VciAhciSdc

1) Functional Description

This component is a single channel SD card controller. From the software point of view, it respect the AHCI standard. On the physical side, it respect the 4bits SD card interface. It can be connected to the SdCard component, that is modeled as a single file stored on the host system, and containing a complete disk image. The block size is fixed to 512 bytes. The VCI burst size is an hardware parameters, defined as a constructor parameter.

According to the AHCI specification, the controller uses a private *Command List* that is handled as a software FIFO, and can register up to 32 *read* or *write* commands. It uses it's DMA capability, to access both the *Command List* and to transfer the data to or from memory.

On the VCI side, it supports both 32 bits and 64 bits data words, and up to 64 bits address width.

An IRQ can be (optionally) asserted as soon as one command in the Command List is completed.

This hardware component checks for segmentation violation, and can be used as a default target.

2) Command List

For each channel, the driver must use a software FIFO to register a command: The Command Descriptor array (32 entries) define the Command List. Each Command Descriptor occupies 16 bytes, and must be aligned on a 16 bytes boundary. It contains mainly the physical address of the associated Command Table. A command Descriptor is defined by the following C structure:

```
typedef struct hba_cmd_desc_s  // size = 16 bytes
{
    unsigned char       flag[2];        // WRITE when bit 6 of flag[0] is set
    unsigned char       prdtl[2];       // Number of buffers
    unsigned int        prdbc;          // Number of bytes actually transfered
    unsigned int        ctba;           // Command Table base address 32 LSB bits
    unsigned int        ctbau;          // Command Table base address 32 MSB bits
} hba_cmd_desc_t;
```

3) Command Table

There is one Command Table for each Command descriptor. For a given command, there is one single LBA (Logic Bloc Address) on the block device, coded on 48 bits, but the source (or destination) memory buffer can be split in a variable number of contiguous buffers. Therefore, the Command Table contains two parts: a fixed size Header, defining the LBA, followed by a variable size array of fixed size buffer descriptors. The Command Table describing one command is defined by the two following C structures:

```
typedef struct hba_cmd_header_s // size = 16 bytes
{
    unsigned int       res0;        // reserved
    unsigned char      lba0;        // LBA 7:0
    unsigned char      lba1;        // LBA 15:8
    unsigned char      lba2;        // LBA 23:16
    unsigned char      res1;        // reserved
    unsigned char      lba3;        // LBA 31:24
    unsigned char      lba4;        // LBA 39:32
    unsigned char      lba5;        // LBA 47:40
```

```
    unsigned char      res1;        // reserved
    unsigned char      res2;        // reserved
    unsigned int       res3;        // reserved
} hba_cmd_header_t;

typedef struct hba_cmd_buffer_s // size = 16 bytes
{
    unsigned int       dba;         // Buffer base address 32 LSB bits
    unsigned int       dbau;        // Buffer base address 32 MSB bits
    unsigned int       res0;        // reserved
    unsigned int       dbc;         // Buffer byte count

} hba_cmd_buffer_t;
```

4) Addressable registers

Regarding the **SD Card configuration**, this component contains three 32 bits Write-Only registers, and one 32 bits Read-Only register:

- **SDC_PERIOD**

This Write-Only register define the ratio between the system clock frequency and the SDC clock frequency. This ratio must be an integer value not smaller than 2.

- **SDC_CMD_ARG**

This 32 bits Write-Oly register must contain the argument used by the command launched by writing in the SDC_CMD_ID register.

- **SDC_CMD_ID**

Writing in this Write-Only register launch a command to the SD card. The command index is defined by the value written in the register. The supported commands are defined below:

Index	argument	action
0	none	SD Card soft reset
2	none	Request SD Card to return CID (Card Identifier)
3	none	Request SD Card to return RCA (Relative Card Address)
7	Card RCA	Toggle SD Card between "stand-by" and "transfer" states
12	Card RCA	Request SD Card to stop a multi-block data transfer
41	specific	Request SD Card to return OCR (Operating Condition)

- **SDC_STATUS**

This Read-Only register contains the status of the SD Card after execution of a command launched by a write in the SDC_CMD_ID register.

Regarding the **AHCI configuration**, this component contains six 32 bits Read/Write registers:

- **HBA_PXCLB**

32 LSB bits of the Command List physical base address. This address must be aligned on a 16 bytes boundary.

- **HBA_PXCLBU**

32 MSB bits of the Command List array physical address.

- **HBA_PXIS**

Channel status, used for error reporting.

```
31 30 29 28.....24    23.........8       7.....1  0
-- R  -- CMD_ID  BUFFER_ID  ------- D
```

Bit[0] : set by hardware when at least one command has been completed. Bit[30] : set by hardware when an error has been detected in a command. Bit[28:24] : index of the faulty command in command list (set by the hardware). Bit[23:8] : index of the faulty buffer in the faulty command (set by the hardware).

When an error is detected for a command, the R bit is set, the channel FSM stops immediately, without handling the remaining commands in the command list, and keep blocked, waiting for a software reset on this PXIS register. Any write access to this register reset all bits to 0, whatever the VCI WDATA value.

- **HBA_PXIE**

This register enables and disables the IRQ reporting the completion (success or error) of the commands for a given channel. Only 2 bits are used:

```
31 30 29 ..................................... 1   0
-- R  ---------------------------------- D
```

Bit 0 : when set, an IRQ is generated when bit0 of AHCI_PXIS is set. Bit 30 : when set, an IRQ is generated when bit30 of AHCI_PXIS is set.

- **HBA_PXCMD**

Boolean : Writing a non zero value activates the polling of the Command List. Writing a zero value makes a soft reset on PXCI, PXIS, PXIE, and PXCMD registers.

- **HBA_PXCI**

Bit-vector, one bit per command in the Command List. These bits are handled as 32 set/reset flip-flops: set by software when a command ha been posted in Command List / reset by hardware when the command is completed. A write command on this register makes a OR between the VCI WDATA field and the current value of the register.

For extensibility issues, the software drivers must use the mnemonics defined here? to access this component .

Even if there is only six registers per channel, each channel sub-segment occupies 4K bytes, and the HBA segment must be aligned on a 32 Kbytes boundary.

5) Component definition & usage

source:trunk/soclib/soclib/module/connectivity_component/vci_ahci_sdc/caba/metadata/vci_ahci_sdc.sd?

6) CABA Implementation

CABA sources

- interface : source:trunk/soclib/soclib/module/connectivity_component/vci_ahci_sdc/caba/source/include/vci_ahci_sdc.h?
- implementation : source:trunk/soclib/soclib/module/connectivity_component/vci_ahci_sdc/caba/source/src/vci_ahci_sdc.cpp?

CABA Constructor parameters

```
VciAhciSdc(
     sc_module_name name,   //  Component Name
     const soclib::common::MappingTable &mt, // MappingTable
     const soclib::common::IntTab &srcid,    // Initiator index
     const soclib::common::IntTab &tgtid,    // Target index
     const uint32_t burst_size ); // burst size (bytes)
```

CABA Ports

- **p_resetn** : Global system reset
- **p_clk** : Global system clock
- **p_vci_target** : The VCI target port
- **p_vci_initiator** : The VCI initiator port
- **p_channel_irq** : Array of interrupt ports (one per channel)

7) TLM-DT Implementation

Not available yet.