YiffCore

A RISC-like SoC architecture for gay critters

General architecture

Provides a general architectural overview

Introduction

YiffCore is a 64-bit System-on-Chip architecture targeting low-power embedded systems within the scope of Yiffware infrastructure. YiffCore utilizes 32-bit fixed-width instructions and a handful of registers used to perform data operations. The bus in-use is to have a word-size of 64-bits and be linearly addressed.

Cores are simply needy kitties who sniff around the address space for opcodes to crunch, therefore, individual processing units are referred to as "kits".

Mandatory access control (MAC)

YiffCore is a tri-MAC (mandatory access control) architecture, in other words, the privilege levels are divided into three rings: 0, 1, and 2. Where 0 is the most privileged and is one-level below the operating system kernel, 1 is reserved for the operating system kernel itself and 2 is the lowest privileged level used for user-level applications.

For scalability, YiffCore employs a MAC-ROM which encodes per-instruction access levels. In other words, every instruction within the MAC ROM is paired with a value indicating what can execute it (if it is higher than what is specified, e.g., an instruction marked with 2 can be executed by rings 2, 1 and 0).

Processor registers

YiffCore utilizes 32 general-purpose registers for various data movement operations, to differentiate between temporaries and other registers, the temporaries are known as CUM stores (the SoC has many holes :3):

ZR	Fixed to zero, writes are ignored	ID=0x00
CUM0	Temporary data	ID=0x01
CUM1	Temporary data	ID=0x02
CUM2	Temporary data	ID=0x03
CUM3	Temporary data	ID=0x04
CUM4	Temporary data	ID=0x05
CUM5	Temporary data	ID=0x06
SP	Stack pointer	ID=0x07
FP	Frame pointer	ID=0x08
TP	Per-thread data pointer	ID=0x09
GP	Global data pointer	ID=0x0A
RA	Return address	ID=0x0B
A0	Function argument 1	ID=0x0C
A1	Function argument 2	ID=0x0D
A2	Function argument 3	ID=0x0E
A3	Function argument 4	ID=0x0F
A4	Function argument 5	ID=0x10
A5	Function argument 6	ID=0x11
S0	Saved register 0	ID=0x12
S1	Saved register 1	ID=0x13
S2	Saved register 2	ID=0x14
S3	Saved register 3	ID=0x15
S4	Saved register 4	ID=0x16
S5	Saved register 5	ID=0x17
S6	Saved register 6	ID=0x18
S7	Saved register 7	ID=0x19
S8	Saved register 8	ID=0x1A
S9	Saved register 9	ID=0x1B
S10	Saved register 10	ID=0x1C
S11	Saved register 11	ID=0x1D
S12	Saved register 12	ID=0x1E
S13	Saved register 13	ID=0x1F

Processor reset state

Upon power-up and/or assertion of the RESET# line, the processor is to enter a reset state in which all registers besides IP, SP, FP, TP, GP and ZP (which are initialized to zero) are to be initialized to a value of 0x1A1A1A1A1A1A1A1A. The instruction pointer (IP) being at zero maps to the platform firmware ROM (PFR).

There are two kinds of resets, a **cold** reset and a **warm** reset. A **cold** reset is when power is first applied to the System-on-Chip (SoC) package and results in the processor entering a low-power state in which IP is inhibited until an implementation specific signal tells it to run. A **warm** reset is when power is already applied to the SoC package and the processor reset line is pulled low, in which case doesn't require IP to be inhibited, thus the processor simply runs.

Instruction set architecture

This chapter describes the instruction set architecture of YiffCore

Instruction listing

Below are the instructions available to the programmer:

MNEMONIC	OPCODE	BRIEF
NOP	0x00	No-operation
MOV R0, R1	0x01	Move register to register
RESERVED	0x02	Reserved, do not use
ADD R0, R1, R2	0x03	Add R1 and R2 and copy result to R0
SUB R0, R1, R2	0x04	Subtract R1 and R2 and copy the result to R0
RESERVED	0x05	Reserved, do not use
RESERVED	0x06	Reserved, do not use
RESERVED	0x07	Reserved, do not use