Wireguard FPGA

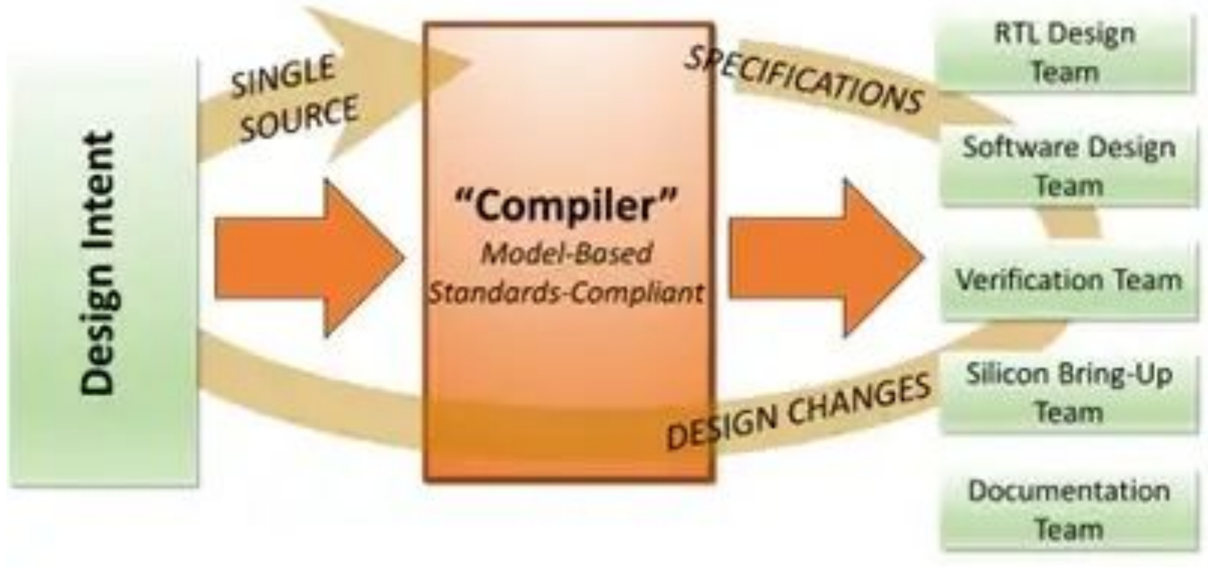
CSR/HAL Generation Flow Meeting

10/29/2024

Outline

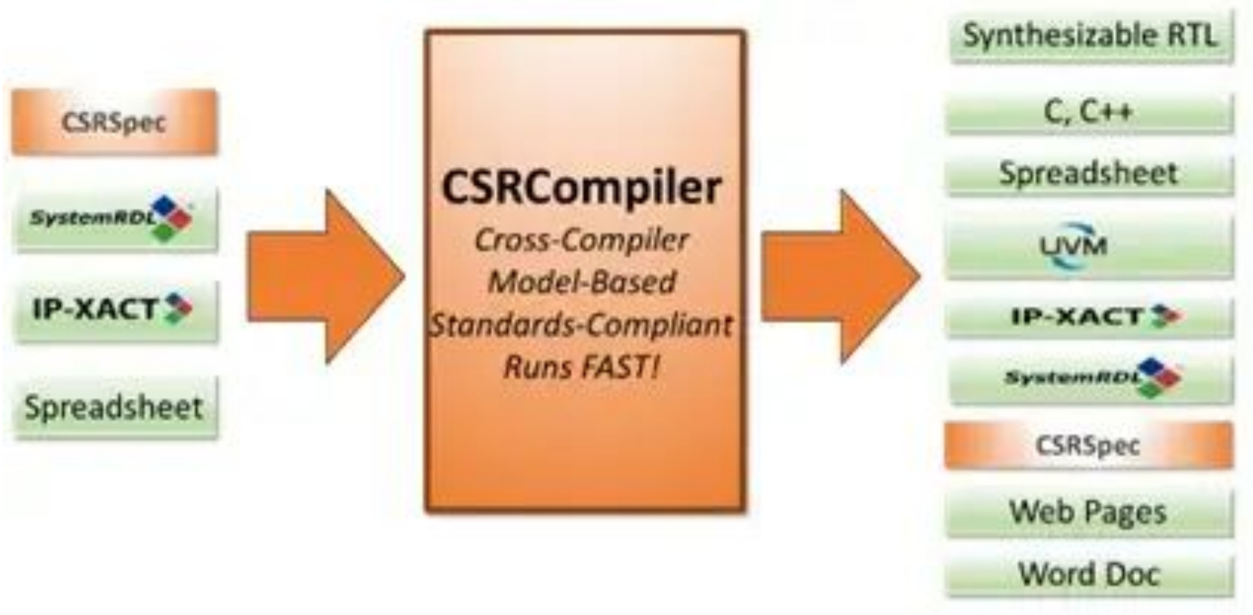
- Automated CSR Generation
- RgGen
- SystemRDL

Automated CSR Generation



- https://semiwiki.com/eda/semifore/319538-stop-writing-rtl-for-registers/

Automated CSR Generation



- https://semiwiki.com/eda/semifore/319538-stop-writing-rtl-for-registers/

RgGen

- MIT Licensed Open-source by Taichi Ishitani (https://github.com/rggen/rggen)
- Based on UVM register model (UVM RAL/uvm_reg)
- Supports standard bus protocols for CSR IF (APB, AXI-Lite, Wishbone)
- Inputs:
  - YAML, JSON, TOML, Spreadsheet (XLSX, ODS, CSV)
- Outputs:
  - SystemVerilog, Verilog, VHDL
  - C header file
  - Register map documents written in Markdown
- Implemented in Ruby (supported by *docker images*)
- Plugins for special bit field types & custom host protocol

RgGen

```yaml
- register_block:
  - name: uart_csr
    byte_size:  32

################################################################
  - register:
    - name: rbr
      offset_address: 0x00
      type: [indirect, [lcr.dlab, 0]]
      comment: |
        Receiver Buffer Register
      bit_fields:
      - { bit_assignment: { lsb: 0, width: 8 }, type: rotrg }

################################################################
  - register:
    - name: thr
      offset_address: 0x00
      type: [indirect, [lcr.dlab, 0]]
      comment: |
        Transmitter Holding Register
      bit_fields:
      - { bit_assignment: { lsb: 0, width: 8 }, type: wotrg, initial_value: 0xFF }

################################################################
  - register:
    - name: ier
      offset_address: 0x04
      type: [indirect, [lcr.dlab, 0]]
      comment: |
        Interrupt Enable Register
      bit_fields:
      - <<:
        - { name: erbfi, bit_assignment: { lsb: 0, width: 1 }, type: rw, initial_value: 0 }
```

```yaml
  - register:
    - name: fcr
      offset_address: 0x08
      comment: |
        FIFO Control Register
      bit_fields:
      - <<:
        - { name: fifoen, bit_assignment: { lsb: 0, width: 1 }, type: wo, initial_value: 0 }
        - comment: |
            FIFO Enable
            1: Enables FIFOs
      - <<:
        - { name: rcvr_fifo_reset, bit_assignment: { lsb: 1, width: 1 }, type: w1trg }
        - comment: |
            Receiver FIFO Reset
            1: Resets RCVR FIFO
      - <<:
        - { name: xmit_fifo_reset, bit_assignment: { lsb: 2, width: 1 }, type: w1trg }
        - comment: |
            Transmitter FIFO Reset
            1: Resets XMIT FIFO
      - <<:
        - { name: dma_mode_select, bit_assignment: { lsb: 3, width: 1 }, type: wo, initial_value: 0 }
        - comment: |
            DMA Mode Select
            0: Mode 0
            1: Mode 1
      - <<:
        - { name: rcvr_fifo_trigger_level, bit_assignment: { lsb: 6, width: 2 }, type: wo, initial_value: 0b00 }
        - comment: |
            RCVR FIFO Trigger Level
            0b00: 1 byte
            0b01: 4 bytes
            0b10: 8 bytes
            0b11: 14 bytes
```

YAML input

RgGen

```systemverilog
module uart_csr
  import rggen_rtl_pkg::*;
#(
  parameter int ADDRESS_WIDTH = 5,
  parameter bit PRE_DECODE = 0,
  parameter bit [ADDRESS_WIDTH-1:0] BASE_ADDRESS = '0,
  parameter bit ERROR_STATUS = 0,
  parameter bit [31:0] DEFAULT_READ_DATA = '0,
  parameter bit INSERT_SLICER = 0,
  parameter bit [7:0] DLL_INITIAL_VALUE = 8'h00,
  parameter bit [7:0] DLM_INITIAL_VALUE = 8'h00
)(
  input logic i_clk,
  input logic i_rst_n,
  rggen_apb_if.slave apb_if,
  input logic [7:0] i_rbr,
  output logic o_rbr_read_trigger,
  output logic [7:0] o_thr,
  output logic o_thr_write_trigger,
  output logic o_ier_erbfi,
  output logic o_ier_etbei,
  output logic o_ier_elsi,
  output logic o_ier_edssi,
  input logic i_iir_intpend,
  input logic [2:0] i_iir_intid2,
  output logic o_fcr_fifoen,
  output logic o_fcr_rcvr_fifo_reset_trigger,
  output logic o_fcr_xmit_fifo_reset_trigger,
  output logic o_fcr_dma_mode_select,
  output logic [1:0] o_fcr_rcvr_fifo_trigger_level,
```

```systemverilog
      if (1) begin : g_rcvr_fifo_reset
        rggen_bit_field_if #(1) bit_field_sub_if();
        `rggen_connect_bit_field_if(bit_field_if, bit_field_sub_if, 1, 1)
        rggen_bit_field_w01trg #(
          .TRIGGER_VALUE  (1'b1),
          .WIDTH          (1)
        ) u_bit_field (
          .i_clk        (i_clk),
          .i_rst_n      (i_rst_n),
          .bit_field_if (bit_field_sub_if),
          .i_value      ('0),
          .o_trigger    (o_fcr_rcvr_fifo_reset_trigger)
        );
      end
      if (1) begin : g_xmit_fifo_reset
        rggen_bit_field_if #(1) bit_field_sub_if();
        `rggen_connect_bit_field_if(bit_field_if, bit_field_sub_if, 2, 1)
        rggen_bit_field_w01trg #(
          .TRIGGER_VALUE  (1'b1),
          .WIDTH          (1)
        ) u_bit_field (
          .i_clk        (i_clk),
          .i_rst_n      (i_rst_n),
          .bit_field_if (bit_field_sub_if),
          .i_value      ('0),
          .o_trigger    (o_fcr_xmit_fifo_reset_trigger)
        );
      end
```

Generated SystemVerilog RTL interface (flat model)

RgGen

```c
#ifndef UART_CSR_H
#define UART_CSR_H
#include "stdint.h"
#define UART_CSR_RBR_BIT_WIDTH 8
#define UART_CSR_RBR_BIT_MASK 0xff
#define UART_CSR_RBR_BIT_OFFSET 0
#define UART_CSR_RBR_BYTE_WIDTH 4
#define UART_CSR_RBR_BYTE_SIZE 4
#define UART_CSR_RBR_BYTE_OFFSET 0x0
#define UART_CSR_THR_BIT_WIDTH 8
#define UART_CSR_THR_BIT_MASK 0xff
#define UART_CSR_THR_BIT_OFFSET 0
#define UART_CSR_THR_BYTE_WIDTH 4
#define UART_CSR_THR_BYTE_SIZE 4
#define UART_CSR_THR_BYTE_OFFSET 0x0
```

```c
typedef union {
  uint32_t rbr;
  uint32_t thr;
  uint32_t dll;
} uart_csr_reg_0x00_t;
typedef union {
  uint32_t ier;
  uint32_t dlm;
} uart_csr_reg_0x04_t;
typedef union {
  uint32_t iir;
  uint32_t fcr;
} uart_csr_reg_0x08_t;
typedef struct {
  uart_csr_reg_0x00_t reg_0x00;
  uart_csr_reg_0x04_t reg_0x04;
  uart_csr_reg_0x08_t reg_0x08;
  uint32_t lcr;
  uint32_t mrc;
  uint32_t lsr;
  uint32_t msr;
  uint32_t scratch;
} uart_csr_t;
```

Generated C header file

SystemRDL

- Standardized by Accellera (https://www.accellera.org/downloads/standards/systemrdl)
- GPL-3.0 Licensed (https://github.com/systemrdl)
- Open-Source Tools:
  - **PeakRDL-regblock** generates synthesizable SystemVerilog RTL
  - **PeakRDL-html** produces intuitive and dynamic HTML documentation
  - **PeakRDL-uvm** generates a UVM register model
  - **PeakRDL-ipxact** lets you import and export IP-XACT XML
  - **PeakRDL-Markdown** export to a Markdown document
  - **PeakRDL-cheader** outputs a software abstraction layer C header
- Implemented in Python
- Extensive documentation

SystemRDL

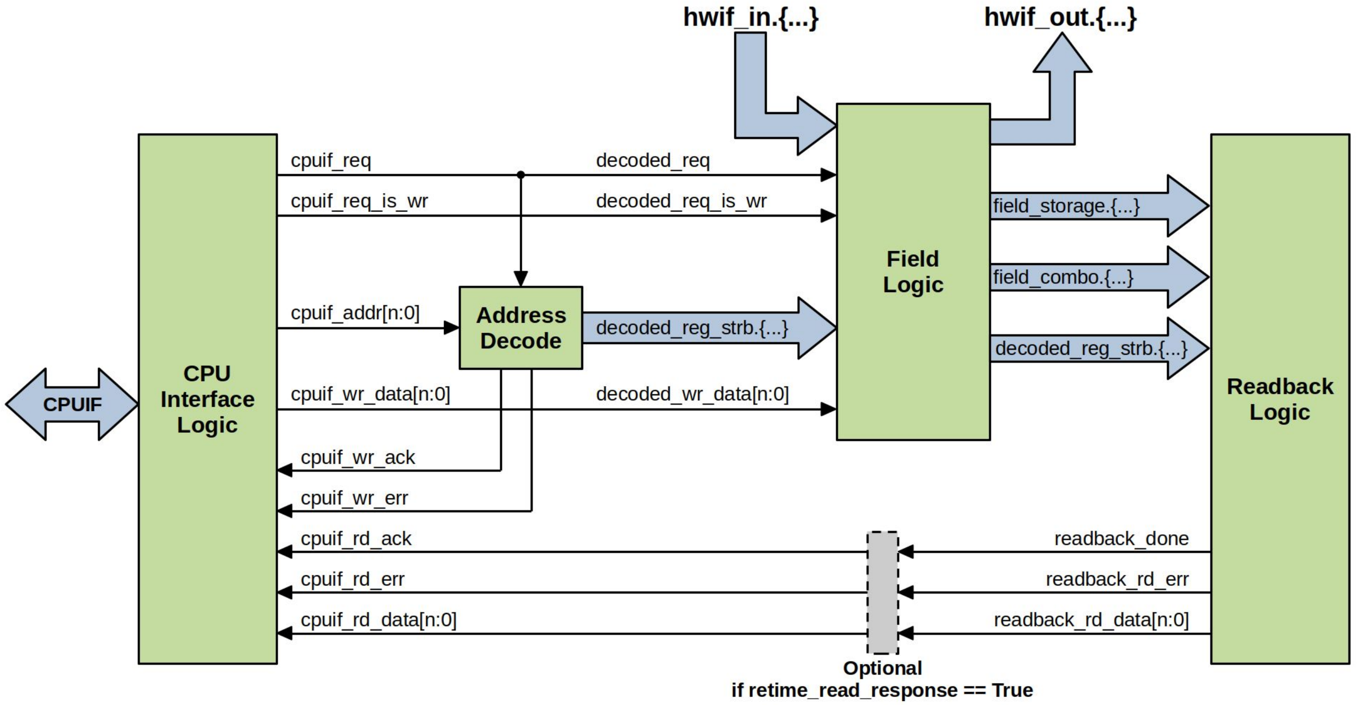
SystemRDL input

```
addrmap my_design {
    reg {
        field {
            sw = rw;
            hw = rw;
            we;
        } my_field;
    } my_reg[2];
};
```

Output - structured HW interface

```
hwif_out.my_reg[0].my_field.value
hwif_in.my_reg[0].my_field.next
hwif_in.my_reg[0].my_field.we
hwif_out.my_reg[1].my_field.value
hwif_in.my_reg[1].my_field.next
hwif_in.my_reg[1].my_field.we
```

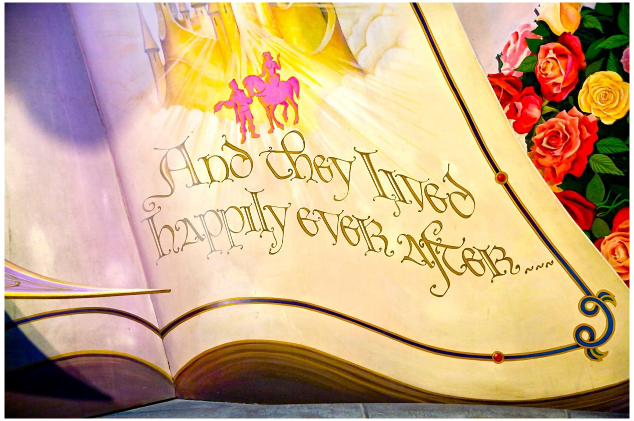
SystemRDL



Generated RTL

SystemRDL

DEMO



Thank you!