d) Xilinx IP cores

The Xilinx EDK toolchain ships with a set of IP cores (devices) for use in Xilinx Spartan and Virtex FPGAs. The devices cover the whole range of standard device types (network, serial, etc.) and miscellaneous devices (gpio, LCD, spi, etc). Also, since these devices are implemented within the fpga fabric every instance of the device can be synthesised with different options that change the behaviour.

Each IP-core has a set of parameters which the FPGA designer can use to control how the core is synthesized. Historically, the EDK tool would extract the device parameters relevant to device drivers and copy them into an 'xparameters.h' in the form of #define symbols. This tells the device drivers how the IP cores are configured, but it requires the kernel to be recompiled every time the FPGA bitstream is resynthesized.

The new approach is to export the parameters into the device tree and generate a new device tree each time the FPGA bitstream changes. The parameters which used to be exported as #defines will now become properties of the device node. In general, device nodes for IP-cores will take the following form:

```
(name): (generic-name)@(base-address) {
        compatible = "xlnx,(ip-core-name)-(HW_VER)"
                     [, (list of compatible devices), ...];
        reg = <(baseaddr) (size)>;
        interrupt-parent = <&interrupt-controller-phandle>;
        interrupts = < ... >;
        xlnx,(parameter1) = "(string-value)";
        xlnx,(parameter2) = <(int-value)>;
};

(generic-name):   an open firmware-style name that describes the
                  generic class of device.  Preferably, this is one word,
such
                  as 'serial' or 'ethernet'.
(ip-core-name): the name of the ip block (given after the BEGIN
                  directive in system.mhs).  Should be in lowercase
                  and all underscores '_' converted to dashes '-'.
(name):           is derived from the "PARAMETER INSTANCE" value.
(parameter#):     C_* parameters from system.mhs.  The C_ prefix is
                  dropped from the parameter name, the name is converted
                  to lowercase and all underscore '_' characters are
                  converted to dashes '-'.
(baseaddr):       the baseaddr parameter value (often named C_BASEADDR).
(HW_VER):         from the HW_VER parameter.
(size):           the address range size (often C_HIGHADDR - C_BASEADDR +
1).
```

Typically, the compatible list will include the exact IP core version followed by an older IP core version which implements the same interface or any other device with the same interface.

'reg', 'interrupt-parent' and 'interrupts' are all optional properties.

For example, the following block from system.mhs:

```
	BEGIN opb_uartlite
		PARAMETER INSTANCE = opb_uartlite_0
		PARAMETER HW_VER = 1.00.b
		PARAMETER C_BAUDRATE = 115200
		PARAMETER C_DATA_BITS = 8
		PARAMETER C_ODD_PARITY = 0
		PARAMETER C_USE_PARITY = 0
		PARAMETER C_CLK_FREQ = 50000000
		PARAMETER C_BASEADDR = 0xEC100000
		PARAMETER C_HIGHADDR = 0xEC10FFFF
		BUS_INTERFACE SOPB = opb_7
		PORT OPB_Clk = CLK_50MHz
		PORT Interrupt = opb_uartlite_0_Interrupt
		PORT RX = opb_uartlite_0_RX
		PORT TX = opb_uartlite_0_TX
		PORT OPB_Rst = sys_bus_reset_0
	END
```

becomes the following device tree node:

```
	opb_uartlite_0: serial@ec100000 {
		device_type = "serial";
		compatible = "xlnx,opb-uartlite-1.00.b";
		reg = <ec100000 10000>;
		interrupt-parent = <&opb_intc_0>;
		interrupts = <1 0>; // got this from the opb_intc parameters
		current-speed = <d#115200>;     // standard serial device prop
		clock-frequency = <d#50000000>; // standard serial device prop
		xlnx,data-bits = <8>;
		xlnx,odd-parity = <0>;
		xlnx,use-parity = <0>;
	};
```

Some IP cores actually implement 2 or more logical devices. In this case, the device should still describe the whole IP core with a single node and add a child node for each logical device. The ranges property can be used to translate from parent IP-core to the registers of each device. In addition, the parent node should be compatible with the bus type 'xlnx,compound', and should contain #address-cells and #size-cells, as with any other bus. (Note: this makes the assumption that both logical devices have the same bus binding. If this is not true, then separate nodes should be used for each logical device). The 'cell-index' property can be used to enumerate logical devices within an IP core. For example, the following is the system.mhs entry for the dual ps2 controller found on the ml403 reference design.

```
	BEGIN opb_ps2_dual_ref
		PARAMETER INSTANCE = opb_ps2_dual_ref_0
		PARAMETER HW_VER = 1.00.a
		PARAMETER C_BASEADDR = 0xA9000000
		PARAMETER C_HIGHADDR = 0xA9001FFF
		BUS_INTERFACE SOPB = opb_v20_0
		PORT Sys_Intr1 = ps2_1_intr
		PORT Sys_Intr2 = ps2_2_intr
```

```
        PORT Clkin1 = ps2_clk_rx_1
        PORT Clkin2 = ps2_clk_rx_2
        PORT Clkpd1 = ps2_clk_tx_1
        PORT Clkpd2 = ps2_clk_tx_2
        PORT Rx1 = ps2_d_rx_1
        PORT Rx2 = ps2_d_rx_2
        PORT Txpd1 = ps2_d_tx_1
        PORT Txpd2 = ps2_d_tx_2
END
```

It would result in the following device tree nodes:

```
opb_ps2_dual_ref_0: opb-ps2-dual-ref@a9000000 {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "xlnx,compound";
        ranges = <0 a9000000 2000>;
        // If this device had extra parameters, then they would
        // go here.
        ps2@0 {
                compatible = "xlnx,opb-ps2-dual-ref-1.00.a";
                reg = <0 40>;
                interrupt-parent = <&opb_intc_0>;
                interrupts = <3 0>;
                cell-index = <0>;
        };
        ps2@1000 {
                compatible = "xlnx,opb-ps2-dual-ref-1.00.a";
                reg = <1000 40>;
                interrupt-parent = <&opb_intc_0>;
                interrupts = <3 0>;
                cell-index = <0>;
        };
};
```

Also, the system.mhs file defines bus attachments from the processor to the devices. The device tree structure should reflect the bus attachments. Again an example; this system.mhs fragment:

```
BEGIN ppc405_virtex4
        PARAMETER INSTANCE = ppc405_0
        PARAMETER HW_VER = 1.01.a
        BUS_INTERFACE DPLB = plb_v34_0
        BUS_INTERFACE IPLB = plb_v34_0
END

BEGIN opb_intc
        PARAMETER INSTANCE = opb_intc_0
        PARAMETER HW_VER = 1.00.c
        PARAMETER C_BASEADDR = 0xD1000FC0
        PARAMETER C_HIGHADDR = 0xD1000FDF
        BUS_INTERFACE SOPB = opb_v20_0
END

BEGIN opb_uart16550
        PARAMETER INSTANCE = opb_uart16550_0
```

```
        PARAMETER HW_VER = 1.00.d
        PARAMETER C_BASEADDR = 0xa0000000
        PARAMETER C_HIGHADDR = 0xa0001FFF
        BUS_INTERFACE SOPB = opb_v20_0
END

BEGIN plb_v34
        PARAMETER INSTANCE = plb_v34_0
        PARAMETER HW_VER = 1.02.a
END

BEGIN plb_bram_if_cntlr
        PARAMETER INSTANCE = plb_bram_if_cntlr_0
        PARAMETER HW_VER = 1.00.b
        PARAMETER C_BASEADDR = 0xFFFF0000
        PARAMETER C_HIGHADDR = 0xFFFFFFFF
        BUS_INTERFACE SPLB = plb_v34_0
END

BEGIN plb2opb_bridge
        PARAMETER INSTANCE = plb2opb_bridge_0
        PARAMETER HW_VER = 1.01.a
        PARAMETER C_RNG0_BASEADDR = 0x20000000
        PARAMETER C_RNG0_HIGHADDR = 0x3FFFFFFF
        PARAMETER C_RNG1_BASEADDR = 0x60000000
        PARAMETER C_RNG1_HIGHADDR = 0x7FFFFFFF
        PARAMETER C_RNG2_BASEADDR = 0x80000000
        PARAMETER C_RNG2_HIGHADDR = 0xBFFFFFFF
        PARAMETER C_RNG3_BASEADDR = 0xC0000000
        PARAMETER C_RNG3_HIGHADDR = 0xDFFFFFFF
        BUS_INTERFACE SPLB = plb_v34_0
        BUS_INTERFACE MOPB = opb_v20_0
END
```

Gives this device tree (some properties removed for clarity):

```
plb@0 {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "xlnx,plb-v34-1.02.a";
        device_type = "ibm,plb";
        ranges; // 1:1 translation

        plb_bram_if_cntrl_0: bram@ffff0000 {
                reg = <ffff0000 10000>;
        }

        opb@20000000 {
                #address-cells = <1>;
                #size-cells = <1>;
                ranges = <20000000 20000000 20000000
                          60000000 60000000 20000000
                          80000000 80000000 40000000
                          c0000000 c0000000 20000000>;

                opb_uart16550_0: serial@a0000000 {
```

```
                        reg = <a00000000 2000>;
                };

                opb_intc_0: interrupt-controller@d1000fc0 {
                        reg = <d1000fc0 20>;
                };
        };
};
```

That covers the general approach to binding xilinx IP cores into the device tree. The following are bindings for specific devices:

i) Xilinx ML300 Framebuffer

Simple framebuffer device from the ML300 reference design (also on the ML403 reference design as well as others).

Optional properties:
- resolution = <xres yres> : pixel resolution of framebuffer. Some implementations use a different resolution. Default is <d#640 d#480>
- virt-resolution = <xvirt yvirt> : Size of framebuffer in memory. Default is <d#1024 d#480>.
- rotate-display (empty) : rotate display 180 degrees.

ii) Xilinx SystemACE

The Xilinx SystemACE device is used to program FPGAs from an FPGA bitstream stored on a CF card. It can also be used as a generic CF interface device.

Optional properties:
- 8-bit (empty) : Set this property for SystemACE in 8 bit mode

iii) Xilinx EMAC and Xilinx TEMAC

Xilinx Ethernet devices. In addition to general xilinx properties listed above, nodes for these devices should include a phy-handle property, and may include other common network device properties like local-mac-address.

iv) Xilinx Uartlite

Xilinx uartlite devices are simple fixed speed serial ports.

Required properties:
- current-speed : Baud rate of uartlite

v) Xilinx hwicap

Xilinx hwicap devices provide access to the configuration logic of the FPGA through the Internal Configuration Access Port (ICAP). The ICAP enables partial reconfiguration of the FPGA, readback of the configuration information, and some control over 'warm boots' of the FPGA fabric.

Required properties:
- xlnx,family : The family of the FPGA, necessary since the
capabilities of the underlying ICAP hardware
differ between different families. May be
'virtex2p', 'virtex4', or 'virtex5'.

vi) Xilinx Uart 16550

Xilinx UART 16550 devices are very similar to the NS16550 but with
different register spacing and an offset from the base address.

Required properties:
- clock-frequency : Frequency of the clock input
- reg-offset : A value of 3 is required
- reg-shift : A value of 2 is required

vii) Xilinx USB Host controller

The Xilinx USB host controller is EHCI compatible but with a different
base address for the EHCI registers, and it is always a big-endian
USB Host controller. The hardware can be configured as high speed only,
or high speed/full speed hybrid.

Required properties:
- xlnx,support-usb-fs: A value 0 means the core is built as high speed
only. A value 1 means the core also supports
full speed devices.