

SZÉCHENYI ISTVÁN EGYETEM - MŰSZAKI TUDOMÁNYI KAR

AUTOMATIZÁLÁSI TANSZÉK

Impulzusválasz mérés MLS módszerrel

TMDK dolgozat

Készítette:

Dömötör Zénó

Belső konzulens: Torda Béla

Külső konzulens: Dr. Kovács Lóránt

Győr

2012

Tartalomjegyzék

1	Bevezetés, technológiai motiváció.....	4
1.1	Rendszerjellemző függvények mérésére használható eljárások.....	5
1.1.1	Frekvenciatartományi módszerek	5
1.1.2	Időtartományi módszerek.....	6
2	Az MLS mérési módszer	7
2.1	Az MLS tulajdonságai és előállítása	7
2.1.1	Definíció	7
2.1.2	MLS előállítása	8
2.1.3	Az MLS tulajdonságai	11
2.2	Az MLS felhasználása impulzusválasz méréséhez	12
2.2.1	Időtartománybeli átlapolódás (Time aliasing)	12
2.2.2	Impulzusválasz-számítás MLS-gerjesztés segítségével	13
2.3	Gyors Hadamard-transzformáció	15
2.3.1	Az Hadamard-tr. felhasználása keresztkorreláció számításhoz.....	16
2.3.2	A permutációs mátrixok meghatározása	17
3	Megvalósítás real-time eszközökön	19
3.1	A megvalósításhoz használt LabVIEW szoftver bemutatása	19
3.1.1	LabVIEW fejlesztőkörnyezet	20
3.2	A megvalósításhoz használt hardverek bemutatása	25
3.2.1	NI USB-6211	25
3.2.2	NI cRIO-9074	26
3.3	A hálózat-analizátort megvalósító VI működése	27
3.3.1	Mérésadatgyűjtő VI.....	28
3.3.2	Számításokat végző VI (Main_RT_level.vi)	30
3.3.3	A hálózat-analizátor font panelje.....	32
4	Mérési eredmények	34

4.1	Soros RLC-kör	34
4.2	RC-tag.....	37
4.3	Hangszóró-mikrofonmérés	38
5	Eredmények értékelése, további fejlesztési lehetőségek.....	40
	Irodalomjegyzék.....	41
	Ábrajegyzék.....	42
	Függelék.....	43

1 Bevezetés, technológiai motiváció

Napjainkban egyik leggyakoribb mérnöki feladat a rendszeridentifikáció: ismeretlen rendszereket leíró rendszerjellemező függvények (modellek) meghatározása. A rendszeridentifikáció szükségszerűsége származhat abból, hogy a szóban forgó függvény túl bonyolult a pontos fizikai leíráshoz, illetve lehet oka ennek állandó változása is (adaptív rendszerek).

A rendszerek leírása történhet frekvencia- és időtartományban, természetesen ez a kettősség figyelhető meg a rendszeridentifikáció területén is: az időtartományi rendszeridentifikáció célja az impulzusválasz meghatározása, míg a frekvenciatartományi módszerek a pólus-zérus elrendezés meghatározására törekszenek.

A dolgozat célkitűzése egy kiemelkedő tulajdonságokkal rendelkező hálózatanalizátor (rendszer identifikációt végző) berendezés kifejlesztése. A legtöbb kihívást talán az akusztikai rendszerek identifikációja jelenti, a rendkívül hosszú tartójú és bonyolult impulzusválaszok előfordulása miatt, így konkrétan a következők teljesítését tűztük célul:

- Időtartományi identifikáció (ennek legfőbb előnye, hogy nemparametrikus mérést tesz lehetővé: közvetlenül az impulzusválaszt adja, amelyből bármelyik parametrikus modell (akár frekvencia-, vagy idő-frekvencia tartományi) egyszerűen származtatható;
- Jó zavarvédetség, jó jel-zaj viszony;
- 1-csatornás mérés (a meghatározóan drága mérőmikrofonok számát érdemes akusztikai rendszerek mérésekor csökkenteni);
- Gyors, valós idejű működés (az impulzusválasz hossza meghatározza a minimális mérési időt, ezalatt elvárás az impulzusválasz kiszámolása)

A fenti célokat a maximális hosszúságú álvéletlen sorozatokat (Maximum Length Sequences – MLS), mint gerjesztőjelet alkalmazó mérési elvvel lehet megvalósítani.

A mérési elv konkrét realizálásához valós-idejű működésre alkalmas HW eszközöket választottam: az analízátort NI cRIO-9074-re implementáltam.

1.1 Rendszerjellemző függvények mérésére használható eljárások

Az átviteli jellemzők mérésére igen sokféle használható eljárás létezik, alapvetően időtartományi (impulzusválasz) és frekvenciatartományi (pólus-zérus elrendezés, vagy Bode-diagram) módszerek ismeretesek. Az alábbiakban a lehetséges alternatívákat tekintjük át.

1.1.1 Frekvenciatartományi módszerek

a) Bode-diagram hagyományos felvétele

A rendszert előre megválasztott frekvenciájú és amplitúdójú szinuszokkal gerjesztjük, a rendszer kimenő kapcsain – követőszűrő alkalmazásával – amplitúdót és fázist (bemenethez képest) mérünk. Az eljárás követőszűrővel jó jel-zaj viszonyt biztosít, ugyanakkor kétcsatornás mérőműszerre van szükség (pl. vektor-voltmérő), és a mérés igen körülményes, hosszú ideig tart, pláne, ha sok pontban szeretnénk mérni az átvitelt.

b) TDS eljárás – multi-színusz

Az előbbi megoldás automatizált változatának nevezhető az angol irodalomban TDS-ként emlegetett (Time Delay Spectrometry) eljárás: egy lassan változó frekvenciájú FM-jellel „pásztázzuk” végig a vizsgálatunk tárgyát képező frekvenciatartományt. A mérés így gyorsan elvégezhető, de a két csatorna, és a követőszűrő továbbra is szükséges.

c) Kétcsatornás FFT-analizátor

A digitális-technika térhódításával lehetővé vált a gyakorlatban kielégítő sebességgel működő Fourier-analizátorok megvalósítása. A mintavételi frekvencia (sávszélesség, időfelbontás) és az FFT-pontszám (frekvenciafelbontás) megválasztása komoly megfontolásokat igényel. Korlátot a számítási kapacitás jelent.

A mérés során az analizátor egyik bemenetére a fehér-zajszerű gerjesztőjelet, a másik csatornájára a mért válasz-jelet vezetjük, s a spektrumok hányadosa adja az átviteli függvényt (komplex formában).

Előnyként a pontosság, hátrányként a nagy számítási kapacitású HW, és a szükséges két mérőcsatorna említhető.

1.1.2 Időtartományi módszerek

Adott rendszer impulzusválasz mérésére közvetett és közvetlen módszert alkalmazhatunk. A közvetlen módszer Dirac-delta gerjesztéssel működik, mely egyetlen végtelen rövid ideig tartó végtelen nagy impulzusból áll. A meghatározás alapján látszik, hogy valós rendszereknél ezt csak megközelíteni tudjuk, ekkor viszont a kicsi energiatartalom miatt a jel-zaj viszony rossz lesz.

A közvetett módszer olyan mérőjelet használ, amelyből számítással határozható meg az impulzusválasz. Az ilyen módszer az impulzust időben szétteríti és így több energiát képes a rendszerbe táplálni, amelynek következménye a jobb jel-zaj viszony.

a) Mérés Dirac-delta impulzussal

Ismert tény, hogy gerjesztésként Dirac-deltát alkalmazva a rendszer kimenetén a rendszer impulzusválasza megfigyelhető. Az elv gyakorlati kivitelezésénél a Dirac-delta előállítás, és annak igen csekély energiatartalma, ugyanakkor nagy amplitúdója jelent problémát. A csekély energiatartalomból fakadó rossz jel-zaj viszony mérsékelhető sok mérés átlagolásával (Gauss-zajt feltételezve).

b) MLS módszer

Az MLS-sel végzett átvitel-mérés legközelebb a Fourier-analizátoros megoldáshoz áll. Lényegében abban különbözik tőle, hogy a rendszerre adott gerjesztőjel álvéletlen zaj, tehát determinisztikus jel, így elegendő a rendszerünk válaszát megmérni, a gerjesztés ismeretében az átvitel számítható. A mérőjel megválasztásától függ a mérés számításigénye. A maximális hosszúságú álvéletlen sorozatok jellemzői nagyon kedvezőek számunkra.

2 Az MLS mérési módszer

Az impulzusválasz meghatározására a korábban bemutatott módszerekből az MLS módszert választottuk, mivel az adott feladat szempontjából ennek vannak a legjobb tulajdonságai. A feladat célkitűzései között szerepel a valósidejű megvalósítás, melynek érdekében a gerjesztőjelből és válaszjelből keresztkorreláció helyett (amely nagy számításigényű lenne) gyors Hadamard transzformációval számoljuk ki az impulzusválaszt.

2.1 Az MLS tulajdonságai és előállítása

Ahogy arra a fentiekben utaltunk a mérendő rendszer gerjesztőjeként MLS jelet használunk, amely néhány egyszerű lépéssel olyan formába hozható, hogy az impulzusválasz gyorsan számolható legyen. Előbb azonban nézzük meg az MLS jel definícióját, előállítását valamint tulajdonságait, melyeket később a számításainknál kihasználunk.

2.1.1 Definíció

MLS-nek nevezzük a $2^m - 1$ ($m = 1, 2, 3, \dots$) szerint periodikus kétállapotú $(-1, +1)$ álvéletlen jelet, melynek autokorrelációs függvénye:

$$\phi_{ss}(0) = 1$$

$$\phi_{ss}(n) = \frac{-1}{L}$$

A digitális számítógépek világában az egyik legelterjedtebb véletlenszám- ill. zajgenerálási módszer a visszacsatolt shift-regiszterrel létrehozott MLS. Bár léteznek más technikák is, az MLS mégis számos előnnyel bír. Melyek ezek? A legfontosabb tulajdonságuk a fent említett speciális autokorrelációs függvényük. Ez egy igen csekély DC-hibától eltekintve tökéletes periodikus impulzus-sorozat. (Vagyis a jel mintái teljesen korrelálatlanok.) A teljesítmény spektrum tehát teljesen egyenletes eloszlású, eltekintve a zérus frekvenciától. Az MLS paraméterei tehát mintegy fehér-zaj tulajdonságúak, mégis determinisztikus, jól reprodukálható jelről van szó. (Amennyiben a mérőjelem determinisztikus, akkor azt nem kell mérnem. Vagyis az építendő analízátorom – már látszik – egy csatornás lesz.) Az MLS másik nagy előnye, hogy egyszerűen generálható, akár célhardveren, akár általános célú számítógépen. Továbbá, mivel a sorozat értékkészlete bináris, ezért a keresztkorreláció (fontosságát lásd később) számítása

nagyon leegyszerűsödik: Ha a két állapot $+1$ és -1 lehet, akkor nincs szükségünk szorzásra, csak összeadásra és kivonásra, ami a legtöbb HW esetén jóval gyorsabban végezhető művelet, mint a szorzás.

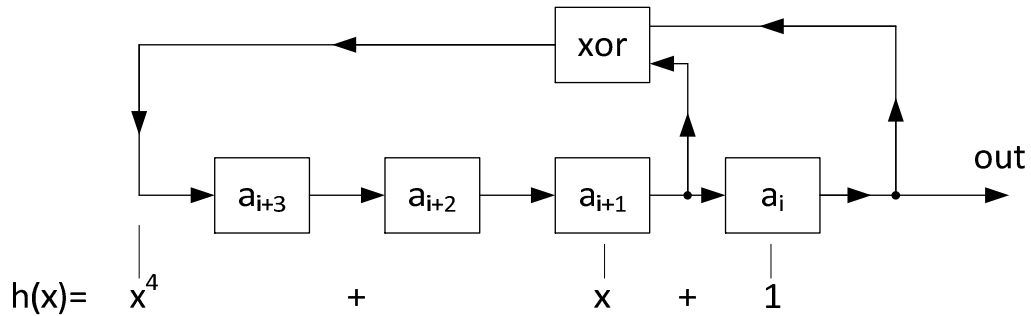
2.1.2 MLS előállítás

Egy $L = 2^m - 1$ hosszúságú MLS-sorozat előállításához szükség van egy m -ed fokú úgynevezett primitív polinomra, jelölje ezt $h(x)$ (forrás: (1)). Bizonyított tény, hogy minden m foksámhoz található legalább egy primitív polinom az 1. ábra-n felsoroljuk az $m = 1$ és $m = 20$ -ad rendű közötti primitív polinomokat.

m	$h(x)$ (primitív polinomok)
1	$x + 1$
2	$x^2 + x + 1$
3	$x^3 + x + 1$
4	$x^4 + x + 1$
5	$x^5 + x^2 + 1$
6	$x^6 + x + 1$
7	$x^7 + x + 1$
8	$x^8 + x^6 + x^5 + x + 1$
9	$x^9 + x^4 + 1$
10	$x^{10} + x^3 + 1$
11	$x^{11} + x^2 + 1$
12	$x^{12} + x^7 + x^4 + x^3 + 1$
13	$x^{13} + x^4 + x^3 + x + 1$
14	$x^{14} + x^{12} + x^{11} + x + 1$
15	$x^{15} + x + 1$
16	$x^{16} + x^5 + x^3 + x^2 + 1$
17	$x^{17} + x^3 + 1$
18	$x^{18} + x^7 + 1$
19	$x^{19} + x^6 + x^5 + x + 1$
20	$x^{20} + x^3 + 1$

1. ábra. MLS jelet előállító shift regiszteres kapcsolást definiáló primitív polinomok

Ezt a polinomot úgy fogjuk fel, mint amely leír egy visszacsatolt shift-regiszter struktúrát, amely m késleltetőt tartalmaz, és a polinomnak megfelelő pontokon visszacsatolást tartalmaz. A bemenetre kerülő összeg modulo 2 (XOR) aritmetikával számítandó (2. ábra).



2. ábra. MLS jelet előállító shift regiszteres kapcsolás

Példaként vegyük a $h(x) = x^4 + x + 1$ polinomot. Ennek megfelel a 2. ábra-n látható kapcsolás, az ábrán lévő négyzetek egy-egy késleltetőt reprezentálnak. Ha példaként az ábrán lévő kapcsolásban szereplő 4 db késleltető az i . időpillanatban rendre:

$$a_{i+3} \quad a_{i+2} \quad a_{i+1} \quad a_i$$

értékeket tartalmazza, akkor az $i + 1$. időpillanatban az:

$$a_{i+4} = a_{i+1} \oplus a_i \quad a_{i+3} \quad a_{i+2} \quad a_{i+1}$$

értékeket fogja (az XOR művelet jele: $\oplus$). Más szavakkal: a visszacsatolt shift-regiszteres kapcsolás egy végtelen $a_i, a_2, \dots, a_n$ sorozatot állít elő az $a_{i+4} = a_{i+1} \oplus a_i$ rekurzió szerint.

Az m késleltetőt tartalmazó shift-regiszternek elméletileg összesen 2^m állapota lehet. A generált sorozat ebből következőleg biztosan periodikus lesz. Ha minden tároló tartalma zérus, akkor a kimeneten 1-es érték nem jöhet létre ($0 \oplus 0 = 0$), a rendszer "megfagy", ezért ennek létrejöttét el kell kerülni. Ezt biztosítja a primitív polinom által definiált visszacsatolás. Ha a regiszter csupa 0-ás állapotát kizárjuk, a kimeneten $2^m - 1$ hosszúságú periodikusan ismétlődő jelsorozatot kapunk. (Mivel ennél hosszabb periódusú sorozat adott hosszúságú shift-regiszterrel nem állítható elő, ezért nevezzük az így generált $2^m - 1$ hosszú sorozatot maximális hosszúságúnak.)

A fentiekből következik, hogy a shift-regiszter bármelyik belső – nem csupa 0 – állapotával inicializálható, a jelfolyamban különbség csak egy kezdeti eltolás formájában figyelhető meg.

A 3. ábra-n lévő táblázatban szerepel a 4-ed rendű shift-regiszteres kapcsolásban lévő shift regiszterek kimenetei az egyes iterációkban, valamint az MLS jel aktuális értéke. Az $m = 4$ esethez: $2^m - 1 = 2^4 - 1 = 15$ állapot tartozik, így az létrejövő jel maximális

hossza 15-nek adódik. Az ábrán megfigyelhető, hogy 15 iteráció után ismétlődik a mintázat.

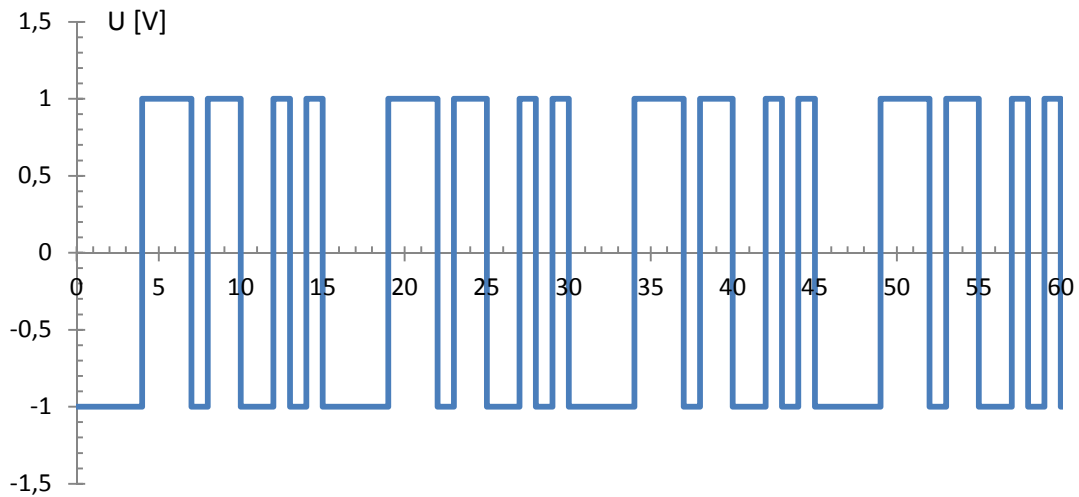
Lépés	Shift regiszterek kimenetei $a_{i+3}, a_{i+2}, a_{i+1}, a_i$	MLS jel (a_i kimenete)
0	1 1 1 1	1
1	0 1 1 1	1
2	0 0 1 1	1
3	0 0 0 1	1
4	1 0 0 0	0
5	0 1 0 0	0
6	0 0 1 0	0
7	1 0 0 1	1
8	1 1 0 0	0
9	0 1 1 0	0
10	1 0 1 1	1
11	0 1 0 1	1
12	1 0 1 0	0
13	1 1 0 1	1
14	1 1 1 0	0
15	1 1 1 1	1
16	0 1 1 1	1
17	0 0 1 1	1
...	...	...

3. ábra. MLS elemek sorozata – $m = 4$ esetén

A maximálisan hosszú sorozatot előállító egység kimenete 0 vagy 1 értékű lehet. A 0-kból és 1-kből a mérőjelet az alábbi megfeleltetés szerint állítjuk elő:

- 1 esetén, az analóg kimeneten: -1 V-ot küldünk ki,
- 0 esetén, az analóg kimeneten: $+1$ V-ot küldünk ki.

A 4. ábra-n látható a -1 V-ból és $+1$ -ből álló gerjesztőjel. Megfigyelhető, hogy a gerjesztőjel 15 elemenként periodikus, viszont egy 15 elemből álló perióduson belül nincs ismétlődés.



4. ábra. MLS jel

2.1.3 Az MLS tulajdonságai

(Az alábbi tulajdonságok a 4. ábra-n megfigyelhetők.)

1. Eltolási tulajdonság

Ha $b_0, b_1 \dots b_{2^m-1}$ maximálisan hosszú sorozat, akkor $b_j, b_{j+1} \dots b_{j+2^m-1}$ is az.

2. Ablakozási tulajdonság

Ha egy m -rendű MLS sorozatra m -hosszú ablakot helyezünk, akkor biztos, hogy minden létező m -hosszú bináris sorozatot pontosan egyszer megfigyelhetünk, kivéve a tiszta 0-t. A sorozat kezdeténél és végénél felmerülő nehézségeket kiküszöbölhetjük azzal, hogy úgy gondoljuk el a fenti állítást, hogy a sorozat körben van írva.

3. „Fele 1, fele 0” tulajdonság

Minden MLS pontosan (2^{m-1}) 1-est és $(2^{m-1} - 1)$ db 0-t tartalmaz. Ennek oka: 2^{m-1} páratlan szám van 1 és 2^m között, melyek 2-es számrendszerben ábrázolva 1-re végződnek, és ugyanebben a tartományban $2^{m-1} - 1$ páros szám van, melyek 0-ra végződnek. Ezeken az állapotokon megy végig a shift-regiszter, a kimenet pedig az utolsó bit (2. ábra).

4. Összeadási tulajdonság

Ha egy MLS-sorozattal összeadjuk önmagának az eltoltját (mod 2), akkor az összeg is MLS-sorozat lesz, olyan sorozat, amely az eredeti eltoltja.

5. Autokorrelációs tulajdonság

Az MLS autokorrelációs függvénye egészen speciális:

$$\Phi_{ss}(0) = 1$$

$$\Phi_{ss}(n) = \frac{-1}{L} \quad 0 < n < L$$

6. Futam tulajdonság

Definiáljuk a „futam”-ot úgy, hogy jelentse az egymásután előforduló ugyanolyan szimbólumok hosszát.

Bármely MLS-ben az előforduló futamok fele 1-nek adódik, negyede 2-nek, nyolcada 3-nak, stb.. A 0-kból és 1-esekből álló futamok száma megegyezik.

2.2 Az MLS felhasználása impulzusválasz méréséhez

2.2.1 Időtartománybeli átlapolódás (Time aliasing)

Definiáljuk a periodikus egység-impulzust:

$$\delta'(n) = \begin{cases} 1, & \text{ha } n \bmod L = 0 \\ 0, & \text{egyébként} \end{cases}$$

A vessző index a jel periodikus voltára utal. A rendszerelméletből ismert, hogy a lineáris rendszer válasza a periodikus egységimpulzus gerjesztésre a konvolúció műveletének periodikus kiterjesztésével számolható:

$$y'(n) = x'(n) \otimes h'(n) = \sum_{k=0}^{L-1} x'(n)h'(n-k),$$

ahol a futó indexek esetén értelemszerűen modulo L aritmetikát kell alkalmazni.

A periodikus Dirac-gerjesztésre adott választ periodikus impulzusválasznak nevezzük és angolból PIR-rel hivatkozunk rá (forrás: (2)).

A PIR tehát kiszámolható a periodikus Dirac-impulzus és az impulzusválasz konvolúciójával:

$$h'(n) = \delta'(n) * h(n) = \sum_{k=-\infty}^{\infty} \delta'(k) h(n-k) = \sum_{k=-\infty}^{\infty} h(n+kL)$$

(A * a konvolúció jele.) Tehát, ahogy természetes, a PIR is L -szerint periodikus.

Látható, hogy a PIR akkor fog jó közelítéssel megegyezni az impulzusválasszal, ha L nagyobb, mint az impulzusválasz hossza. Ellenkező esetben, mivel a képlet értelmében az impulzusválasz L -hosszú darabjait egymás fölé tolva és összeadva adódik a PIR (egy

periódusa), az impulzusválasz részei „összekeverednek”, és onnantól szétválaszthatatlanok. Mivel ez a jelenség nagyon hasonlít a frekvenciatartományban, a mintavételi tétel tárgyalásakor megfigyelt aliasing jelenséghez, ezért nevezzük ezt most time-aliasing-nek, vagyis időtartománybeli átlapolódásnak. (Látható, hogy amennyiben L nagyobb, mint az impulzusválasz hossza, akkor a PIR számításakor minden pontnál csak egyetlen impulzusválasz-értéket veszünk figyelembe, mert a többi már mind 0.) (Forrás: (3).)

A továbbiakban feltételezzük, hogy a gerjesztés periódusát úgy választottuk meg, hogy ne léphessen föl time-aliasing, azaz a PIR jó közelítéssel megegyezik az impulzusválasz függvénnyel. Az „impulzusválasz” szóhasználat pedig egyben jelölni fogja a vele megegyező periodikus impulzusválaszt is.

2.2.2 Impulzusválasz-számítás MLS-gerjesztés segítségével

Az MLS-t visszacsatolt shift-regiszterrel állíthatjuk elő. A regiszter kimenetén megfigyelt 1-es bitet -1 -es, a 0-s értéket (konvenció szerint) $+1$ -szintként adjuk a vizsgálandó rendszerre, hogy 0-ra szimmetrikus legyen a mérőjelünk. Mint láttuk, az MLS autokorrelációs függvénye egy egész kicsi DC komponenstől eltekintve periodikus Dirac-sorozat. Ha $(L + 1)$ -gyel normalizáljuk az autokorrelációs függvényt (az ilyen módon definiált autokorrelációs függvényeket a továbbiakban Ω -val jelöljük), akkor elérjük azt, hogy értéke a nulla helyen pont egységnyivel legyen nagyobb, mint máshol:

$$\Omega_{ss}(n) = \frac{1}{L+1} \sum_{k=0}^{L-1} s'(k) s'(k+n)$$

$$\Omega_{ss}(0) = \frac{L}{L+1}$$

$$\Omega_{ss}(n) = \frac{-1}{L+1} \quad \text{ha } 0 < n < L$$

Ebben az esetben a normalizált autokorreláció felbontható egy Dirac-sorozatra és egy hozzáadódó DC-komponensre:

$$\Omega_{ss}(n) = \delta'(n) - \frac{1}{L+1}$$

Látható, hogy L -t növelve a DC-tartalom 0-hoz tart, vagyis az egész kifejezés egy ideális periodikus impulzus-sorozathoz. Mivel az akusztikai rendszerek zöme AC-csatolt, ezért ez a kicsiny DC-hiba még érdektelenebb (forrás: (4)).

Ha most a $h'(n)$ periodikus impulzusválaszú rendszerünket egy $s'(n)$ MLS-sel gerjesztjük, az $y'(n)$ válasz a két jel periodikus konvolúciójából kiszámolható:

$$y'(n) = s'(n) \otimes h'(n) = \sum_{k=0}^{L-1} s'(k) h'(n-k)$$

$y'(n)$ és $s'(n)$ ismert, tehát $h'(n)$ számolható, méghozzá a gerjesztés és a válasz periodikus keresztkorrelációjával (ez utóbbit is, hasonlóan az autokorrelációhoz, $(L+1)$ -gyel normalizáljuk. A K szimbólum a periodikus keresztkorreláció műveletet jelöli).

$$\begin{aligned} \Omega_{sy}(n) &= s'(n) K y'(n) = \\ &= s'(n) K [s'(n) \otimes h'(n)] = \\ &= [s'(n) K s'(n)] \otimes h'(n) = \\ &= \Omega_{ss} \otimes h'(n) = \\ &= \left[\delta'(n) - \frac{1}{L+1} \right] \otimes h'(n) = \\ &= h'(n) - \frac{1}{L+1} \sum_{k=0}^{L-1} h'(k) = \\ &= h'(n) - \frac{1}{L} \sum_{k=0}^{L-1} h'(k) + \frac{1}{L(L+1)} \sum_{k=0}^{L-1} h'(k) \end{aligned}$$

Ez utóbbi kifejezés második tagja a PIR középértékét, azaz DC-komponensét jelenti, a harmadik tag ugyanennek $(L+1)$ -ed részét.

Eredményül tehát a PIR egy kvázi AC-csatolt verzióját kapjuk meg, ugyanis a DC-komponenszt levonjuk, és csak egy $(L+1)$ -gyel leosztott DC-komponens szivárogoz át a rendszeren. Nagy L esetén ez belevész a zajba. ($L = 16\,384$ esetén ez 84 dB csillapítást jelent.) Ráadásul, ha a rendszerünk AC-csatolt, akkor ez a kis DC-komponens sem jut át rajta.

Vizsgáljuk most mérési elvünket a frekvenciatartományban.

A periodikus Dirac-impulzussorozat teljesítményspektruma az autokorrelációs függvényének Diszkrét Fourier Transzformációjával határozható meg:

$$DFT\{\Phi_{\delta\delta}\} = \frac{1}{L}$$

Az MLS teljesítményspektruma:

$$DFT\{\Phi_{\delta\delta}\} = \begin{cases} \frac{1}{L}, & \text{ha } n = 0 \\ \frac{L+1}{L}, & \text{ha } n \neq 0 \end{cases}$$

Látható, hogy a DC kivételével mindkettő minden frekvencián konstans értékű, de az MLS $(L + 1)$ -szer nagyobb értékű. Vagyis azonos amplitudójú gerjesztés mellett az MLS $L + 1$ -szer nagyobb teljesítményű, mint a periodikus Dirac-impulzus.

Ha az MLS DFT-jének amplitúdója után kiszámítjuk a fázisát is, a számított fázisértékek egyenletes valószínűség-sűrűség eloszlást mutatnak $-\pi$ és $+\pi$ között. (Szemben a periodikus Dirac-deltával, amely DFT-jének fázisa minden helyzetben zérus.)

2.3 Gyors Hadamard-transzformáció

Az Hadamard-transzformáció mátrix-szorzás formájában írható fel legegyszerűbben. A transzformációs mátrixot, amely a bemeneti vektort transzformálja, nevezzük H_n -nek, ahol n a sor és oszlopszámot jelöli. A mátrix elemei $+1$ és -1 , olyan elrendezésben, hogy az alábbi egyenletet kielégítse

$$H_n H_n^T = nI_n.$$

A jelenlegi feladatunkhoz alkalmazott algoritmus még ezen belül is egy speciális típusú Hadamard-mátrixot igényel, amit Sylvester-mátrixnak hívnak és rekurzívan definiálható:

$$H_1 = [1]$$

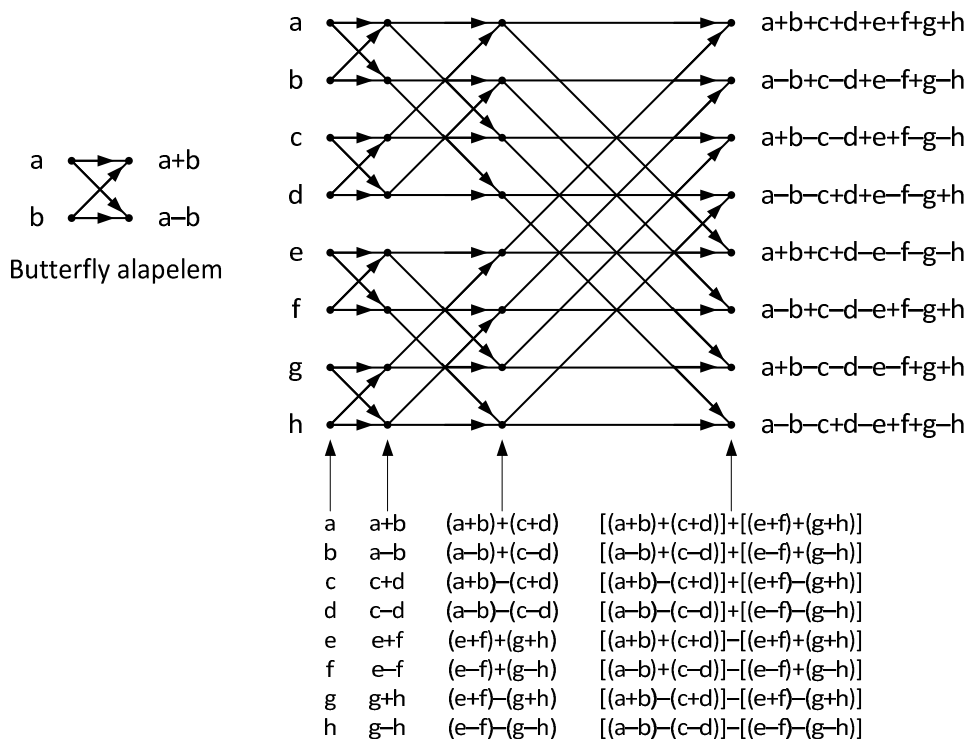
$$H_{2i} = \begin{bmatrix} H_i & H_i \\ H_i & -H_i \end{bmatrix}$$

Az 5. ábra-n látható egy 8×8 -as Sylvester-mátrix, valamint a mátrixszal történő transzformáció.

$$H_8 = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} a + b + c + d + e + f + g + h \\ a - b + c - d + e - f + g - h \\ a + b - c - d + e + f - g - h \\ a - b - c + d + e - f - g + h \\ a + b + c + d - e - f - g - h \\ a - b + c - d - e + f - g + h \\ a + b - c - d - e - f + g + h \\ a - b - c + d - e + f + g - h \end{bmatrix}$$

5. ábra. Transzformáció 8×8 -as Sylvester-mátrixszal.

A 6. ábra-n látható a transzformáció alapeleme az úgynevezett Butterfly alapelem, valamint az alapelemből rekurzíván építkező transzformáció folyamatábrája.



6. ábra. 8-pontos Hadamard-transzformáció folyamatábrája

A feladatunkban szereplő cirkuláris MLS gerjesztés-mátrix nem tesz eleget a fenti követelményeknek. A fő probléma, hogy a mérete nem 2-hatványa. A következő alfejezetekben bemutatunk egy olyan eljárást, amellyel a cirkuláris MLS-mátrix Sylvester-mátrixszá alakítható.

2.3.1 Az Hadamard-tr. felhasználása keresztkorreláció számításához

Néhány egyszerű lépéssel a cirkuláris MLS-mátrix Sylvester-mátrixszá alakítható. A műveletet az alábbi kifejezés írja le:

$$N_n = P_2 S_2 H_{n+1} S_1 P_1,$$

ahol P_1 és P_2 permutációs mátrixok. Céljuk az MLS-mátrix sorainak átrendezése, míg S_1 és S_2 célja a mátrix 2-hatványú méretre növelése (1 sor és egy oszlop hozzáadásával).

Ha ezek a mátrixok megtalálhatóak, elegendő egyszer megkeresni a gyakran használt MLS-hosszakhoz tartozó mátrixokat, fájlban tárolhatók, tehát a méréseink idejét nem fogja P_1 és P_2 keresése megnyújtani.

A korábbi eredmények behelyettesítésével a következő formulához juthatunk:

$$\Omega_{sy} = \frac{1}{n} P_2 \left(S_2 \left(H_{n+1} (S_1 (P_1 Y)) \right) \right)$$

A kifejezés szavakkal a következőt jelenti: A vizsgált rendszer kimenetén mért Y válaszvektort P_1 szerint permutáljuk, és a vektor elejére egy 0 elemet helyezünk. Az így kapott vektor 2^m elemet tartalmaz, ezen végezzük el az Hadamard-transzformációt. Ezután az első elemet eldobjuk, a végső eredményt pedig a P_2 -szerinti permutálással és normalizálással kapjuk meg.

2.3.2 A permutációs mátrixok meghatározása

A permutációs mátrixok meghatározásakor az N -mátrixból indulunk ki, és lépésről lépésre haladunk a H -mátrix felé. Vezessük be az alábbi jelölést:

$$G_n = S_2 H_{n+1} S_1 = P_2^{-1} N_n P_1^{-1}$$

A most definiált G -mátrix tehát nem más, mint az eredeti H_{n+1} -mátrix – az első sora és oszlopa nélkül.

Jegyezzük meg, hogy az alább következő algoritmus P mátrixok inverzét adja eredményül, de az invertálás ebben az esetben ekvivalens a transzponálással. (Minden sor és minden oszlop egyetlen 1-est tartalmaz, a többi eleme mind zérus.)

Ha egyszer meghatároztuk az MLS-t generáló shift-regiszterünk kezdeti állapotát, akkor ezzel egyértelműen meghatároztuk a regiszter soron következő $2^m - 1 = n$ állapotát is. Mint korábban bemutattuk, minden állapot pontosan egyszer fordul el, és ezek a bináris szavak nyilvánvalóan megfigyelhetők az N (MLS – noise) mátrix oszlopainak értékeiben is. Ha ezekhez a bináris szavakhoz súlyt rendelünk (kettes számrendszerbeli számként értelmezzük), akkor minden egyes oszlopot azonosítani tudunk a súlyával. P_1 -t úgy kapjuk meg, ha az alábbi 7. ábra szerinti súlyképzés alapján járunk el. Ez módszer azt a permutációs mátrixot eredményezi, amelyik N oszlopait úgy cseréli fel, hogy a sorok súlyaik alapján növekvő sorrendbe kerüljenek.

$$N = \begin{matrix} & 1 & 3 & 7 & 6 & 5 & 2 & 4 \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} & \rightarrow & N' = \begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$

7. ábra. N mátrix oszlopainak rendezése P_1 permutáló mátrixszal

Most következhet P_2 meghatározása. Ez a mátrix N' -t fogja G -vé permutálni. Most a súlyok képzéséhez szolgáló szavakat a H -mátrix rekurzív tulajdonságából kiindulva kell meghatározni. Keressünk olyan „azonosító” szavakat egy H -mátrix soraiban, melyek soraihoz növekvő súly értékeket rendelnek. Ezt a rekurzív tulajdonságból beláthatóan a sorok 2^k -dik ($k = 0, 1, \dots, m-1$) pozícióiban lévő értékekből képzett bináris szavak fogják biztosítani.

Ezek után P_2 inverzét úgy kapjuk meg, ha a fentebb említett súlyképzési szabállyal képezzük N' sorainak súlyait, és megkeressük azt a permutációt, amely a súlyokat növekvő sorrendbe viszi át. A 8. ábra-n látható, hogy a P_2 permutáló mátrix N' sorait úgy rendezi, hogy azok a súlyuk szerint növekvő sorrendbe kerüljenek.

$$N' = \begin{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 4 \\ 3 \\ 6 \\ 7 \\ 5 \end{matrix} & \rightarrow & G = \begin{matrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} & \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} \end{matrix}$$

8. ábra. N' mátrix sorainak rendezése P_2 permutáló mátrixszal

3 Megvalósítás real-time eszközökön

A célnak leginkább megfelelő, munkahelyemen rendelkezésre álló eszköz, amit a fejlesztés platformjául választottam, a National Instruments NI cRIO-9074 típusú Compact vezérlője. A választást az is motiválta, hogy ez az eszköz grafikus adatfolyam-alapú programnyelven LabVIEW-ban programozható, amely lehetővé tette a gyors és professzionális minőségű mérőszoftver fejlesztést. Ezen túlmenően a CompactRIO beágyazott rendszerként is működhet, amely lehetőséget ad az autonóm működésre is (előny terepi méréseknél).

3.1 A megvalósításhoz használt LabVIEW szoftver bemutatása

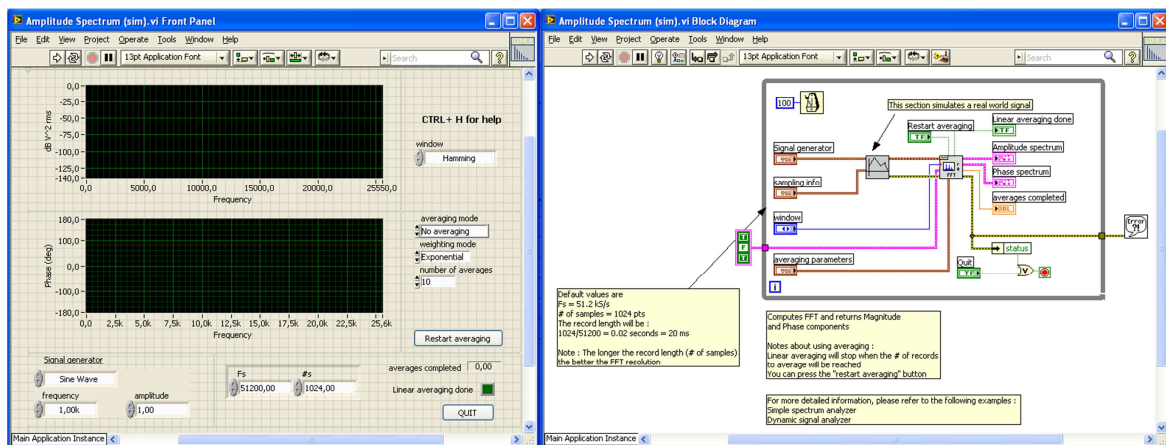
A megvalósításhoz szükséges szoftver és hardver eszközöket a mérés és automatizálás témakörben élen járó National Instruments Corporation fejleszti és gyártja. A cég által kifejlesztett LabVIEW nevű szoftver elődje virtuális műszert kezelni és létrehozni képes szoftver volt. Folyamatos fejlesztés során a LabVIEW számos eszközzel bővült, mára a mérnökök számára mérési, automatizálási, folyamatirányítási, szimulációs és tesztelési célokra egyaránt használható. A programozás során virtuális műszereket (Virtual Instruments – VI) hozunk létre. A létrehozott virtuális műszerek mind az iparban, mind pedig az oktatásban eredményesen használhatók.

A virtuális műszerek kialakulását és elterjedését a személyi számítógépek rohamos fejlődése tette lehetővé. Használatuk során erős és viszonylag olcsó hardvert kapunk, (az egyre alacsonyabb ár mellett, egyre nagyobb teljesítményre képes PC-k segítették elő az ilyen rendszerek elterjedését és fejlődését). A PC – megfelelő mérőkártya és szoftver társítása után – a mérnöki világban szükséges műszerként használható. A virtuális műszer használata közben jelenlévő PC könnyíti a kommunikációt, naplózást, elemzést, stb.. A technológia sajátossága a testre szabható mérőműszer. A széles körben történő elterjedésének kritériuma a mérőkártyák szabványos illesztő felülete, melyet a National Instruments jól és pontosan definiál eszközeiben.

3.1.1 LabVIEW fejlesztőkörnyezet

A LabVIEW grafikus programozási környezet, amely adatfolyam elven, G-nyelven programozható. A programfejlesztés front panel (előlap) és blokk diagram (grafikus programot tartalmazó rész – kvázi forráskód) segítségével történik.

A LabVIEW-val létrehozott program a VI (Virtual Instruments – virtuális műszer). A VI három komponensből áll, melyek a következők: az előlap (front panel), blokk diagram (block diagram), ikon és konnektor (icon and connector pane). A 9. ábra bal oldalán látható a front panel, melyen megfigyelhetők a kontrollok és indikátorok, az 9. ábra jobb oldalán pedig a blokk diagram, amely a „forráskódot” tartalmazza.



9. ábra. LabVIEW front panelje és blokk diagramja

(Forrás: LabVIEW/NI Example Finder)

3.1.1.1 Front panel, blokk diagram

Az előlap nem más, mint a mérést végző virtuális műszer kezelőfelülete, ezen helyezhetjük el a felhasználó által testre szabható virtuális be- és kimeneti perifériákat, más szóval a front panelbeli elemeket. A frontpanelbeli elemek a kontrol (control) és indikátor (indicator) elnevezést kapták szoftverben. A kontrol lehet nyomógomb, kapcsoló, csúszka, forgatógomb, szám vagy szöveg bevitelére alkalmas mező és egyéb bemeneti eszköz. Az indikátor lehet LED, grafikon, szám és szöveg kijelzésére alkalmas mező és egyéb kimeneti eszköz.

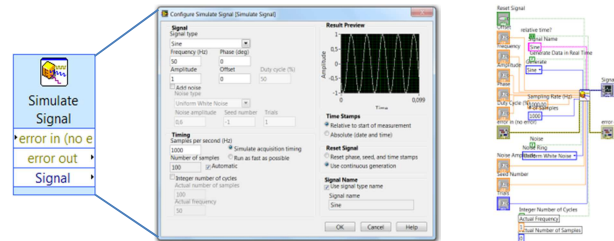
Az előlaphoz tartozik egy blokk diagram, amely az előlapon szereplő elemek működését szabja meg. Ez tartalmazza a forrásnyelvi leírást, amely G nyelven készíthető el. A blokk diagramon megjelennek az előlapra helyezett elemek szimbólumai, melyek segítségével programozhatjuk az ezek közötti kapcsolatot. Függvényeket, struktúrákat,

aritmetikai és logikai műveleteket és egyéb eszközöket használhatunk programunk elkészítéséhez. A 9. ábra-n látható front panelen (bal oldali panel) megfigyelhetők a kontrollok és indikátorok, míg a blokk diagramon (jobb oldali panel) a forrásnyelvi leírás.

3.1.1.2 Modularitás (SubVI)

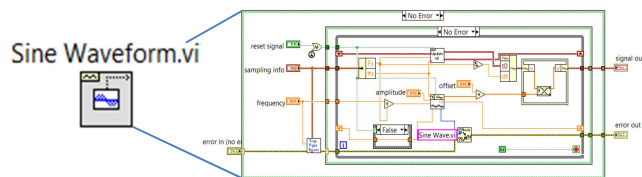
Express VI-ok:

dialógusablakban
konfigurálható,
interaktív VI-ok



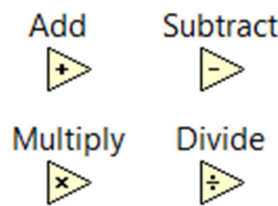
SubVI-ok:

huzalozással testre szabható,
modularizált VI-ok



Függvények:

a LabVIEW alapvető elemei,
nincs előlapjuk és
blokkdiagramjuk



A fenti műveletek hierarchia szintjüket tekintve lehetnek magas és alacsony szintűek. A magas szintű műveletek az expressz VI-ok. Komplet feladat ellátására készültek, ezért könnyíti a programozó feladatát, viszont csak bizonyos célfeladatok ellátására alkalmasak. Vegyük a fájlkezelő műveleteket példaként. A magas szintű fájlművelet teljesíti a fájlkezelésnél előforduló mindhárom fájlműveletet, jelesül a fájlmegnyitást, fájlolvasást-írást és fájllezárást¹.

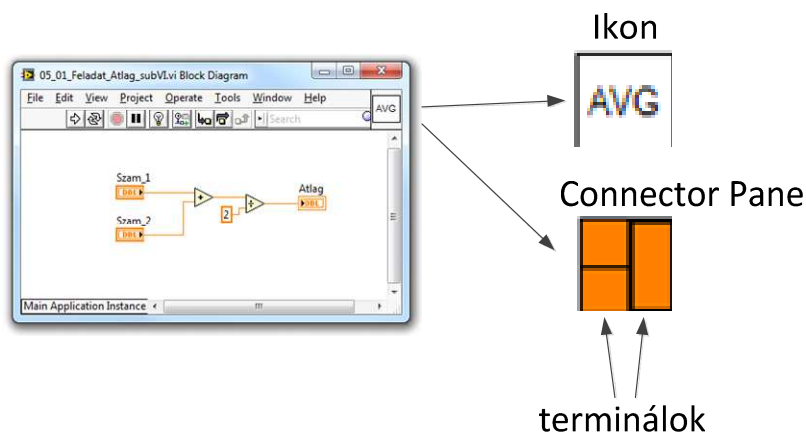
Az alacsonyszintű műveletek a függvények és a SubVI-ok². Ezeket a műveleteket használva a programozó alapelemekből építkezhet. Meghatározhatók az egyéni feladatrészek működése, rugalmasságot biztosít a programozásban, viszont nagyobb programozói tudást igényel. Fontos szem előtt tartani, hogy az alacsonyszintű műveletek

¹ Negyedik műveletként a magas szintű fájlművelet elvégzi a hibakezelést.

² A SubVI a magas- és az alacsonyszintű műveletek között helyezkedik el, mivel jelen írásban a műveleteket két szintre bontjuk, így a SubVI az alacsonyszintű műveletek közé soroljuk.

hatékonyabbak és eredményesebbek, mint a magas szintűek. A fájlkezelő eszköztárat példaként tekintve a fájlmegnyitás vagy létrehozás, fájl bezárás, stb. művelet tartozik ebbe a kategóriába.

A SubVI helyes használatával a LabVIEW-ban megírt program modulárisrá tehető. A SubVI-ok úgy működnek, mint a konvencionális programnyelvek szubrutinjai vagy függvényei. Használatához szükség van ikonra és Connector Pane-re (kapcsolófelületre) (10. ábra).



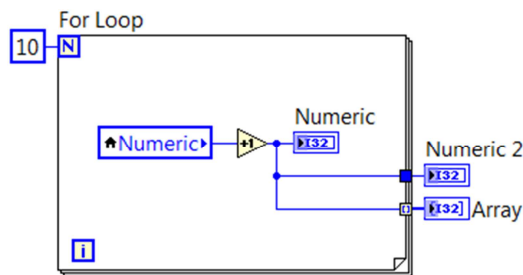
10. ábra. Ikon és Connector Pane

Az ikon a VI grafikus reprezentációja, ez jelképezi a VI-t más blokkdiagramokban. A SubVI rendelkezik alapértelmezett ikonnal, viszont ennek használata kerülendő, mivel nem utal a SubVI működésére (átláthatatlan programot eredményez). A Connector Pane a VI kapcsolófelülete, be- és kimeneti terminálokat tartalmaz. A VI a terminálokon keresztül adja át és kapja meg az adatokat más VI-októl.

3.1.1.3 Struktúrák

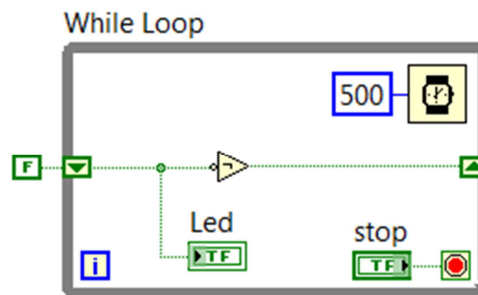
A LabVIEW-ban használhatunk struktúrákat, mint a legtöbb programozási nyelvben. A blokkdiagram funkció palettáján belül a struktúrák alpalettáján található az elérhető struktúrák.

For ciklus (11. ábra): Elöltesztelő ciklus, az „N” bemeneti terminálba (Count Terminal) kötött numerikus adat határozza meg a lefutásra kerülő ciklusok számát. Rendelkezik ciklusszámlálóval (i, Iteration Terminal), amely 0 indexről indul. Az „N” bemeneti terminálba nullát kötve a ciklus egyszer sem fut le.



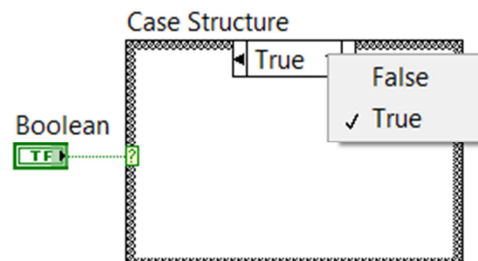
11. ábra. For Loop ciklus

While ciklus (12. ábra): Hátralétesztelésű ciklus, így legalább egyszer lefut. A feltétel terminálba (Conditional Terminal) érkező logikai értéktől függ a futása (a 12. ábra-n látható While ciklus a stop gomb megnyomásáig fut).



12. ábra. While-Loop struktúra

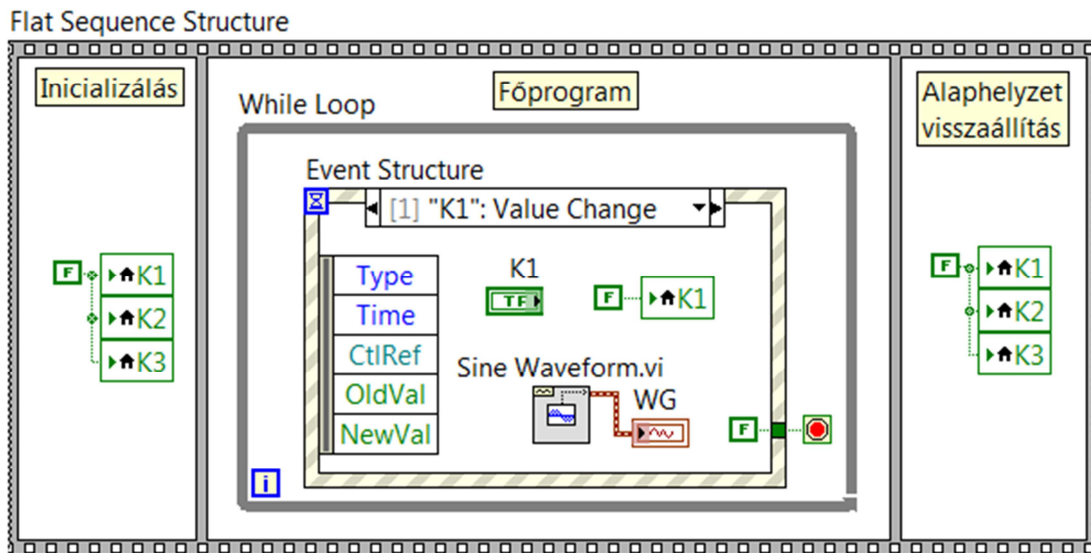
Case struktúra (13. ábra): Elágazás – a Case Selector-ba (kérdőjellel jelölt terminál) beérkező adat értékétől függ, hogy melyik Case ág fut le. A Selector³ felirata szerinti Case ág jeleníődik meg, a többi ág e mögött, rejtett formában van jelen. A Case struktúra egyszer fut le, (ha nincs egyéb struktúrába foglalva), oly módon, hogy mindenképpen lefut az egyik Case ága, mivel ha egyik feltétel sem teljesül, akkor a kötelezően kiválasztandó default ág kerül lefutásra.



13. ábra. Case struktúra

³ A Case struktúra felső részének közepén helyezkedik el, segítségével kiválaszthatjuk az egyes Case ágakat.

Flat Szekvencia Struktúra (14. ábra): A Flat szekvencia kereteket (Frame) tartalmaz, amely kereteken belül elhelyezett kódrészlet mindenképpen lefut és mindenképpen csak egyszer fut. A struktúrához új Frame-k adhatók a struktúra szegélyén történő jobb egérgomb kattintás segítségével.



14. ábra. Flat Szekvencia Struktúra

A fenti struktúrák esetében létrehozhatók be- és kimeneti Tunnel-ek adatok adott struktúrába történő be- és kivezetése érdekében. A Tunnel a struktúra szegélyén négyzetként jelenik meg, használatára láthatunk egy példát a 11. ábra-n, ahol a „Numeric 2” és „Array” nevű indikátorok Tunnel-eken keresztül kapják meg értékeiket. A ciklikus működésre képes struktúrák Tunnel-ei beállíthatók, hogy indexeljék⁴ az adatokat vagy sem⁵. A kimeneti Tunnel az őt tartalmazó struktúra lefutása után továbbítja az adatot.

A For és a While ciklus –, tehát a ciklikus működésre képes struktúrák – szegélyén Shift regiszter hozható létre. A Shift regiszter jobb és a bal oldali részből áll. Aktuális ciklusban a Shift regiszter jobb oldali termináljába adatot töltve, megjelenik az adat a következő ciklusban a Shift regiszter bal oldali termináljában.

⁴ Ebben az esetben azt jelenti az indexelés, hogy a Tunnel-be beérkező adatokat ciklusonként tömbbe fűzi.

⁵ Ha a Tunnel indexelése nem engedélyezett, akkor a ciklusok lefutása után csak a legutolsó adat áll rendelkezésre, a többi pedig eldobásra kerül.

3.1.1.4 Egyebek

A megírt VI-ok, SubVI-ok, globális változók, stb. projektbe rendezhetők. A Getting Started ablakban az Empty Project menüpontot választva létrejön egy új és üres projekt. A felugró Project Explorer ablakban jelennek meg a projekthez hozzáadott alkotóelemek, itt hozzáadhatunk, eltávolíthatunk részeket.

A LabVIEW-ban használhatunk többféle változótípust, úgymint lokális változó, változóként használt, rejtett indikátor, globális változó, funkcionális globális változó, stb. A felsoroltak közül a globális változó használatakor feltétlen szükséges projektet létrehozni, mivel ez a változó egy speciális, LabVIEW által létrehozott könyvtárba kerül (a létrehozott könyvtár a PC fájlrendszerében egy fájlként látszik).

A tabulátor kontrol lapokból áll, amely lapok a front panelen jelennek meg. A lapokon elhelyezhetőek front panelbeli elemek. A tabulátor kontrol felső felén található Page Selector segítségével választhatjuk ki a megjeleníteni kívánt lapot. A nem látható lapon lévő frontpanelbeli elem futásidőben aktívan működik, általában a helytakarékoság, tagoltság biztosítására használjuk (elrejtünk bizonyos elemeket).

3.2 A megvalósításhoz használt hardverek bemutatása

Ebben a fejezetben bemutatásra kerülnek a hálózat-analizátor megvalósításához szükséges hardvereszközök, melyek a következők: NI USB-6211, NI cRIO-9074 (modulok: NI 9215, NI 9263).

3.2.1 NI USB-6211

A TMDK dolgozatban bemutatott mérési folyamatot első lépésként az NI USB-6211-es mérés-adatgyűjtő kártyára implementáltuk. Az algoritmus helyes működését követően kezdtük el Real-Time eszközre átültetni a folyamatot.

Az NI USB-6211 16-bites felbontású, 250 kS/s mintavételi sebességű, multifunkciós mérés-adatgyűjtő eszköz. Az adatgyűjtő 16 analóg inputot, 2 analóg outputot, 2 db 32-bites számlálót valamint 4 + 4 digitális I/O pontot tartalmaz (forrás: (5)).

Az analóg input ± 10 V tartományba eső jelet tud mintavételezni, az analóg output pedig 0 V - 5 V-ig képes jelet kiadni – csatornánként 5 mA áramerősség mellett.

3.2.2 NI cRIO-9074

Az NI cRIO-9074 integrált rendszer kombinálja a real-time processzor és az újrakonfigurálható FPGA (field-programmable gate array) előnyeit, valamint ugyanabban a „keretben” található beágyazott rendszer és felügyeleti alkalmazás (forrás: (6)). A hardver elnevezésében található cRIO a Compact RIO szavakból tevődik össze (RIO: Real-Time Input Output). A cRIO tartalmaz egy 400 MHz-s ipari real-time processzort (Freescale MPC5200), valamint egy 2 millió kapuval rendelkező FPGA chip-et (Spartan-3 2000). A 15. ábra-n látható az NI cRIO-9074-es, az ábra bal oldalán látható rész tartalmazza a real-time processzort, csatlakozókat, az ábrán látható, hogy hardver gyakorlatilag egy keret (chassis), melybe 8 db NI C sorozatú I/O modul helyezhető bele.



15. ábra. NI cRIO-9074.

A robusztus alkalmazások érdekében a cRIO működési hőmérséklete -20°C - 55°C , melyet a jelen alkalmazásban nem használtunk ki, viszont ebből az adatból is látszik, hogy ipari körülményekre lett tervezve ez a hardver. Tápforrásként egy egyszerű 19 - 30 V közötti DC táp szükséges. 128 MB DRAM áll rendelkezésre az hardveren futó alkalmazás számára, ezen kívül 256 MB nemillanó memória az adatgyűjtés számára.

A cRIO-9074-es két 10/100 Mb/s Ethernet portot tartalmaz, így az eszköz vezérelhető a benne található programozható kommunikációs hálózaton, a beépített Web (http) és file (FTP) szervereken keresztül. Továbbá a 2-es jelzésű Ethernet port segítségével további bővítő és megosztott I/O-k adhatók a rendszerhez. Például az egyik Ethernet csatlakozó használható PC-vel vagy vállalati rendszerrel történő hálózati kommunikációra, míg a másik csatlakozó I/O bővítésre (ehhez egyszerűen csatlakoztatni kell egy további CompactRIO-t vagy más Ethernet alapú kiegészítő I/O eszközt).

A cRIO-ban lévő FPGA portjai a keretbe helyezhető modulokra csatlakoznak, így az onnan érkező adatokat az FPGA közvetlen eléri. Az eszközt Scan Interface módban

használva egy alapértelmezett FPGA kód generálódik, mely 1 kHz-en tud mintavételezni/mintát generálni, feladata a keretbe csatlakoztatott modulok nagysebességű kiszolgálása. A TMDK dolgozatban tárgyalt téma 50 kHz-es frekvenciát követel meg, így a Scan Interface mód nem használható, ezért az eszközt FPGA módban használjuk.

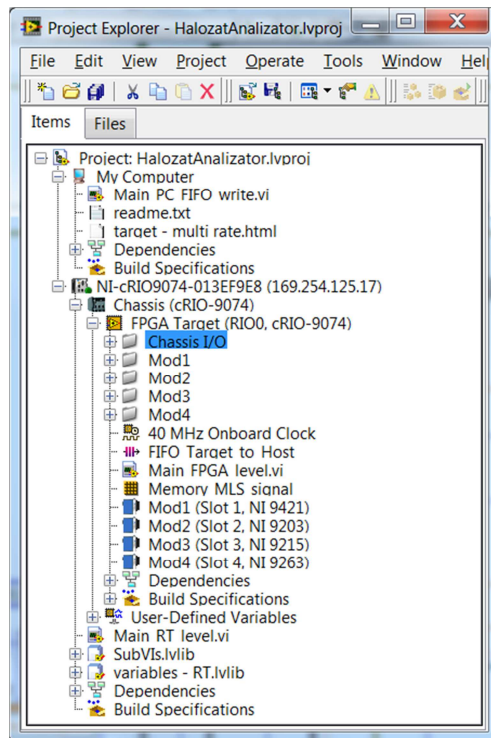
A CompactRIO-t FPGA módban használva MHz-s nagyságrendű működési sebesség érhető el, vagyis ez bőven lefedi az általunk igényelt 50 kHz-es frekvenciát. FPGA módban saját fejlesztésű kód írandó, mellyel a felhasználó felprogramozza az FPGA chipet.

3.3 A hálózat-analizátort megvalósító VI működése

A hálózat-analizátort megvalósító szoftvereket célhardverekre fejlesztettük. Az alkalmazott hardverek 3 részből állnak: FPGA, real-time processzor, PC. Az FPGA és a real-time processzor a Compact RIO két fő egysége. A hardver egységekre a szoftver fő funkciói az alábbiak szerint kerültek implementálásra:

- FPGA: gerjesztőjel előállítás, adatgyűjtés
- RT processzor: számítások elvégzése
- PC: adatok megjelenítése, fájlba mentése

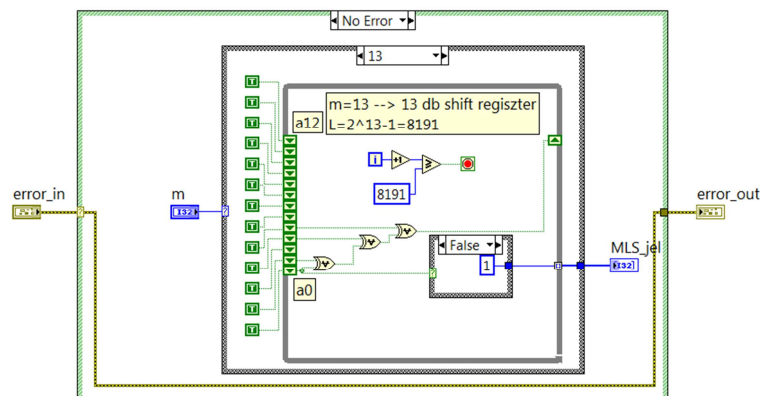
A LabVIEW-ban fejlesztett VI-okat egy HalozatAnalizator.lvproj nevű projekt fájl tartalmazza, a 16. ábra-n látható a Project Explorer ablak, amely segítségével kezelhetők a projektben lévő fájlok. A projekt legmagasabb szintén maga projekt áll, amely alatt a PC (My Computer), valamint a NI cRIO-9074 (NI-cRIO9074-013EF9E8 (169.254.125.17)) célhardverek szerepelnek. A megjelenítést a Main_PC_FIFO_write.vi végzi, a 16. ábra-n látható módon a PC szint alatt szerepel.



16. ábra. Project Explorer ablak

3.3.1 Mérésadatgyűjtő VI

Az MLS mérőjel előállítását a – projekten kívüli – MLS_generation_SubVI.vi nevű VI állítja elő off-line módon, az eredményt file-ban tároljuk (a kód a 17. ábra-n látható). Az elektronikus mellékletek között a LabVIEW\MLS elérési úton található e VI segítségével generált MLS kódok, valamint ennek autokorrelációs függvénye. A DA konverziót, illetve a válaszjel AD átalakítását (azaz a mérés-adatgyűjtési feladatot) az FPGA végzi, amelyet a Main_FPGA_level.vi-ban írtunk meg.



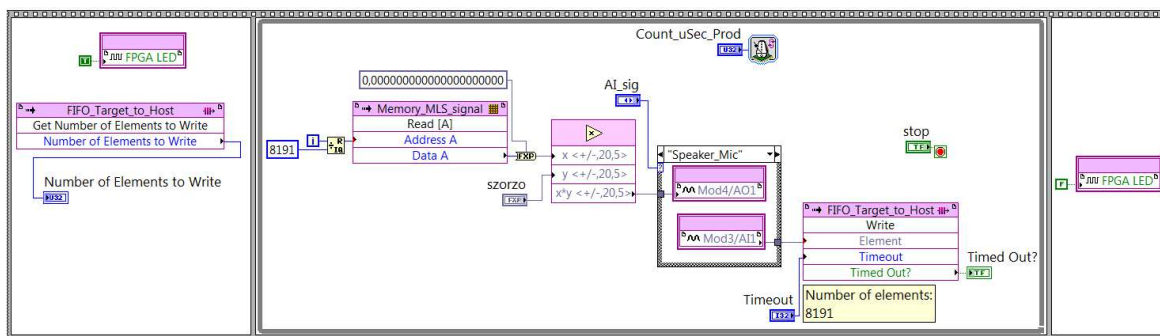
17. ábra. MLS_generation_SubVI.vi

A 17. ábra-n megfigyelhető, hogy a shift regiszterek inicializálására a tiszta magas szintet választottuk. Az MLS_generation_SubVI.vi-on belül a logikai magas és logikai alacsony szintek +1 és -1-re való konvertálása is megtörténik egy korábbi fejezetben tárgyalt szabály szerint.

Az MLS_generation_SubVI.vi-egy másik VI hívja meg, melynek funkciói a következők: MLS jel és MLS jel autokorrelációs függvényének előállítása futásidőben történő megjelenítéssel együtt, valamint az előállított jelek fájlba mentésének fölkínálása a felhasználó felé. Az így előállított MLS autokorrelációs függvény az elméleti részben tárgyalt várakozásoknak megfelelő.

Az FPGA szintem létrehoztunk egy 8192 elem⁶ tárolására alkalmas memóriát, amely ennek következtében magán az FPGA chipen jön létre. A memóriába egy konfigurációs VI segítségével tölthetjük bele a tartalmat. A konfigurációs VI a korábban tárgyalt módon előállított MLS jellel tölti fel a memóriát.

A 18. ábra-n látható Main_FPGA_level.vi nevű VI-t az FPGA chipre fejlesztettük. A cRIO használata során, nem feltételen szükséges megírni az FPGA szintű VI-t. Az eszköz használható Scan Interface módban is, ekkor azonban a mintavételezés maximális frekvenciája 1 kHz. Az eszközünket FPGA módban használva megszűnik az 1 kHz-s korlát.



18. ábra. Mérésadatgyűjtést végző, FPGA szintű VI

A Compact RIO újrakonfigurálható FPGA-t tartalmaz. A Main_FPGA_level.vi-ból a LabVIEW az erre a célra dedikált compiler-e segítségével elkészíti a konfigurációs bitmintát, amellyel a chip konfigurálása történik. A konfigurálás gyakorlatilag nem vesz időt igénybe, viszont a fordítás 20 percet vesz igénybe. A fordításhoz szükséges idő nem

⁶ A feladatunkhoz 8191 elem tárolására alkalmas memória elegendő, viszont a létrehozható memória méretének 2 egész kitevőjű hatványának kell lennie.

számít soknak figyelembe véve az FPGA felépítését, valamint azt a tényt, hogy FPGA chipet – közvetett módon – grafikus nyelven programozhatunk, (viszont a fejlesztést, debugolást ennek figyelembevételével kell megtervezni).

A hálózat-analizátor megvalósítása során szükséges analóg input és analóg output a cRIO keretébe behelyezett modulok formájában elérhető. A modulokat a fizikai behelyezés után a Project Explorer-ben hozzá kell rendelni az FPGA szinthez. Az általunk használt modulok a 16. ábra-n látható Mod3 (analóg input) és Mod4 (analóg output) névvel vannak ellátva.

Az FPGA szinten lévő VI-ból úgy érjük el az adott analóg be- és kimenetet, hogy a projektben már megfelelő helyen lévő modul legördülő menüjéből kiválasztjuk a szükséges portot. A kiválasztott port FPGA I/O node-ként jelenik meg a VI-ban, amely segítségével a gerjesztőjel megjeleníthető az analóg kimeneten, valamint a válaszjel monitorozható a megfelelő analóg bemeneten. Az FPGA szintem kizárólag fixpontos számokkal dolgozhatunk.

A Main_FPGA_level.vi adott iterációjában kivesszük a memóriából az MLS jel aktuális értékét, majd ezt kitesszük az analóg kimenetre. Az analóg kimenet kezelésével párhuzamosan mintavételezzük az analóg bemenetet, amelyen a vizsgált rendszer válasza érkezik. A mérési folyamat 50 kHz-es frekvenciával fut, amelyet egy 20 μ s-mal konfigurált real-time időzítővel érünk el. A cRIO keretén található LED-et felhasználtuk arra, hogy jelezze a mérési folyamatot.

Az analóg bemenet mintavételezésével nyert adatokat az FPGA szinten létrehozott FIFO_Target_to_Host névvel ellátott FIFO-ba töltjük (16. ábra). A FIFO target to host típusú, a target az FPGA chip, a host pedig a cRIO-ban lévő RT vezérlő. A target to host beállítás lehetővé teszi a cRIO-n belüli kommunikációt, amelyre azért van szükség, mert a szükséges számításokat a real-time vezérlőn végezzük.

3.3.2 Számításokat végző VI (Main_RT_level.vi)

A számításokat a Main_RT_level.vi végzi, amely a Compact RIO kereten belüli legmagasabb szinten található, amelyből az látszik, hogy ennek végrehajtását az RT vezérlő végzi (16. ábra).

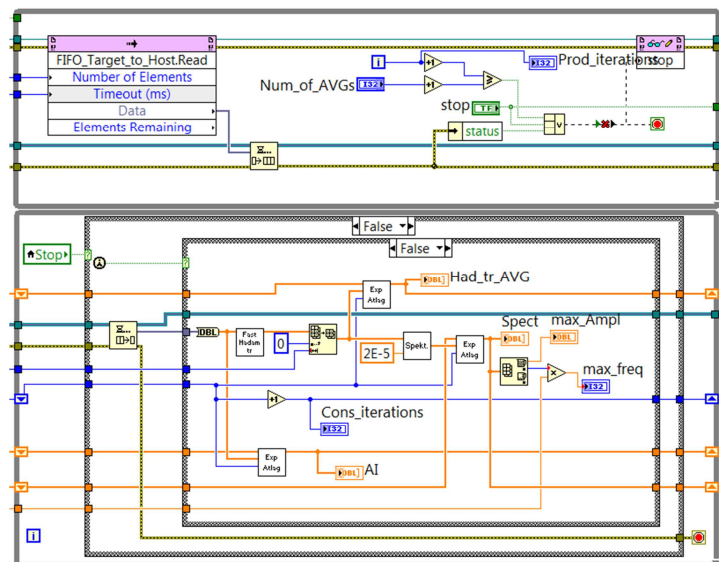
A számítást végző VI a producer-consumer (termelő-fogyasztó) tervezési mintát követi. A termelő és a fogyasztó közötti kommunikáció – a LabVIEW

fejlesztőkörnyezetben definiált – Queue végzi. A Queue egy puffertelt, FIFO adatáramlást megvalósító csatorna, amely adattöbbszörözés és adatvesztés nélküli kommunikációt tesz lehetővé.

A FIFO_Target_to_Host FIFO-t tehát az FPGA szinten lévő VI – 50 kHz-es frekvenciával futva – tölti fel adatokkal, az RT vezérlőn lévő VI pedig kiolvassa onnan az adatokat. A kiolvasást végző VI termelő-fogyasztó struktúrája a 19. ábra-n látható. Az ábra felső része a termelő ciklus, amely 8191 hosszú tömböként olvassa ki az adatokat a FIFO-ból. A ciklus a következő frekvenciával és periódusidővel fut:

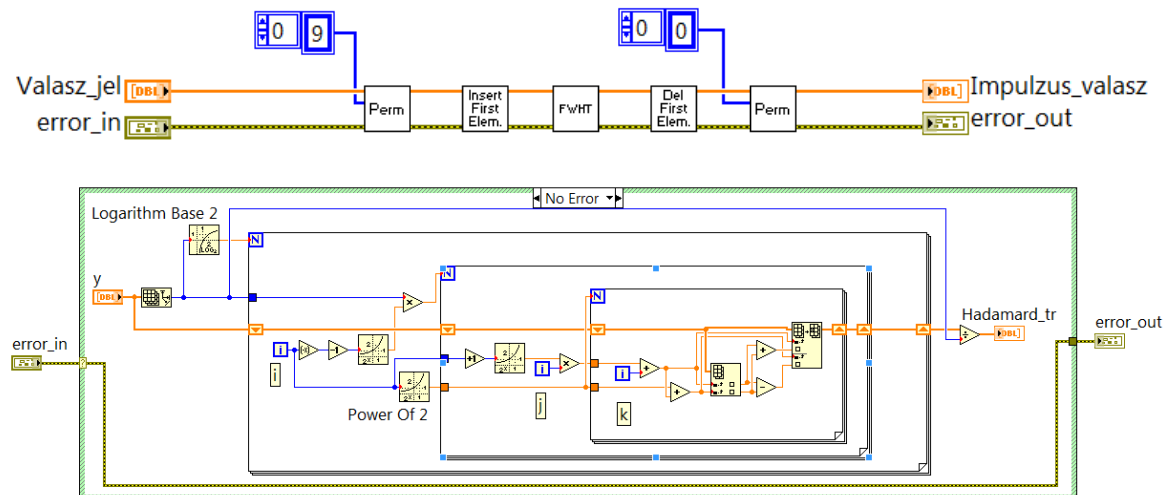
$$f = \frac{50000}{8191} = 6,1043 \text{ Hz} \rightarrow T = \frac{1}{f} = \frac{8191}{50000} = 0,16382 \text{ s.}$$

A 19. ábra alsó részében a termelő-fogyasztó struktúra, fogyasztó ciklusa látható. A két ciklus közötti kommunikáció Queue-val történik. Mivel a Queue-val 8191 hosszú tömböket viszünk át, így a fogyasztó ciklus is ~ 6,1 Hz-vel fut. A fogyasztó ciklusnak tehát 163,82 ms áll rendelkezésére a szükséges számítások elvégzésére, annak érdekében, hogy a feldolgozás real-time maradjon. (A kiegyenlítettebb erőforrásigény érdekében megnövelhetnénk a fogyasztó ciklus frekvenciáját, hogy egyszerre kisebb méretű tömbön kelljen dolgoznia, viszont ez ebben az esetben nem tehető meg, mivel az Hadamard-transzformációt végző VI-nak a 8191 hosszú tömböt egészben kell átadni.)



19. ábra. Producer-consumer struktúra

A 20. ábra felső részében látható az impulzusválasz számoló VI blokk diagramja, (a blokk diagram SubVI-okat tartalmaz). Az ábra alsó részében az előbbi SubVI-okból az Hadamard-transzformációt megvalósító VI blokk diagramja látható.



20. ábra. Az impulzusválaszt számoló VI blokk diagramja

A 20. ábra alsó részében szereplő VI az elméleti részben leírt algoritmus szerint számolja ki a bemenő vektor Hadamard-transzformáltját.

3.3.3 A hálózat-analizátor front panelje

A következő fejezetben lévő 24. ábra-n látható front panel bal felső sarkában végezhetjük el a mérést befolyásoló beállításokat. A mérési adatok elmenthetők, a mentés egy dedikált elérési utat használ, így az elérési út futásidőben nem változtatható. Az adatok mentése text fájlba történik, így azok külső programmal megnyithatók. A mentés során, a front panelen lévő három diagram adatai három különálló fájlba kerülnek lementésre. A mérési adatokat tartalmazó fájlok automatikusan létrejönnek és a fájl neve is automatikusan generálódik le, oly módon, hogy a fájlnevében szerepel a front panelen látható mérési sorszám (Num_of_Meas) is. A mentési metódus természetesen ettől eltérő módon is kialakítható lett volna, a jelenlegi kialakítás a szoftver kezelésének leegyszerűsítésére optimalizált.

A mérés során az AI_sig nevű legördülő menüből a Speaker_Mic, Four_terminal és MLS_signal lehetőségek közül választhatunk. Az MLS_signal lehetőséget választva az FPGA-n futó VI az MLS jelet tartalmazó memóriából az adatokat közvetlen a FIFO-ba tölti,

így egy elméleti, 1-szeres erősítésű rendszert szimulálhatunk. A további két beállítással a négyfólyus vagy a mikrofon-hangszórómérés végezhető el.

A front panelen lévő valamennyi diagram átlagolt adatokat jelez ki. Az átlagolás folyamatosan frissülve történik, mindig az aktuális állapot kijelzésével. A Num_of_AVGs kontrol segítségével beállítható, hogy egy mérési folyamat hány mérésből álljon. Az átlagolás során ezeknek a méréseknek az adatit átlagoljuk. (A Num_of_AVGs kontrol értékét 1-re állítva átlagolás nélkül mérhetünk.)

A kezelőfelületen lévő Prod_iterations indikátor kijelzi a termelő ciklus, a Cons_iterations indikátor pedig a fogyasztó ciklus által végrehajtott iterációk számát. E két indikátor segítségével a real-time működés debug-olható, (real-time működés esetén az indikátorok értékei szinkronban növekednek).

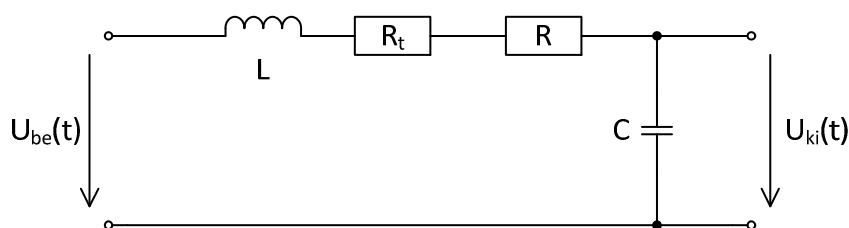
A front panelen a jobb felső sarokban kerülnek kijelzésre az analóg bemeneten érkező minták, a jobb alsó sarokban az impulzusválasz, a bal alsó sarokban pedig a Bode-diagram. Az impulzusválaszt Hadamard-transzformációval számoljuk, a Bode-diagramot pedig az impulzusválasz FFT-jével. A Bode-diagram számolásánál a kezelőfelületen található Subset kontrollal beállíthatjuk, hogy az impulzusválasz függvény első hány elemének FFT-je kerüljön kiszámításra.

4 Mérési eredmények

Az elkészült rendszerrel próbaméréseket végeztünk, amelyek segítségével verifikáltuk az új rendszer tulajdonságait. Elsőként néhány egyszerű elektronikus áramkör impulzusválasztát mértük meg, amelyek esetén a várható eredmény analitikus meghatározása sem okoz gondot, később akusztikai hálózat mérését is elvégeztük, a célkitűzésnek megfelelően.

4.1 Soros RLC-kör

A következő méréssel RLC szűrőkörnek határoztuk meg az átvitelét (21. ábra). Az ábrán látható kapcsolás R ellenállásból, L tekercsből, C kondenzátorból, valamint R_t -ből – a tekercs soros ellenállásából – áll. Az egymással sorba kapcsolt hálózati elemekre rákapcsolt gerjesztésre adott választ a kondenzátoron mérjük.



$$C = 73 \cdot 10^{-9} \text{ F}, \quad L = 7 \cdot 10^{-3} \text{ H}, \quad R_t = 1,7 \, \Omega, \quad R = 10,4 \, \Omega$$

21. ábra. RLC szűrő – C kimenettel

Az RLC szűrő a 21. ábra-n szereplő elrendezésben egy aluláteresztő szűrőt valósít meg. A két energiatároló következtében a Bode-diagramon a töréspont utáni görbe meredeksége várhatóan: $-40 \frac{\text{dB}}{\text{dekád}}$ lesz.

A rendszer átviteli függvénye a kimenetre felírt feszültségosztóból számolható (, ahol $R_e = R + R_t$):

$$\overline{a(\omega)} = \frac{\frac{1}{j\omega C}}{\frac{1}{j\omega C} + j\omega L + R_e} = \frac{1}{1 + j\omega R_e C + (j\omega)^2 LC}.$$

Vezessük be τ időállandót és ξ csillapítási tényezőt:

$$\tau = \sqrt{LC}, \quad \xi = \frac{R_e}{2} \sqrt{\frac{C}{L}} \quad \left(R_e C = 2\xi\tau \rightarrow \xi = \frac{R_e C}{2\sqrt{LC}} = \frac{R_e}{2} \sqrt{\frac{C}{L}} \right).$$

Helyettesítsük be τ -t és ξ -t az eredeti átviteli függvénybe, valamint térjünk át frekvenciatartományból operátortartományba:

$$\overline{a(s)} = \frac{1}{1 + 2\xi\tau s + \tau^2 s^2} = \frac{1}{1 + 8,833 \cdot 10^{-7} s + 5,11 \cdot 10^{-10} s^2}$$

Az időállandó értéke: $\tau = \sqrt{LC} = 0,0226$ ms, amelyből meghatározható a rezonancia-körfrekvencia:

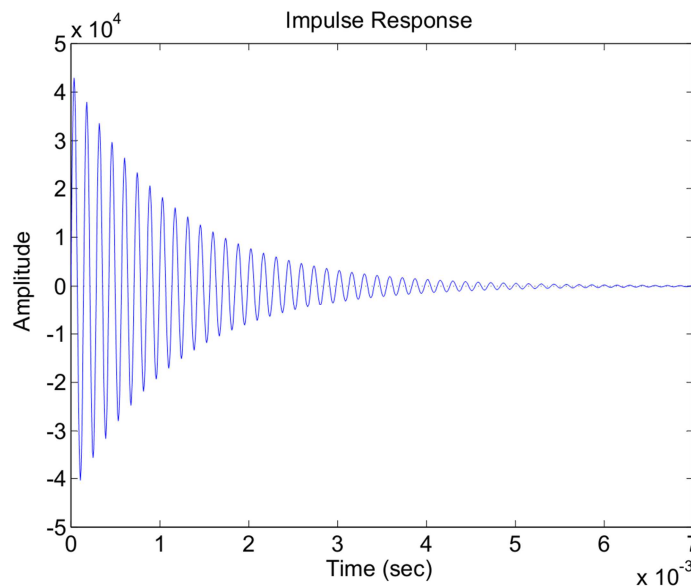
$$\omega_0 = \frac{1}{\tau} = \frac{1}{\sqrt{LC}} = 44237,3955 \frac{\text{rad}}{\text{s}},$$

és a rezonanciafrekvencia:

$$f = \frac{\omega_0}{2\pi} = 7040,6 \text{ Hz.}$$

A csillapítási tényező a fentiekben szereplő kifejezésből számolható, értéke: $\xi = 0,02855$. A csillapítási tényező értékéből arra következtethetünk, hogy a négypólus bemenetére kapcsolt Dirac impulzus hatására a kimeneten lengés jön létre, valamint a Bode-diagramon a töréspontnál egy csúcs lesz megfigyelhető.

Az RLC rezgőkör impulzusválaszát (22. ábra) és Bode-diagramját (23. ábra) első lépésként Matlab-bal határoztuk meg.

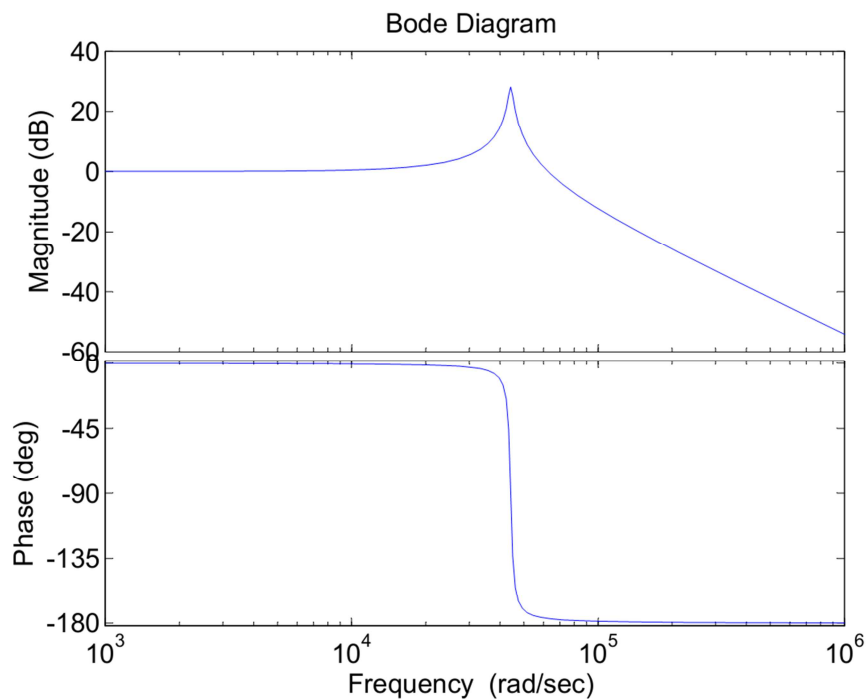


22. ábra. RLC-tag impulzusválasza (Matlab)

A 22. ábra-n látható, hogy RLC-tag impulzusválasza a korábbi várakozásainknak megfelelően lengő jellegű. Az ábráról leolvasható, hogy 1 ms alatt közelítőleg 7 lengés

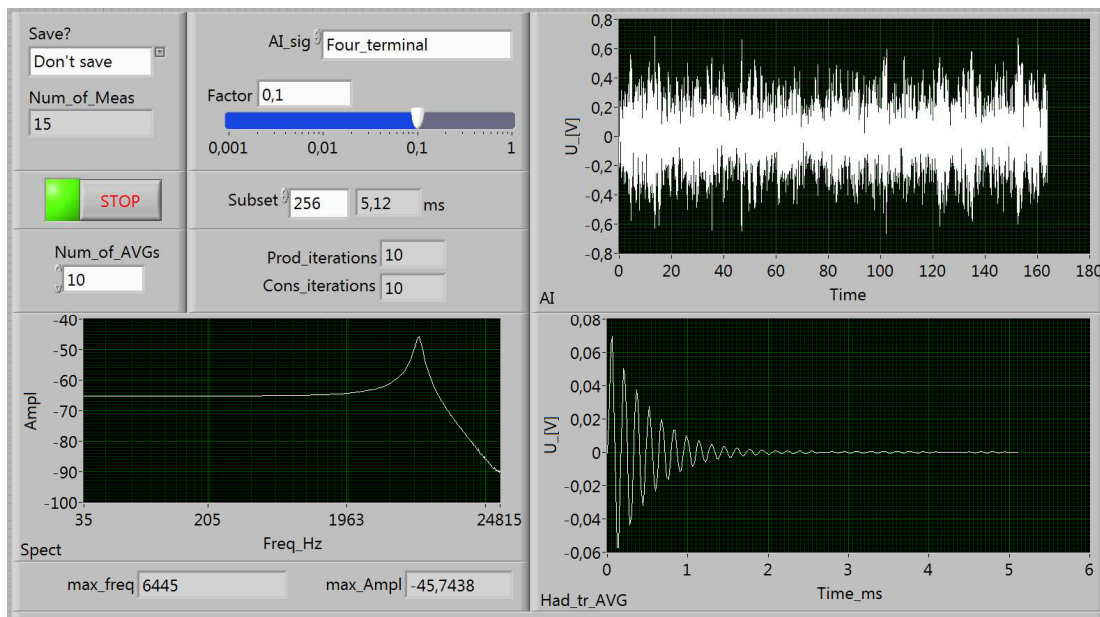
(periódus) jön létre. (A szabad szemmel történő leolvasásból a frekvencia: $f' = 7000$ Hz-nek adódik.)

A 23. ábra-n látható a Bode-diagram Matlab-bal történő felvétel. A Bode-diagramon megfigyelhető a töréspontnál lévő csúcs, valamint töréspont utáni $-40 \frac{\text{dB}}{\text{dekád}}$ meredekségű görbe.



23. ábra. RLC-tag Bode-diagramja (Matlab)

A Matlab-os szimuláció után elvégeztük a valós méréseket a TMDK munka során kialakított rendszerrel. A mérési eredmények a 24. ábra-n láthatóak.

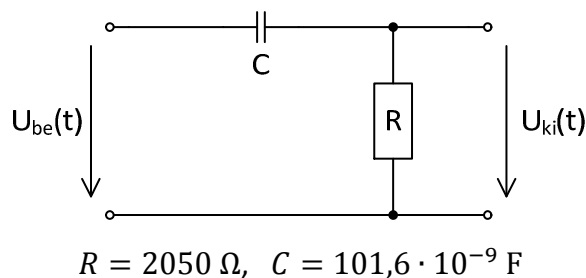


24. ábra. C kimenetű RLC-tag mérése (front panel)

A 24. ábra-n szereplő mérési eredmények megfelelőek a Matlab-os szimulációból származó eredmények alapján elvárt eredményeknek. Az kezelőfelületen a max_freq indikátoron megfigyelhetjük a vizsgált rendszer rezonanciafrekvenciáját, amely nem tér el durva mértékben az elméleti értéktől.

4.2 RC-tag

A következő méréseket a 25. ábra-n látható RC szűrőn végeztük el. A szűrőkör egy kondenzátorból és egy ellenállásból áll. Mivel a kimenetet az ellenállásról vesszük le, így felüláteresztő szűrőhöz jutunk.

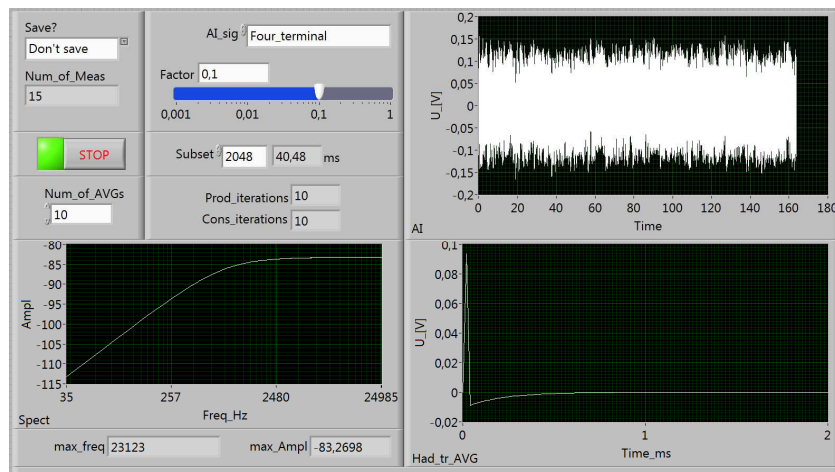


25. ábra. RC-tag – R kimenettel

A 25. ábra-n lévő kapcsolás egyszerűsége révén könnyen kitalálható az elrendezés Bode-diagramja. A töréspont előtti görbe meredeksége $20 \frac{\text{dB}}{\text{dekád}}$, mivel a szűrő elsőfokú.

A törésponti körfrekvencia: $\omega_0 = \frac{1}{RC} = \frac{1}{2050 \cdot 101,6 \cdot 10^{-9}} = 4801,23 \frac{\text{rad}}{\text{s}}$, a törésponti

frekvencia: $f = \frac{\omega}{2\pi} = \frac{4801,23}{2\pi} = 764,14 \text{ Hz}$. A frekvenciára kapott érték megegyezik a 26. ábra-ról leolvasható értékkel.



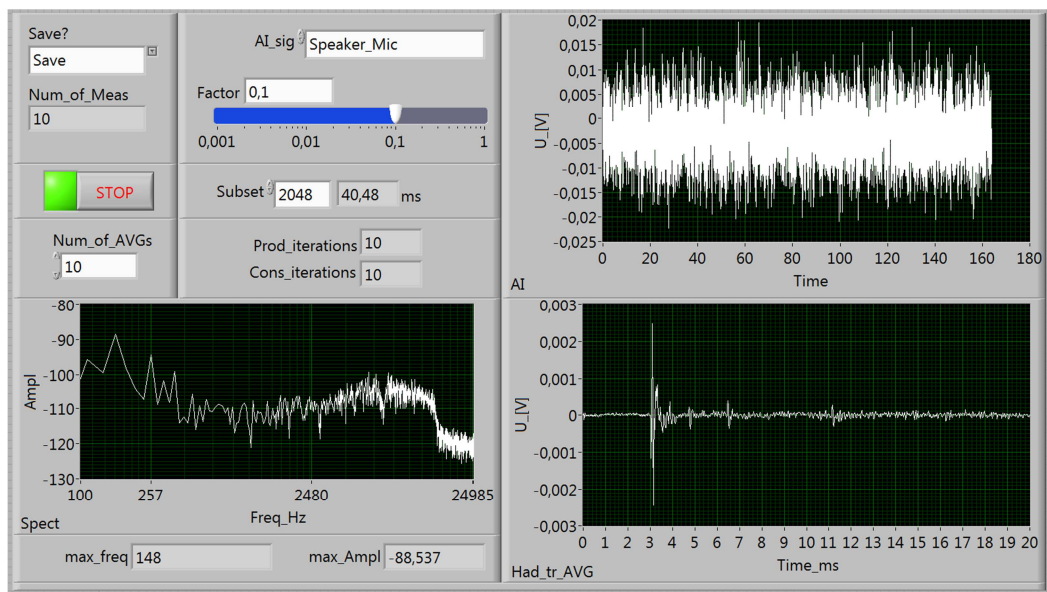
26. ábra. R kimenetű RC-tag mérése (front panel)

A 26. ábra bal alsó sarkában látható a vizsgált szűrő Bode-diagramja. A Bode-diagram a várakozásoknak megfelelő $(20 \frac{\text{dB}}{\text{dekád}})$ -os meredekség, $f = 764,14 \text{ Hz}$ törésponti frekvencia). A mérés során beállítható az átlagolandó minták száma, melyet a jelen mérésnél 10-re állítottunk. Minden mérési metódusban az elsőként mintavételezett 8191 db elemből álló tömböt eldobjuk, mivel a nulladik időpillanattól egy bizonyos ideig még nem érkezhethet válasz a gerjesztésre. A mérés során az impulzusválaszból kivágott első 2048 ponton végeztük el az FFT-t, melynek eredményeként kaptuk a Bode-diagramot.

4.3 Hangszóró-mikrofonmérés

A mérési elrendezés egy hangszóróból, egy mikrofonból állt, a mérés egy átlagos elektronikai laboratóriumban folyt, mely akusztikai szempontból nem tekinthető semlegesnek. A mérési összeállítás következménye, hogy a hangszóróból, a labor szobából és a mikrofonból álló rendszernek mértük ki az átviteli függvényét.

A hangforrásként egy aktív stúdió monitor hangfalat használtunk, melynek teljesítménye 35 W. A mikrofonként egy közönséges dinamikus mikrofont használtunk, melynek jelét erősítés után kötöttük a real-time mérőkártyába. A mérési eredmények a 27. ábra-n figyelhetők meg.



27. ábra. Mikrofon-hangszórómérés (front panel)

A mikrofont és a hangszórót 1 m-re helyeztük el a mérések során. A hang terjedési sebességét $c = 340 \frac{\text{m}}{\text{s}}$ -mal közelítve, az 1 m távolság megtételéhez $\frac{1 \text{ m}}{340 \frac{\text{m}}{\text{s}}} = 2,94 \text{ ms}$ -ra van szüksége a hangnak. A 27. ábra jobb alsó sarkában lévő impulzusválasz függvényben megfigyelhető $\sim 3 \text{ ms}$ -os késleltetés nem más, mint az 1 m távolság megtételéhez szükséges idő. A függelékben található 28. ábra-n egy olyan mérést figyelhetünk meg, amelynél a mikrofon és hangszóró távolsága 2-szeres (2 m), ebben az esetben a késleltetési idő is duplázódik.

5 Eredmények értékelése, további fejlesztési lehetőségek

Mint minden fejlesztési munka, így a jelen TMDK dolgozat sem véglegesen lezárt egység. Számtalan továbbfejlesztési lehetőség adja magát, amelyek közül a legfontosabbak, melyeket elsősorban szeretnék megvalósítani, az alábbiak:

- A jelenlegi fix hosszúságú mérőjel helyett a felhasználó szabadon dönthessen az MLS rendjéről. Ennek a jelentősége elsősorban az, hogy a mérendő impulzusválasz hosszához adaptálható a mérés. A túl hosszú mérési idő fölösleges, a túl rövid pedig időtartományi átlapolódást okoz.
- Az impulzusválasz mérésének az egyik legfőbb előnye (a frekvenciatartományi módszerekkel szemben), hogy rendkívül sokféle más rendszerjellemző függvény származtatható. A dolgozatban bemutatott változat jelenleg csak az amplitúdó menet kirajzolására képes, a palettát szeretném elsősorban idő-frekvencia tartományi módszerekkel bővíteni (Gabor-transzformáció, Wigner-disztribúció), amelyeknek nagy jelentősége van akusztikai alkalmazásokban.
- Az akusztikai mérések legdrágább komponense a mérőmikrofon. Ez kiküszöbölhető azáltal, ha egy nem mérési célú, olcsó (de azért elfogadható átvittel rendelkező) mikrofonnal dolgozunk: ugyanazt a forrást megmérjük egyszer kalibrált mérőmikrofonnal, majd az olcsóbbal, keresztspektrumot számolva az olcsó mikrofon átvitele becsülhető, majd későbbi (már csak olcsó mikrofonnal végzett mérések esetén) ezzel az átvittel korrigálható a mérési eredmény. Ennek a referenciatartományi átlapolódást okoz.
- A HW lehetővé teszi több csatorna szinkron mintavételezését. Ez a lehetőség kihasználható lenne keresztspektrum mérésére.

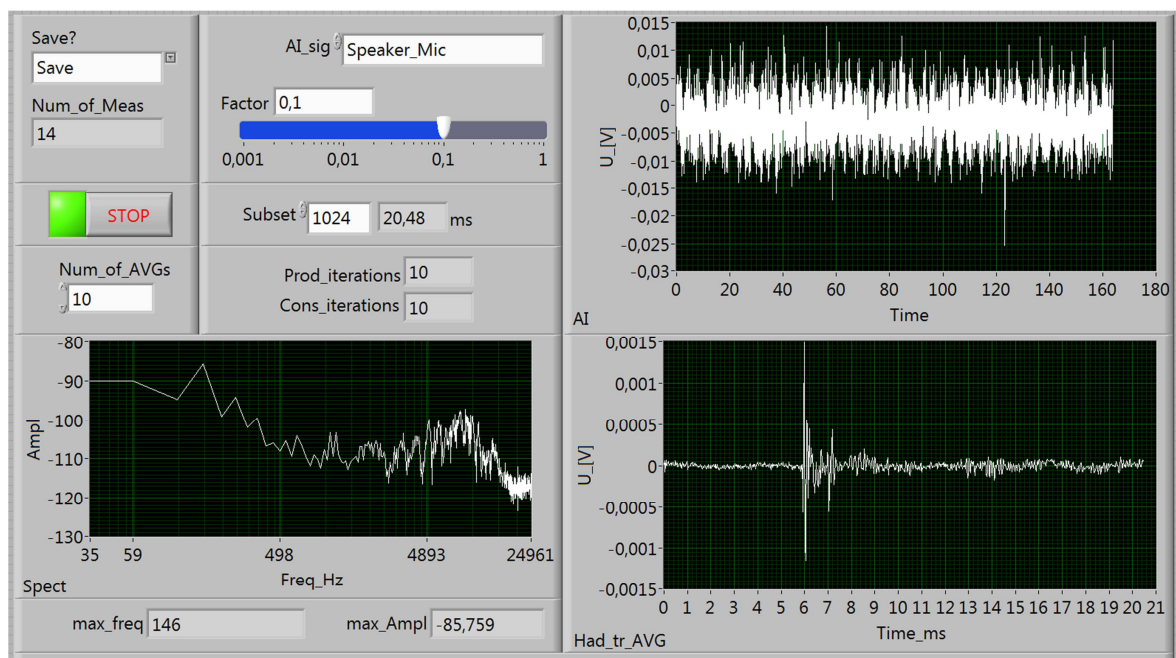
Irodalomjegyzék

1. **J. MacWilliams, N. J. A. Sloane.** Pseudo-Random Sequences and Arrays. *Proceedings of the IEEE*. 1976.. december, old.: 1715-1729.
2. **Simonyi Ernő.** *Digitális szűrők. A digitális jelfeldolgozás alapjai.* Budapest : Műszaki könyvkiadó, 1984.
3. **J. Borish, J. B. Angell.** An Efficient Algorithm for Measuring the Impulse Response Using Pseudorandom Noise. *J. Audio Eng. Soc.* 1983.. július, old.: 478-487.
4. **D. D. Rife, J. Vanderkooy.** Transfer-Function Measurement with Maximum-Length Sequences. *J. Audio Eng. Soc.* 1989.. június, old.: 419-444.
5. **National Instruments.** NI USB-6211 data sheet. *www.ni.com*. [Online] 2012. <http://sine.ni.com/ds/app/doc/p/id/ds-9/lang/hu>.
6. **National Instruments.** NI cRIO-9074 data sheet. *www.ni.com*. [Online] 2012. <http://sine.ni.com/ds/app/doc/p/id/ds-204/lang/hu>.

Ábrajegyzék

1. ábra. MLS jelet előállító shift regiszteres kapcsolást definiáló primitív polinomok	8
2. ábra. MLS jelet előállító shift regiszteres kapcsolat	9
3. ábra. MLS elemek sorozata – $m = 4$ esetén.....	10
4. ábra. MLS jel	11
5. ábra. Transzformáció 8x8-as Sylvester-mátrixszal.	15
6. ábra. 8-pontos Hadamard-transzformáció folyamatábrája	16
7. ábra. N mátrix oszlopainak rendezése $P1$ permutáló mátrixszal.....	18
8. ábra. N' mátrix sorainak rendezése $P2$ permutáló mátrixszal	18
9. ábra. LabVIEW front panelje és blokk diagramja	20
10. ábra. Ikon és Connector Pane.....	22
11. ábra. For Loop ciklus	23
12. ábra. While-Loop struktúra.....	23
13. ábra. Case struktúra.....	23
14. ábra. Flat Szekvencia Struktúra	24
15. ábra. NI cRIO-9074.....	26
16. ábra. Project Explorer ablak.....	28
17. ábra. MLS_generation_SubVI.vi	28
18. ábra. Mérésadatgyűjtést végző, FPGA szintű VI.....	29
19. ábra. Producer-consumer struktúra	31
20. ábra. Az impulzusválaszt számoló VI blokk diagramja.....	32
21. ábra. RLC szűrő – C kimenettel	34
22. ábra. RLC-tag impulzusválasza (Matlab).....	35
23. ábra. RLC-tag Bode-diagramja (Matlab)	36
24. ábra. C kimenetű RLC-tag mérése (front panel).....	37
25. ábra. RC-tag – R kimenettel	37
26. ábra. R kimenetű RC-tag mérése (front panel).....	38
27. ábra. Mikrofon-hangszórómérés (front panel).....	39
28. ábra. Mikrofon-hangszórómérés (front panel) – 2 m -es távolsággal	43

Függelék



28. ábra. Mikrofon-hangszórómérés (front panel) – 2 m-es távolsággal