T-EMU: AMBA Bus and Plug-and-Play

Prepared

Mattias Holm
Technical Manager

Checked

Dan Søren Nielsen
QA Manager

Approved

Michela Alberti
General Manager

Record of Changes

Author	Description	Rev	Date
Mattias Holm	Auto gen tables	1.1	2016-05-12
Mattias Holm	Initial Version	1.0	2015-03-01

Table of Contents

1. Introduction

T-EMU provides support for AMBA plug-and-play as used in the Gaisler GRLIB. The AMBA bus support and interfaces are defined in "temu-c/Bus/Amba.h". In addition to the interfaces implemented by device models, the AhbCtrl and ApbCtrl classes are provided.

2. Interfaces

The interesting interfaces are defined in the `temu-c/Bus/Amba.h` header. This header also defines constants for various vendor identifiers and inline helper functions to populate the PnP info structs.

```
typedef struct {
  uint32_t IdentReg;
  uint32_t UserDef[3];
  uint32_t Bar[4];
} temu_AhbPnpInfo;

typedef struct temu_AhbIface {
  temu_AhbPnpInfo* (*getAhbPnp)(void *Obj);
} temu_AhbIface;

typedef struct {
  uint32_t ConfigWord;
  uint32_t Bar;
} temu_ApbPnpInfo;

typedef struct temu_ApbIface {
  temu_ApbPnpInfo* (*getApbPnp)(void *Obj);
} temu_ApbIface;
```

3. Classes

There are two important classes provided, the AhbCtrl and ApbCtrl classes. These are available in libTEMUAhbCtrl.so and libTEMUApbCtrl.so. When configuring a non-standard LEON3 / LEON4 based processor, the AHB and APB controllers must be instantiated and the controllers should be connected to any devices implementing the plug and play interfaces. For the AhbCtrl class, the property to connect is the masters and slaves arrays, and for the ApbCtrl class, only the slaves property exist.

3.1. AhbCtrl

3.1.1. Properties

Name	Type	Description
masters	[64 x iref]	
object.timeSource	object	Time source object (a cpu or machine object)
slaves	[64 x iref]	

3.1.2. Interfaces

Name	Type	Description
DeviceIface	DeviceIface	
MemAccessIface	MemAccessIface	

3.1.3. Ports

Prop	Iface	Description
-	-	-

3.2. ApbCtrl

3.2.1. Properties

Name	Type	Description
object.timeSource	object	Time source object (a cpu or machine object)
pnp.bar	[4 x uint32_t]	
pnp.identReg	uint32_t	

Name	Type	Description
pnp.userDef	[3 x uint32_t]	
slaves	[512 x iref]	

3.2.2. Interfaces

Name	Type	Description
AhbIface	AhbIface	
DeviceIface	DeviceIface	
MemAccessIface	MemAccessIface	

3.2.3. Ports

Prop	Iface	Description
-	-	-

4. Examples

The first example shows how to create and connect the AHB and APB bus controllers to different objects implementing the plug-and-play interfaces.

```
import AhbCtrl
import ApbCtrl

# Create two bus objects
object-create class=AhbCtrl name=ahbctrl0
object-create class=ApbCtrl name=apbctrl0

# Map to the normal addresses
memory-map memspace=mem0 addr=0x800ff000 length=0x1000 object=apbctrl0
memory-map memspace=mem0 addr=0xfffff000 length=0x1000 object=ahbctrl0

# Connect various APB devices to the APB controller
connect a=apbctrl0.slaves b=ftmctrl0:ApbIface
connect a=apbctrl0.slaves b=apbuart0:ApbIface
connect a=apbctrl0.slaves b=irqMp0:ApbIface
connect a=apbctrl0.slaves b=gpTimer0:ApbIface
connect a=apbctrl0.slaves b=ahbstat0:ApbIface

# Connect various AHB devices to the AHB controller
connect a=ahbctrl0.masters b=cpu0:AhbIface
connect a=ahbctrl0.slaves b=ftmctrl0:AhbIface
connect a=ahbctrl0.slaves b=apbctrl0:AhbIface
```

The next example shows how to implement a simple APB device.

```
#include "temu-c/Bus/Amba.h"

// This is the model type, we need to add the Pnp info.
typedef struct MyDevice {
  temu_ApbPnpInfo Pnp;
  // ...
} MyDevice;


// Implement the APB PNP interface
temu_ApbPnpInfo*
getApbPnp(void *Obj)
{
   MyDevice *Dev = (MyDevice*)Obj;

   return &Dev->Pnp;
}

temu_ApbIface ApbIface = {
  .getApbPnp = getApbPnp
};


// Define functions to allocate and destroy the object
void*
create(int Argc, const temu_CreateArg *Argv)
{
  MyDevice *Dev = malloc(sizeof(MyDevice));
  memset(Dev, 0, sizeof(MyDevice));

  // PNP init
  temu_apbSetVendorId(&MyDevice->Pnp, 0x99);
  temu_apbSetDeviceId(&MyDevice->Pnp, 0x001);
  temu_apbSetVersion(&MyDevice->Pnp, 1);

  temu_apbSetAddr(&MyDevice->Pnp, 0);
  temu_apbSetCP(&MyDevice->Pnp, 0);
  temu_apbSetMask(&MyDevice->Pnp, 0xfff);
  temu_apbSetType(&MyDevice->Pnp, 1); // APB I/O space

  return MyDevice;
}

void
dispose(void *Obj)
{
  MyDevice *Dev = (MyDevice*)Obj;
```

```
  free(Irq);
}

// Define the device interface
void
reset(void *Obj, int ResetKind)
{
}

void
mapDevice(void *Obj, uint64_t Addr, uint64_t Len)
{
  MyDevice *Dev = (MyDevice*)Obj;
  temu_apbSetAddr(&Dev->Pnp, Addr);
}


temu_DeviceIface DeviceIface = {
  reset, // Called on resets
  mapDevice, // Called when a device is mapped to a memory location.
};



TEMU_PLUGIN_INIT
{
  temu_Class *cls = temu_registerClass("MyClass", create, dispose);

  temu_addInterface(cls, "ApbIface", "ApbIface", &ApbIface);
  temu_addInterface(cls, "DeviceIface", "DeviceIface", &DeviceIface);

}
```