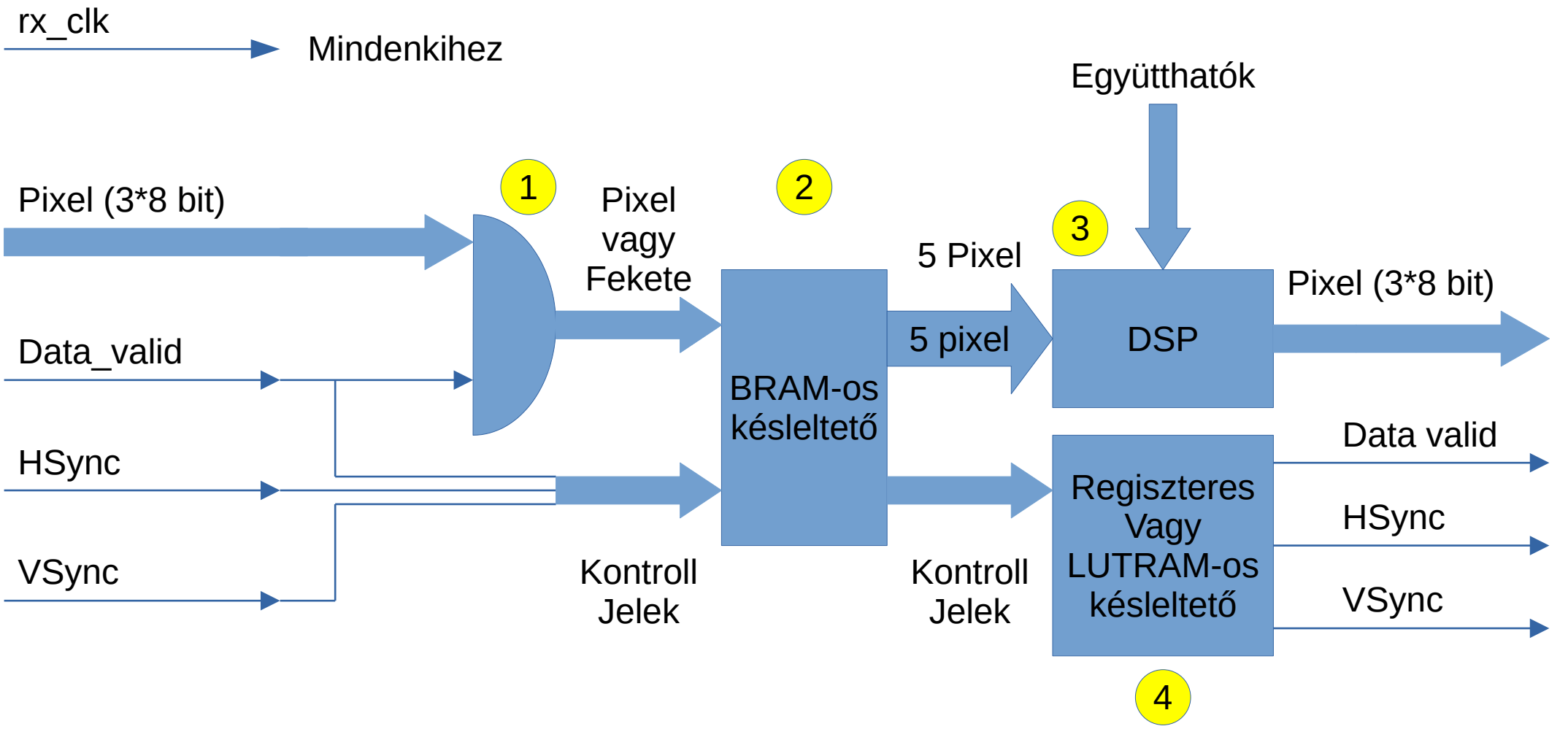
Logikai tervezés – Házi feladat blokkvázlat
Bálint Gergely
Szilágyi Gábor

Teljes blokkvázlat:

<u>bemenetek</u>

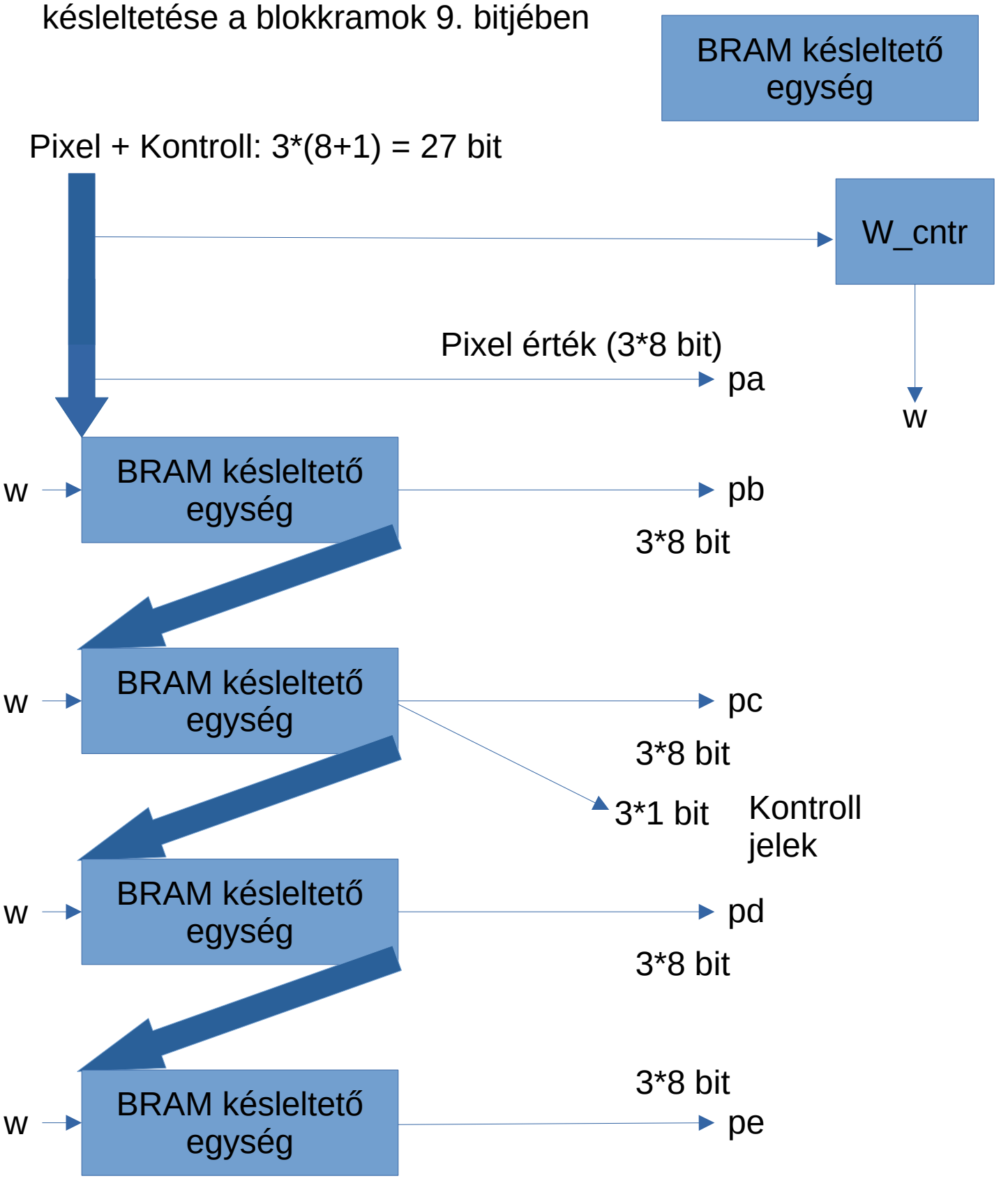
<u>kimenetek</u>



1. Blokk részletesen: sok ÉS kapu, a bejövő Data_valid jelet a bejövő Pixel bitjeivel egyenként összeéseli, így fekete pixellel számolunk tovább, ha a bejövő pixel érték érvénytelen.

Esetleg kell még az elejére egy pipeline regiszter-réteg, mert a data valid jel fanoutja itt 25 (26 a továbbvitt, késleltetett példányával együtt)

2. Blokk részletesen: beérkező pixel mentése és 5 különböző késleltetésű változat (pa, …, pe) előállítása a DSP rész bemenetéhez, valamint színcsatornánként 1-1 kontroll jel késleltetése a blokkramok 9. bitjében

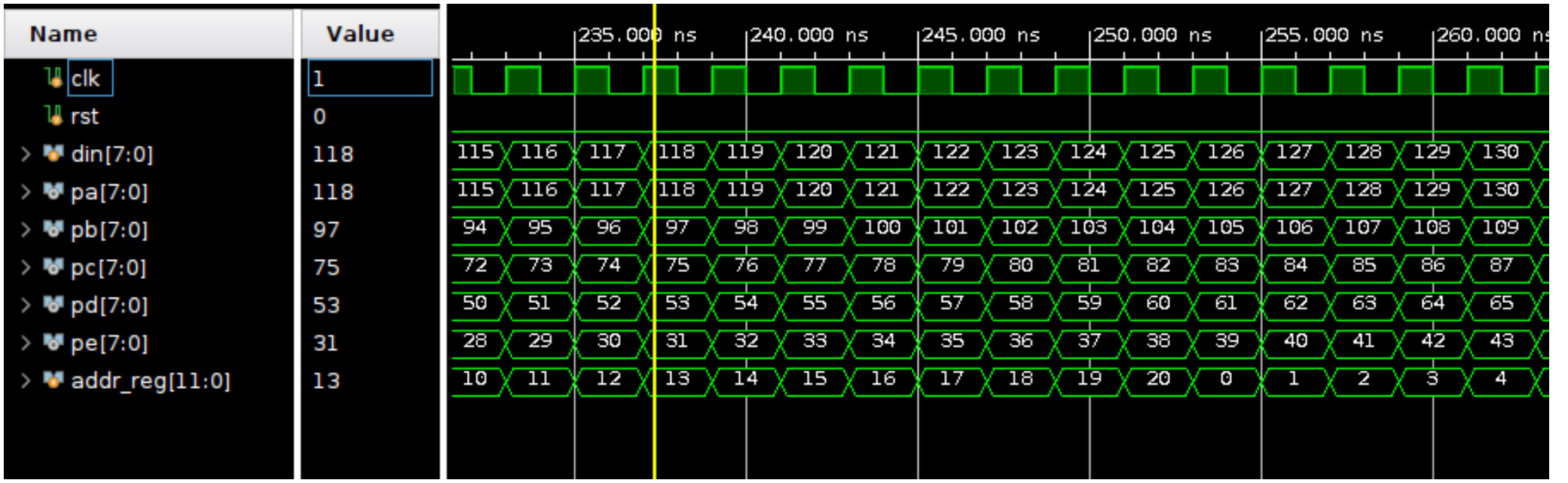


3 db párhuzamos, 4k x 9 bit blokkram, ciklikus címzés single port, read first mód, 8 bit (1 színcsatorna) + 1 bit (data_valid/hsync/vsync egy-egy színcsatorna mellé)

Egy számláló+ regiszter, ami megszámolja a két Hsync felfutó él között eltelő órajelciklusokat, vagyis a kép szélességét az invalid pixelekkel együtt (ez a w, ami 12 bites, mert 1080p esetén ennyibe fér bele a kép szélessége). Az előállított w értéket használják a BRAM-ok a ciklikus címzés reszeteléséhez.

Így minden sor hosszát az előző sor hosszának feltételezünk. Működés közbeni felbontásváltás esetén ez nem probléma, mert a hiba csak egy sorig áll fent, ami szabad szemmel nem látható.

2. Blokk szimuláció:
A címet fix értékig, 20-ig számítottam, így a BRAM-ok 21 mélyek voltak (0..20). A bemenetre növekvő számokat adtam, és a pa, pb, stb. kimeneteket vizsgáltam.

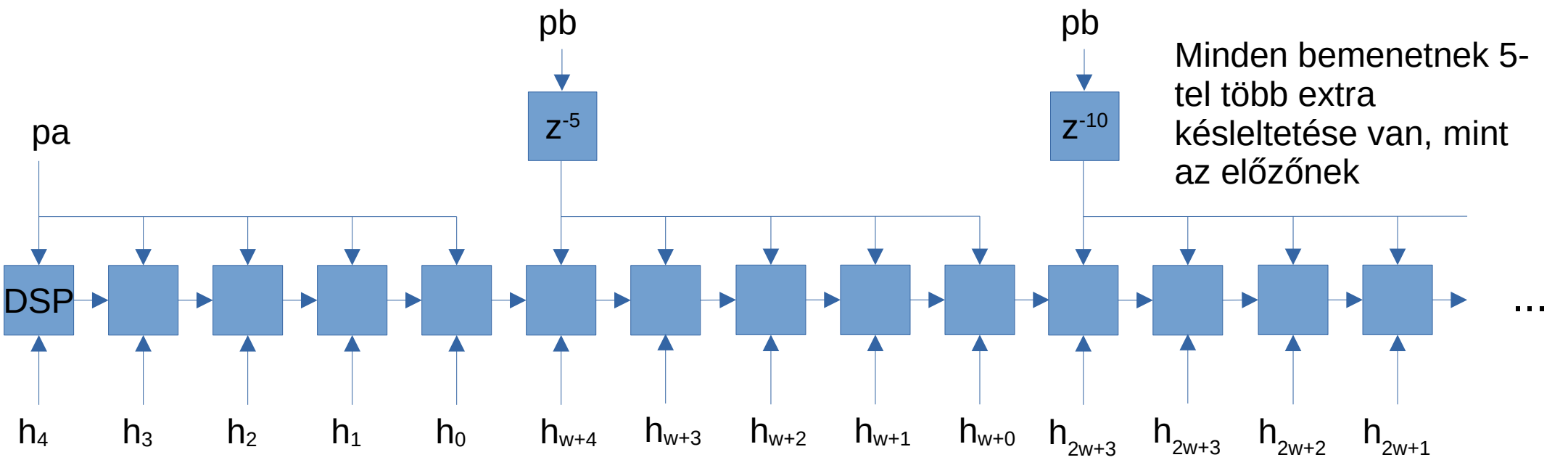


A címek 20-nél resetelődnek, ez látszik az utolsó sorban.
A pa kimenethez a bemenetet nem késleltetjük, így ezek megegyeznek.
A pb bemeneten lévő értékek 21-gyel kisebbek a pa-knál.
A pc, pd, pe kimenetek 22-vel kisebbek az előttük lévőnél. Ez első ránézésre hibás működésnek tűnt, ezért beraktam még egy késleltetést a rendszerbe, amellyel ezeket kompenzáltam. Ez azonban nem volt jó megoldás, mert a valós hardveren való tesztnél kiderült, hogy úgy elcsúsznak egymáshoz képest a sorok, így a szűrők nem működnek megfelelően. Így (tehát a 22-es késleltetésekkel) viszont megfelelően működött a rendszer.

3. Blokk részletesen: DSP

Szorzás: a súlyozómátrix sorait meg lehet valósítani kasztkád struktúrával, mert az csak ugyanannak a pixel streamnek a különböző késleltetésű változatait adja össze.

Egy DSP blokk elvileg 3 órajel késleltetést jelent (ha minden belső pieline regiszterjét használjuk) VERIFIKÁLNI



1 új pixelérték kiszámításához a 25 együtthatót 25 különböző idővel késleltetett pixelértékkel kell szorozni, és ezeket összeadni (késleltetések és együtthatók a következő oldalon). Az együtthatóablak egy sorában lévő pixelek késleltetése egyesével növekszik, ez azt jelenti, hogy az a sor elején lévő pixelt az aktuális és következő 4 órajelben kell felhasználnunk, majd egy sorhossznyi késleltetés múlva újra.

Fent látható, hogy az adott pixelt a kaszkádosított szorzó-összeadó blokkba emiatt egyszerre 5 helyen csatoljuk be, a +4 késleltetésről a számítás gondoskodik.
A pb pixelt 5 hellyel később csatoljuk be a kaszkádszámításba, így elcsúszna a hozzá tartozó pa pixeltől. Ennek a megoldására az előző oldalon látható 5-ös késleltetések szolgálnak. Ezekkel a később becsatolt pixelek késleltetését “kiegyenlítjük”.

z^{-n}

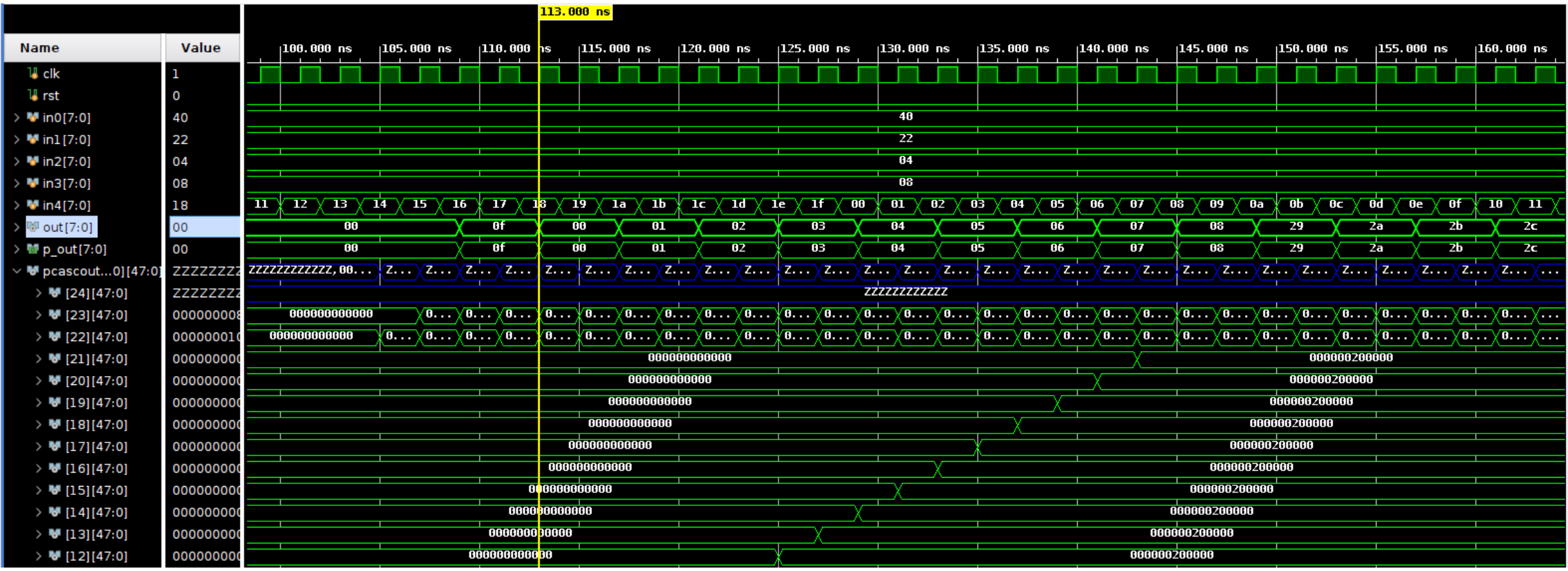
n órajellel késleltető
FF-halom, vagy
BRAM

3. Blokk szimuláció:

A DSP blokk szimulációjához egy egyszerű együtthatókészletet választottam. $h_{4w+2}=h_{4w+2}=1/2$, minden más együttható 0 értékű. A gerjesztéshez a következők a bemeneti pixelértékek:

pa=8’b01000000
pe=8’b00000000

És ezen kívül pe minden órajelben egyet fölfelé számol. A kimeneten látszik, ahogy két órajelenként számol fölfelé egyet, ez az ½ értékű együtthatók miatt van így. Látszik ezen kívül az ábrán, ahogy az egyes láncba kapcsolt DSP slice-ok bemenetén végigcsorog a reset után az első nem 0 bemenet.



Késleltetések a konvolúcióhoz:

4+4W	3+4W	2+4W	1+4W	0+4W
4+3W	3+3W	2+3W	1+3W	0+3W
4+2W	3+2W	2+2W	1+2W	0+2W
4+W	3+W	2+W	1+W	0+W
4	3	2	1	0

Együtthatók nevei:

h_{4w+4}	h_{4w+3}	h_{4w+2}	h_{4w+1}	h_{4w+0}
h_{3w+4}	h_{3w+3}	h_{3w+2}	h_{3w+1}	h_{3w+0}
h_{2w+4}	h_{2w+3}	h_{2w+2}	h_{2w+1}	h_{2w+0}
h_{w+4}	h_{w+3}	h_{w+2}	h_{w+1}	h_{w+0}
h_4	h_3	h_2	h_1	h_0

4. Blokk részletesen: Státusz jelek késleltetése

A státuszjeleket a pixelekkel együtt késleltetjük a második blokk végéig. Itt a pixelek a DSP kaszkádba kerülnek, így a státuszjeleket a DSP kaszkád késleltetésével megfelelő mértékben kell késleltetni. Ez BRAM-mal és LUT-tal is megvalósítható, mivel viszonylag rövid késleltetésről van szó.

Specifikáció

A megvalósítandó feladat egy olyan IP létrehozása volt, ami egy videó adatfolyam képkockáin 5*5-ös kernellel konvolúciót hajt végre valós időben. A konvolúciós együtthatók széles keretek között fordítási időben állíthatóak, de futási időben 9 db. együtthatókészletből lehet választani kapcsolók segítségével. Az együtthatók 18 bites előjeles számok, amelyeknek az egészrész bitszáma is fordítási időben állítható. A két szélsőséges formátum:

s.0.17 bit (1 bit előjel, 17 bit törtrész); s.17.0 (1 bit előjel, 17 bit egészrész, 0 bit törtrész)

A tervezésnél korlátot elsősorban a maximális elérendő működési frekvencia jelentett, amit a maximális felbontás határoz meg. Úgy terveztünk, hogy a 1920*1080p felbontás 60 FPS mellett ne okozzon gondot, ez a Wikipedia szerint 148.5 MHz működési frekvenciát jelent, ami megfelel a pixelfrekvenciának.

Erőforrásigény

Mivel az RGB pixelek három komponenséhez külön-külön el kell végezni a 25 szorzás-összeadás műveletkombinációt minden órajelciklusban, ezért várhatóan 75 DSP slice-t használ a design.

Egy színcsatorna egy sorszélességnyivel való késleltetéséhez egy BRAM-ot használunk és összesen 4 sonyival kell késleltetni a pixelfolyamot, így ebből várhatóan 12 db-ot használ a design.

A tényleges erőforrásigények (a teljes projektre a HDMI interfésszel együtt):

[db]	DSP	BRAM	LUT	FF
Synth.	75	13	579	752
Impl.	75	13	563	755

Az erőforrásigények a HDMI interfész nélkül:

[db]	DSP	BRAM	LUT	FF
Synth.	75	12	232	302
Impl.	75	12	218	302