Alkalmazás-specifikus perifériavezérlő fejlesztése az ACU mikrokontrollerhez

I^2C interfészáramkör
Dokumentáció

Mikroelektronikai Laboratórium

Frisch Attila
Molnár Martin Péter

2021

Utoljára frissítve: 2021. 05. 12.

Bevezetés

A félév során egy I2C interfészáramkör tervezése a feladat. Az I2C egy multi-master, multi-slave, csomagkapcsolt soros busz. Esetünkben csak az általunk tervezett áramkör lehet master a buszon. Az I²C jellemzően kis távolságú, viszonylag alacsony sebességű IC és fedélzeti rendszerek közötti kommunikációra szokás alkalmazni.

A feladatkiírásban az alábbi követelményeknek szerepelnek:

- Az SDA vonal háromállapotú meghajtóját az áramkör NE tartalmazza. Helyette az áramkör interfésze tartalmazzon egy „enable_output” jelet, amely a külső háromállapotú meghajtót engedélyezi.
- Szimbólumsebesség: futásidőben konfigurálható: 100 kb/s vagy 400 kb/s

I2C Master Transceiver

Az I2C Master Transceiver modul egy párhuzamos-soros, illetve soros-párhuzamos átalakítást megvalósító, egy állapotgépből álló egység, mely a soros oldalon I2C protokollt megvalósítva kommunikál.

Ez a kommunikáció "acknowledge" biteken alapuló handshake módszerrel történik.

Az előírt soros üzenet formátuma a következő:

start feltétel → címkeret → R/W (írás vagy olvasás) → ACK (handshake jel) → adatkeret 1 → ACK → ... → adatkeret N → ACK → stop feltétel

Az adatátvitel szimbólumsebessége futásidőben konfigurálható, egy lassú (100 kbps) és egy gyors (400 kbps) móddal rendelkezik, mely a további I2C interfésszel rendelkező eszközökkel való kommunikáció esetén érvényes. A mód kiválasztása az ACU által a szintézis paraméterként meghatározott "address_speed_ctrl" címre kiküldött adattal történhet.

Az I2C Master címbitjeinek száma (7 vagy 10) szintén hasonló módon választható az "address_address_frame" szintézis paraméter segítségével.

Az alábbiakban a modul ki- és bemeneteit tüntettük fel azok irányával, méretével és funkciójával, emellett megtalálható a modul szintézis paramétereinek leírása, az elvárt hullámformák, illetve az eszköz blokkvázlata.

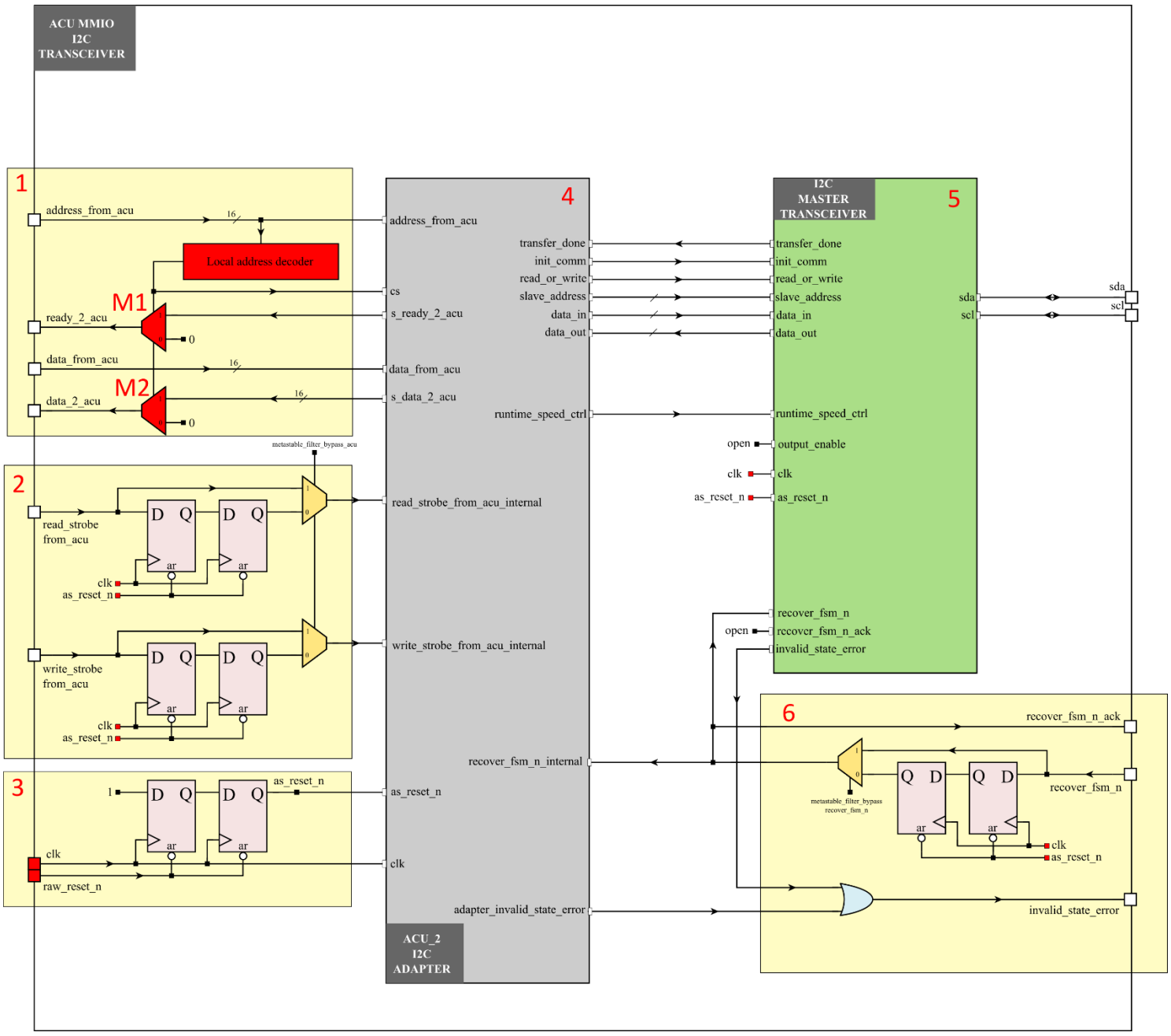
Portlista – I2C transceiver

Port neve	Port iránya	Port mérete	Port funkciója
SDA_in	I	1	Soros adat bemenet
SDA_out	O	1	Soros adat kimenet
SCL	I/O	1	Az I2C busz órajele
output_enable	O	1	Külső háromállapotú meghajtó engedélyezőjele (magas-aktív)
runtime_speed_ctrl	I	1	Működési mód kiválasztójele (0 = slow mode, 100kbps; 1 = fast mode, 400kbps)
init_comm	I	1	Magas-aktív vezérlőjel, melynek segítségével a beágyazó környezet adatátvitelt (írás / olvasás) kezdeményezhet, mely a stop feltétel teljesüléséig tart
transfer_done	O	1	Magas-aktív állapotjel, mely jelzi, ha az adatátvitel véget ért
read_or_write	I	1	Vezérlőjel, mely meghatározza, hogy a periférián írás vagy olvasás történik (0 = olvasás, 1 = írás)
slave_address	I	paraméter	Annak a slave eszköznek a címe, amelyből olvasást, vagy amelybe írást kezdeményez a beágyazó környezet
data_in	I		A beírandó adatbitek
data_out	O		A kiolvasandó adatbitek
invalide_state_error	O	1	Magas-aktív állapotjel, mely jelzi, ha az I2C adóvevő modul állapotgépe érvénytelen állapotba kerül
recover_fsm_n	I	1	Alacsony-aktív vezérlőjel, melynek segítségével az I2C adóvevő modul állapotgépe alaphelyzetbe állítható
recover_fsm_n_ack	O	1	Alacsony-aktív nyugtázójel, mely "recover_fsm_n"-hez tartozik

Szintézis paraméterek

Paraméter neve	Paraméter értéke	Paraméter funkciója
clk_in	50000000	A beágyazó rendszer órajel-frekvenciája (Hz-ben)
address_frame	7-10 közötti egész	A slave eszközök címbitjeinek száma
data_frame	8	A küldendő / fogadandó adatbitek száma
metastable_filter_init_comm	bináris (1/0)	Az init_comm jel metastabilitás-elkerülését segíti elő
metastable_filter_recover_fsm_n	bináris (1/0)	Az recover_fsm_n jel metastabilitás-elkerülését segíti elő

Blokkvázlat



1. ábra - Blokkdiagram

Az áramkör részei

1. **Lokális címdekóder:** Az ACU mikrokontroller elosztott címdekódolási sémát valósít meg. Minden perifériavezérlő saját címdekódert tartalmaz, amely a processzor címbuszának értéke alapján előállít egy lokális chip-select (cs) jelet. E chip-select jel logikai magas szintű, ha a processzor címbuszán lévő cím az adott perifériavezérlő címterébe tartozik. Esetünkben a perifériavezérlő címterébe a következő címek tartoznak:
    - address_init_comm
    - address_data
    - address_speed_ctrl
    - address_slave_address

    A lokális chip-select jellel vezérelt M1 és M2 jelű multiplexerek biztosítják, hogy a perifériavezérlő a ready_2_acu és a data_2_acu kimeneteket mindaddig logikai alacsony értéken tartsa, ameddig a processzor „meg nem szólítja” a perifériát. A lokális címdekóder egy kombinációs hálózat.

2. **MMIO strobe jelek metastabil szűrői:** Az MMIO perifériavezérlők és a processzormag aszinkron órajeltartományokat képviselhetnek, ezért a strobe jeleket metastabil szűrőkkel kell ellátni. Ezek a szűrő fokozatok nem valósítják meg a jel megérkezését nyugtázó visszacsatolást, mert ugyanezt az információt az MMIO interfész ready_2_acu jele is hordozza.

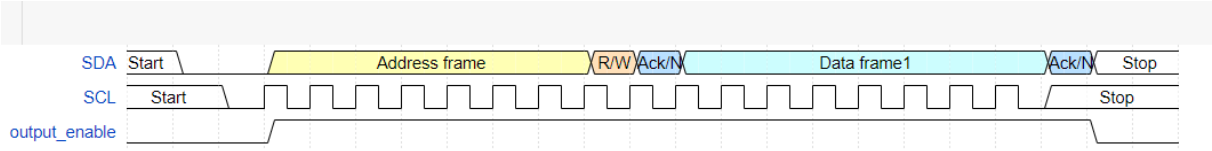
3. **Double-flop reset-szinkronizáló áramkör:** Az ACU mikrokontrollerben minden almodul (a processzormag és a különböző perifériavezérlők) saját reset-szinkronizáló fokozattal van ellátva, ami megkönnyíti a toplevel-integrációt, ahol már nem kell a reset-hierarchia kialakításával foglalkozni, elegendő az almodulokat példányosítani. Az almodulonkénti reset-szinkronizáció a reset jel terheléskiegyenlítését is szolgálja.

4. **MMIO adapter FSM:** Az MMIO interfész írás/olvasás kéréseit egy állapotgép fordítja le az I2C Master „nyelvére”; az MMIO írás/olvasás kéréseket értelmezve vezérli a perifériaáramkört. Az adapter attól függően, hogy melyik címen szólította meg az ACU, adatot továbbít vagy kér az I2C Master-től.

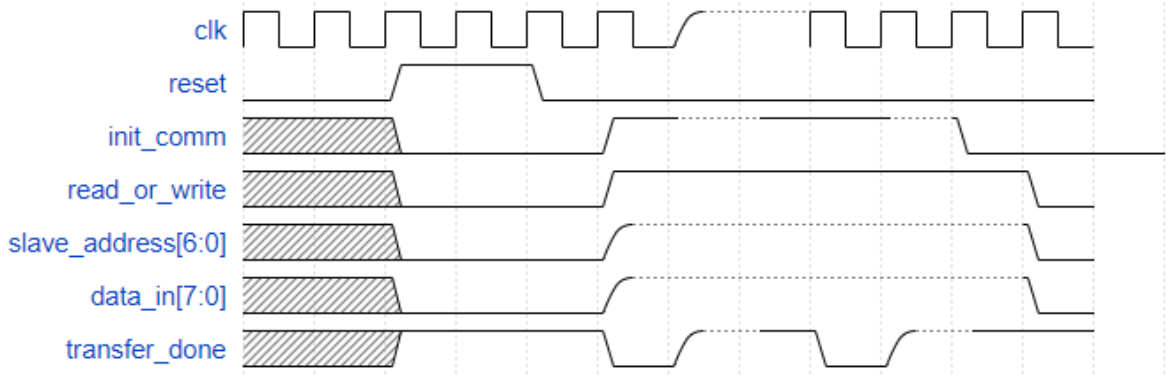
5. **I2C Master:** Az egyedi interfésszel rendelkező, újra felhasználható modul, amelyet a 1. ábrán bemutatott egyéb áramköri részletekkel teszünk ACU MMIO kompatibilissé.

6. **Állapothelyreállító interfész:** Az ábrán vázolt elrendezés két állapotgépet feltételez, egyik maga a USER LOGIC jelű áramkör, a másik pedig az MMIO interfész-adapter modul. Mindegyik állapotgép jelzi, ha valamilyen zavar miatt érvénytelen állapotba került. A USER LOGIC jelű modul természetesen nem feltétlenül kell, hogy állapotgép legyen, de ha az, akkor a fenti sémát érdemes követni az implementáció során. Mivel a hibadetektáló és naplózó alrendszer ugyancsak különálló órajeltartományt képviselhet, ezért az állapotgépeket alapállapotukba visszaállító bemenet (recover_fsm_n) esetén is szükség van a szintézis során kiiktatható double-flop szinkronizáló fokozatra.

I²C busz hullámforma



I2C Master és az ACU MMIO interfész adapter közötti kommunikáció



Áramkör fejlesztése

A rendszerterv megalkotása után elkezdhettük az áramkör fejlesztését, amely során egy VHDL nyelven megfogalmazott szintetizálható RTL modell készítése volt a cél. Ahogy a rendszertervben is láttuk, az áramkört érdemes két nagyobb részre bontani, magára az I2C Masterre és egy MMIO adapterre, ami megteremti a kapcsolatot az ACU és az I2C adó között.

MMIO adapter:

Az adapter feladata a kapcsolatteremtés az ACU és az I2C adó között. Mivel az ACU felől csak írási/olvasási kérelmek érkeznek, ezért szükség van generic paraméterként néhány fontosabb címet lefoglalni. Így az ACU ezekre a címekre küldött írási műveletekkel tud kommunikációt

kezdeményezni az I2C Masterrel. Az ’address_slave_address’ megcímzésével tud az ACU külső slave eszközt címezni, ilyenkor a ’data_from_acu’ bemenetre érkező 7 vagy 10 bites címet (szintézis paraméter) továbbítja az adapter az adó felé.

Futási időben konfigurálható az I2C adó szimbólumsebessége. Ennek változtatását az ’address_speed_ctrl’ megcímzésével teheti meg az ACU. Ilyenkor az adapter a ’data_from_acu’ bemenetére érkező adat LSB bitjét továbbítja az I2C Master felé.

Kommunikáció kezdeményezéséhez az ACU-nak először meg kell címeznie a külső slave eszközt, majd az ’address_data’ megcímzésével át kell adnia a küldendő adatot az adapternek, ami továbbítja az I2C adónak. És ezek után szabad csak megcímeznie az ’address_init_comm’ címet, aminek hatására az adó elkezdi kiküldeni az adatot.

I2C Master Transceiver:

A transceiver egység felépítése egyszerű, egy állapotgépből, számlálókból, illetve metastabilitás-szűrő, sorba kapcsolt flip-flopokból áll – természetesen reset-logikával kiegészülve -. Metastabilitás szűrésére a kommunikáció kezdeményezésére szolgáló ’init_comm’ és a hiba állapotból való visszatérésre használatos ’recover_fsm_n’ jelek esetében van szükség. A számlálók a külső eszközökkel való adatátvitelhez szükséges órajel előállításában, illetve a bitenkénti transzfer során történő indexelésben játszanak szerepet.

Az állapotgép hat különböző állapotot képes felvenni, melyek: ’wait_for_input’, ’change_config’, ’transaction_start’, ’addressing’, ’data_transfer’ és ’error’.

Wait_for_input állapot esetén a rendszer az ACU felől érkező jelre vár. Kétféle állapotátmenet lehetséges: amennyiben a ’runtime_speed_ctrl’ bemeneten 1-es érték érkezik, akkor change_config állapotba lép át a gép, ha ez nem történik meg, az ’init_comm’ értéke viszont logikai magas, akkor az SDA vonal értéke logikai alacsonyra vált, ezzel jelezve a kommunikáció kezdetét. Ezt követően egy órajelnyit késleltet, majd az eszköz transaction_start állapotba kerül. Eltérő esetekben az állapot nem változik.

A **change_config** állapot a futásidőben történő sebességváltoztatásra alkalmas, ekkor a rendszer kiolvassa a ’data_in’ adatbuszon érkező érték legkisebb helyiértékű bitjét. Ha ez a bit 0, akkor 100 kbps adatátviteli sebességre konfigurálja magát, ha 1, akkor a transzfer sebessége 400 kbps lesz. Ezt követően a rendszer wait_for_input állapotba tér vissza.

A **transaction_start** állapot szerepe, hogy engedélyezze a kimeneti háromállapotú buffert, valamint, hogy az SCL vonalat alacsony feszültségszintre húzza, majd egy órajelperiódusnyi idő elteltével addressing állapotba térjen át.

Addressing állapot esetén a rendszer bitenként kiküldi a 'slave_address' bemeneten kapott címet az SDA adatvonalra, ezzel párhuzamosan pedig a korábbi beállításnak megfelelő frekvenciájú órajelet állít elő az SCL vonalon. A slave címének kiküldését követően egy, az adatátvitel irányát meghatározó bitet is küld, majd vár egy nyugtázó jelre a megfelelő slave-től. Amennyiben ez a nyugtázójel nem érkezik meg, error állapotba kerül a transceiver, ellenkező esetben pedig data_transfer állapotba lép tovább, és megkezdődhet az adatátvitel.

Data_transfer állapotban az előző esetben említett órajel generálása folytatódik, az eszköz pedig írás / olvasás esetén továbbítja / buffereli az adatot az SDA vonalon keresztül. A fogadó eszköznek minden adatbyte-ot követően nyugtázójelet kell küldenie a küldő fél felé, ellenkező esetben a folyamat megszakad, és a küldő fél hiba állapotba lép. Amennyiben az adatátvitel befejeződött, az I2C Transceiver a 'transfer_done' jel által tudatja ezt az ACU-val. Az ACU is jelezhet az I2C Transceiver számára az 'init_comm' jel segítségével, hogy szeretné-e folytatni a kommunikációt a kijelölt slave eszközzel (ezesetben nincs szükség újabb címzésre), vagy befejezné azt. Ha a kommunikáció befejeződött, az eszköz ismét wait_for_input állapotba kerül.

Az **error** állapot szerepe a hibák jelzése, ekkor az eszköz vezérlőjelei újrainicializálódnak (reset). Ebből az állapotból a gép recover_fsm_n jel segítségével léphet vissza wait_for_input állapotba.

Reset esetén 'transfer_done', SDA és SCL logikai magas értékre kerülnek, az állapotgép állapota wait_for_input lesz, a többi kimenet, valamint a számlálók pedig 0 értéket vesznek fel.

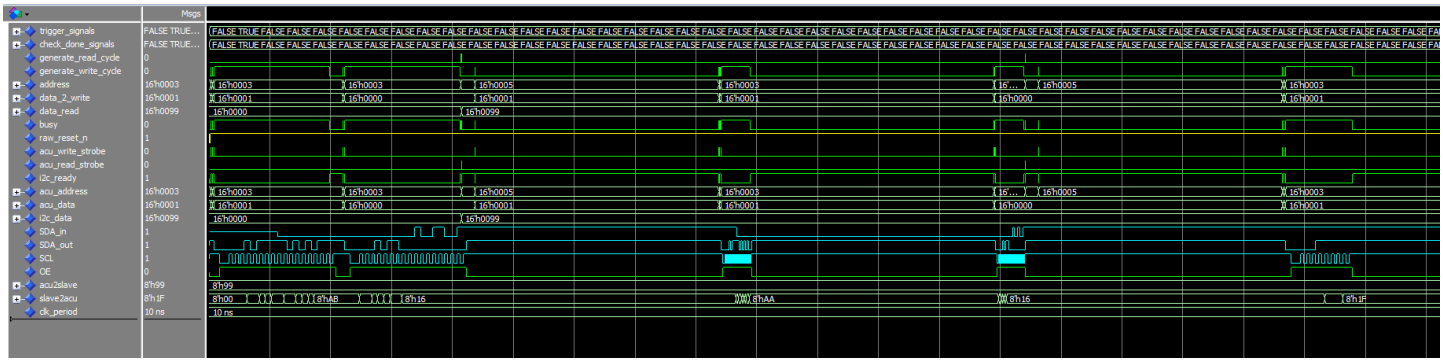
Eltérések a specifikációhoz képest:

Praktikussági okokból a szimbólumsebességet jelző 'symbol_rate' és az I2C busz órajelfrekvenciáját indikáló 'clk_bus' nem szintézisparaméterként, hanem lokális változóként kerültek megvalósításra, emellett a kétirányú SDA vonalat két egyirányú SDA_in és SDA_out portra bontottuk szét, az 'ack_or_nack' szintézisparaméter pedig eltávolításra került.

Funkcionális verifikáció:

A verifikációhoz automatizált regressziós tesztelésre alkalmas HDL tesztkörnyezeteket hoztunk létre. Ehhez szükség volt egy 'acu_mmio_bfm.vhd' fájlra, ami az ACU-t hivatott szimulálni a megfelelő jelek előállításával. Ezen kívül készítettünk egy I2C slave-t, ami pedig

a buszra kapcsolt külső eszközt szimulálta a verifikáció során. A testbench-ben mivel általunk kitalált bemenetekkel hajtottuk meg az áramkört, ezért tudtuk, hogy minek kell megjelennie a kimeneten. Így megfelelő report-ok elhelyezésével könnyedén végezhettünk szimulációt, hiszen nem kellett minden esetben a hullámformát vizsgálni. Az alábbi ábrákon látható a report-ok eredménye a kész RTL modell szimulációja után, valamint a hullámforma is, ahol például jól megfigyelhető a szimbólumsebesség változtatásának hatása az SCL jelen.



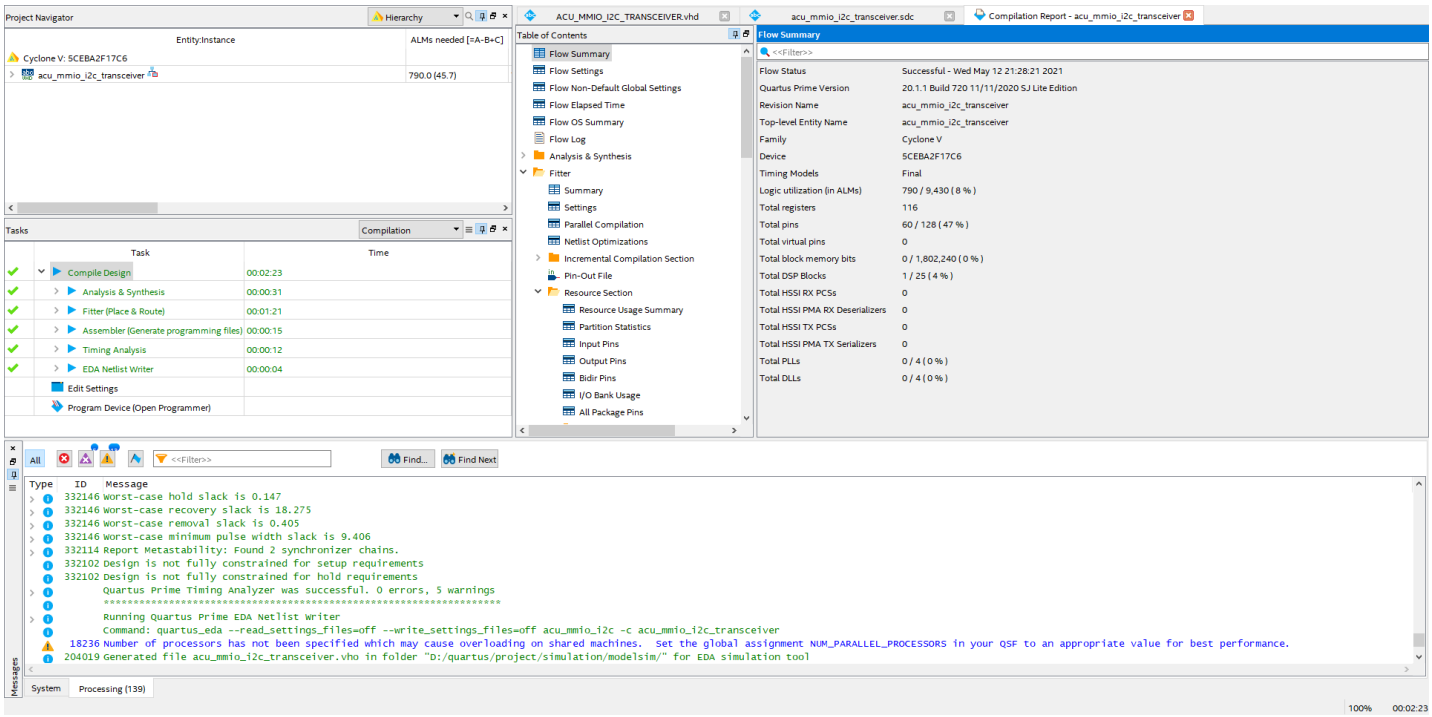
2. ábra – verifikáció során kapott hullámforma

```
VSIM 3> run
# ** Note: ------------------------------------------------------------------------> check  #1 OK
#    Time: 235 ns  Iteration: 0  Region: /tb_acu_mmio_bfm/L_CHECKS
# ** Note: Successful slow write operation
#    Time: 110030 ns  Iteration: 0  Instance: /tb_acu_mmio_bfm
# ** Note: Successful slow read operation
#    Time: 218390 ns  Iteration: 0  Instance: /tb_acu_mmio_bfm
# ** Note: Successful fast write operation
#    Time: 644610 ns  Iteration: 0  Instance: /tb_acu_mmio_bfm
# ** Note: Successful fast read operation
#    Time: 681710 ns  Iteration: 0  Instance: /tb_acu_mmio_bfm
# ** Note: Expectations met
#    Time: 950190 ns  Iteration: 0  Instance: /tb_acu_mmio_bfm
```

3. ábra – Automatizált tesztelés üzenetei

Automatizált RTL szintézis

Az automatizált RTL szintézist a Quartus Prime Lite Edition program segítségével végeztük. Itt kiválasztottunk egy létező FPGA család egyik eszközét, majd erre végeztük el az automatizált szintézist. A Compile Design gomb megnyomása után a szintézernek sikerült megvalósítania a tervezett áramkört az adott FPGA-ra. Ezután lehetőségünk van megtekinteni a kész áramkör részletes adatait, mint például pin kihasználtság, maximális működési frekvencia, felhasznált erőforrások. Annak érdekében, hogy az áramkör fogyasztását is meg tudjuk becsülni, szükség van a Power Analyzer Tool használatára, így a szintézis után ezt is megvizsgáltuk. A következő képernyőképek segítségével mutatjuk be a fontosabb paramétereit az áramkörnek.



4. ábra – Sikeres szintézis eredménye – felhasznált erőforrások

Fitter Resource Usage Summary

	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	790 / 9,430	8 %
2	ALMs needed [=A-B+C]	790	
1	[A] ALMs used in final placement [=a+b+c+d]	797 / 9,430	8 %
1	[a] ALMs used for LUT logic and registers	38	
2	[b] ALMs used for LUT logic	749	
3	[c] ALMs used for registers	10	
4	[d] ALMs used for memory (up to half of total ALMs)	0	
2	[B] Estimate of ALMs recoverable by dense packing	18 / 9,430	< 1 %
3	[C] Estimate of ALMs unavailable [=a+b+c+d]	11 / 9,430	< 1 %
1	[a] Due to location constrained logic	0	
2	[b] Due to LAB-wide signal conflicts	0	
3	[c] Due to LAB input limits	11	
4	[d] Due to virtual I/Os	0	
3			
4	Difficulty packing design	Low	
5			
6	Total LABs: partially or completely used	104 / 943	11 %
1	-- Logic LABs	104	
2	-- Memory LABs (up to half of total LABs)	0	
7			
8	Combinational ALUT usage for logic	1,441	
1	-- 7 input functions	3	
2	-- 6 input functions	116	
3	-- 5 input functions	380	
4	-- 4 input functions	317	
5	-- <=3 input functions	625	
9	Combinational ALUT usage for route-throughs	4	
10			
11	Dedicated logic registers	116	

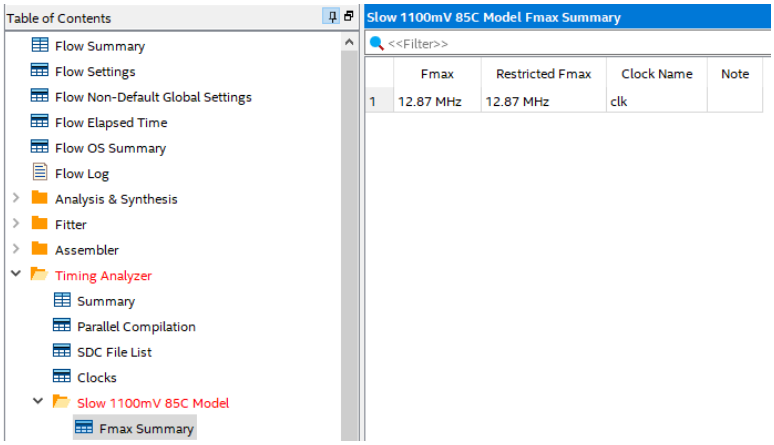
Fitter Resource Usage Summary

	Resource	Usage	%
11	Dedicated logic registers	116	
1	-- By type:		
1	-- Primary logic registers	94 / 18,860	< 1 %
2	-- Secondary logic registers	22 / 18,860	< 1 %
2	-- By function:		
1	-- Design implementation registers	94	
2	-- Routing optimization registers	22	
12			
13	Virtual pins	0	
14	I/O pins	60 / 128	47 %
1	-- Clock pins	4 / 5	80 %
2	-- Dedicated input pins	0 / 11	0 %
15			
16	M10K blocks	0 / 176	0 %
17	Total MLAB memory bits	0	
18	Total block memory bits	0 / 1,802,240	0 %
19	Total block memory implementation bits	0 / 1,802,240	0 %
20			
21	Total DSP Blocks	1 / 25	4 %
22			
23	Fractional PLLs	0 / 4	0 %
24	Global signals	1	
1	-- Global clocks	1 / 16	6 %
2	-- Quadrant clocks	0 / 88	0 %
25	SERDES Transmitters	0 / 68	0 %
26	SERDES Receivers	0 / 68	0 %
27	JTAGs	0 / 1	0 %
28	ASMI blocks	0 / 1	0 %
29	CRC blocks	0 / 1	0 %

27	JTAGs	0 / 1	0 %
28	ASMI blocks	0 / 1	0 %
29	CRC blocks	0 / 1	0 %
30	Remote update blocks	0 / 1	0 %
31	Oscillator blocks	0 / 1	0 %
32	Impedance control blocks	0 / 3	0 %
33	Average interconnect usage (total/H/V)	1.5% / 1.5% / 1.6%	
34	Peak interconnect usage (total/H/V)	14.9% / 14.8% / 15.1%	
35	Maximum fan-out	116	
36	Highest non-global fan-out	89	
37	Total fan-out	6097	
38	Average fan-out	3.62	

5. ábra – Felhasznált erőforrások részletesen

A következő ábrákon a maximális működési frekvencia értéke, a becsült fogyasztás, valamint a szintézer által készített RTL modell blokksémája látható:



6. ábra – Maximális órajel

Power Analyzer Summary

<<Filter>>

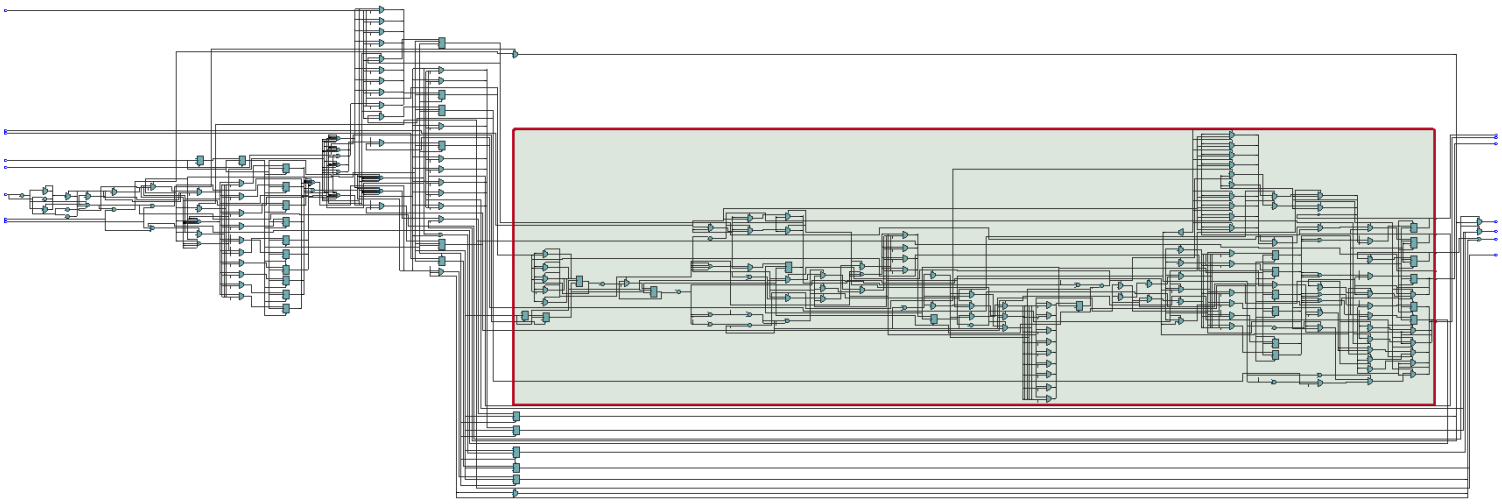
Power Analyzer Status	Successful - Wed May 12 21:49:22 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	acu_mmio_i2c_transceiver
Top-level Entity Name	acu_mmio_i2c_transceiver
Family	Cyclone V
Device	5CEBA2F17C6
Power Models	Final
Total Thermal Power Dissipation	208.54 mW
Core Dynamic Thermal Power Dissipation	5.36 mW
Core Static Thermal Power Dissipation	194.05 mW
I/O Thermal Power Dissipation	9.13 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Parallel Compilation

<<Filter>>

	Processors	Number
1	Number detected on machine	4
2	Maximum allowed	4
3		
4	Average used	1.30
5	Maximum used	4
6		
7	Usage by Processor	% Time Used
1	Processor 1	100.0%
2	Processor 2	10.0%
3	Processor 3	10.0%
4	Processor 4	9.9%

7. ábra – Becsült fogyasztás



8. ábra – Az áramkör RTL modellje