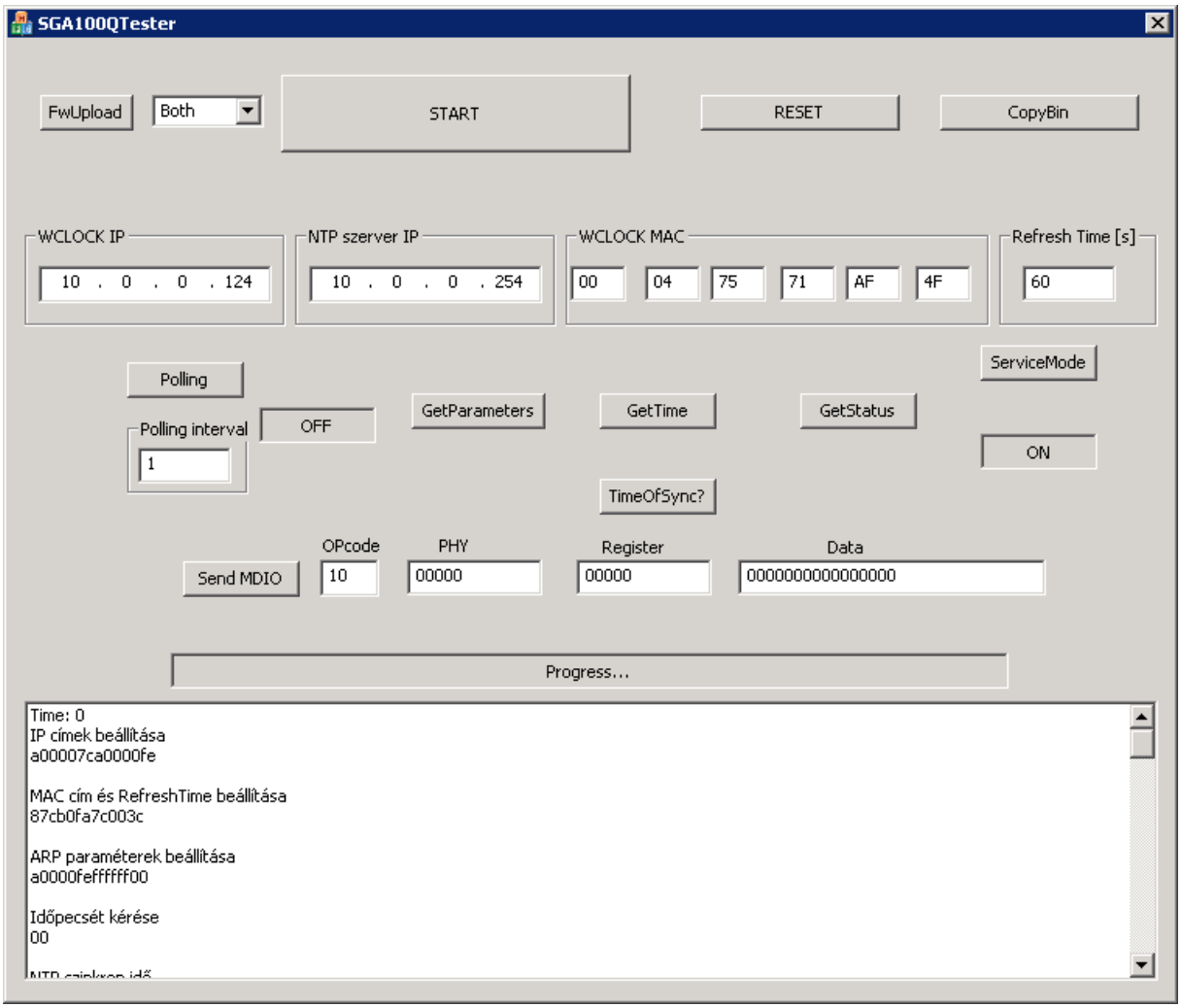
36. ábra: A teszteléshez használt elrendezés

A WCLOCK kártyát egy általános PC-be raktam be, így biztosított volt a tápfeszültség. A monitorozó kártya szerepét az AITIA International SGA-100Q kártyája töltötte be [24]. Ez a cég egy régebbi terméke, egy PCI portos interfészkártya, mely szintén FPGA alapú, rendelkezésemre álltak hozzá a driverek és ugyanolyan *Feature Connector* található rajta, mint a legújabb monitorozó kártyákon. Az én szempontomból tehát teljesen azonos paraméterekkel rendelkezik, mint egy most használatban lévő eszköz. A cég munkatársaitól kapott driver és mintaprogramok alapján létrehoztam egy saját MFC alapú grafikus felületet, mely egyszerű IO hívásokkal kommunikált a kártyával, így minden, a WCLOCK által küldött üzenetet meg tudtam jeleníteni és fájlba tudtam menteni.

Az NTP szerver szerepét egy helyi alhálózaton elérhető, Linuxot futtató gép töltötte be (természetesen a WCLOCK működik bármilyen, interneten elérhető NTP szerverrel, csak az IP címét kell ismerni). A GPS egy GlobalSat EM-406 típusú kompakt egység volt integrált antennával [25]. Ez egy eszközön belülre szánt modul, a kiadott jelszintek LVTTL szabvány szerintiek voltak, így szintillesztés nélkül az FPGA-hoz vezethettem azokat. A későbbiekben kiválasztásra kerül egy kültéri antenna, azonban ez a firmware-t lényegesen nem befolyásolja. Ha netán olyan vevőt kellene illeszteni, ami nem képes NMEA üzenetek küldésére (ez egyébként nem jellemző), akkor csak az NMEA dekódert kellene lecserélni SiRF Binary Protocol dekóderre.

A tesztelés folyamata igen hosszadalmas volt, a következőkben érintőlegesen bemutatom a nagyobb állomásokat, a hangsúlyt az időmérés hatékonyságára fektetem. Először a WCLOCK – monitorozó kártya kommunikációt ellenőriztem. Erre a legkézenfekvőbb lehetőségem az volt, hogy minden keretet, amit a két kártya váltott egymást közt, felküldtem a kezelői felületnek. Amennyiben

érvényes volt a keret fejléce, a program kiírta annak típusát és a keretben levő 64 bites adatot. Minden információt automatikusan szövegfájlba is elmentett, így a későbbiekben visszakereshetőek voltak az eredmények. A grafikus kezelői felületet bekapcsolás után a 37. ábra mutatja.



37. ábra: A kezelői felület indítás után

A felület felső része tesztelési, szerviz célokat szolgál (SGA-100Q kártya felprogramozása, újraindítás, stb.). Az alatta lévő mezőkben lehet megadni az NTP szerver szükséges paramétereit, az ez alatt levő gombok a tesztelés során használt vezérlők, a legalsó mező pedig a Realtek RTL 8212 hálózati interfész IC menedzsment felületéhez készült. A program az alsó ablakba folyamatosan kiírja a monitorozó és a WCLOCK közötti üzenetek fejlécét és az adatot. A 37. ábra egy indítási folyamatot mutat, ahol a monitorozó kártya elküldi a WCLOCK kártyának az NTP-hez szükséges paramétereket.

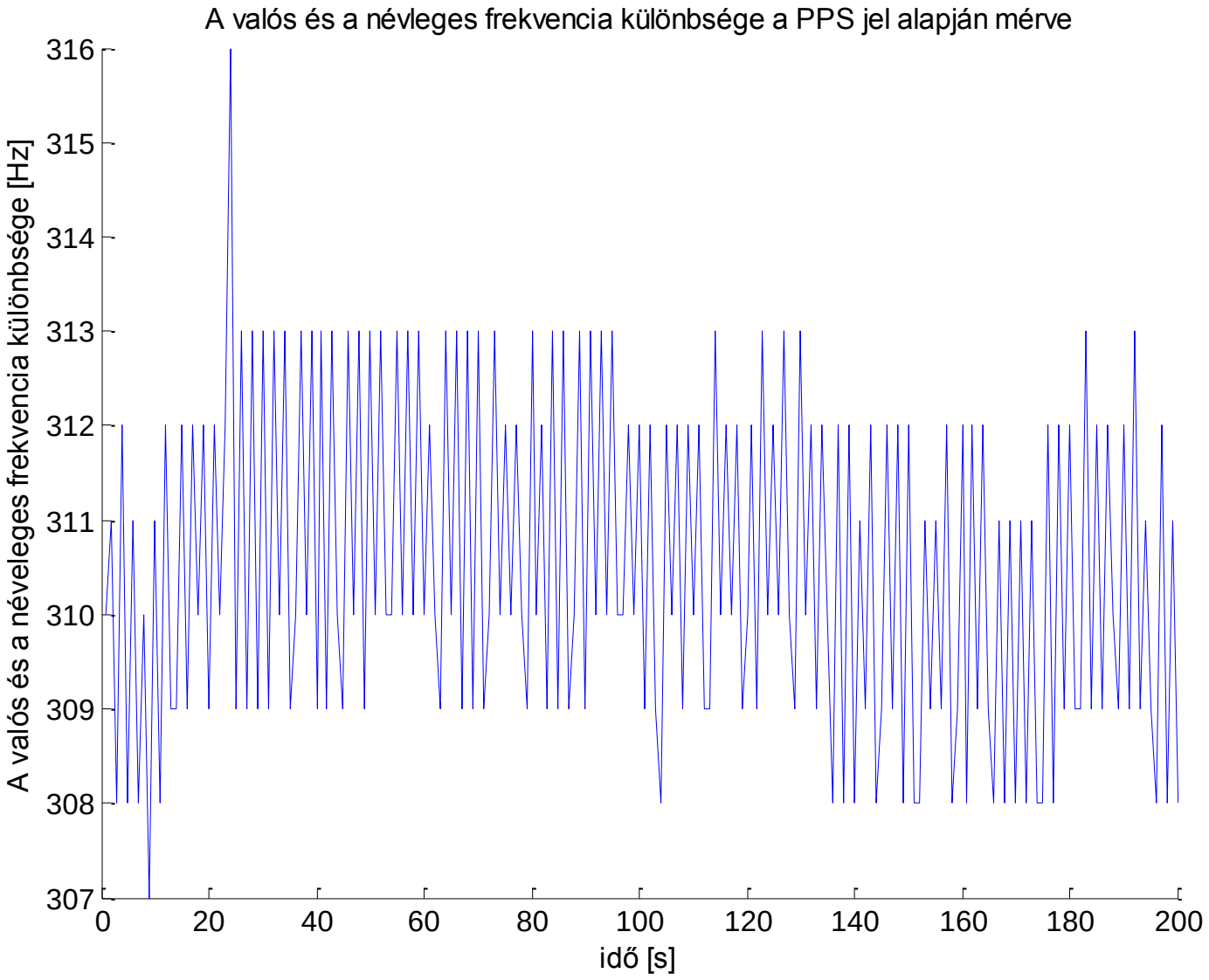
Ezzel a funkcióval bármilyen információt el tudtam küldeni a PC-nek a WCLOCK kártyából, így a továbbiakban csak a tesztelési, mérési eredményeket mutatom be. A feldolgozáshoz szükséges C kódokra és az esetleges teszteléshez szükséges FPGA firmware módosításokra nem térek ki.

5.2.1 A helyi oszcillátor hibájának vizsgálata

A belső időmérő akkor működhet hosszútávon helyesen, ha a belső oszcillátor frekvenciája stabil, azaz a valós érték eltérhet a névleges értéktől, de azt stabilan kell tartania. Ez állandó környezeti paraméterek mellett a 2.3.1 fejezet alapján teljesül, azonban ki is lehet mérni ebben az

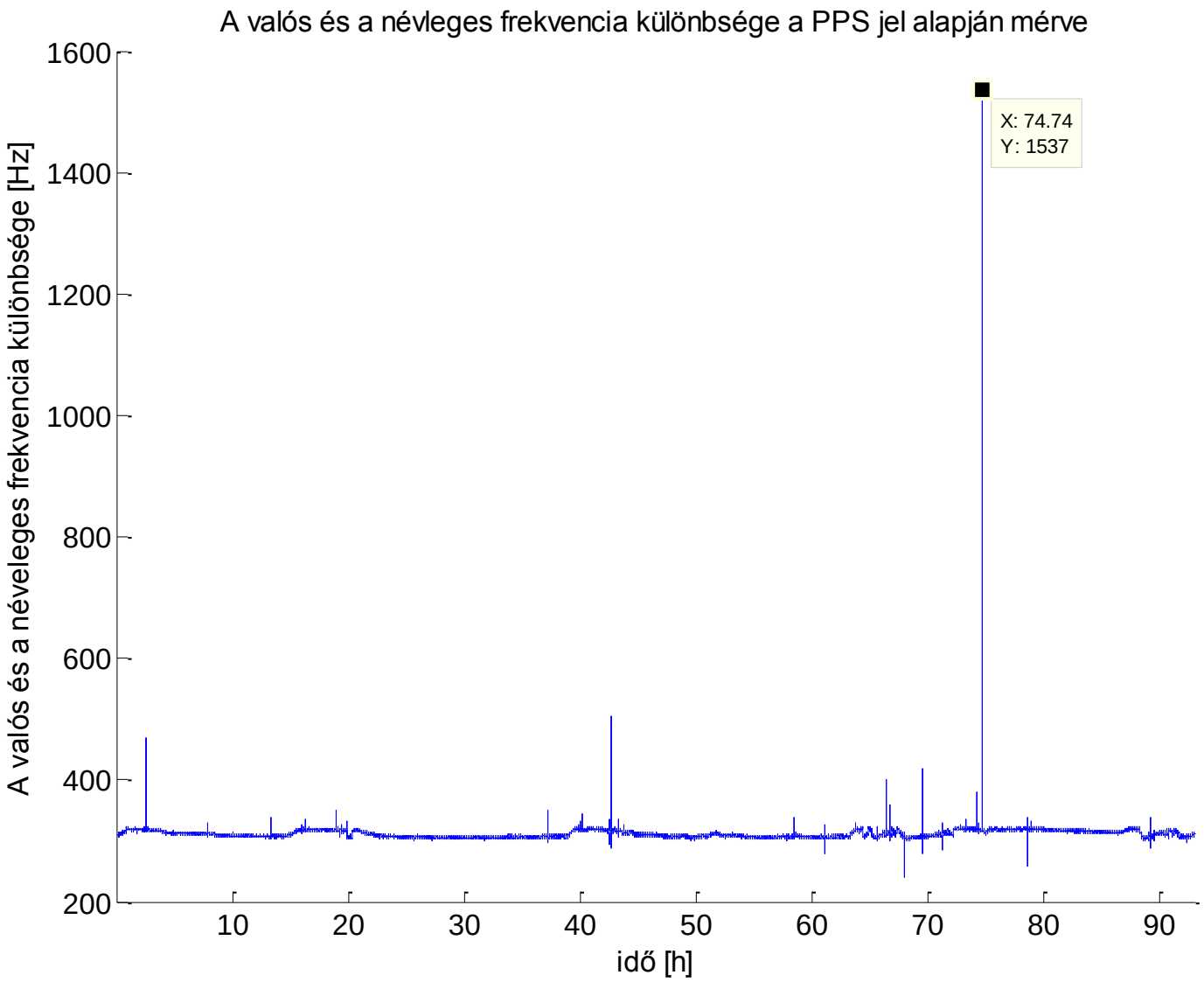
elrendezésben a kvarc frekvenciájának alakulását rövid- és hosszútávon. A PPS jelet kihasználva lehetőség van megszámolni az órajel-periódusok számát másodpercenként mind a WCLOCK kártyán, mind a monitorozó kártyán. A kettő közül a felhasználás szempontjából a monitorozó frekvencia-stabilitása fontosabb, hiszen monitorozáskor a csomagokra kerülő időpecsét a monitorozó kártya órájától származik. Tehát az itt működő időmérésnek ugyanolyan pontosnak kell lennie, mint a WCLOCK kártyán működőnek, ugyanakkor a kettő teljesen egy és ugyanaz a komponens, csak különböző FPGA-n futnak. Ezért az innentől bemutatott eredmények a monitorozó kártyán belüli mérésekre vonatkoznak. A két időmérés viselkedése között mindössze a kezdeti állapotban van különbség, mert a GPS üzenetből származó időpecsétnek át kell jutnia a WCLOCK-tól a monitorozó kártyára az USRT összekötetésen. Ez után viszont egyszerre jár a két óra, hiszen a monitorozó kártya késedelem nélkül megkapja a WCLOCK kártyától a PPS impulzusokat. Ezen okok miatt teljesen egyenértékűnek tekinthetők a WCLOCK kártyán és a monitorozó kártyán futó óra mérési eredményei.

Az monitorozó kártyán egy 50 MHz +/-50 ppm-es órajelforrás működik, azaz névleges frekvencián két PPS jel között $5x10^7$ periódusa kell, hogy elteljen. Ennek kimérésére egy számlálót használtam, amely minden órajelre eggyel növelte az értékét, a PPS jelre pedig nullázta magát, de előtte elküldte az eredményt a PC-nek. Az értékekből kivontam $5x10^7$-t, hogy jobban ábrázolhatóak legyenek. Az eredményt a 38. ábra mutatja.



38. ábra: A valós és a néveleges frekvencia különbsége

Látható, hogy a valós frekvencia valamelyest eltér a névlegestől, átlagosan kb. 311-el többet üt az óra másodpercenként, mint kellene. Ez nagyjából 311/50 = 6,22 ppm, ami bőven a megengedett tartományon belül van. E néhány perces mérés alapján stabilnak mondható a kvarc frekvenciája elenyésző a bizonytalansága. A 39. ábra mutatja ugyanezt a mérést hosszabb távon.

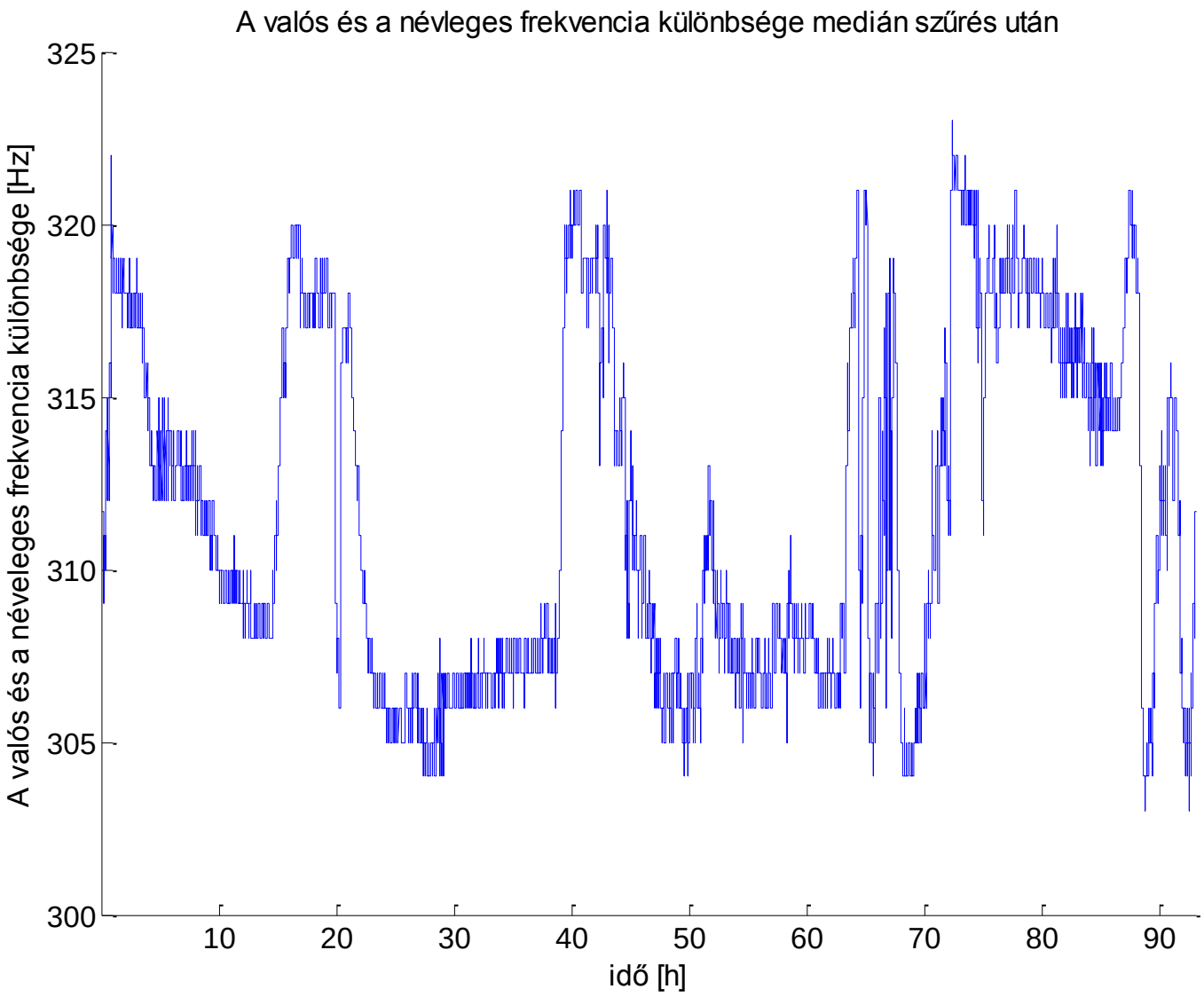


39. ábra: A valós és a névleges frekvencia különbsége több napon keresztül

Látható, hogy napokon keresztül stabilan tartotta a kvarc a frekvenciáját néhány tüskét leszámítva. Ezek egyedi, ritka eltérések, a frekvencia nem tolódott el jelentősen egyik irányba sem. A legnagyobb kiugrás pillanatában a névleges értéktől való eltérés 1537 Hz volt, ami 1537/50 = 30,74 ppm, ez még mindig a megengedett határokon belül van. Ezeknek az egyedi, kiugróan magas hibáknak a magyarázata lehet akár a tápfeszültség ingadozása, vagy a környezeti paraméterek pillanatnyi megváltozása (pl. erős huzat). Ugyanakkor megvan az a kellemes tulajdonságuk, hogy könnyű őket detektálni és kiszűrni. Az időmérő jelentősen elállítódhatna, ha egy ilyen egyszeri hiba alapján mért periódusidővel működne egy másodpercig, aztán újra visszaállna a normális értékre. Ezért a tesztelési tapasztalatok alapján beépítettem egy szűrő jellegű logikát, ami az egyedüli, a

korábbiaktól jelentősen eltérő értékeket eldobja, így nem állítódik el jelentősen a felhasznált periódusidő egy – egy ilyen tüske miatt.

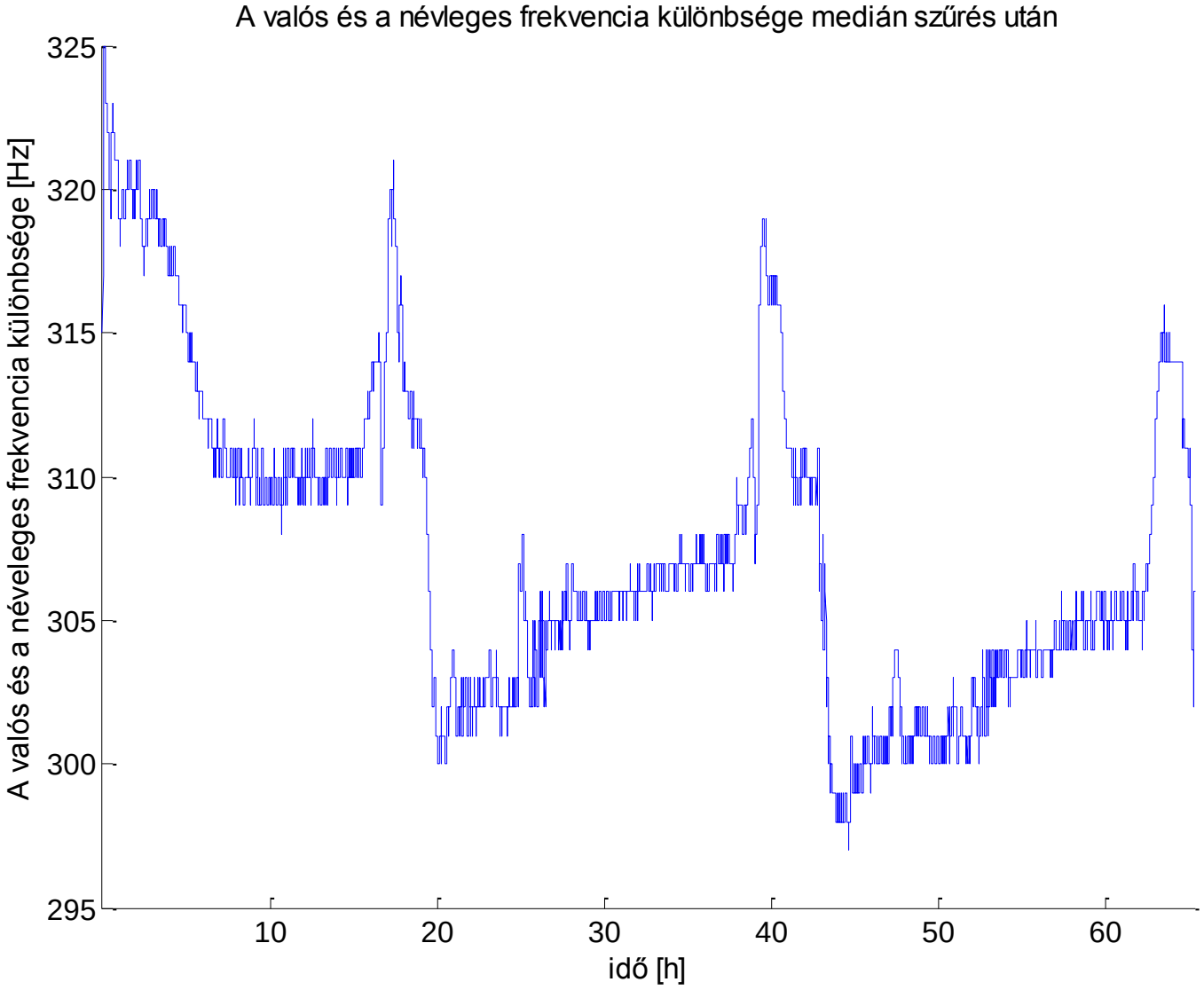
A mérés eredményén mediánszűrést alkalmazva el lehet távolítani a kiugró tüskéket és meg lehet figyelni a környezeti hatások (pl. hőmérséklet) frekvenciára gyakorolt hatását. Ezt mutatja a 40. ábra.



40. ábra: A valós és a névleges frekvencia különbsége mediánszűrés után

Megfigyelhető egyfajta periodikusság: nagyjából 15, 40 és 65 óránál láthatunk egy hirtelen, meredek növekedést a frekvenciák különbségében. Ennek magyarázata a mért objektumok elhelyezkedésében keresendő. A GPS vevő közvetlenül egy erkélyajtó tövében volt a padlón (azért, hogy tiszta rálátása legyen a műholdakra), a teszteléshez használt PC, benne pedig a WCLOCK és a monitorozó kártya a vevő mellett, szintén a padlón volt elhelyezve. Az erkélyajtó nagyjából keletre nézett, a tesztelt eszközökre így közvetlenül rásütött a nap kb. reggel 8-9 óráig, ez látszik a grafikonon. A mérést reggel 9 körül indítottam el, jól megfigyelhető a következő három hajnali napsütés hatása. A harmadik utáni reggel változik a grafikon, a leszálló ág után nem marad 305-310 között az érték, mint addig, hanem hamar megnő 315-320 körülire. Ezt az okozta, hogy aznap

reggeltől én is a helyiségben voltam és kinyitottam az erkélyajtót, így ekkor a légmozgás és a kinti levegő változtatta meg kismértékben a kvarc frekvenciáját. Egy másik, hasonló eredményű mérést mutat a 41. ábra.



41. ábra: Egy másik mérés során tapasztalt frekvencia ingadozás

Érdemes megfigyelni, nagyjából milyen mértékben változik az óra periódusideje csak a hőmérséklet-ingadozásnak köszönhetően. A pontosság igénye nélkül mondjuk azt, hogy a grafikonon 320 és 300 közötti értékeket láthatunk. Egyik esetben 1/50000320 = 19,999872 ns a periódusidő, másik esetben 1/50000300 = 19,99988 ns. Ha az első eset állna fenn, de a második eset periódusideje lenne az időmérés alapja, akkor egy mp alatt a hiba mértéke:

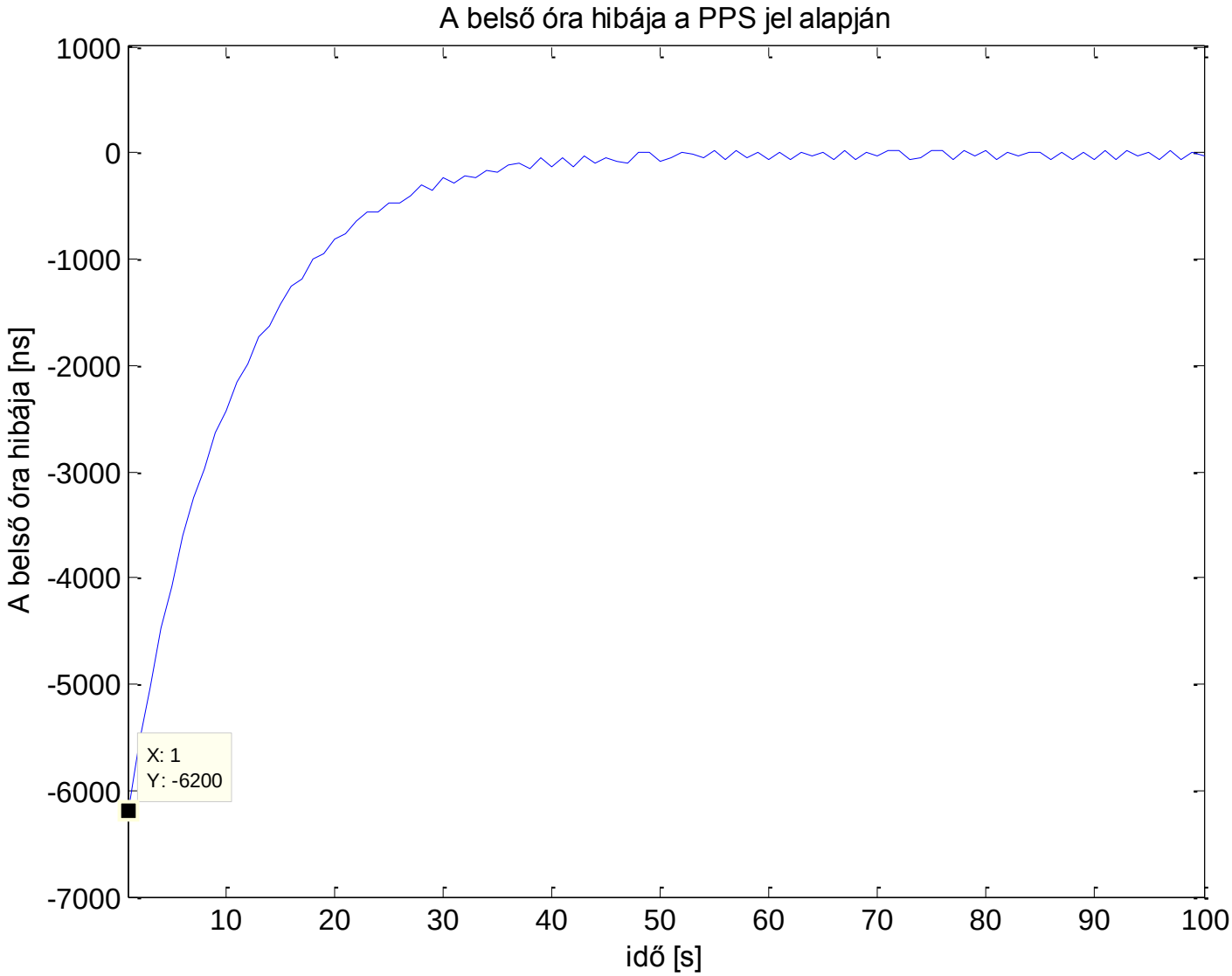
$$10^9 - 50000320 \times 19{,}99988 = 400\ ns$$

Ez már jelentős hiba lenne 10 Gbps Ethernet hálózat monitorozásakor, és ez csak néhány nap hőmérséklet-ingadozása. Ezek a mérések tehát két fontos információval szolgáltak:

- az egyszeri, kiugró változásokat nem szabad figyelembe fenni a periódusidő számításakor,
- jogos a folyamatos periódusidő ellenőrzés, mert az órajel jelentős mértékben változik a hőmérséklet függvényében.

5.2.2 A kompenzált időmérés hatékonysága

Miután kimértem a helyi oszcillátorok tulajdonságait, következett az időmérés működésének vizsgálata. Ami igazán érdekes volt, hogy a PPS jelek vételekor mekkora a belső óra hibája, ezzel lehetett vizsgálni a valós periódusidő számításának működését. Ezért nem a konkrét időpecséteket küldtem el a PC-nek (az nem szolgált volna semmilyen formációval), hanem a PPS jelhez mért különbségeket. Az indítás utáni állapotot a 42. ábra mutatja.



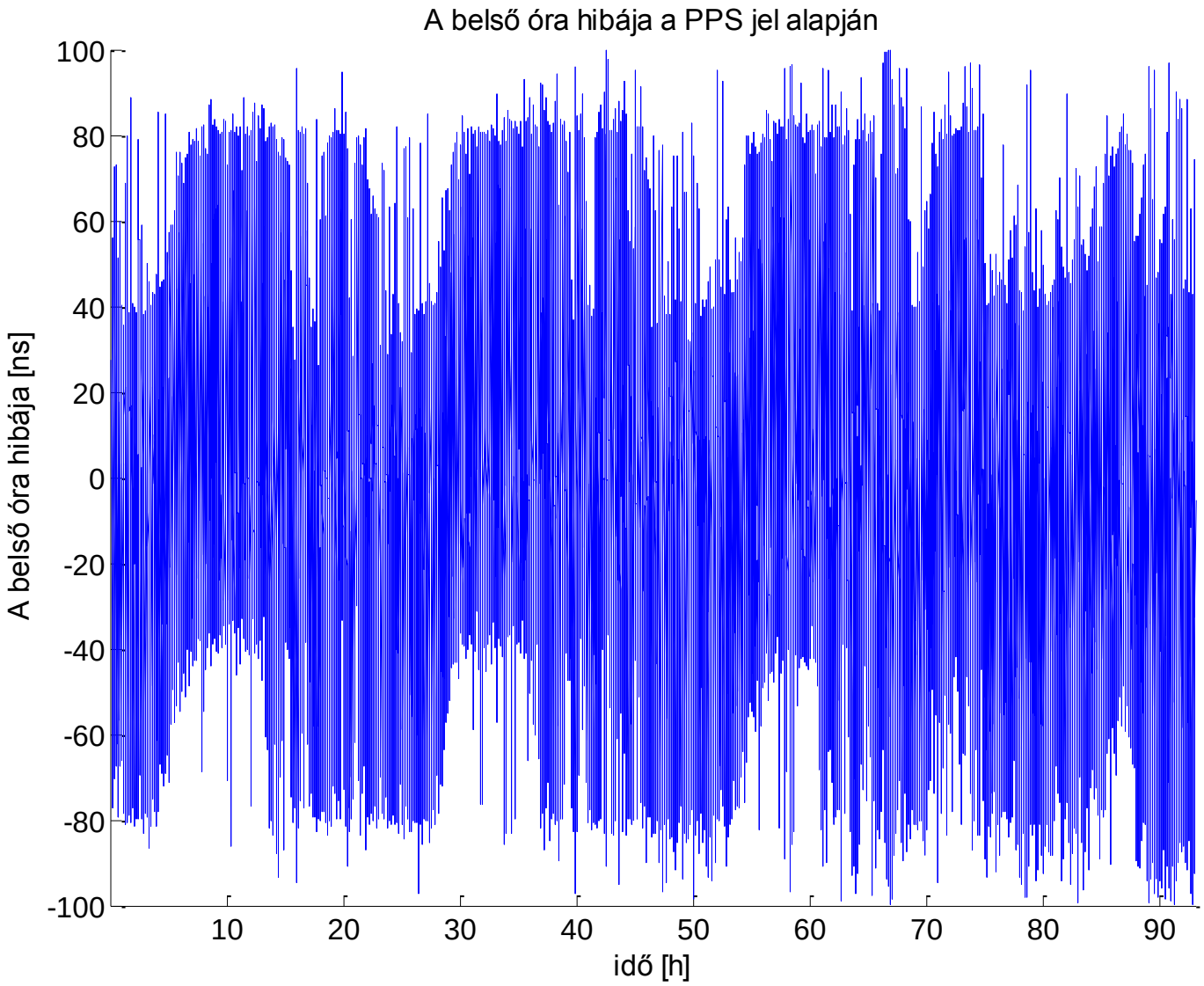
42. ábra: A belső óra hibája indítás után

Az ábráról jól látszik, hogy az eleinte néveleges periódusidővel számoló óra hogyan áll be a PPS jel alapján a valós értékre. Az eddigiek alapján nem tévedünk nagyot, ha a valós órajelet a névleges 50

MHz-nél nagyjából 310 Hz-el nagyobbnak becsüljük. Ha ennyit üt az óra két PPS között, de a névleges periódusidővel számol, akkor egy mp alatt a hiba:

$$10^9 - 50000310 \times 20 = 6200\, ns$$

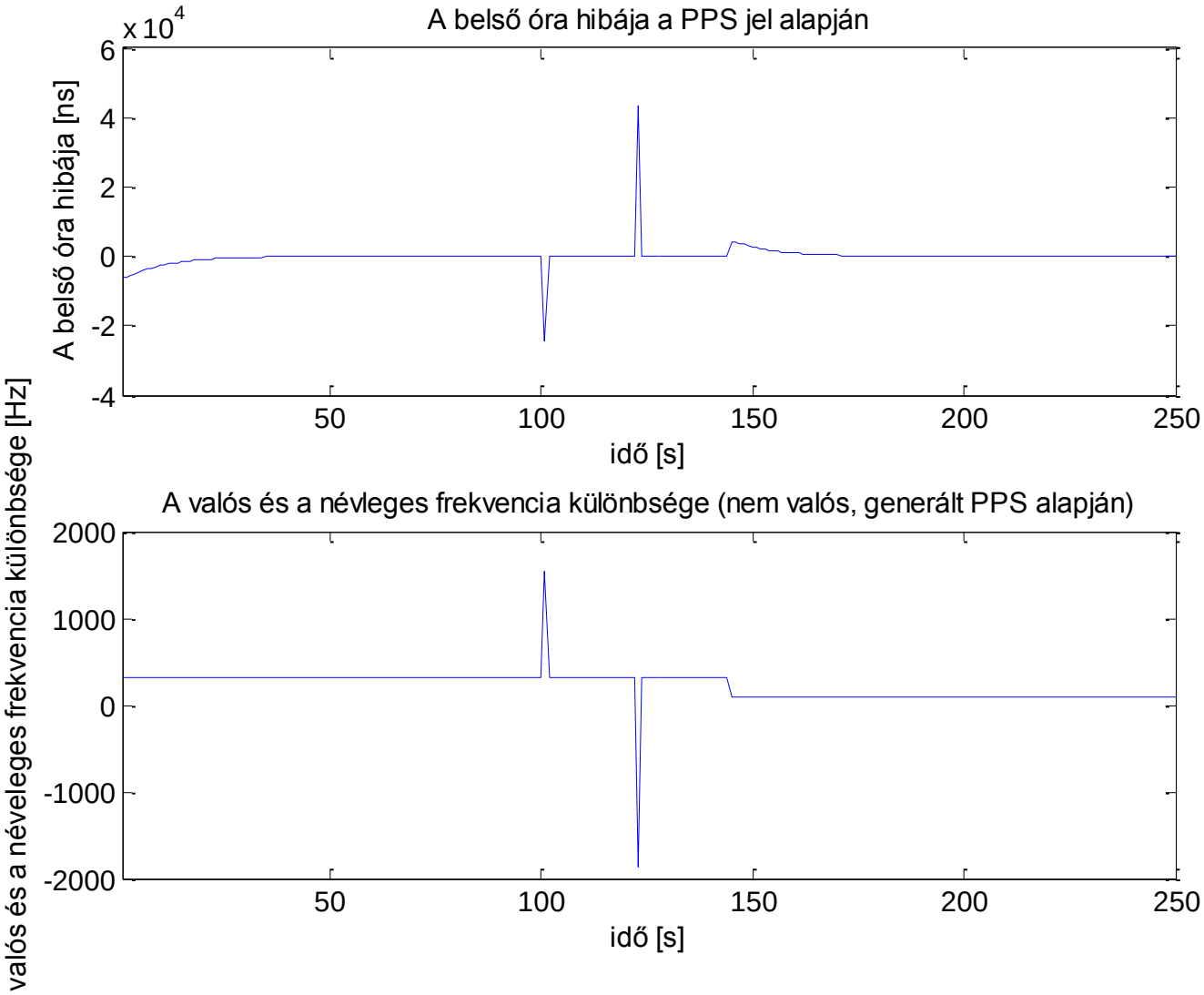
Pontosan ezt az értéket lehet leolvasni az ábráról, mint kezdeti hibát, ami aztán fokozatosan csökken, amíg be nem áll egy nulla körüli értékre. Egy ilyen stabil állapotot mutat a 43. ábra.



43. ábra: A belső óra hibája hosszú távon

Látható, hogy az időmérés napokon át stabilan működött, az abszolút hiba egyszer sem volt 100 ns felett. Ez a hiba elegendő a gyakorlatban akár 10 Gbps Ethernet hálózat monitorozása esetén is, még ha az elméletileg lehetséges legkisebb idő, amivel a csomagok követhetik egymását, kisebb (61,6 ns), mint a garantálható pontosság.

Az egyszeri, kiugró különbségek szűrését egy mesterségesen generált PPS jel segítségével is ellenőriztem. Az időmérő PPS bemenetére a GPS jele helyett egy erre a célra írt komponens kimenetét kötöttem, ez szolgáltatta a szükséges PPS impulzusokat. Az eredményt a 44. ábra mutatja.

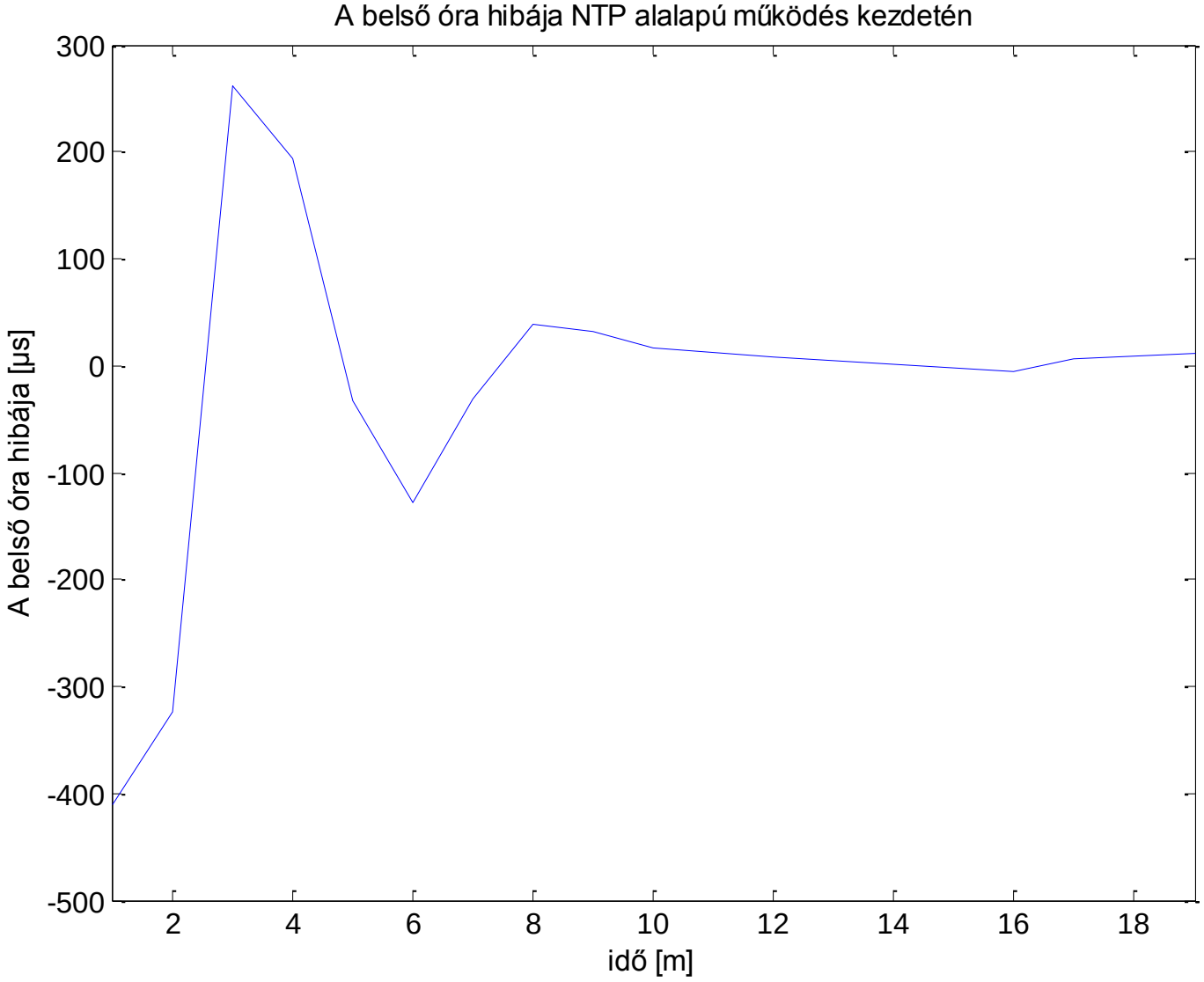


44. ábra: Az időmérő működése mesterséges PPS jel esetén

Látható, hogy a belső óra hibája a teszt elején beáll nullára, és tartja ezt egészen kb. a 100. másodpercig, ahol a két PPS vétele közötti időkülönbség nagymértékben különbözik az eddigiektől. Itt a hiba ennek megfelelő mértékű, utána ahogy a PPS-ek között eltelt idő ismét a korábbiakkal megegyező, a hiba értéke újra nulla lesz. Ha az időmérő nem szűrné ezt az egyszeri hibát, akkor ez alapján átállítaná az időmérés alapjaként szolgáló periódusidő értékét, és a hiba nem lenne utána egyből nulla, hanem exponenciális jelleggel lecsengene. Ugyanez látható, amikor ellenkező irányban mutat nagy, egyedi változást a két PPS között eltelt idő, ezt is sikeresen kiszűri az időmérő. A teszt végén olyan esetet szimuláltam, amikor tartósan megváltozik a helyi oszcillátor frekvenciája (pl. hőmérsékletváltozás miatt), erre is megfelelően reagál az időmérő, látszik, hogy fokozatosan „megtanulja” az új periódusidőt és a hiba beáll nullára.

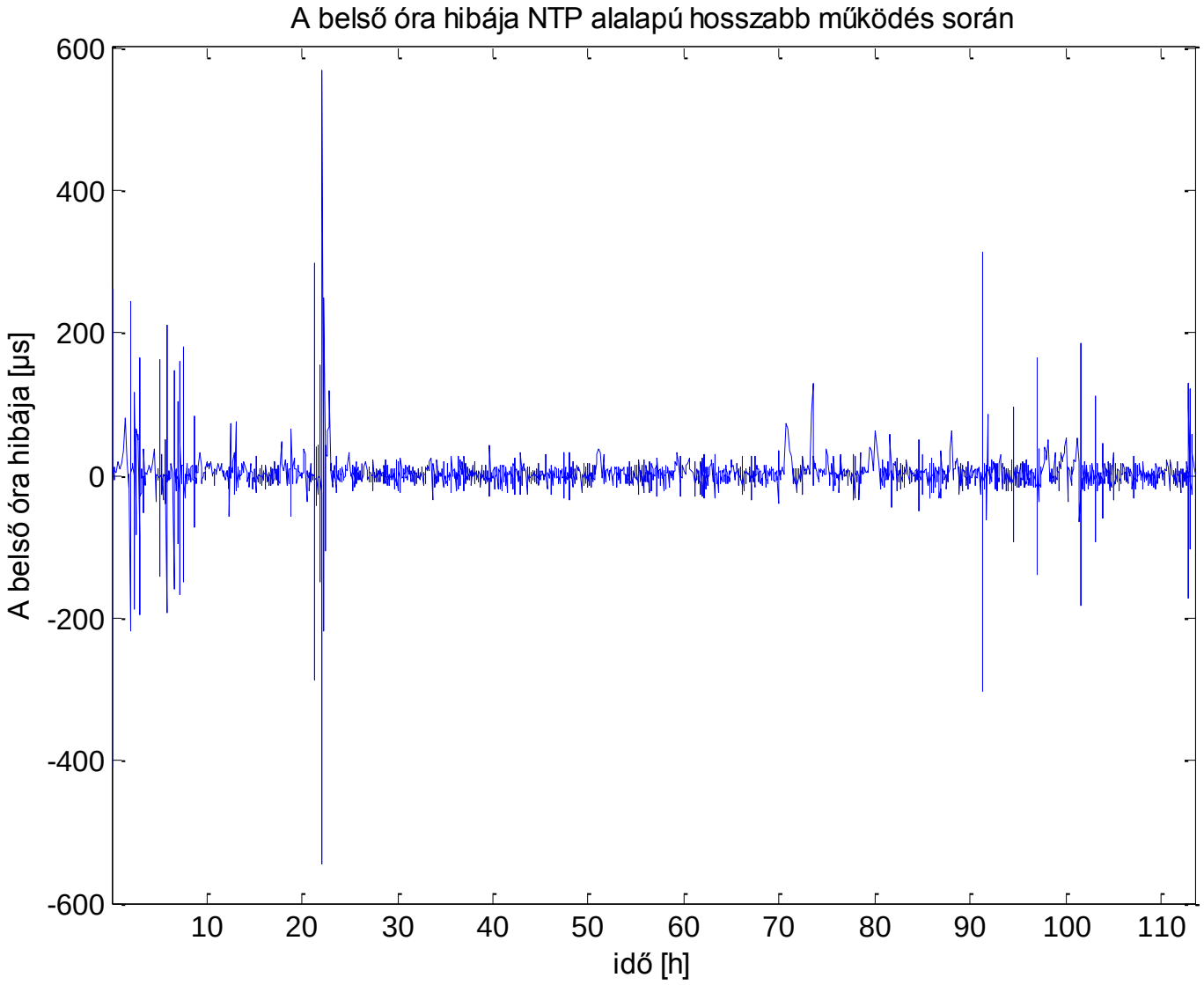
5.2.3 Az NTP alapú működés

A GPS-hez hasonlóan azt is ellenőriztem, hogyan működik együtt a WCLOCK és a monitorozó az NTP szerverrel, pontosságot biztosít így a belső időmérő. A kezdeti állapotot a 45. ábra mutatja.



45. ábra: A belső óra hibája indítás után az NTP üzenetek alapján

Az NTP szervertől percenként kért a WCLOCK időpecsétet, így lassabban tudott csak rászinkronizálni. A pontosság az NTP-ből adódóan itt µs nagyságrendű szemben a GPS ns nagyságrendjével, de szépen látszik, ahogy az óra megtanulja a valódi periódusidőt az NTP szerver üzeneti alapján. Egy hosszabb működésről készült grafikont mutat a 46. ábra.



46. ábra: A belső óra hibája NTP alapú hosszabb működés során

Ennél a mérésnél már csak 15 percenként kért időpecsétet a WCLOCK a szervertől. Az időmérőbe beépített logika 1 percről folyamatosan növeli a kérések között eltelt időt 15 percig, miután már rászinkronizált az NTP szerver idejére. Ha kiesik a szinkronból, újra 1 percről indul a kérések gyakorisága. Így nem terheli le fölöslegesen a szervert, de ha pontatlanságot tapasztal, gyorsabban tud reagálni. Azt látjuk, hogy hosszabb időszakokban néhányszor 10 µs volt az óra hibája, azonban vannak nagyobb különbséget mutató időpontok és úgy általában nagyobb szórás jelentkezik, mint ami a GPS alapú óra esetében megfigyelhető volt. Ez persze természetes, hiszen az NTP szerver egy PC volt, aminek a WCLOCK hálózatból jövő kéréseit fel kellett dolgozni, és válaszolni kellett rá. Ha éppen más feladatai is voltak a gépnek, akkor ezt nyilván jóval lassabban tette meg, mint amikor minden erőforrása szabad volt. Ez a mérés jól mutatja, hogy az NTP (a szoftveres feldolgozás, az operációs rendszerek miatt) nem alkalmas hosszútávon 10 Gbps hálózatok monitorozásánál időpecsét forrásnak. Ez az eredmény így is elég jónak mondható, köszönhetően az üzenetek hardveres feldolgozásának és a hálózati késleltetés hardveres mérésének [19].

6 Összefoglalás, fejlesztési lehetőségek

A dolgozatban bemutattam egy teljes egészében általam tervezett és implementált FPGA alapú megoldást, amely GPS-t használva időforrásként képes nagypontosságú időpecsétet biztosítani hálózatmonitorozó eszközöknek. A PPS impulzusok felhasználásával folyamatos és kellő pontosságú időinformációt nyújt bármilyen, erre felkészített monitorozó eszköznek. A helyi oszcillátor hibáját kompenzálva földrajzilag elosztott rendszerek időforrásaként is alkalmazható, akár 10 Gbps Ethernet hálózatok monitorozása esetén is. Egy korábbi munkámat felhasználva integráltam a firmware-be egy NTP klienst is, így az időpecsételő képes NTP szervert használni időforrásként. Az így jelentkező megnövekedett bizonytalanság teljes egészében az NTP szerver és a hálózati elemek megbízhatatlanságából adódik. A megvalósított funkció egy erre a célra fejlesztett hardveren fut, a WCLOCK bővítőkártyán, melynek tervezésében is fontos részem volt.

A tesztelés során megmutattam a beépített időmérő funkció hatékonyságát. A megoldás a GPS-től (vagy az NTP-től) származó információk alapján képes folyamatosan kompenzálni a helyi oszcillátor néveleges és a valós periódusidejének eltérését. A kompenzált időmérés fontosságát elméleti számításokkal bizonyítottam, melyek tudomásom szerint ebben a formában nem találhatók meg a szakirodalomban. A teszteredmények alapján megfigyelhető egy kvarc frekvenciájának alakulása rövidtávon (néhány perc) és hosszútávon (néhány nap). Ez alapján jogos az a feltételezés, hogy rövidtávon a valós frekvencia ugyan nem pontos, de precíz, hosszútávon azonban a környezeti paraméterek változásának hatására jelentősen is változhat az értéke. Ennek megfelelően az időmérő az egyszeri, nagy kilengéseket kiszűri, a folyamatos változásokat viszont képes a feladat által megkövetelt hibahatáron belül követni. Így az eszköz állandóan szinkronban marad az aktuális rendszeridővel.

Ahhoz, hogy a WCLOCK kártyát együtt lehessen használni a monitorozó kártyákkal, utóbbiak firmware-ét is módosítani kellett, mely a kommunikációs modul és az időmérő modul integrálását jelentette a monitorozó eszközökbe. Így ezeken belül is folyamatosan ugyanaz az időpecsét áll rendelkezésre, mint amit a WCLOCK kérésre elküld a kártyáknak. Ennek az integrációnak a tesztelése folyamatban van az AITIA International Zrt. forgalomban lévő eszközeivel. A monitorozók eddig használt időpecsét-forrásának lecserélése az FPGA-tól a monitorozó szoftverig számos réteget érint, így a vállalat több munkatársának is módosítania kell a monitorozó rendszer általa fejlesztett és karbantartott részét.

A WCLOCK ebben a formában gyakorlatilag egy kész termék, funkcióját teljesen mértékben képes betölteni. Meg lehet fontolni más globális navigációs rendszer használatát is időforrásként, ezek azonban vagy nem üzemelnek még (pl. az európai Galileo vagy a kínai Compass), vagy csak korlátozottan elérhetőek (pl. az orosz GLONASS). Ipari igény esetén lehetőség van az NTP mellett egy PTP-t (Precision Time Protocol) megvalósító funkció kialakítására is. További esetleges fejlesztési lehetőségként felmerült egy Stratum 1-es NTP szerver szolgáltatás megvalósítása, így a WCLOCK a GPS-t használva a lehető legnagyobb pontosságú NTP szerverként viselkedhetne.

7 Hivatkozások

[1] SGA Interface Cards, AITIA International Zrt. http://www.aitia.hu/eng/telecom/products/interface_cards [2011. október]

[2] IEEE Standard 802.3z, Gigabit Ethernet, 1998 http://standards.ieee.org/findstds/standard/802.3z-1998.html [2011. október]

[3] IEEE Standard 802.3ab, Gigabit Ethernet over copper, 1999 http://standards.ieee.org/findstds/standard/802.3ab-1999.html [2011. október]

[4] IEEE Standard 802.3ae, 10 Gigabit Ethernet, 2002 http://standards.ieee.org/findstds/standard/802.3ae-2002.html [2011. október]

[5] IEEE Standard 802.3an, 10 Gigabit Ethernet over copper, 2006 http://standards.ieee.org/findstds/standard/802.3an-2006.html [2011. október]

[6] IEEE Standard 802.3ba, 40 and 100 Gigabit Ethernet, 2010 http://standards.ieee.org/getieee802/download/802.3ba-2010.pdf [2011. október]

[7] IEEE P802.3ba Task Force 5 Criteria, 2008 http://www.ieee802.org/3/ba/PAR/P802.3ba_5C_0908.pdf [2011. október]

[8] Intel White Paper, 10 Gigabit Ethernet Technology Overview, 2003 http://www.intel.com/network/connectivity/resources/doc_library/white_papers/pro10gbe_lr_sa_wp.pdf [2011. október]

[9] John D'Ambrosia, David Law, Mark Novell, *40 Gigabit Ethernet and 100 Gigabit Ethernet – Technology Overview*, Ethernet Alliance, June 2010 http://www.ethernetalliance.org/files/document_files/40G_100G_Tech_overview.pdf

[10] Elliot D. Kaplan, Christopher J. Hegarty (Editors), *Understanding GPS – Prnciples and Applications*, Second Edition, Artech House, Norwood, 2006

[11] David W. Allan, Neil Ashby, Clifford C. Hodge, *The Science of Tiemkeeping*, Application Note 1289, Hewlet Packard, USA June 1997 http://www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf [2011. október]

[12] GPS and CDMA time synch modules, Endace http://www.endace.com/gps-and-cdma-receivers.html [2011. október]

[13] Accuracy of the Network Time Synchronization, Research for Netwrok, CES.NET http://www.ces.net/project/qosip/publications/2003/ntp-meas/ [2011. október]

[14] Trimble Acutime Gold GPS Smart Antenna Datasheet http://trl.trimble.com/docushare/dsweb/Get/Document-366428/022542-002_Acutime_DS_0207_lr.pdf [2011. október]

[15] John R. Big, *Introduction to Quartz Frequency Standards - Accuracy, Stability, and Precision*, IEEE UFFC, October 1992 http://www.ieee-uffc.org/frequency_control/teaching.asp?vig=vigaccur [2011. október]

[16] *Fundamentals of Quartz Oscillators*, Application Note 200-2, Hewlett Packard, 1997, USA http://www.leapsecond.com/pdf/an200-2.pdf [2011. október]

[17] *SiRF Binary Protocol – Reference Manual Rev 2.4*, SiRF Technology Inc, 2008, San Jose, USA http://gpsd.googlecode.com/files/SiRF-SiRF-v2_4.pdf [2011. október]

[18] *NMEA Reference Manual*, SiRF Technology Inc, 2008, San Jose, USA http://www.usglobalsat.com/store/downloads/NMEA_commands.pdf [2011. október]

[19] Fendler Tamás, *NTP alapú idő-szinkronizáció megvalósítása FPGA-val*, BME TDK 2010

http://tdk2010.aut.bme.hu/Files/TDK2010/NTP-alapu-idoszinkronizacio.pdf [2011. október]
[20] Gplanar interface card overview, AITIA International Inc.
http://www.fpganetworking.com/gplanar/ [2011. október]
[21] SGA-10GED interface card overview, AITIA International Inc.
http://www.fpganetworking.com/10ged/ [2011. október]
[22] EuroQuartz 3EQHM57 oscillator Data Sheet
http://www.farnell.com/datasheets/75562.pdf [2011. október]
[23] FNEPON125 125 MHz Oscillator Data Sheet
http://www.pericom.com/pdf/datasheets/se/FNEPON125.pdf [2011. október]
[24] SGA-100Q: 10/100 Ethernet Card Introduction, AITIA International Zrt.
http://www.aitia.hu/eng/telecom/products/sga-100q [2011. október]
[25] EM-406 GPS Receiver User Manual, GlobalSat Co.
http://www.globalsat.com.tw/globalsat_admin/new_file_download.php?Pact=FileDownLoad&Pval=991 [2011. október]