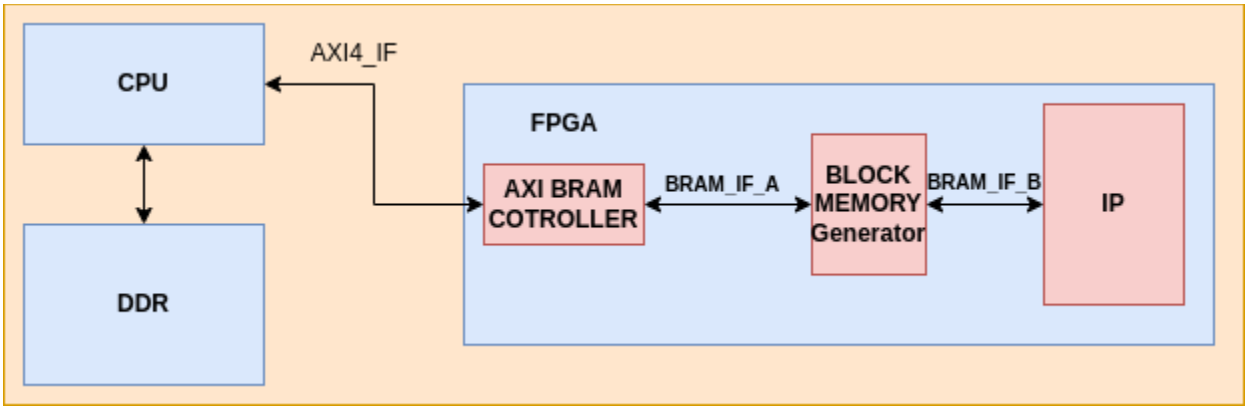
Vežba10 - BRAM Controller i BRAM generator

UVOD

Za većinu digitalnih sistema koji se implementiraju u FPGA delu čipa treba obezbediti podatke iz DDR memorije. Jedan od načina da se to uradi jeste da procesor prosledi te podatke direktno digitalnom sistemu preko AXI interfejsa. Na sledećoj slici prikazan je sistem koji omogućava da se ovako nešto izvrši:



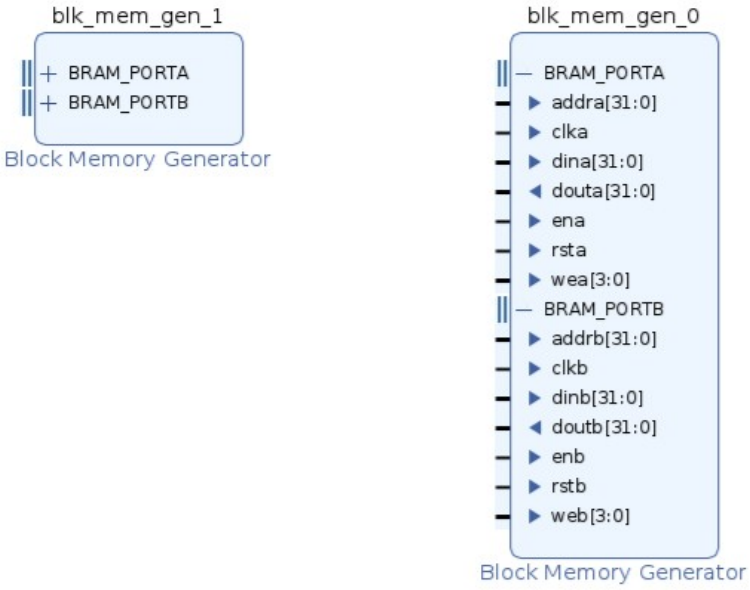
Slika 1. Komunikacija procesora i logike u FPGA delu čipa

AXI BRAM Controller i *BLOCK Memory Generator* su već gotove komponente kompanije Xilinx koje olakšavaju ovu komunikaciju.
Osnovna ideja jeste da se podaci koji se nalaze u DDR memoriji prebace u BRAM memoriju na FPGA delu čipa kako bi IP mogao da ih procesira. Da bi se to omogućilo neophodno je iskoristiti BRAM ćelije i to može da se uradi na dva načina:

- Korišćenjem gotovih modula
- Kucanjem HDL koda koji će se mapirati na BRAM ćelije.

BLOCK MEMORY GENERATOR

Mi ćemo u ovom materijalu koristiti prvu opciju, odnosno koristićemo BLOCK Memory Generator modul koji automatski pravi i konfiguriše BRAM ćelije:

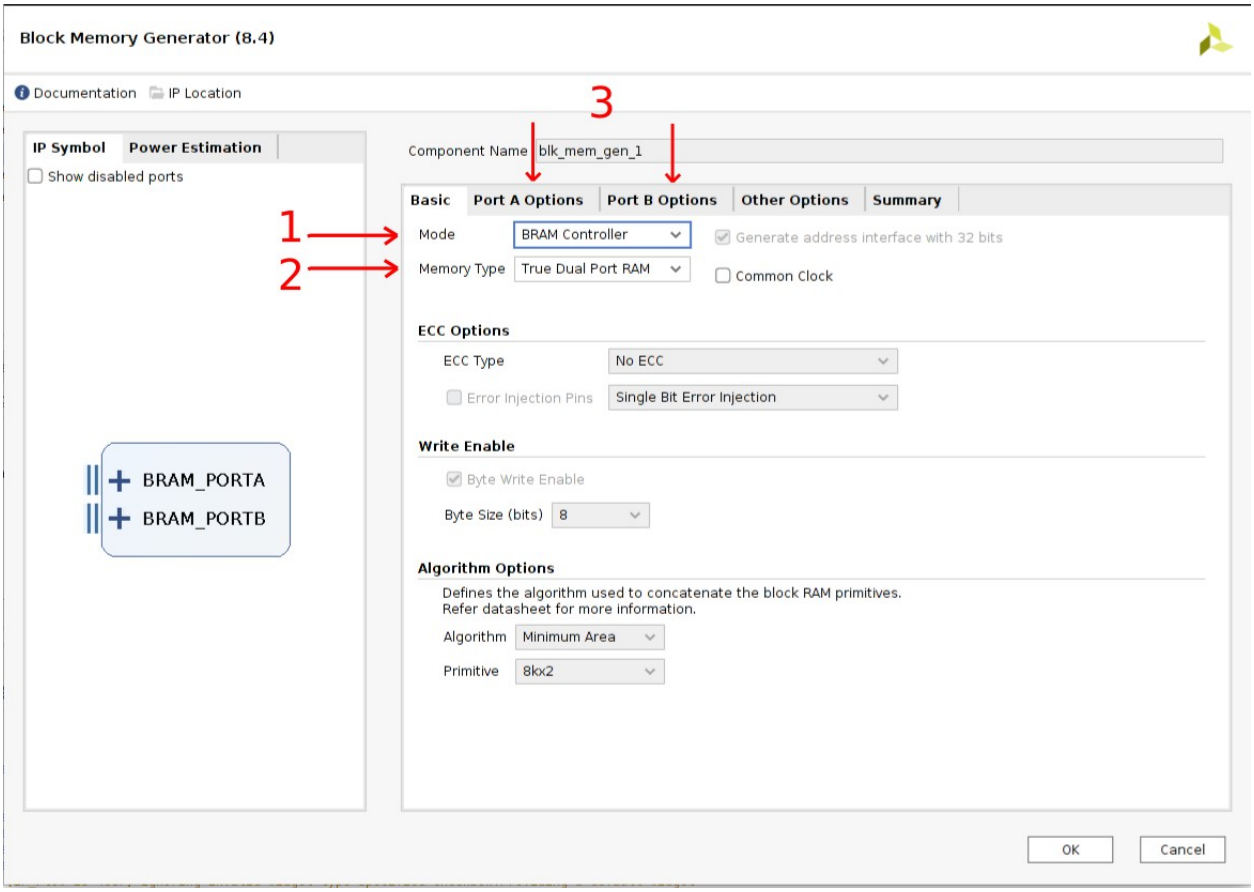


Slika 2. Block memory generator

Sa prethodne slike se može videti da ova komponenta može da se konfiguriše da ima dva interfejsa (BRAM PORTA i BRAM PORTB) preko kojih istovremeno može da se radi i čitanje i pisanje (Takozvana *True Dual Port* konfiguracija). Portovi koji čine ova dva interfejsa su:

- Addr - Adresa sa koje se čita iz memorije ili na koju se upisuje.
- Dinb - Podatak koji se upisuje u memoriju
- Doutb - Podatak koji se čita iz memorije
- Enb - Dozvola rada memorije
- Web - Dozvola upisa u memoriju. Ovaj port je na slici 2 četvorobitan i on određuje koji bajtovi 32-bitnog podatka će biti upisani u memoriju. Ukoliko je ovaj signal postavljen na 1111 sva četiri bajta biće upisana, a ako je postavljen na 0001 biće upisan samo LSB bajt. Takođe, samo ukoliko je Web = 0, može da se vrši čitanje iz BRAM memorije.

Dvoklikom na ovu komponentu u IP integrator mogu da se vrše konfiguracije Block Memory Generator komponente:



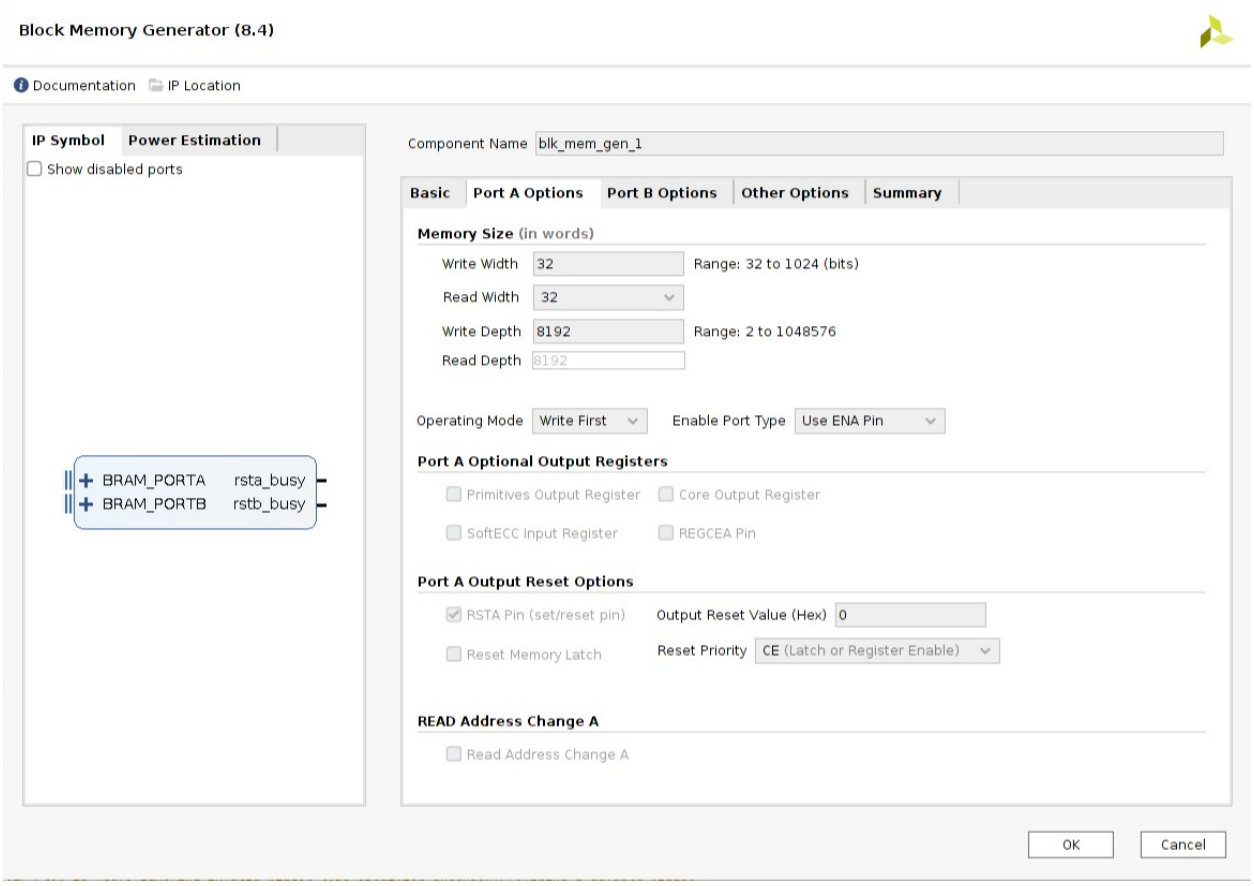
Slika 3. Konfiguracija *Block Memory Generator* komponente

Najbitnije konfiguracije su prikazane na prethodnoj slici:

1. ***Mode*** ima dva izbora:
    - *BRAM controller* - Ukoliko je ona odabrana, onemogućena je konfiguracija BRAM Memory Generator komponente od strane korisnika već će sva podešavanja biti određena podešavanjima unutar BRAM Controller komponente.
    - *Stand Alone* - Ovo omogućava korisniku da sam konfiguriše BRAM Memory Generator komponentu.
2. ***Memory Type*** ima četiri izbora i ovo određuje koliko interfejsa BRAM memorije će biti iskorišćeno i kako će se oni ponašati:
    - Single Port Ram - BRAM će imati samo jedan interfejs preko koga će moći da se radi ili čitanje ili upis
    - True Dual Port - BRAM će imati dva interfejsa preko kojih će moći da se radi ili čitanje ili upis.
    - Single Port ROM - BRAM će se ponašati kao ROM koji ima samo jedan interfejs za čitanje podataka

- Dual Port ROM - BRAM će se ponašati kao ROM koji ima dva interfejsa za čitanje podataka

3. Opcije za Port A i Port B - Ukoliko se BRAM konfiguriše Kao ***Stand Alone*** moguće je konfigurisati portove, odnosno menjati dubinu do koje može da se dođe preko određenog porta, širinu podataka koji se čitaju ili upisuju preko određenog porta, itd. Na sledećoj slici je to prikazano:



Slika 4. Konfigurisanje određenog BRAM porta

Za detaljnije informacije o BRAM Memory Generator komponenti pogledati: Link

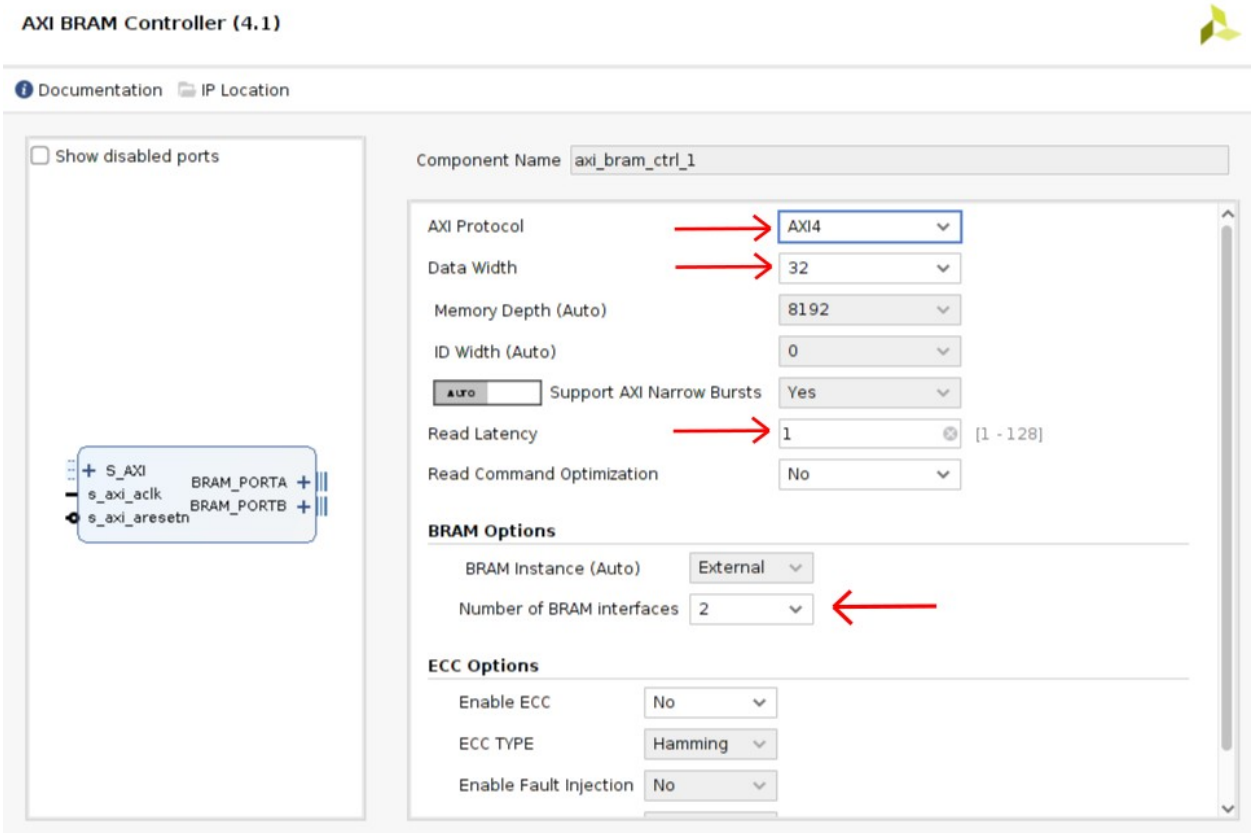
AXI BRAM CONTROLLER

Ovo je komponenta koja omogućava da procesor pošalje podatke prema logici u FPGA delu čipa i na sledećoj slici je prikazan ovaj blok u IP integratoru:

Slika 5. IP integrator, *BRAM controller*

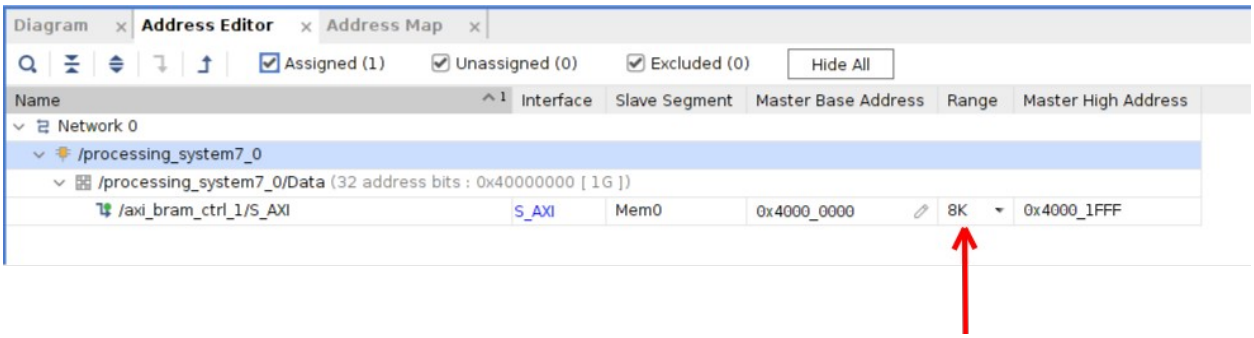
Može se videti da on sa jedne strane ima AXI Full interfejs, a sa druge BRAM interfejs. Ukoliko procesor treba da pošalje nešto u BRAM memoriju, neophodno je njegov AXI interfejs povezati sa AXI interfejsom BRAM controller komponente i BRAM controller povezati sa BLOCK memory generator komponenentom.

Dodatne konfiguracije ove komponenta se takođe mogu otvoriti dvoklikom na blok u IP integrator (Strelicom su naznačene opcije koje se najčešće menjaju):



Slika 6. AXI Bram controller konfiguracija

Konfiguracija koja ne može da se menja tek tako jeste *Memory Depth,* na prethodnoj slici postavljena na 8192. Da bi se ona promenila neophodno je u Adress editor prozoru proširiti broj memorijski mapiranih lokacija BRAM controller komponente:

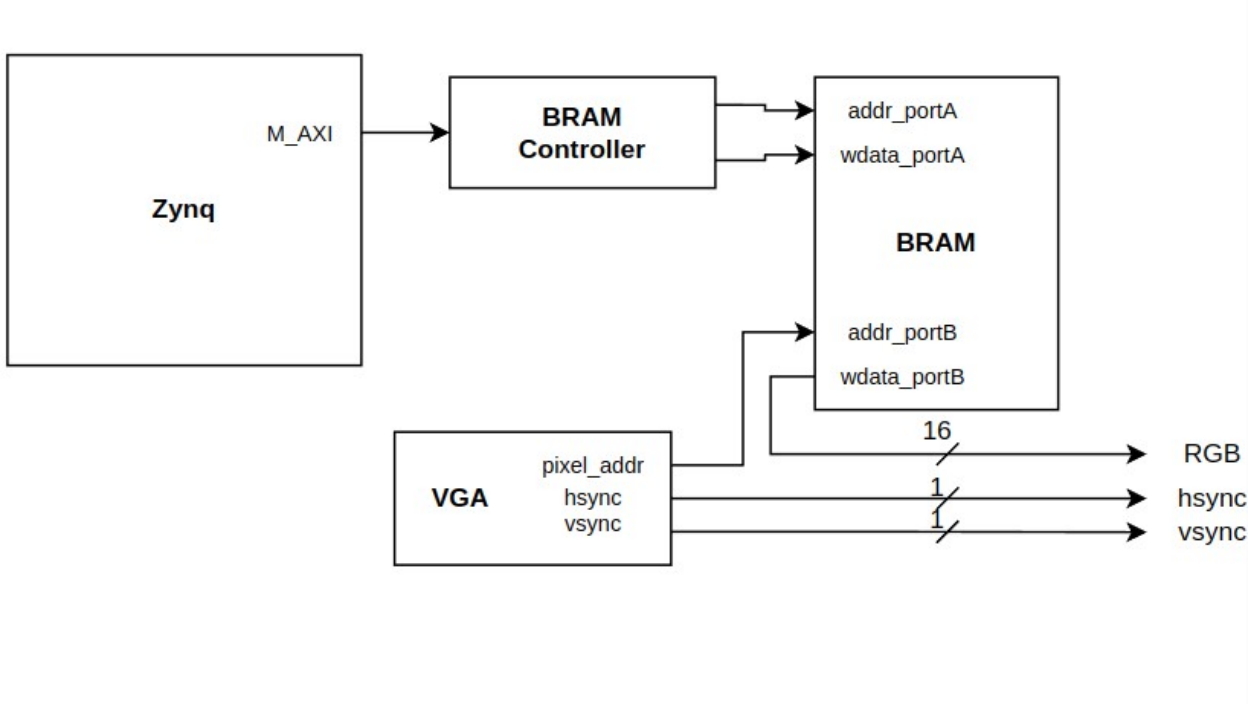


Slika 7. Memorijsko mapiranje BRAM controller komponente.

Na prethodnim laboratorijskim vežbama pomenuto je da svaka komponenta koja je preko AXI interfejsa povezana sa procesorom dobija memorijski mapirani adresni opseg preko koga procesor može da pristupi toj komponenti. Isto važi i za BRAM controller. Kako je BRAM controller povezan sa BLOCK memory generator komponentom, svaki podatak kome on može da pristupi je memorijski mapiran na određenu adresu. Ukoliko je potrebno pristupiti većem broju podataka memorijski opseg sa prethodne slike se treba proširiti (podrazumevana vrednost je 8Kb).

VGA

Da bi se demonstrirao način korišćenja prethodno opisanih komponenti realizovaćemo logiku za prikaz slike na monitoru preko VGA interfejsa na način prikazan na sledećoj slici:



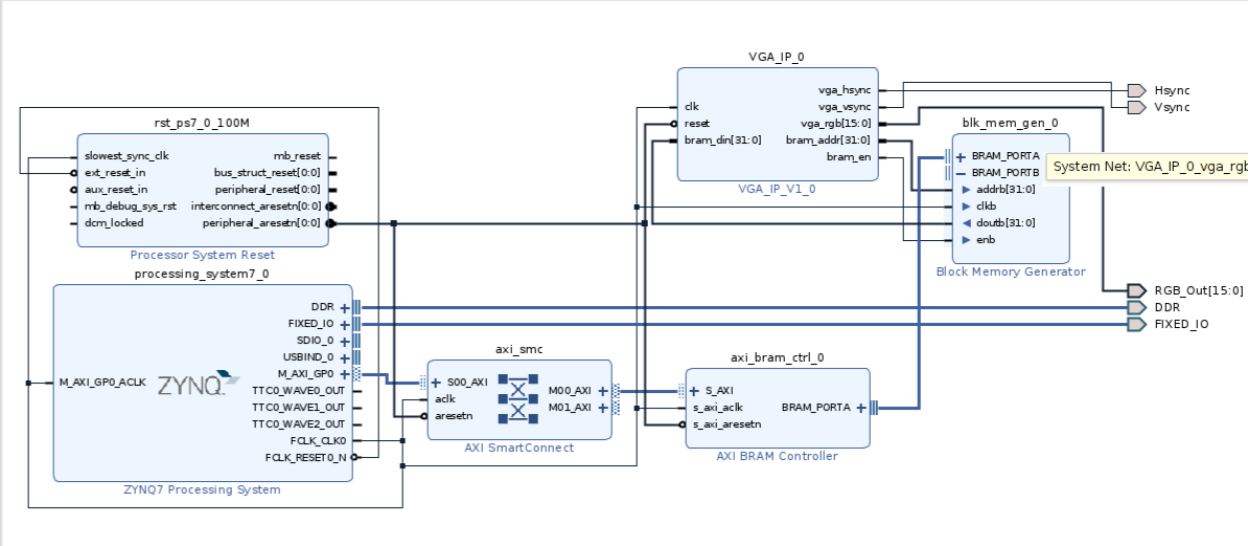
Slika 8. Prikaz slike pomoću VGA modula

BRAM memoriju se smešta slika koju je potrebno prikazati na monitoru. Dubina BRAM memorije iznosi 9600 32-bitne lokacije. Razlog za to je rezolucija slike koja iznosi 160*120 piksela. Kako je jedan piksel predstavljen sa 16 bita, to znači da se na jednoj lokaciji nalaze dva piksela. Zbog ovakvog načina organizacije memorije i zbog toga što je BRAM memorija bajt adresabilna, svaki piksel je pomeren za 2 adresne lokacije u odnosu na piksele oko sebe, odnosno prvi piksel se upisuje na adresu 0x0000, drugi piksel se upisuje na adresu 0x0002, treći na 0x0004 itd. U ovom slučaju BRAM komponenta je konfigurisana kao dvopristupna memorija (portA i portB), pri čemu se slika upisuje preko porta A, a čita preko porta B.

U priloženom materijalu se nalazi **VGA** direktoriju unutark koga se nalazi Master.tcl fajl. To je skripta koja automatski pravi Vivado projekat, ubacuje sve neophodne komponente, konfiguriše ih, povezuje ih i pokreće sintezu i implementaciju. Da bi se ta skripta pokrenula, otvorite vivado i idite na opciju:

Tools -> Run Tcl

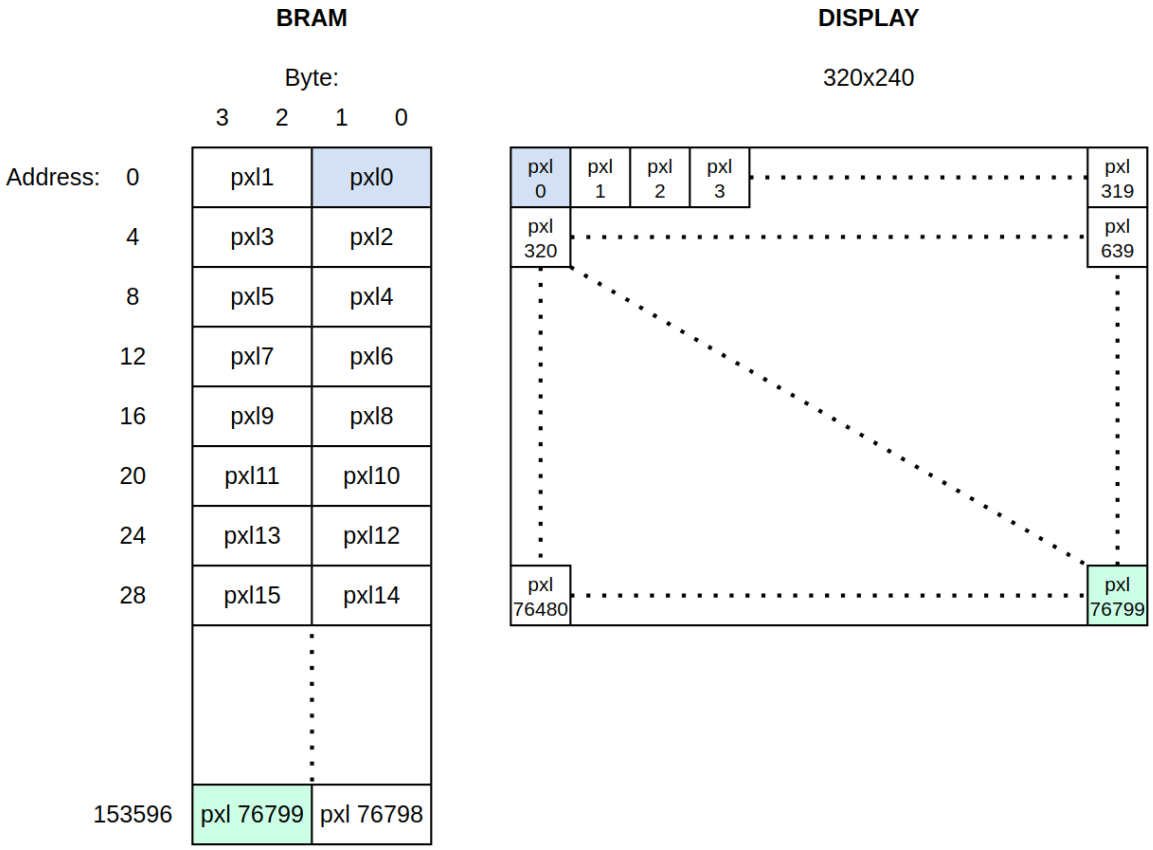
Pronađite master.tcl fajl i pritisnite ok. Nakon par minota sinteza i implementacija će se završiti i možete pokrenuti Vitis kako bi ste programirali ploču.



Slika 9. Sistem nakon pokretanja .tcl skripte

Vitis

U Vitisu napraviti projekat na isti način kao i na prethodnim vežbama. Način na koji treba popunjavati memoriju kako bi se iscrtavala slika je prikazan u nastavku:



Slika 10. Način popunjavanja BRAM memorije.

Primer koda koji popunjava BRAM memoriju je dat u nastavku:

```
#include <stdio.h>
#include "xparameters.h"
#include "platform.h"
#include "xil_printf.h"
#include "xil_cache.h"
#include "xil_io.h"
#include "xil_mmu.h"
#include "xbasic_types.h"
#include "sleep.h"

//#define USE_MMAP

int main()
{
      int i =0;
      int j =0;
    init_platform();

    print("Hello World\n\r");
    Xil_DCacheDisable();
    Xil_ICacheDisable();

    //TESTING VGA with blue-red

    unsigned char red = 0;
    unsigned char blue= 0;
    unsigned char green = 0;
    u16 colour =0;
    for(j=0; j<240; j++)
    {

      for(i = 0; i<320; i++)
      {
            red   = i;
            blue  = j;
            green = i+j;

            colour = (red << 11) | (green << 5) | blue;
            Xil_Out16(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR +
(j*320+i)*2, (u16)colour);
      }
    }
```

```
      while(1)
         usleep(1000000);

    cleanup_platform();
    return 0;
}
```

Upis u memoriju se radi pomoću *Xil_out16* funkcije kako bi se slali 16-bitni podaci. Svaki poziv ove funkcije iniciraće AXI lite transakciju od procesora ka BRAM controller komponenti. Ovo, i ako funkcionalno, je neefikasno rešenje jer su AXI lite transakcije jako spore, jer se šalje podatak po podatak. A kako je procesor spojen sa BRAM controllerom preko AXI full interfejsa moguće je slati u jednoj transakciji više podataka i ubrzati proces upisa u memoriju. To se postiže na sledeći način:

```
Xil_SetTlbAttributes(XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR, 0x15de6);

memcpy((void *)XPAR_AXI_BRAM_CTRL_0_S_AXI_BASEADDR, lokalni_niz ,broj_bajtova_za_prenos);
```

Ideja je da se pikseli smeste u lokalni niz, a zatim se koristi memcpy da se sve informacije u burst-ovima prenesu u BRAM, pri čemu se iskorištava AXI FULL protokol.