Создание многопроцессорной системы в Vitis: Zynq + MicroBlaze

Автор оригинала: **Adam Taylor**

Перевел: **dvorozhbit**

Рецензент: **KeisN13**

2020 г.

Оглавление

Инфо:

Данная статья представляет собой перевод. Оригинал статьи находится здесь https://www.hackster.io/news/microzed-chronicles-microblaze-and-vitis-317aea401937

Поддержите статью **Adam Taylor** - автора **оригинального руководства** и следите за выходом его **новых руководств**

Также не забудьте поддержать автора перевода

Хотите предложить материал для публикации? Напишите нам на **fpga-systems@yandex.ru**

Аннотация

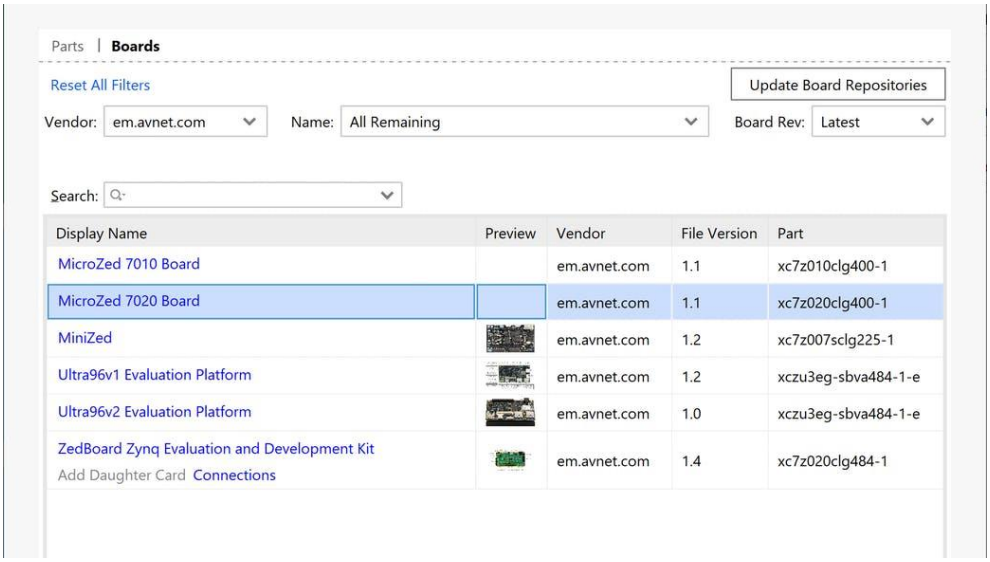
Ранее мы рассматривали то, как мы можем использовать Vitis для встроенных систем на Zynq и Zynq MPSoC.

Другим важным аспектом Vitis является возможность создать платформенное решение, которое включает в себя различные элементы обработки в PS и даже в PL.

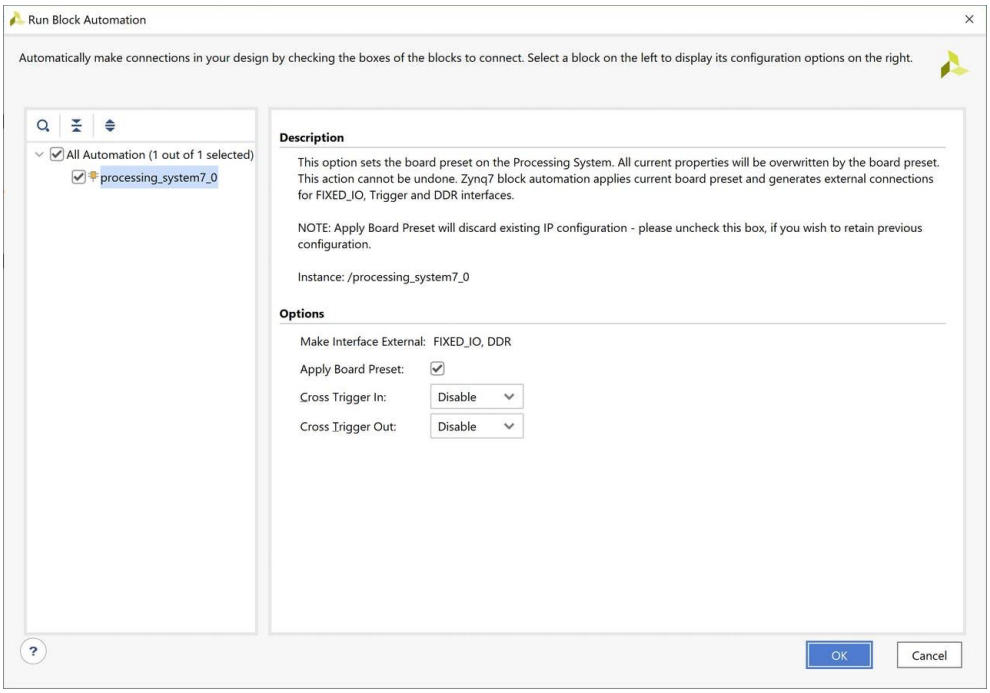
Сегодня мы рассмотрим, как мы можем использовать Vitis для разработки платформенного решения на Zynq, которое использует и Cortex A9 в PS, и MicroBlaze в PL. Нашей целевой платой будет MicroZed 7020.

Создание нового проекта

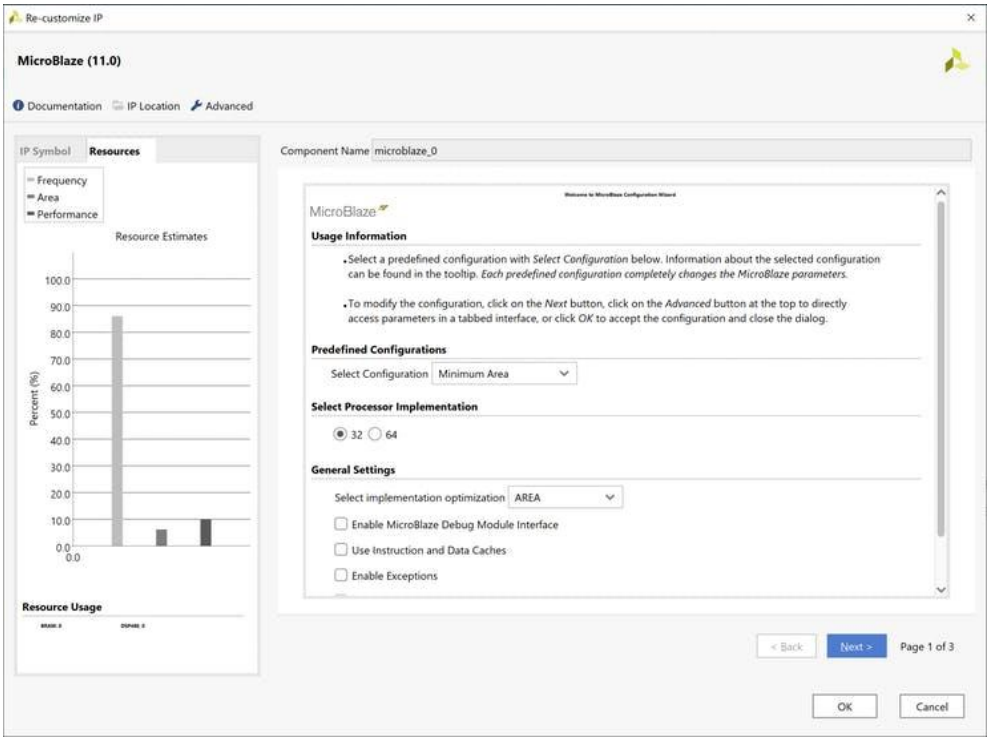
Первое, что нам нужно сделать, это создать новый проект, ориентированный на MicroZed 7020. После создания проекта следующим шагом будет создание блок-схемы и добавление в неё Zynq PS.

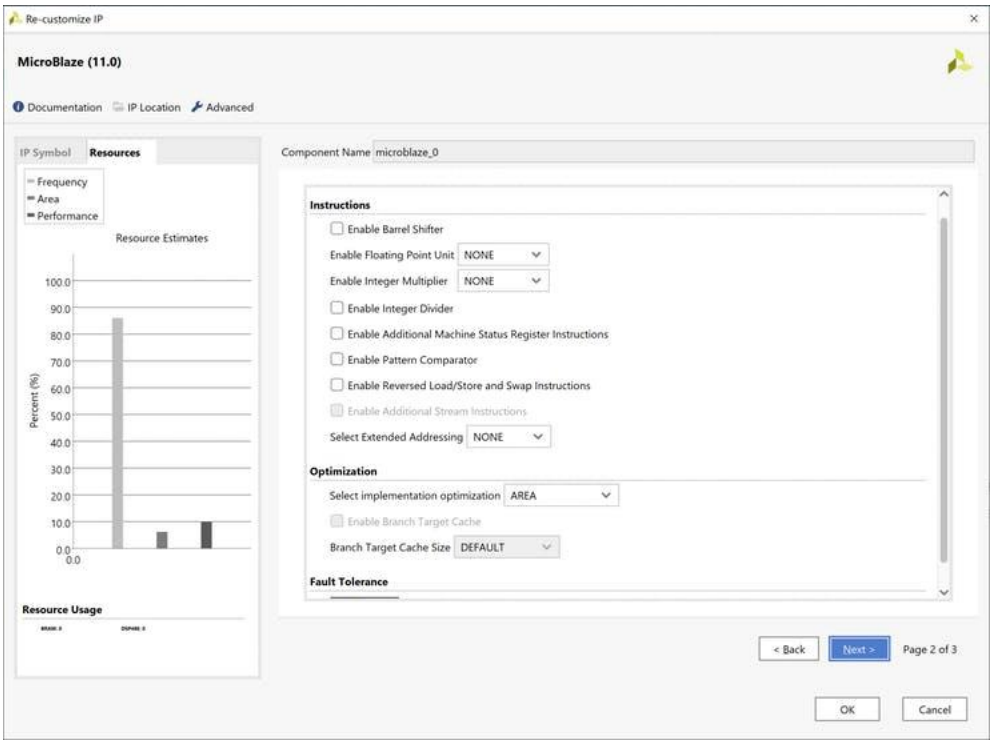


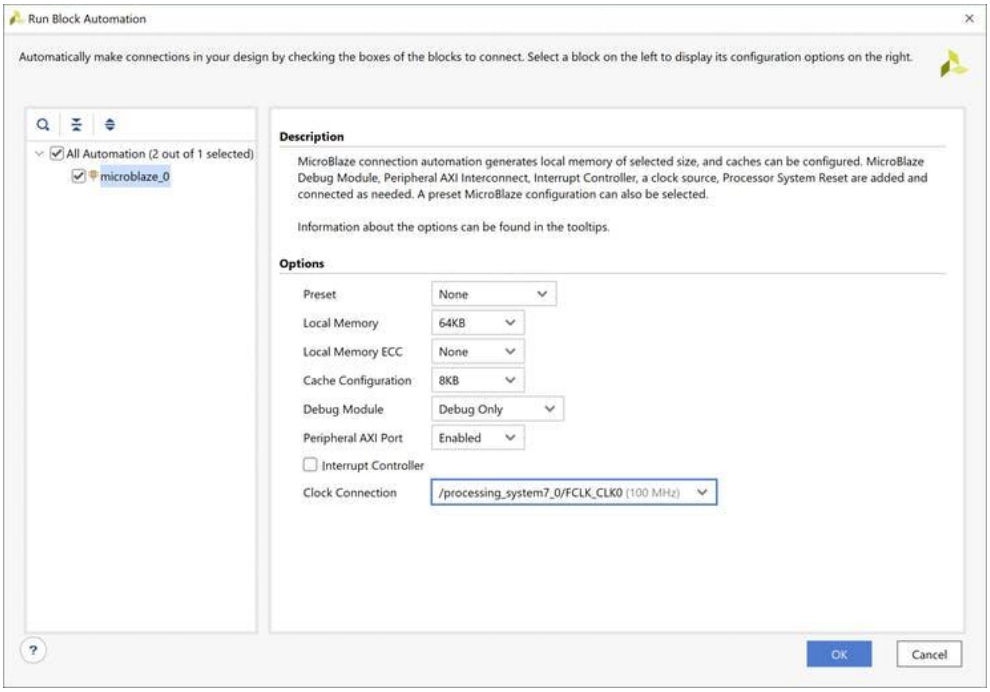
После добавления системы обработки запустите автоматизацию блоков, чтобы сконфигурировать Zynq PS в PL для реализации MicroZed 7020.



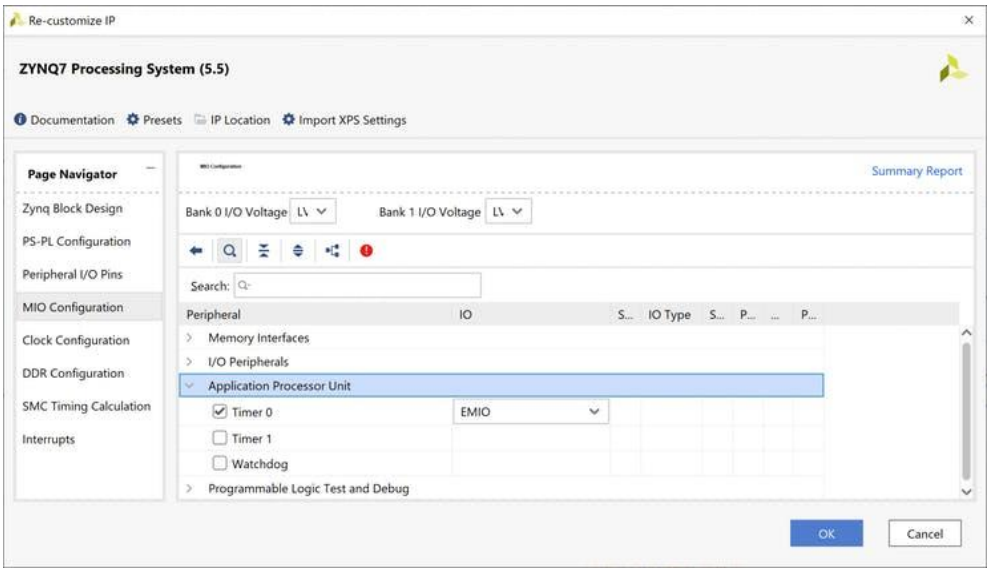
Следующим шагом является добавление блока MicroBlaze Processing и настройка его на минимальную площадь.

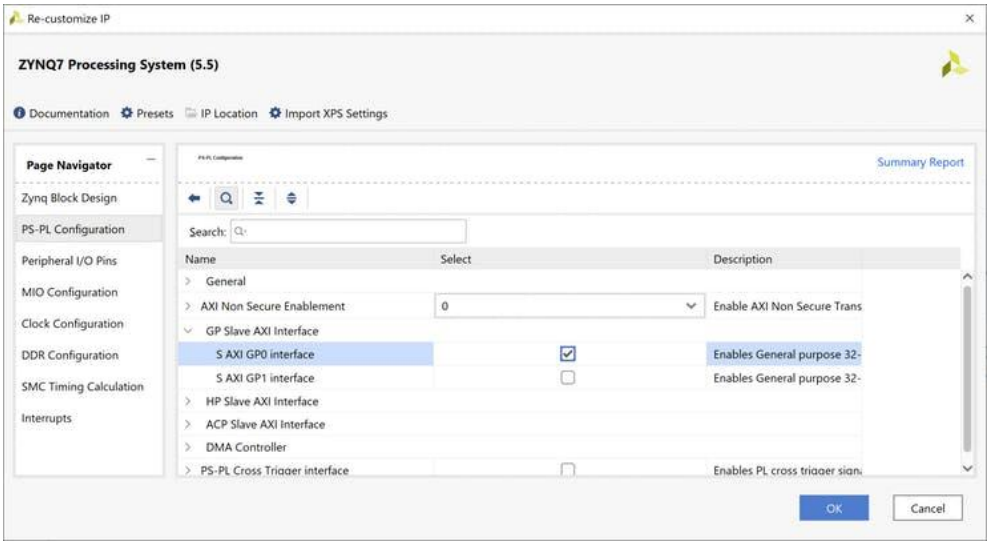






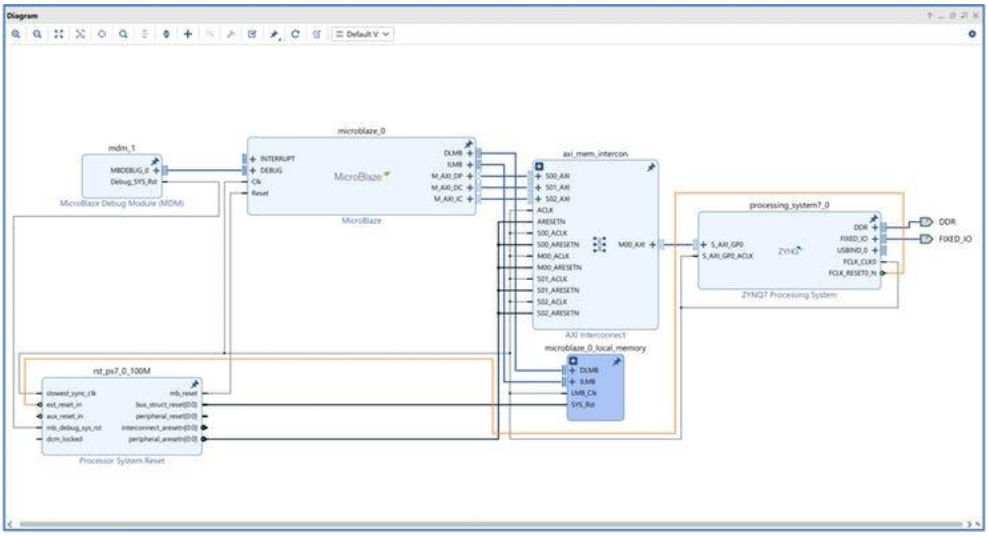
После добавления MicroBlaze следующим шагом будет настройка Zynq A9 - это включает в себя отключение таймера APU и включение порта S_AXI_GP0 в PS - PL конфигурации.





Затем мы можем запустить автоматизацию соединения, которая создаст базовую систему MicroBlaze и подключит ее к Zynq PS.

Когда мы подключаем порт MicroBlaze Peripheral AXI к порту ведомого (slave) GP на Zynq PS, MicroBlaze получает доступ к нескольким периферийным устройствам и памяти, которые доступны для Zynq PS.

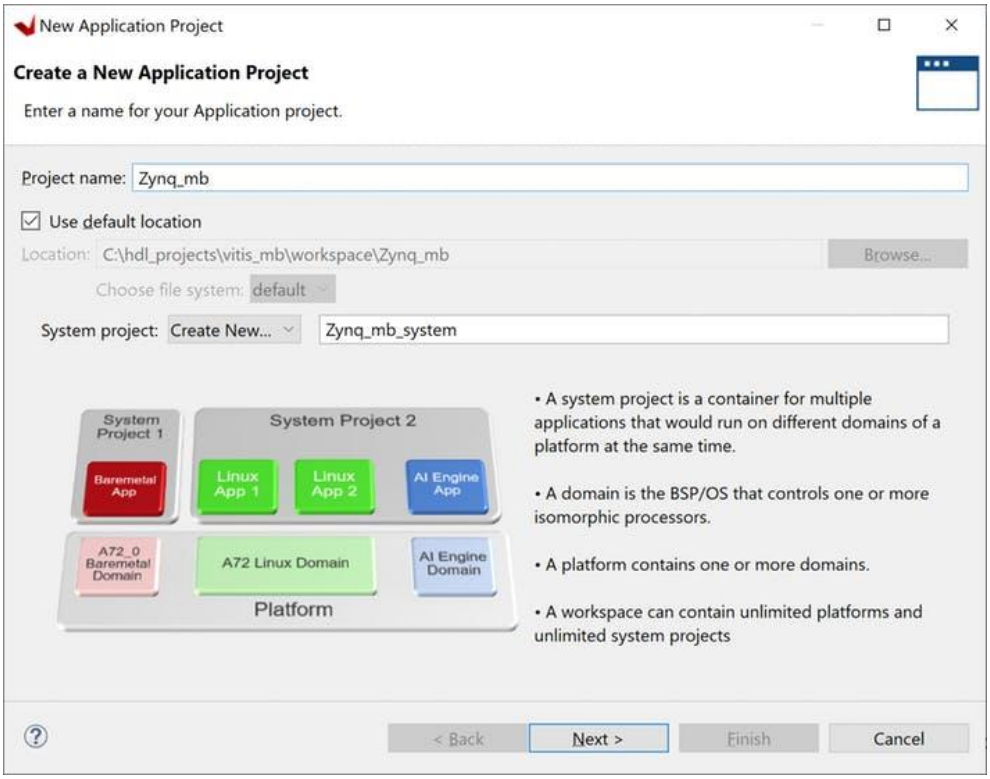


Завершающими этапами являются создание обёртки HDL, генерация битстрима и экспорт XSA.

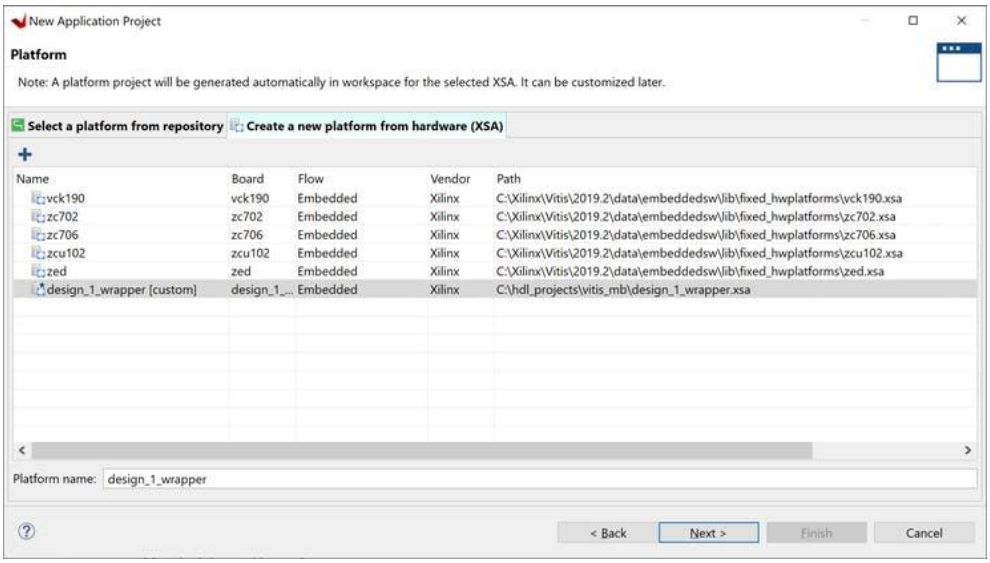
Vitis Platform

После запуска Vitis первое, что мы хотим сделать, это создать проект приложения. Это также создаст новый системный проект - именно к этому системному проекту мы добавим домены.

Первый домен будет создан в платформе автоматически при создании проекта приложения.



Добавьте платформу: нажмите кнопку «+» и укажите в диалоговом окне экспортируемый XSA (только что созданный с помощью Vivado). Когда будет предложено выбрать процессор для домена, выберите процессор Cortex A9_0.

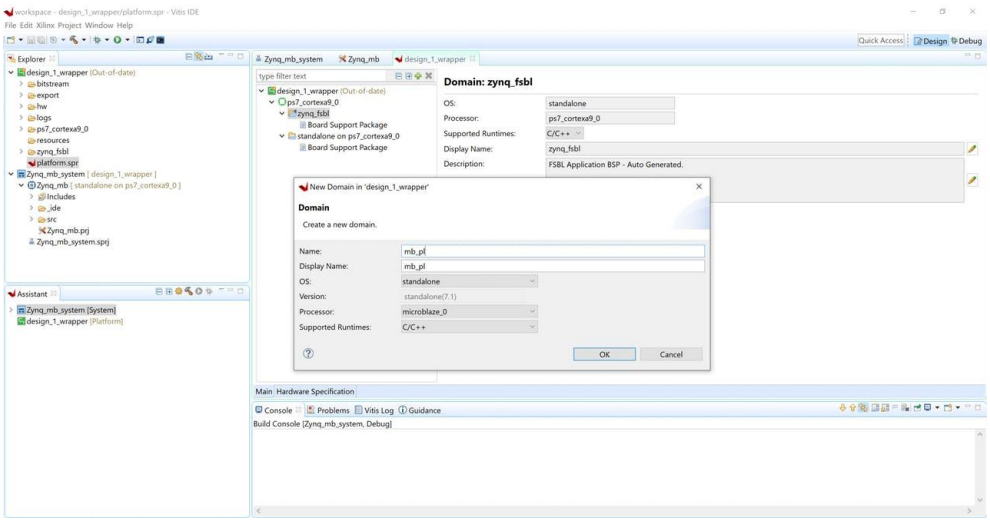


Следуйте подсказкам в диалоговом окне и создайте пример "hello world", нацеленный на ядро Cortex-A9 0.

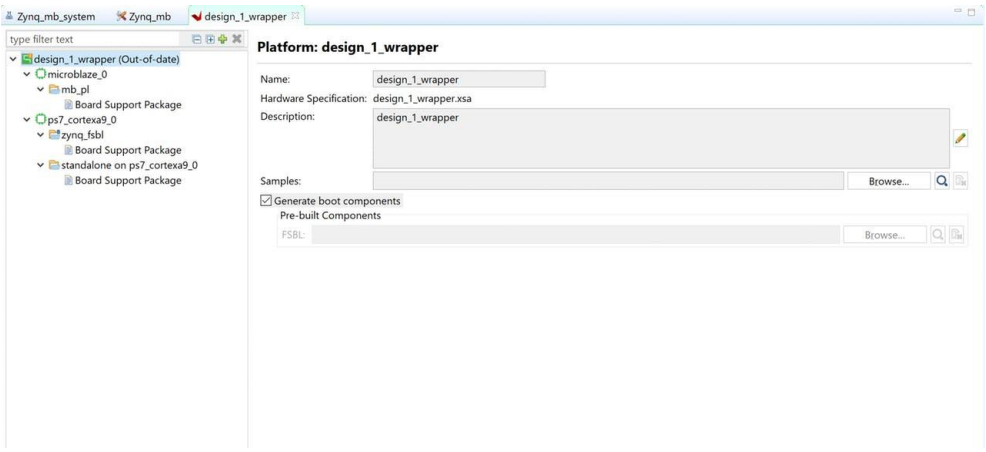
Это откроет новый проект приложения и системный проект для платформы. Под платформой вы увидите, что домен Zynq уже существует вместе с FSBL.

Чтобы иметь возможность добавить приложение MicroBlaze, нам нужно добавить новый домен в платформу. Откройте файл проекта platform.spr и нажмите кнопку +, чтобы добавить новый домен.

Введите имя для домена и выберите процессор MicroBlaze.

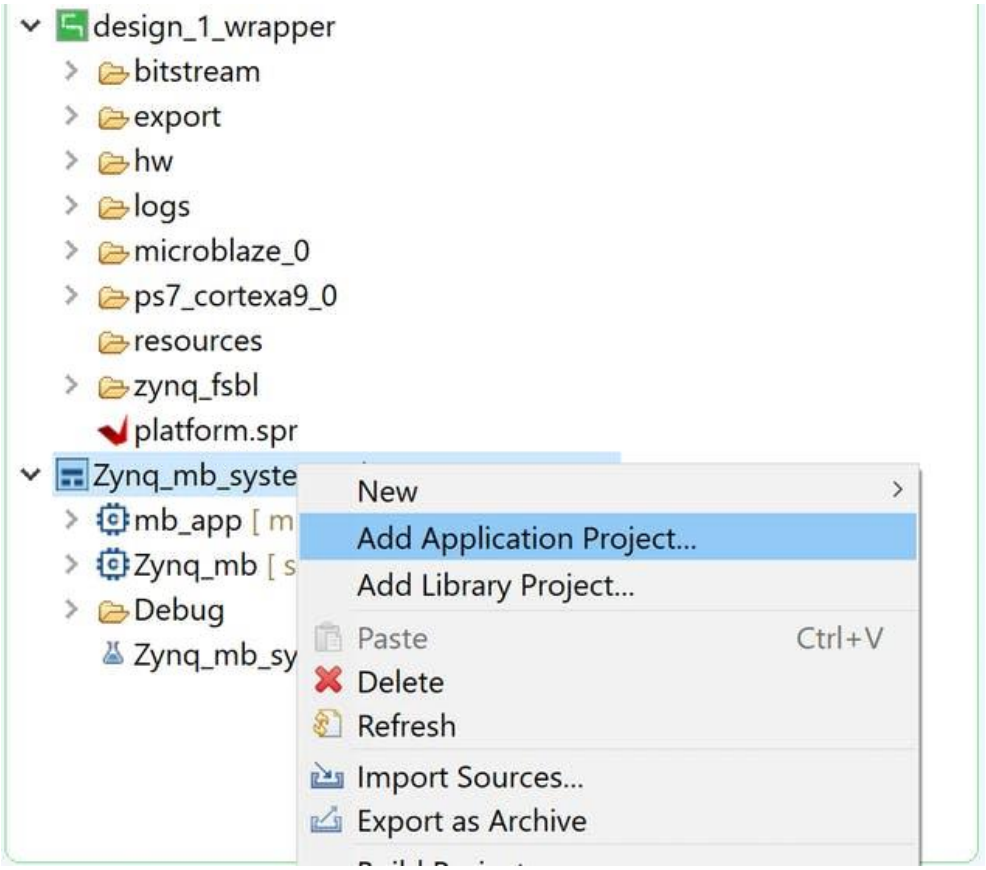


Как только это будет выполнено, под платформой вы увидите два домена - один для Zynq A9, а другой для MicroBlaze.

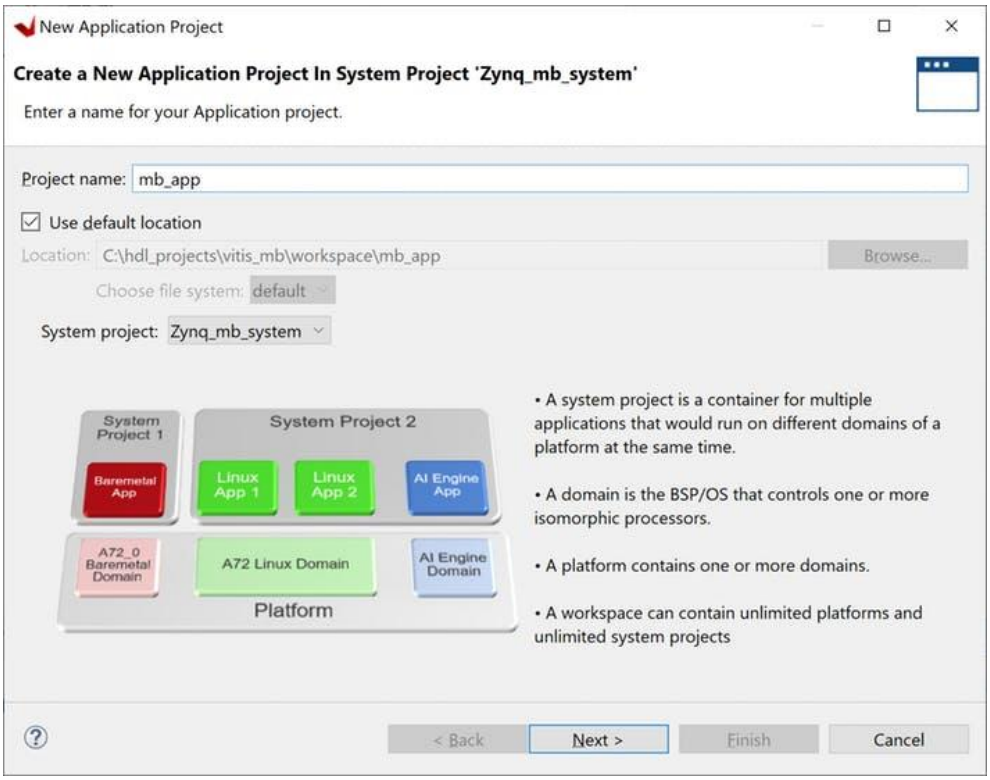


Теперь мы можем обновить платформу и перекомпилировать BSP и FSBL.

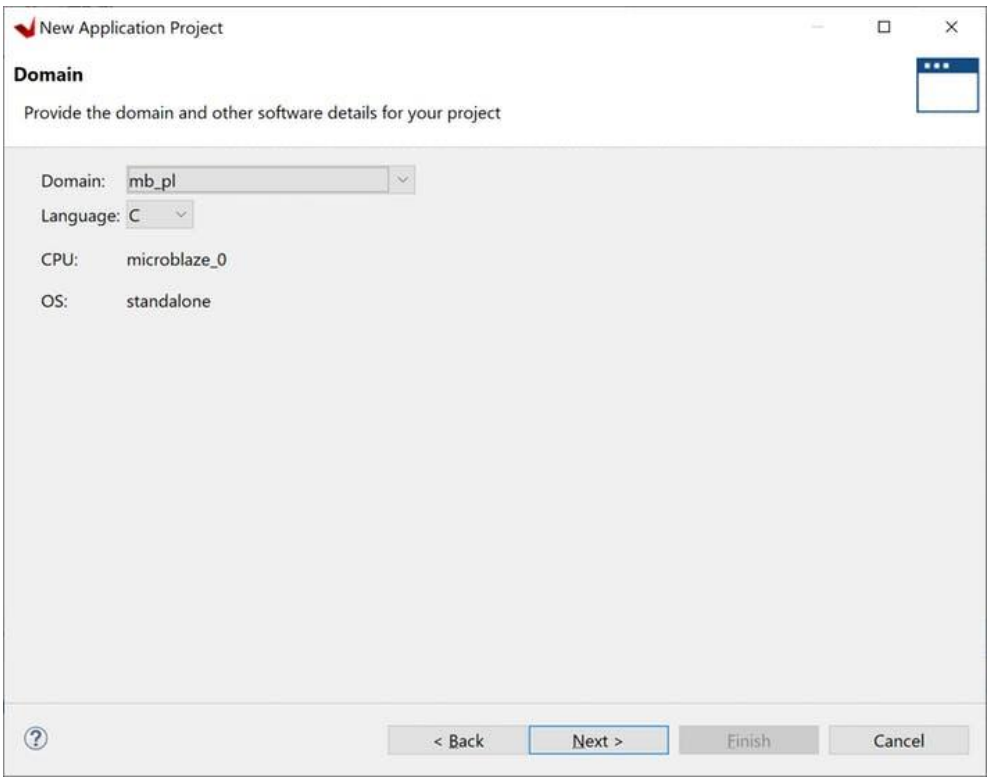
Чтобы создать второе приложение для MicroBlaze, выберите системный проект и щелкните его правой кнопкой мыши, чтобы добавить проект приложения.



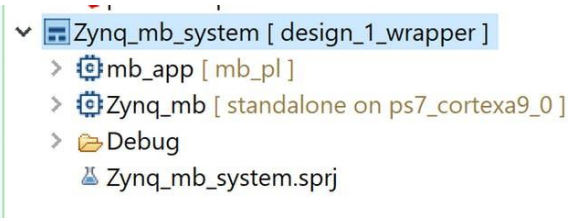
Откроется диалоговое окно создания проекта. Введите имя проекта - имя системного проекта должно быть заполнено автоматически.



Когда будет предложено выбрать домен, выберите только что созданный домен MicroBlaze и снова выберите пример «Hello World».



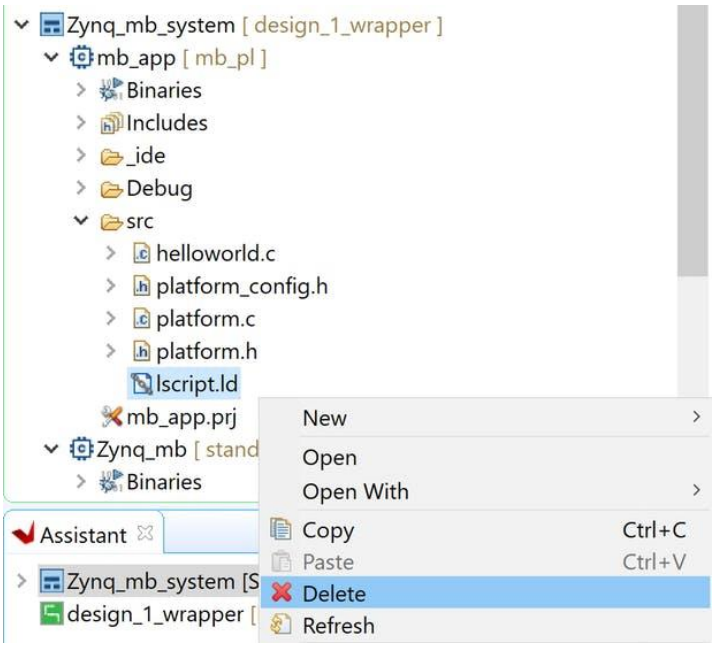
Эти шаги приведут к созданию двух проектов приложений в рамках системного проекта.



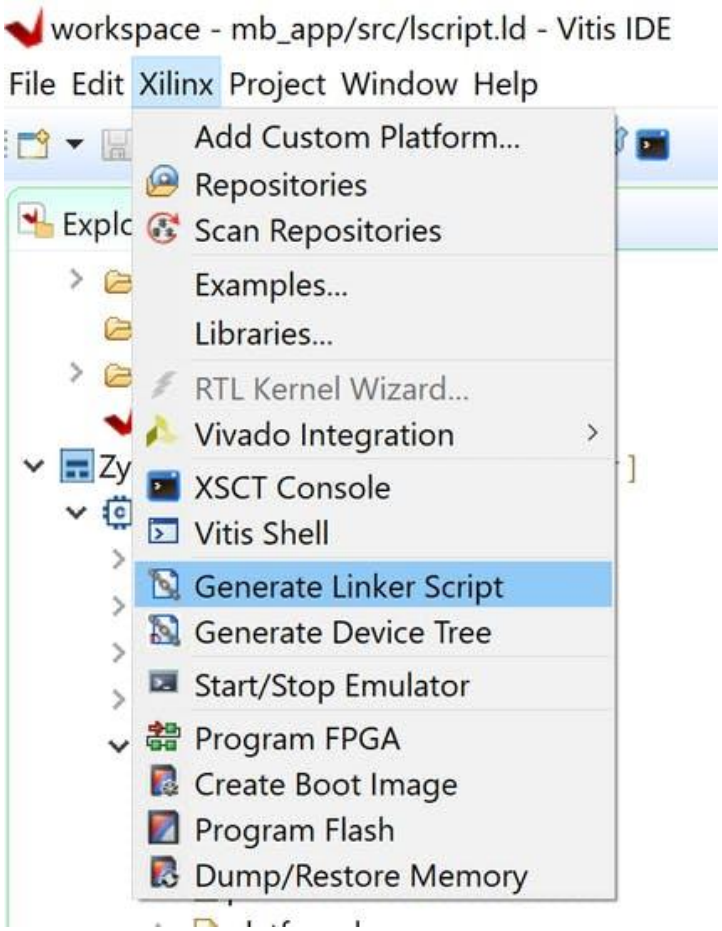
Измените приложение hello world в обоих проектах, чтобы отразить, с какого процессора отправляются сообщения.

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
int main() {
  init_platform();
  print("Hello World From Zynq\n\r");
  cleanup_platform();
  return 0;
}
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
int main() {
  init_platform();
  print("Hello World From MicroBlaze\n\r");
  cleanup_platform();
  return 0;
}
```

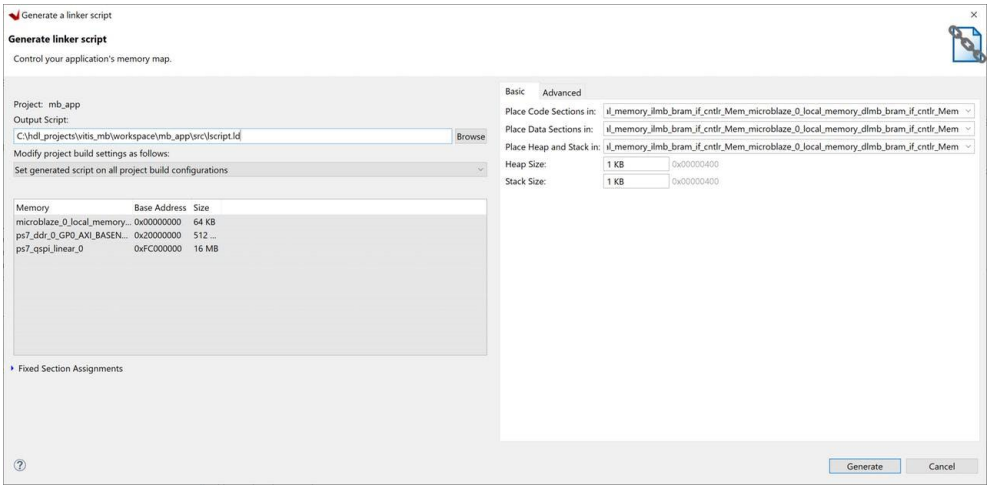
Мы хотим запустить код MicroBlaze из его внутренней BRAM. Выберите файл компоновщика (lscript.ld), доступный в исходном каталоге, и удалите его.



В меню Xilinx выберите параметр «Создать сценарий компоновщика».

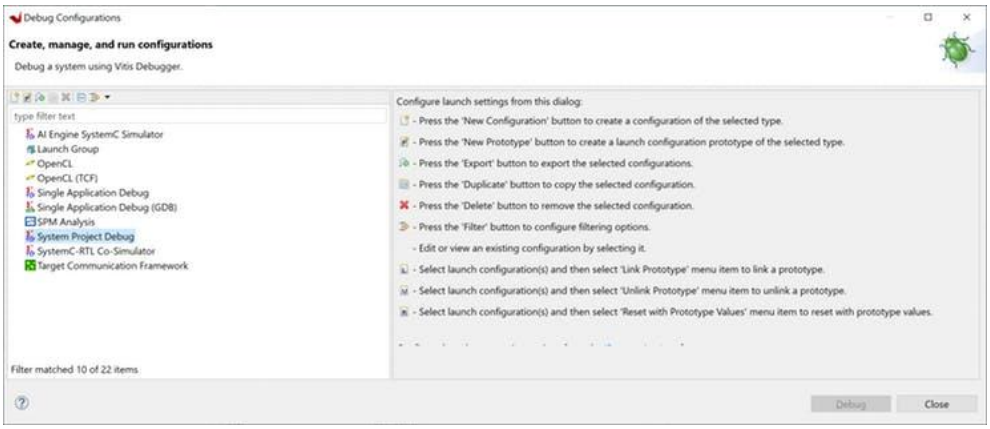


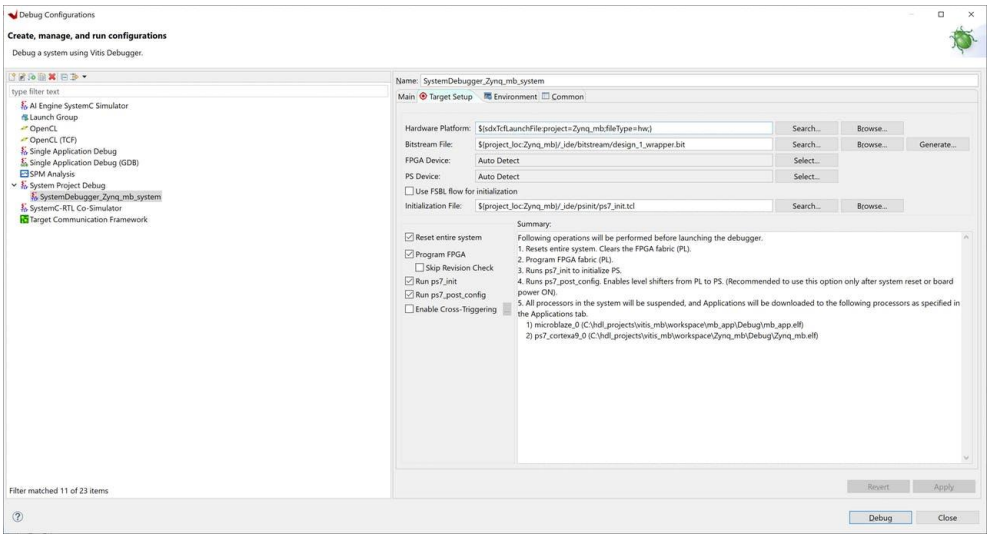
Откроется диалоговое окно, в котором будет создан новый скрипт компоновщика. Убедитесь, что Code, Data, Heap и Stack находятся в локальной памяти BRAM.



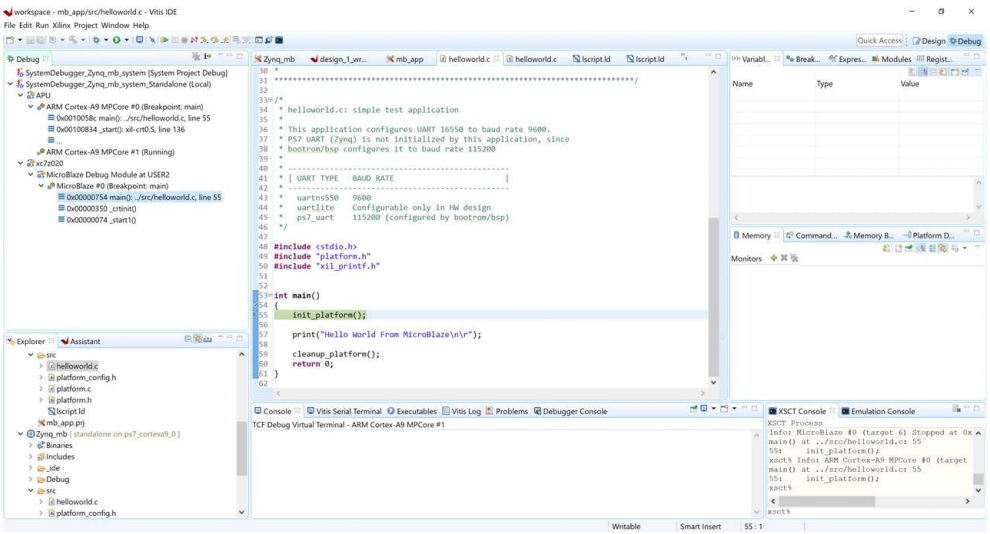
Теперь мы можем собрать системный проект и загрузить его для тестирования на MicroZed.

Для этого нам нужно создать новую конфигурацию отладки. Выберите System Project Debug и создайте новую конфигурацию отладки.

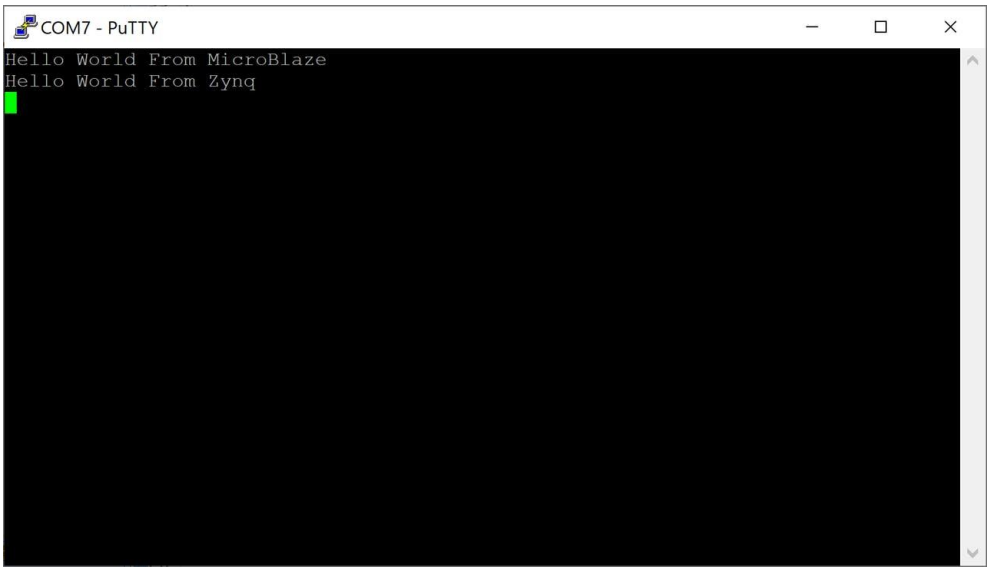




После того как MicroZed настроен, мы можем выполнить код по шагам или запустить процессоры в соответствии с требованиями приложения.



Конечно, в этом примере оба процессора используют один и тот же UART, поэтому я запускал один процессор за другим.



Заключение

Это очень простой пример того, как многопроцессорная система может быть определена и разработана с использованием Vitis. Это также показывает способность разрабатывать системные решения на уровне платформы и выполнять соответствующую отладку.

По мере развития нашего приложения Vitis, я уверен, что мы будем использовать этот подход в более сложных приложениях и проектах в будущем!

Понравилась статья? Не забудьте поддержать автора перевода

Хотите предложить материал для публикации? Напишите нам на **fpga-systems@yandex.ru**