uprocessor App.

Class Introduction

목차 1. uprossecor 2. FPGA 3. SoC 4. DDR 5. SRAM 6. Focus Point 7. Goal

1. uprossecor

- Model : Xilinx ZYNQ7020
- C 등 코드를 올려서 작동. 다만 해당 보드의 명령어들과 명령어들의 구조를 잘 숙지해둘 것.(중간 출제 가능성 多)
- 주로 소프트웨어 파트를 맡음
- 개발환경이 FPGA에 비해 빠름
- C와 assembly를 서로 변환할 줄 알아야 됨.
- goal : ucontroller/ARM CortexA9(시스템 제작)
- 기본 프로그램은 제공해주되 해당 프로그램을 최적화하는 방법이 중요하다. Cache Memory에 어떤 data를 넣을지 고민해둘 것.

2. FPGA

- 베릴로그 등 코드를 받아 작동하는 programable logic array
- 주로 하드웨어 파트를 담당.
- 하드웨어 파트를 직접 설계할 수 있으며 uprossecor에 비해 개발 기간이 길다.

3. SoC

- uprossecor + FPGA = SoC

4. DDR

- D RAM
- 전원을 끄면 사라지는 메모리
- 용량 大, 가격 低 but 시간, 전력 소모 大
- 칩 밖에 따로 존재함.
- 점퍼(전선)을 통해 통신하기 때문에 데이터를 가져올때 보다 긴 사이클이 필요하다.

5. SRAM

- on-chip-memory
- 칩 내부에 존재하며, D RAM과는 달리 매우 짧은 사이클 내에 정보 통신 가능.
- 해당 보드는 Cache Memory가 SRAM으로 구현되어 있다 보면 된다.

6. Focus Point

- instruction set
- Coprocessor : 벡터 연산을 보다 빠른 사이클에 해결
- Memory/Cache
- interrupt/DMA(Direct Memory Access)

- I/O peripherals
- Assembly Programming
- Code optimization
- Bus(AXI)

etc)

1. 일반적인 벡터 연산

```
a0 + b0 = c0
a1 + b1 = c1
a2 + b2 = c2
```

세로를 하나의 벡터로 칠때, 일반적으로 세 사이클에 걸쳐 계산을 진행한다.

2. DMA

```
A[0xAF]
B[0x01]
```

- A의 데이터를 B로 옮길때, 비트를 하나씩 load & store하며 데이터를 연산.
- CPU를 이런 연산에 사용하는 것은 비효율적이므로, 해당 연산을 DMA에 맡기고 CPU는 다른 작업을 수행.

7. Goal

- 기본적으로 같은 프로그램을 제공.
- 이를 기반으로 code optimization을 통해 보다 빠른 식별 프로그램 제작.
- 어떤 방식으로 통신하고, 어떤 메모리를 언제 SRAM에 올릴지 결정하는 것이 중요하다.