UART Registers

Usecases

There are cases where you want to dynamically control your FPGA from external interface, in such cases many people uses `SPI` or `I2C` controlled registers to configure `CSR` (Configuration and Status Registers)

This repo contains `uart_reg` module which can be used to set configuration registers, currently only manually read/write is possible from external interface, other modules can only read values from this registers. Writing to these registers from internal will be future implementation.

`uart_core` is slighly adopted version from `libfpga`

How to Use

It is very simple to import this module into your design, just copy `uart_core.v` and `uart_regs.v` into your workspace folder, and the use the `uart_regs` module from `uart_regs.v` file.

`uart_regs` module

```
module uart_regs #(

    parameter W_REG = 32

)(
    input wire clk,
    input wire rst_n,

    input wire rx,
    output wire tx,
    input wire [9:0] div_int,
    input wire [3:0] div_frac,

    output wire [W_REG-1:0] Reg1,
    output wire [W_REG-1:0] Reg2,
    output wire [W_REG-1:0] Reg3,
    output wire [W_REG-1:0] Reg4,
    output wire [W_REG-1:0] Reg5,
    output wire [W_REG-1:0] Reg6,
    output wire [W_REG-1:0] Reg7,
    output wire [W_REG-1:0] Reg8
);
```

As defined in module definition, `Reg{1..8}` are configuration registers which is read by other

modules. `tx` and `rx` should be mapped to FPGA pins and `clk` and `rst_n` need to be conneced. `div_frac` and `div_int` can be set internally to set `UART` baudrate as per following expression:

`baud_rate` = `fCLk`/(`8`*(`div_int` + `div_frac/16`))

Implementation

`uart_regs` needs `uart_core` module which is `UART` state-machine, whenever we have have `read/write` request as byte in `RX FIFO` `rx_irq` is asserted which triggers `uart_regs` state-machine:

1. If `read_reg` is asserted