Программа **TestApp.c** предназначена для взаимодействия с FPGA (Field-Programmable Gate Array) через PCI Express (PCIe) и предоставляет информацию о версиях IP-ядер, статусе часов и PPS (Pulse Per Second), а также позволяет отслеживать текущие состояния сигналов.

**Основные функции программы**

1. Аргументы командной строки
   * Программа принимает один аргумент — базовый адрес PCIe (например, **0xA0000000**).
   * Если аргумент не указан или имеет неверный формат, программа завершается с ошибкой.
2. Отображение памяти FPGA в пространство пользователя
   * Используется системный вызов **mmap** для отображения области памяти FPGA, начиная с указанного базового адреса.
   * Размер отображаемой памяти определяется константой **pci\_addr\_range** (0x02000000).
3. Чтение версий IP-ядер
   * Программа считывает версии различных IP-ядер FPGA:
     + Версия образа FPGA (**image\_verion\_addr**).
     + Версии ядер для работы с часами, временными метками (TS), PPS и TOD.
   * Адреса регистров для чтения версий определены в заголовочном файле **nettimelogic.h**.
4. Проверка состояния часов
   * Программа проверяет, включены ли часы, и выводит текущий источник тактирования (например, TOD, PTP, RTC и т.д.).
5. Чтение состояния GPIO
   * Программа считывает состояние GPIO для мониторинга входных и выходных сигналов, связанных с PPS и часами.
6. Циклический мониторинг
   * В бесконечном цикле программа:
     + Читает текущее состояние GPIO.
     + Читает ширину импульса PPS.
     + Выводит эти данные на экран каждую секунду.

**Детальное описание ключевых частей кода**

**1. Отображение памяти FPGA**

pcie\_base = mmap(0, mapped\_size, PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, target & ~(off\_t)(page\_size - 1));

* **mmap** используется для отображения физической памяти FPGA в виртуальное адресное пространство процесса.
* **target** — это базовый адрес PCIe, переданный как аргумент.
* **mapped\_size** — размер отображаемой памяти (в данном случае **pci\_addr\_range**).

**2. Чтение версий IP-ядер**

virt\_addr = (char\*)pcie\_base + NTL\_CLK\_REGSET\_BASE + NTL\_CLK\_CLKVERSION\_REG;

read\_result = \*(volatile uint32\_t\*)virt\_addr;

printf("Clock IP Core Version = 0x%X\n", (unsigned int)read\_result);

* Программа считывает значение регистра версии из соответствующего адреса.
* Регистры определены в **nettimelogic.h** (например, **NTL\_CLK\_REGSET\_BASE**, **NTL\_CLK\_CLKVERSION\_REG**).

**3. Проверка состояния часов**

virt\_addr = (char\*)pcie\_base + NTL\_CLK\_REGSET\_BASE + NTL\_CLK\_CLKCONTROL\_REG;

read\_result = \*(volatile uint32\_t\*)virt\_addr;

if (read\_result & NTL\_CLK\_CONTROL\_ENABLE\_BIT != NTL\_CLK\_CONTROL\_ENABLE\_BIT) {

printf("Clock is not enabled \n\r");

}

* Проверяется бит включения часов (**NTL\_CLK\_CONTROL\_ENABLE\_BIT**).
* Если бит не установлен, выводится сообщение о том, что часы выключены.

**4. Определение источника тактирования**

switch ((read\_result >> 16) & 0xff) {

case NTL\_CLK\_SELECT\_NONE: printf("Selected Clk Source is: NONE \n"); break;

case NTL\_CLK\_SELECT\_TOD: printf("Selected Clk Source is: TOD \n"); break;

// Другие случаи...

}

* Программа определяет текущий источник тактирования (например, TOD, PTP, RTC) и выводит его на экран.

**5. Мониторинг состояния GPIO**

virt\_addr = (char\*)pcie\_base + gpio\_sel\_addr + gpio\_in;

read\_result = \*(volatile uint32\_t\*)virt\_addr;

printf("PPS and Clock Status is = 0x%X\n", (unsigned int)read\_result);

* Считывается состояние входных GPIO для мониторинга сигналов PPS и часов.
* Значение выводится в шестнадцатеричном формате.

**6. Чтение ширины импульса PPS**

virt\_addr = (char\*)pcie\_base + NTL\_PPSS\_REGSET\_BASE + NTL\_PPSS\_PULSEWIDTH\_REG;

read\_result = \*(volatile uint32\_t\*)virt\_addr;

printf("PPS pulse width is = %d\n", (unsigned int)read\_result);

* Программа считывает ширину импульса PPS из соответствующего регистра.

**Вывод программы**

**Пример вывода программы: ```**

**PCIe Base Address is set to 0xA0000000**

**FPGA Image Version = 0x1234**

**Clock IP Core Version = 0x5678**

**Signal TS IP Core Version = 0xABCD**

**PPS Master IP Core Version = 0xEF01**

**PPS Slave IP Core Version = 0x2345**

**TOD Slave IP Core Version = 0x6789**

**Selected Clk Source is: PTP**

PPS and Clock Selection is = 0x000000FF

PPS and Clock Status is = 0x00000001

PPS pulse width is = 100

PPS and Clock Status is = 0x00000001 PPS pulse width is = 100