Software Development

Using Hardware Timers

LECTURE 6

IOULIIA SKLIAROVA

Typical Errors

Positive VHDL data subtype

- `subtype positive is integer range 1 to integer'high;`
- **signal** count : positive **range** 0 **to** 9;

Integer ranges

- `signal count : integer range 0 to 9;`
- What happens if `count` is incremented past 9 or decremented below 0?
- Will the synthesis tools force the signal to wrap around?
- The tool assumes that the signal stays within range **by design**, so the case where this is not true, circuit can give any output on the internal bus used to implement the integer value. In this case for 0 to 9 it will be a 4 bit bus, and if the design does not keep the value in range 0 to 9, then even "illegal" values in range 10 to 15 may be presented in the circuit made by synthesis.

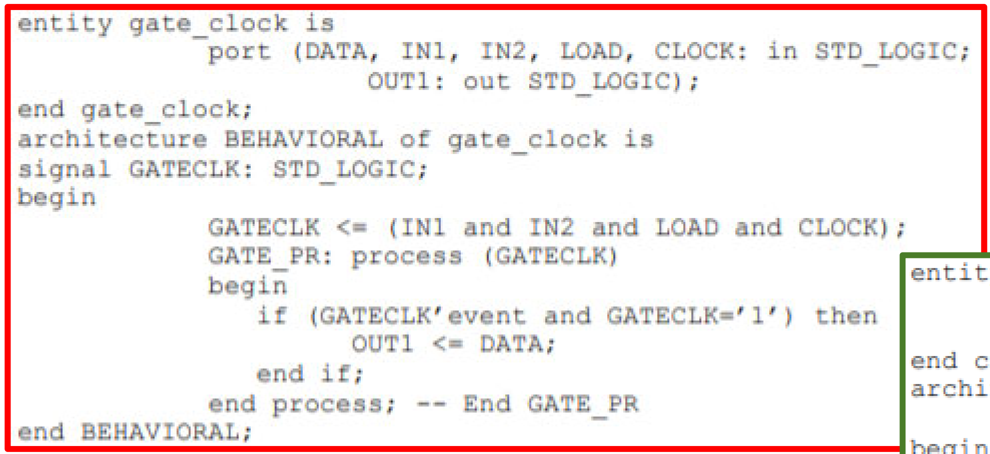
Rem and / operators and timing constraints

- `signal count : unsigned(7 downto 0);`
- …
- `LSsec <= count rem 10;`
- `MSsec <= count / 10;`

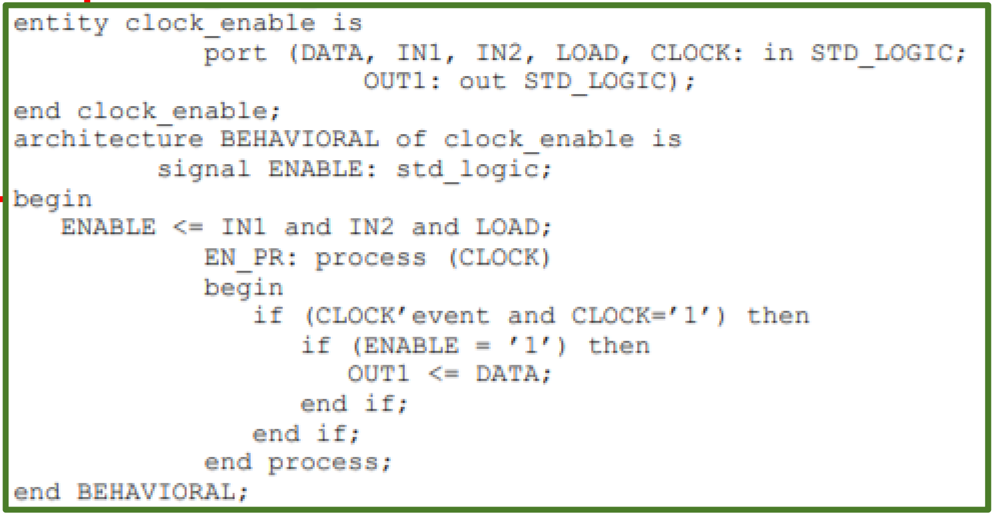
Typical Errors

Gated clock

- The warning is being issued because one of the clock net is sourced by a combinational logic, making it a **gated-clock**.
- Xilinx highly recommends that you use the clock enable pin instead of gated clocks.

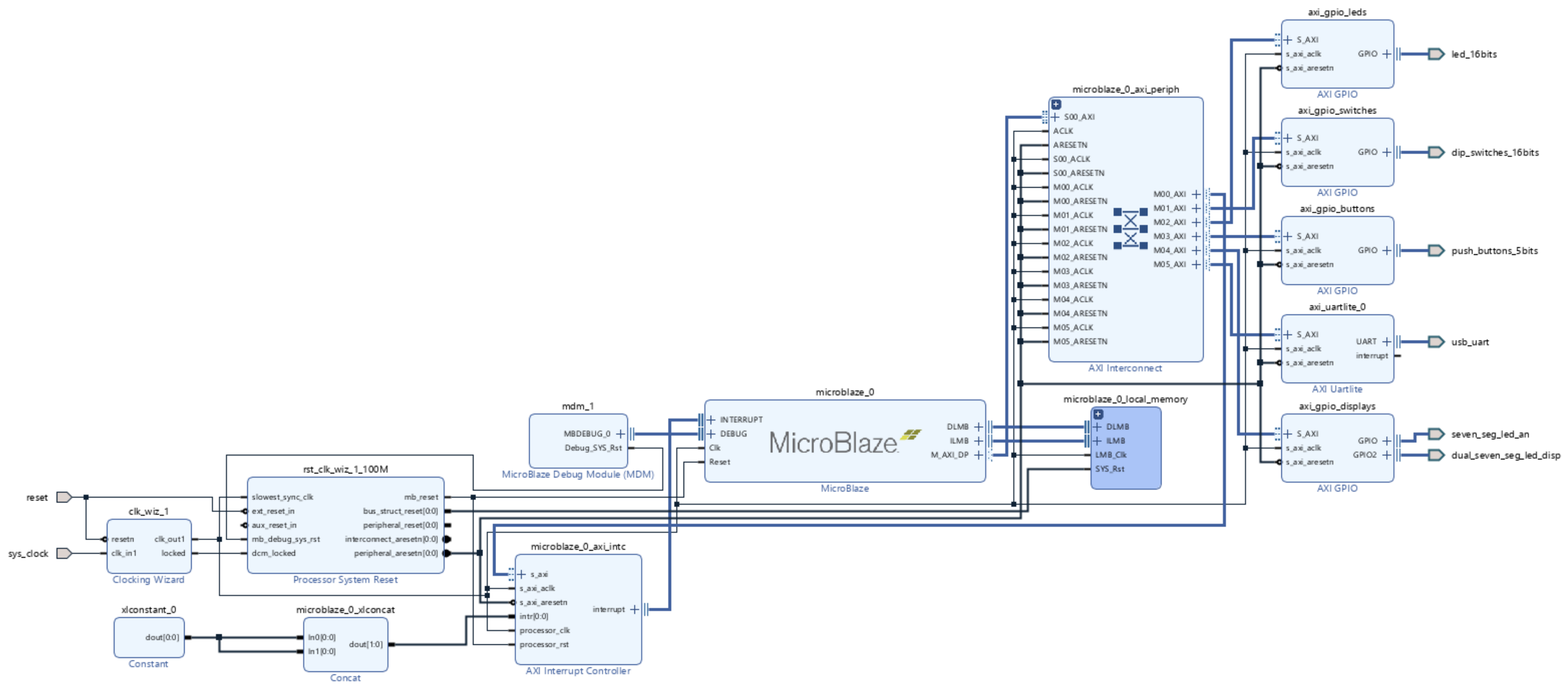


```
entity gate_clock is
            port (DATA, IN1, IN2, LOAD, CLOCK: in STD_LOGIC;
                      OUT1: out STD_LOGIC);
end gate_clock;
architecture BEHAVIORAL of gate_clock is
signal GATECLK: STD_LOGIC;
begin
            GATECLK <= (IN1 and IN2 and LOAD and CLOCK);
            GATE_PR: process (GATECLK)
            begin
               if (GATECLK'event and GATECLK='1') then
                     OUT1 <= DATA;
               end if;
            end process; -- End GATE_PR
end BEHAVIORAL;
```



```
entity clock_enable is
            port (DATA, IN1, IN2, LOAD, CLOCK: in STD_LOGIC;
                      OUT1: out STD_LOGIC);
end clock_enable;
architecture BEHAVIORAL of clock_enable is
         signal ENABLE: std_logic;
begin
   ENABLE <= IN1 and IN2 and LOAD;
            EN_PR: process (CLOCK)
            begin
               if (CLOCK'event and CLOCK='1') then
                  if (ENABLE = '1') then
                     OUT1 <= DATA;
                  end if;
               end if;
            end process;
end BEHAVIORAL;
```

Block Design (BD)



Implementing Hardware

Validate design

- you can run a comprehensive design check on the design.

Generate output products (Global)

- After the BD is complete and the design is validated, you must generate output products for synthesis and simulation, to integrate the BD into a top-level RTL design. The source files and the appropriate constraints for all the IP are generated and made available in the Vivado® Design Suite (IDE) Sources window.
- Generating the output products generates the top-level netlist of the BD. The netlist is generated in the HDL language specified by the Settings → General → Target Language for the project.

Create HDL Wrapper

- This command generates a top-level HDL file with an instantiation template for the IP integrator BD.

Generate Bitstream

Problems and Results

If a synthesis warning will appear regarding the clock period of the processor’s local memory (BRAM), execute the following TCL commands:

- `set_property CONFIG.Port_A_Clock 100 [get_bd_cells /microblaze_0_local_memory/lmb_bram]`
- `report_property [get_bd_cells /microblaze_0_local_memory/lmb_bram]`

Open the synthesized design and execute the following TCL commands:

- `set_property CONFIG_VOLTAGE 3.3 [get_designs synth_1]`
- `set_property CFGBVS VCCO [get_designs synth_1]`
- Then close the synthesized design and save an XDC file with any name

Development Tools

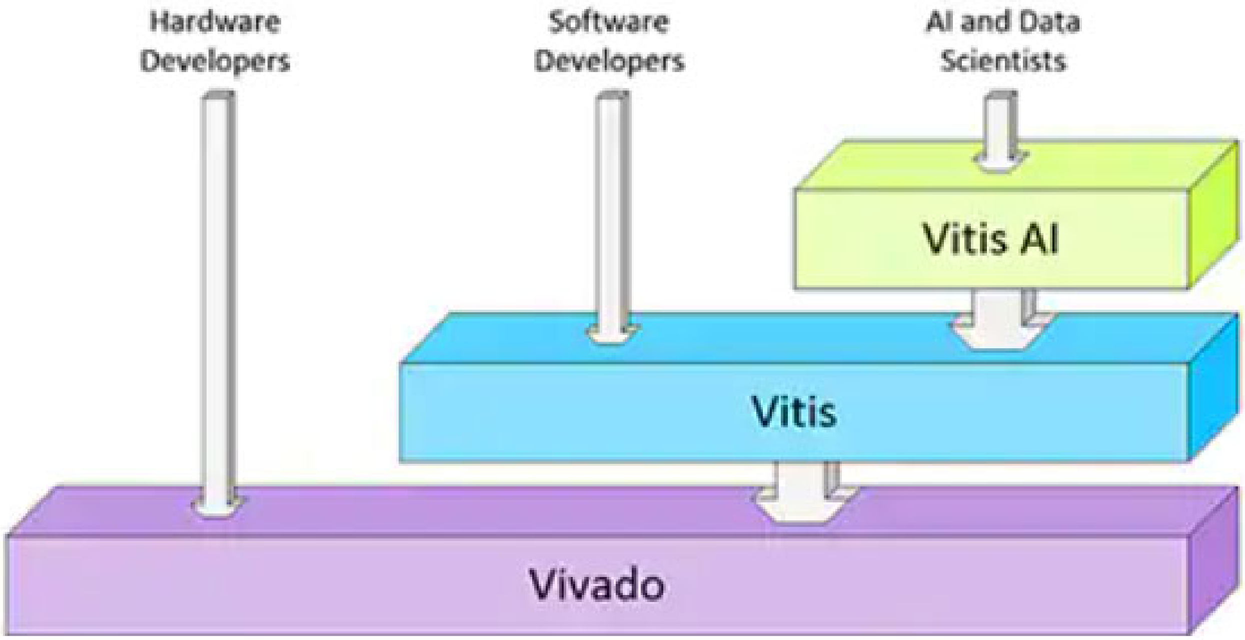


Figure 6: A high-level view of the Xilinx Vivado and Vitis design tool stack reflects how users can work with the tools at the most appropriate levels of abstraction. Hardware designers work with Vivado, software developers work with Vitis, and AI and data scientists work with Vitis AI. (Image source: Max Maxfield)

Software Development

Export hardware

Launch Vitis

- Create a Vitis workspace

Create new application project

- Create a new platform from hardware (XSA)
- Create a “helloworld” project

Program device

Launch Vitis serial terminal (or any other similar application)

Build the project

Execute the software application

Hardware Timer IPs

AXI Timer

- AXI Timer/Counter is a 32/64-bit timer module that interfaces to the AXI4-Lite interface.

Fixed Interval Timer (FIT)

- The FIT core is a peripheral that generates a strobe (interrupt) signal at fixed intervals and is not attached to any bus.

AXI Timer

Two programmable interval timers with interrupt, event generation, and event capture capabilities.

Each timer module has an associated load register that is used to hold either the initial value for the counter for event generation or a capture value, depending on the mode of the timer.

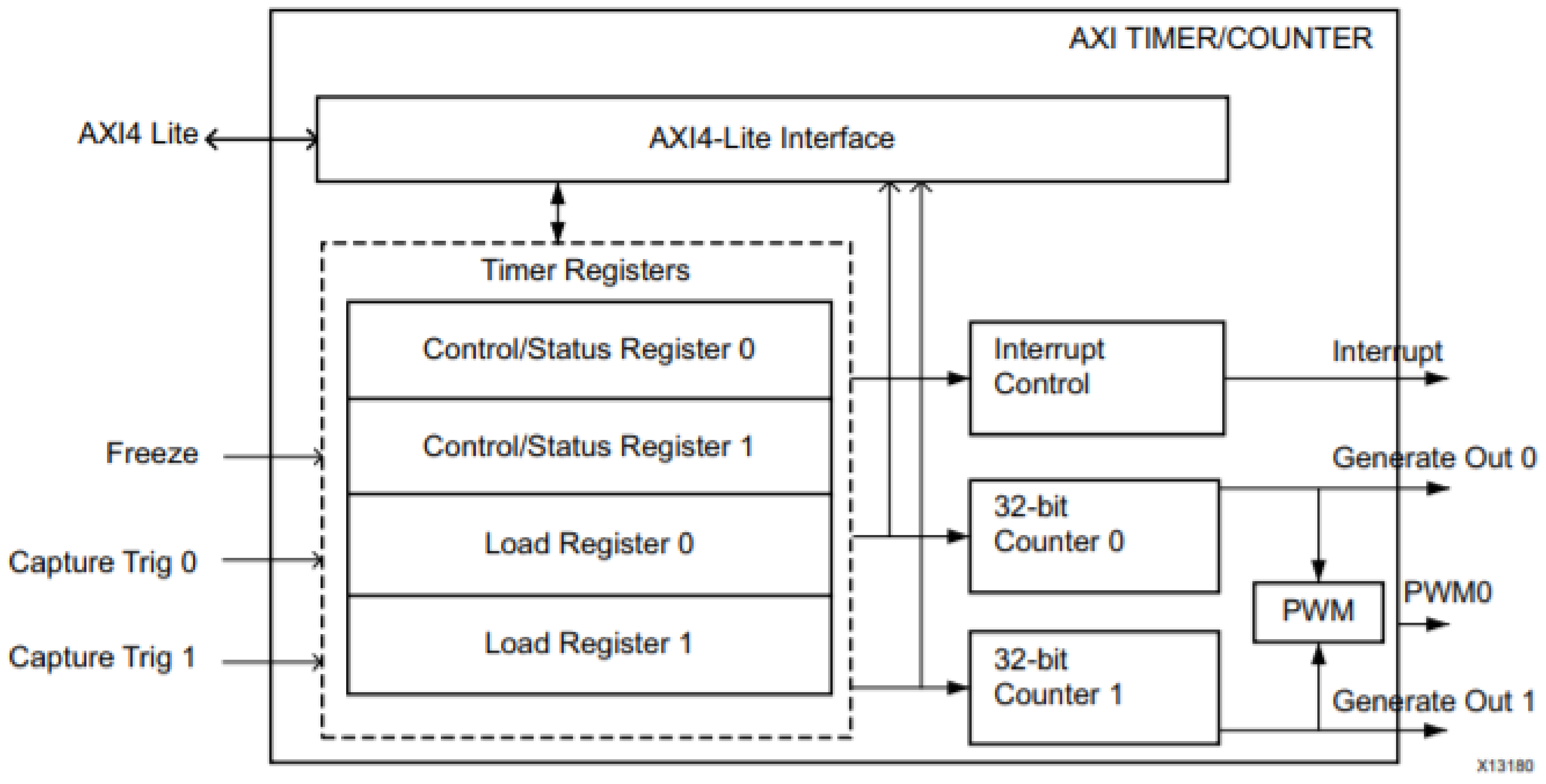
In the Generate mode, the value in the load register is loaded into the counter. The counter, when enabled, begins to count up or down, depending on the selection of the Up/Down Count Timer (UDT) bit in the Timer Control Status Register (TCSR).

On transition of the carry out of the counter, the counter stops or automatically reloads the generate value from the load register and, after reaching the timeout value, continues counting as selected by the Auto Reload/Hold (ARHT) bit in the TCSR.

The Timer Interrupt Status (TINT) bit is set in TCSR and, if enabled, the external GenerateOut signal is driven to 1 for one clock cycle.

If enabled, the interrupt signal for the timer is driven to 1 when reaching the timeout value. Clear the interrupt by writing a 1 to the Timer Interrupt register. Use this mode for generating repetitive interrupts or external signals with a specified interval.

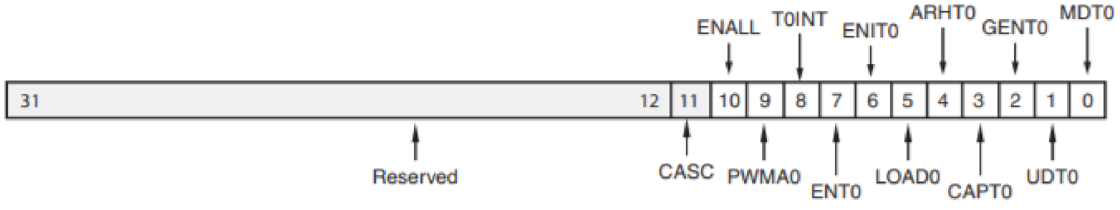
AXI Timer's Block Diagram



AXI Timer Control/Status Register

T0INT - Timer 0 Interrupt Indicates that the condition for an interrupt on this timer has occurred.

- Read:
  - 0 = No interrupt has occurred
  - 1 = Interrupt has occurred
- Write:
  - 0 = No change in state of T0INT
  - 1 = Clear T0INT (clear to 0)



ENT0 - Enable Timer 0

- 0 = Disable timer (counter halts)
- 1 = Enable timer (counter runs)

LOAD0 - Load Timer 0

- 0 = No load
- 1 = Loads timer with value in TLR0

ARHT0 - Auto Reload/Hold Timer 0 (when the timer is in Generate mode, this bit determines whether the counter reloads the generate value and continues running or holds at the termination value).

- 0 = Hold counter
- 1 = Reload generate value

UDT0 – Up/Down Count Timer 0

- 0 = Timer functions as up counter
- 1 = Timer functions as down counter

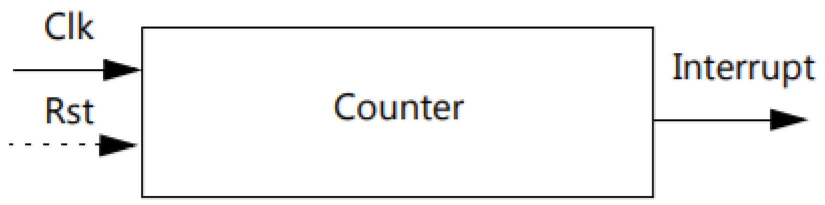
Fixed Interval Timer (FIT)

The FIT core is a peripheral that generates a strobe (interrupt) signal at fixed intervals and is not attached to any bus.

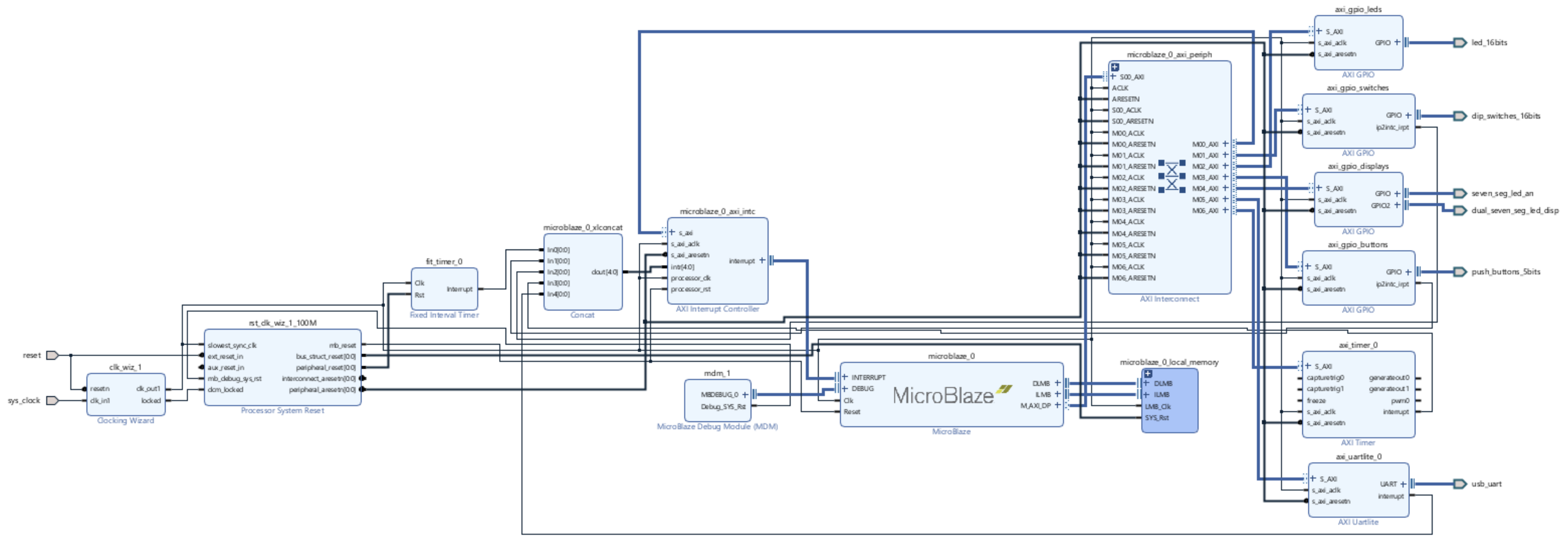
The FIT core generates an interrupt every C_NO_CLOCKS.

The interrupt signal is held high for one clock cycle.

The core begins operation immediately after device configuration if the reset is not connected.



Block Design (BD)



Universidade de Aveiro

Final Remarks

At the end of this lecture you should be able to:

- create a base hardware platform with the MicroBlaze, GPIOs and timers
- create and execute a simple Vitis project

To do:

- Construct the considered hardware platform
- Test the considered applications in Vitis