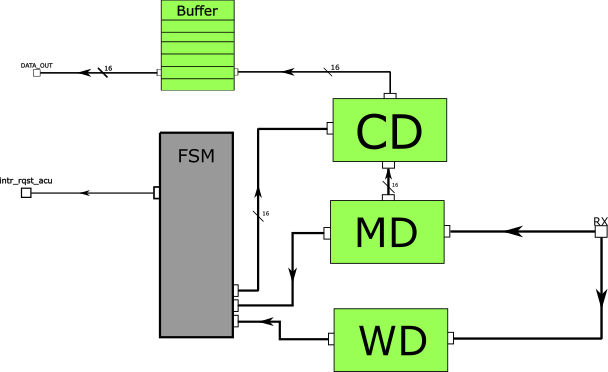
Mikroelektronika labor

PS2 perifériavezérlő rendszerterv

A picture containing shape

Description automatically generatedKészítette: Kovács Patrik  
Végh Gerzson



MD(Messege Decoder): A bemeneten érkező adatról eldönti, hogy az speciális karakter pl. Shit,Ctrl vagy pedig egy értelmes karakter mint pl 0-9 vagy a karakter.

WD(Whachdog): Ez egy időzítő, aminek a feladata az, hogy ha beérkezett egy karakter a bemeneten, akkor elkezd visszaszámolni, ha azelőtt érkezik egy karakter mielőtt lejárna a számláló akkor azt újra indítjuk. Ha lejárt az időzítő, akkor az állapotgépnek jelezzük, hogy a beérkezett karaktert folyamatosan lenyomott állapotban van.

CD(Character decoder): A feladata, hogy az állapot gép aktuális állapotától függően előállítson egy 16bit-es adatot aminek az első 8bit-je felgek amik tartalmazzák hogy egy gomb folyamatosan van e elnyomva vagy nem, üres-e a buffer, van-e error a működésben, az alsó 8 bit-re pedig a beérkezett karakter ASCII kódját helyezi.

## Implementáció

### Szintézis paraméterek:

* Érkezett karakter kiolvasás read címe
* Kiolvasható karakter(ek) elérhetősége read cím
* watchdog timeout ideje, az el nem engedett gombok kezelésére

### Az áramkör

#### bemenet:

#### *Rx* az uart bemenete, ps/2 billentyűzet adatvonal kimenete

1. ábra példa hullámforma, ami megjelenhet az rx bemeneten.

A billentyűzet 8o1 protokollal kommunikál.

#### kimenet:

a külvilág felé nincs kimenete

#### Periféria illesztő felépítése

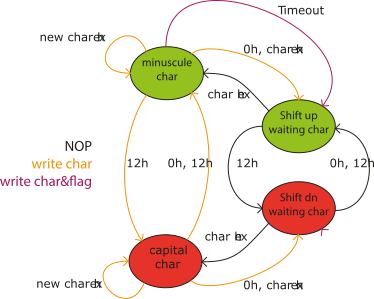
##### UART\_TRANSCEIVER

Az Rx fogadására a szolgáltatott uart\_transceiver hardware modelt használtuk fel, csak fogadó, paritás ellenőrzést letiltva 9 adatbit konfigurációval. A páros paritést a 8+1 ”adat” bitből számoljuk ki. A fogadott adat megjelenik a 8 bites párhuzamos kimenetén.

#### CHARACTER\_TY\_DECODER

A ps/2 billenytű kód byte-ot a character\_ty\_decoder dolgozza fel, pontosabban lefordítja a gombnak megfelelő kis- vagy nagybetű ascii kódjára, illetve érzékeli a különleges billenytű parancsokat. Ezeket 8 bites buszon és minden egyes spec karakternek saját vezetékén jelzi. Egy további logikai bit bemenete segítségével dől el a betű mivolta.

##### CHARACTER\_STATE\_MACHINE

Ez a core követi nyomon az érkező beviteleket, előéletét állapotváltozóban tárolja majd ezek és az elköveetkező bemenő ascii segítségével dönti el, hogy a 16 bites kimenetén milyen karakter jelenjen meg milyen mellékes flagekkel, valamint, hogy mikor kerüljön karakter beírásra az ideiglenes FIFO tárolóba. 

##### WATCHDOG

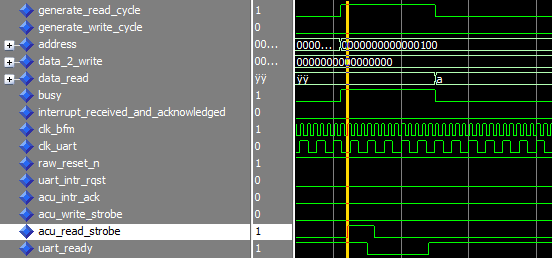
A watchdog figyeli, hogy van-e kommunikáció. Ha szintézisparaméterként megadott idejig nem érkezik új üzenet, a ”beragadt” karakter a megfelelő flag kíséretében kiírásra kerül a CSM által; erre a parancsot egy timeout bit állítása adja ki, és erre egy nyugtázást vár.

##### EDAC\_PROTECTED\_FIFO

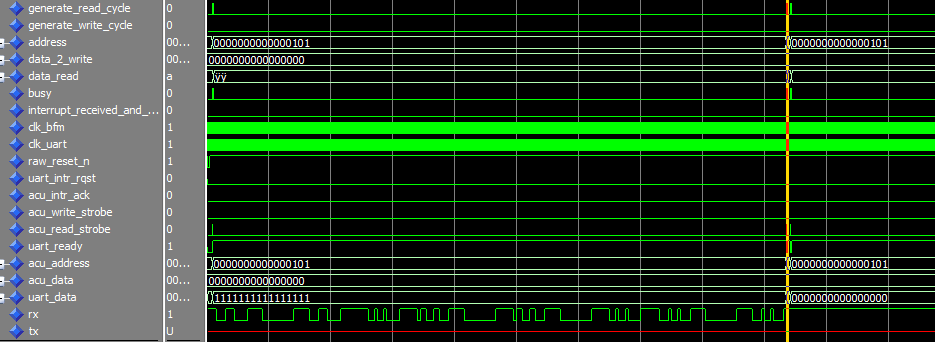
Az ideiglenes tároló megvalósításásra a szolgáétatott edac\_protected\_fifo IP core-t használtuk fel. Az architektúra szolgáltatásaiból a beírást (CSM által), pop-olást használjuk adat kezelésre. A core empty flagjét felhasználjuk az eltárolt karakterek meglétének jelzésére, ennek negatív átmenetével interrupt jelet generálunk a processzor felé, illetve a flag értéke ki is olvasható, hogy megyőződjünk róla, hogy kiürítettük azt.

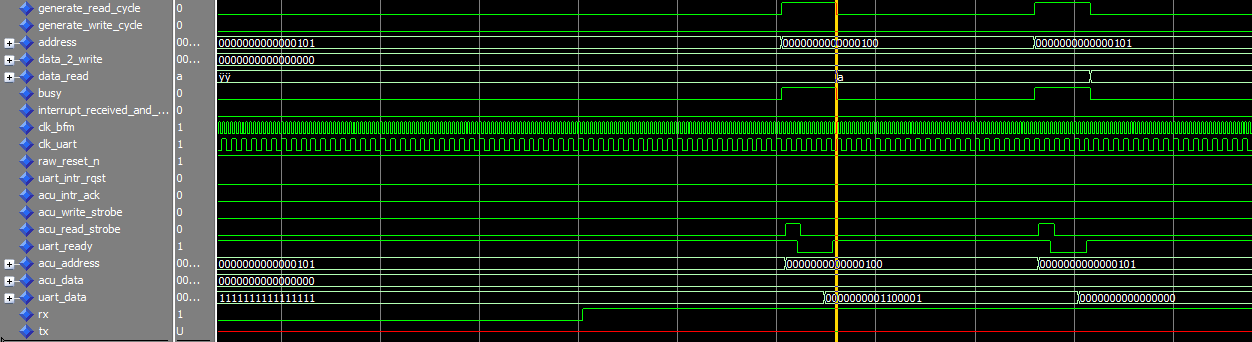
##### ACU\_MMIO\_PERIPHERIAL\_ADAPTER FINITE STATE MACHINE

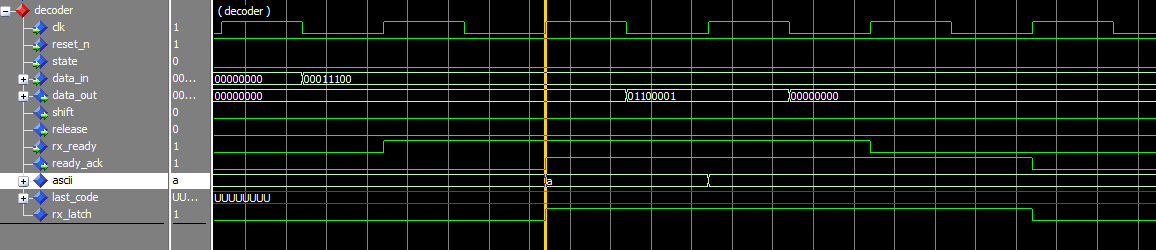
A processzor által kiadott memória műveletek (MMIO) lefordítása a periféria műveleteire a séma állapotgépének a feladata. A két elfogadott művelet reard\_req és flag\_req. Adott címről olvasás műveletre időzítést betartva a FIFO-ból egy 16bites adat (ascii + flagek) kiolvasása történik meg. Másik adott memória címről olvasásra, időzítéseket betartva a FIFO ”ürességéről” kérhetünk információt, logikai érték 16 bitre kiterjesztve. További címekről olvasási és bármilyen írás műveletre nem reagálunk.



2. ábra Olvasási ciklus időzítés diagramja.







dekoder jelváltozás