```
uint8_t cntr1 = 1;
uint8_t cntr2 = 10;
while (cntr1 != 0)
{
    cntr1 = cntr2 & 0x03;
    if (cntr1 > 0)  -> ez pl. lehetne branch if greater than zero
     { cntr2--; }
}   cntr2--;
```

jövendölés
ha cntr1 != 0 amíg nemmulla ⇒T
ha cntr1 > 0 amíg nagyobb ⇒T

	1.	2.	3.	4.	5.
állapotgép	00	01	10	11	10
cntr1	1	2	2	0	0
cntr2	10	10	8	8	7
jövendölés	N	N	T	T	T
tényleges megvalósulás	T	T	T	N	N

↓ innentől átugorja a ciklust
↳ vége

1. while pillanata
   1 != 0 ⇒ T

2. 10 -> 1010
   3 -> 0011
   0010 = 2 = cntr1
   ⇓ 2 > 0 ⇒ T
   cntr2 = 10-1-1 = 8

3. cntr1 = 2 cntr2 = 8
   2 != 0 ⇒ T

4. 8 = 1000 cntr2 = 8
   3 = 0011
   0000 = cntr1
   0 > 0 ⇒ nem igaz ⇒ N

5. cntr2 = 8-1 = 7
   cntr2 = 7
   cntr1 = 0
   cntr1 != 0
   0 != 0 ⇒ N

Teljesülési arány: $\frac{1}{5}$ ⇒ 20 %

Teljesítőképesség: $a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3 + a_4 \cdot b_4$!

2db 4x4-es mátrixot akarunk összeszorozni. A mátrixhoz LD, ST, ADD, MUL utasításokat használunk.

- Van egy sima proci: minden összeadást és szorzást külön el kell végezni
- Van egy SIMD: egy ütemben össze tud szorozni 4 széles tömböt egy másik 4 széles tömbbel, illetve van olyan művelete ami egy 4 széles tömb elemeinek összegeit egy ütemben kiszámolja.

(Adat felhozás, adatkiírás egy ütem)

a) Sima proci-nál hány művelet van külön-külön és egyben?

4x4 mátrix => 16 elem

LD: 2db 4x4-es = 32
ST: 1 mátrix szorzata + írjuk ki = 16
ADD: fent: 3 összeadás => 3.16 = 48
MUL: fent: 4 szorzás => 4.16 = 64

összesen 160

b) SIMD proci-nál hány műv. külön-külön és egyben?

LD: 2db 4x4 betöltése => 32
ST: 1 mátrixot írunk csak ki => 16
MUL: van 16 elem => eddig egy MUL-hoz 4 szorzás kellett => de most 1 lépésben el tudja végezni => 16

ADD: 4 számot össze tud adni egy sterne, 16 elemhez eddig elemenként 3 összeadás kellett (4 szám) => már nem kell => 16

összesen: 80

c) SIMD mennyivel gyorsabb a simánál, ha csak mátrix szorzást csinál?

$\frac{160}{80} = 2x$ gyorsabb

d) Mennyivel gyorsabb a SIMD proci ha csak az idő 50%-ban végez mátrix műveleteket?

$$S = \frac{1}{(1-p)+\frac{p}{N}} = \frac{1}{(1-0,5)+\frac{0,5}{2}} = \frac{4}{3} = 1,34$$

1, Adatfüggőség I-H
2, Jövendölés I-H
3, GPU, API, OpenCV I-H
4, GPU neurális háló I-H
5, Interface tulajdonságok ABCDEF
6, PCI Express I-H
7, Flip-flop tca, tsetup, thold
8, Xilinx 7 LUT RAM memóriás
9, FPGA I-H
10, AXI C kód
11, Soft CPU Xilinx vs. Intel
12, Xilinx SoC I-H
13, LNKO algoritmus -> Ismerd fel, milyen arch., töltsd ki a táblázatot
14, 8b/10b

1, CISC, RISC cisc/risc/ő2/egyik sem

2, párhuzamosítás (FMT, ILP, CMP, Inter wehalem, Intel hyper) I-H

3, Assembly párhuzamosítás

4, Amhdal (nis ga)

5, Interfészstruktúrajdonságok párosítás ABCDEFGHI

6, USB interjace I-H

7, Thunderbolt I-H

8, kijelző/megjelenítő esztk. I-H

9, PCI busz jelátvitel I-H

10, PCI express I-H

11, FPGA áramkör fejlesztés I-H

12, AXI olvasás átvitel ábra : ADDR mintavétel hanyadik Clk?
hány adatcsomagot fogad a master?
ki hajtja - AVALID
-RVALID
-RLAST
master v. slave?

13, Verilog kód

a) 100 MHz elég e $f_{max} < \frac{1}{t_{comb}+t_{rca}+t_{setup}}$

b) x-y megvalósít a) még mindig ua. e?

c) milyen hatása lenne ...

d) mit valósít meg always blokk (LUT, FF...)

c) RTL leírás

14, OpenCL kernel kód + mielő kérdés

- hány dimenzió? miből tudni?
- mire szolgál w argumentum? hogyan lehet kernel argumentumnak értéket adni?
- hány workitem?
- mire van get_global_id?
- melyik memóriában van xy?
- milyen típusú memóriára utal OpenCL?
- milyen regiszterbe képet le xy-t?
- mit jelentenek a regiszter bitjei
- hogyan éri el xy memót. CPU?
- kell e barrier(), miért?

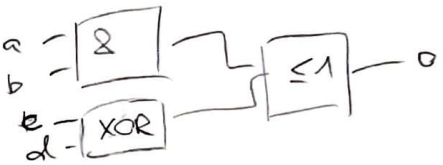
```
wire in;
reg a;
reg [7:0] out;

always@(posedge clk)
begin
  if(rst)
    a <= 1'b0;
  else if
    a <= in
end
assign o = ((a&b)|(c^d));
always@(posedge clk)
begin
  if(rst)
    out <= 8'b0;
  else
    out <= {out[6:0], o};
end
```

szinkron rst

assign o ⇒ wire típus: kombi log.
always@(a or b ...) ⇒ reg típus: kombi log.
always@(posedge clk) ⇒ reg típus: kombi + tároló

∧ - XOR



→ balra shiftelő reg

$t_{comb} = 7ns \quad t_{setup} = 3ns \quad t_{ac} = 1ns$

a) 100 MHz jó-e?

$$f_{clk} < \frac{1}{t_{comb} + t_{setup} + t_{ac}} = 90{,}90\ MHz \Rightarrow 100\ MHz < 90{,}904$$

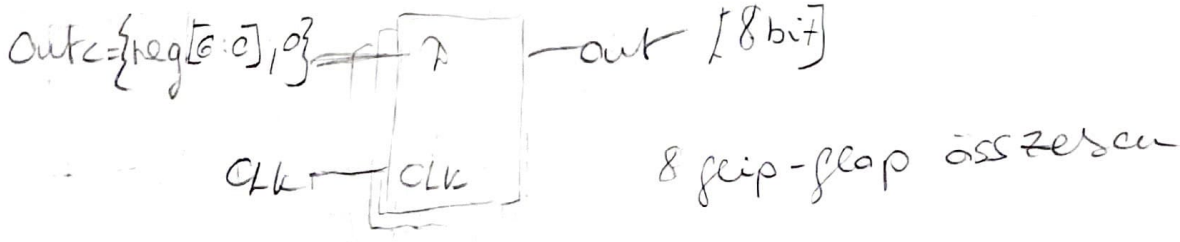
⇒ Nem

b) assign o=(a&b)|(c^d)) cseréli assign o=(a&b)|(c&d)) -re a?
előtte is 1 LUT és utánna is 1 LUT van.
↳ Nem változik mivel 4 2-höz 1 LUT kell.

c) Milyen hatása lenne ha * lenne az always blokknak?
4 változóra érzékeny ⇒ 4 változásnál újra számol ⇒ asztikron lesz.

d) 2. always mit valósít meg? mivel lehet?
Kombinációs (összefűz) ⇒ FF

e) RTL?



A R G B
8 8 8 8 = 32 bit
átlátszóság

Egy példa OpenCL kernel képfeldolgozást véget, a memóriában a kép soronként van eltárolva (row-major order). Egy képpontot (pixel) egy unisgned int típus reprezentál ARGB formátumban (A a legmagasabb, B a legalsó 8 biten). [A R G B]

```
__kernel void zebra( __global unsigned int *imagein, const int stride,
__global unsigned int *imageout) {

const int zebra_size = 30;

const float y_threshold = 235.0f;

int gid = get_global_id(0);

// Képpont koordináták meghatározása
int u = gid % stride;
int v = gid / stride;
int rgb = imagein[ gid ];
int r = (rgb >> 16) & 0xff;
int g = (rgb >> 8) & 0xff;
int b = (rgb) & 0xff;

// Világosság érték kiszámítása
float y = 0.30f * r + 0.59f * g + 0.11f * b;

if (y > y_threshold){
int zebra = (u+v) % zebra_size < zebra_size / 2 ? 255 : 0;
r = zebra;
g = zebra;
b = zebra;
}
imageout[gid] = 0xff000000 | (int)r << 16 | (int)g << 8 | (int) b;
}
```

zebra csíkok szélessége

határérték

aktuális workitem indexét kérdezi le

0xff ff ff : fehér

stride : sor hossza

x, y pixel koordináták

színek kiszedése

R | G | B
8 8 8
→ 16 bit
→ 8 bit

fehér-feketébe kiszámolás (0; 255)

ha a világosság > határérték

kiszámolja az új zebra változóval

kimenethez összefűzi RGB-t

- Mi utal az openCL-re?
  - -- global
  - -- kernel

de ha most csak get_global_id(0) van akkor nem 1 dimenziós?

- Helyes működés eléréséhez hány dimenzióban kell a work-itemeketet ütemezni? Honnan tudhatjuk?

2 dimenzióban. get_global_id(dim) -el lehet lekérni az adott dimenzióhoz tartozó indexet.

⇒ get_global_id(0) = x koordinátát get_global_id(1) = y koordinátát adja vissza

- Mi a funkciója a „stride” nevű argumentumnak? Hogyan adhatunk meg értéket?

stride: sorok számát adja meg.
Az értéket a kernel futtatásakor meg kell adni kernel argumentumaként: clSetKernelArg(...) fgv.

- W x H felbontású kép esetén hány work-itemet ütemezzünk, hogy helyes működést kapjunk?

Legalább W·H work-itemre

- Fizikailag milyen/melyik memóriában helyezkedik ez az „rgb” nevű változó? Milyen típusú memória ez OpenCL nevezéktan szerint?

int rgb: __globális memóriában van ⇒
GDDR RAM ⇒ private SGPR ?

- GCN architektúra esetén milyen regiszter segíti az if (y>y_threshold) elágazás megvalósítását, mit jelentenek az egyes bitjei?

VCC ink.

SCC - scalar condition code regiszter
Összehasonlítás eredményét workitemenként.

kisebb v. = = nagyobb v. = always ért?

- Az „imageout” puffer milyen típusú és melyik memóriában helyezkedik el? Hogyan éri el a CPU?

imageout: unsigned int típusú
__global memóriában van (ez tartalmazza a kimenetet mindig)
A CPU „read_image(..)” fgv.-t vagy „enqueueReadBuffer(..)” fgv.-t használja a globális memóriában lévő adatok olvasásához.

• kell e barrier()?

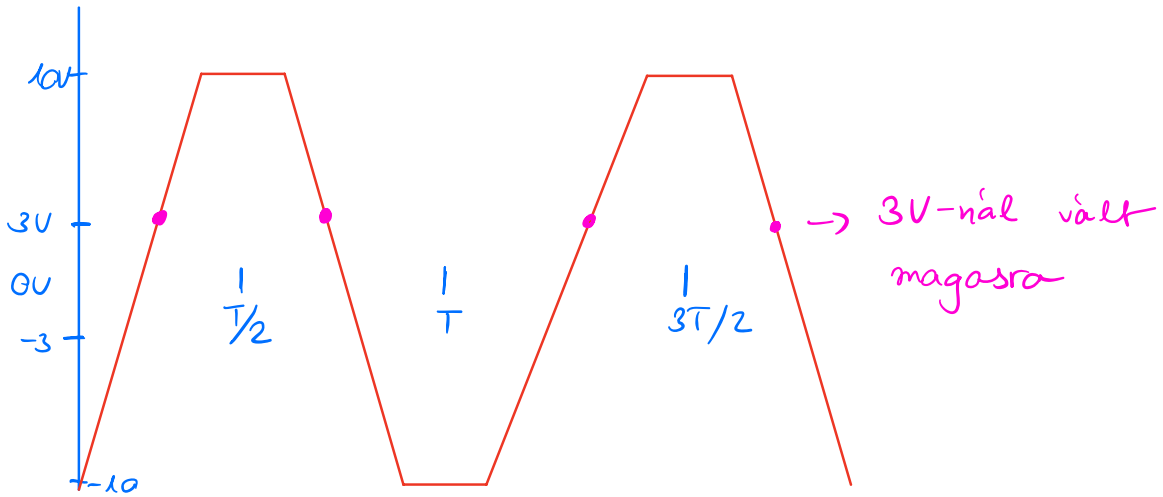
Akkor kell, ha több wavefrontunk van, mivel, ha az 1. elfutott de a 2. még nem került el akkor hiba lesz.

Barrier vár amíg minden WI eléri a barriert.

↳ workgroup > wavefront => szinkronizáció kell.

1, Utasítás készlet vs. mikroarch
2, Utasítás csővez. függőségek 1-H
3, Elágazásjóslás 1-H
4, ARMv7A vs. ARMv8A
5, GPU generációk 1-H
6, GPU RAM 1-H
7, OpenCL esetén honnan tudja a workitemet feldolgozó kernel, hogy milyen adatokat kell feldolgozni 1-H
8, PHIPS
9, USB interface 1-H
10, PCI buszcsalád 1-H
11, Verilog nyelven HDL fejlesztés
12, Xilinx Zynq, Xilinx MicroBlaze kiválasztás
13, Aszinkron, szinkron jelátvitel:
   a) (aszinkr) jelterjedési sebesség
   b) (szinkr) SR
   c) (szinkr) jelterjedési sebesség
   d) melyik PCIe lehet
14, AHB, AXI, AXI stream kód
   a) összehas.: topológia, sebesség, átvitel
   b) FIFO melyikkel, ha számít a gyorsaság
   c) Cortex M-hez milyen buszokat: RAM/FLASH, DMA, GPIO, Ethernet
      ↳ rajzold le, milyen topológiát használnál?
   d) AXI vagy AXI Stream a kód, miért?

13/ Asszimetrikus



a) SR = 3V/µs (T/2, 3T/2 ...) -ban ???

jelterjedési sebesség?

-10V → 3V = 13V

$\frac{13V}{3V/\mu s} = T$ ↓ fél periódus ↳ ·2

$f = \frac{1}{T} = \frac{1}{2 \cdot T}$

b) szimmetrikus ábra $V_{diff} = 2V$ 20% -80% – 15ns

- jelterjedési sebesség:

$$f = \frac{1}{(15ns \cdot 0{,}6) \cdot 2}$$

↓ egész periódus

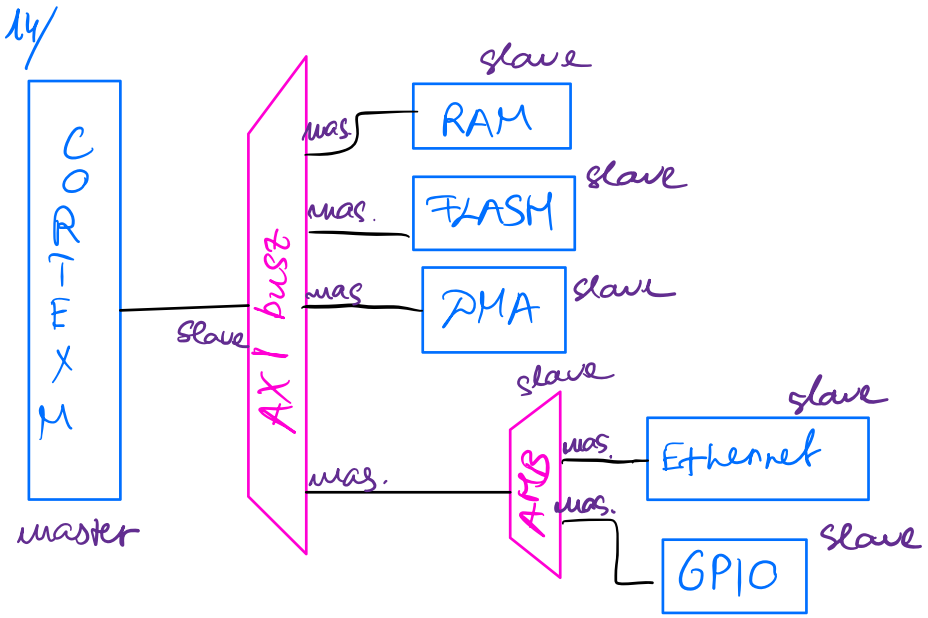
- $SR = \frac{2V}{15ns} \cdot 0{,}6$

a) $UI = 400\,ps$ $V_{TX} = V_{RX} = 4 \cdot 0{,}125$

Milyen SPIexpress ehhez ez?

$$\frac{10^6}{400} = 2500 \rightarrow 2{,}5\ Gbit/sec$$

14/



	AHB	AXI	AXI stream
topológia:	több master több slave	egy master egy slave	egy master egy slave
sebesség:	lassú	gyors	nagyon gyors
átvitel:	• pipeline • felfutóélnél mintavételez • burst	• part-part • pipeline • van címzés • burst	• nincs címzés • korlátlan burst

FIFO: a lényeg, hogy csak be kell rakni az adatot, nincs címzés

↳ AXI stream a leggyorsabb

Ha van + offset a kódban ⇒ AXI, ha nincs ⇒ AXI stream