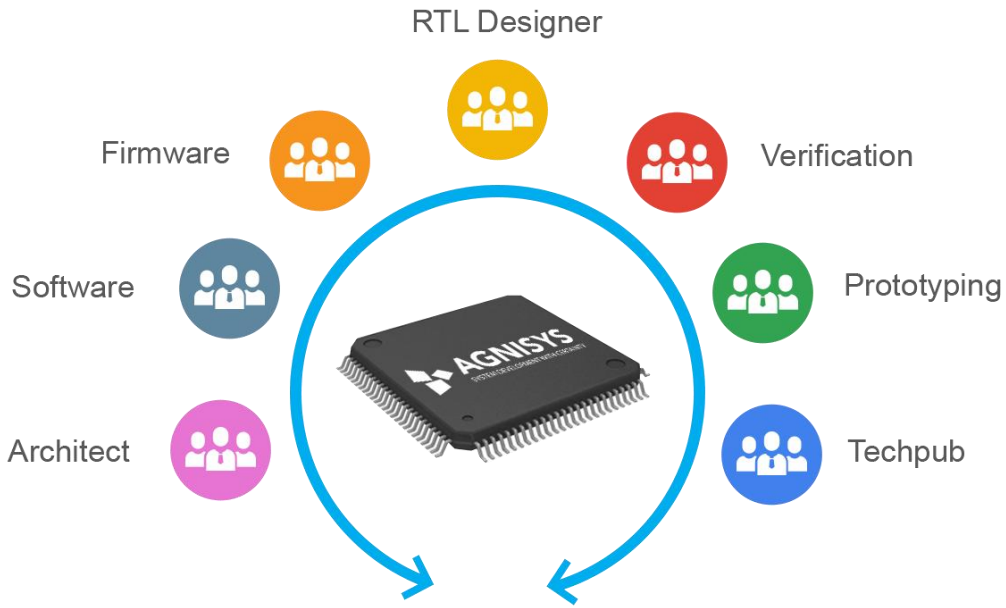
ソリューション

標準言語もしくはカスタム・データからRTL、UVM、Cコード、ヘッダファイル、ドキュメント等を自動生成

System RDL, IP-XACT, XML, YAML, RALF, CSV, Custom XML, GUI, Word, Excel, Open Office

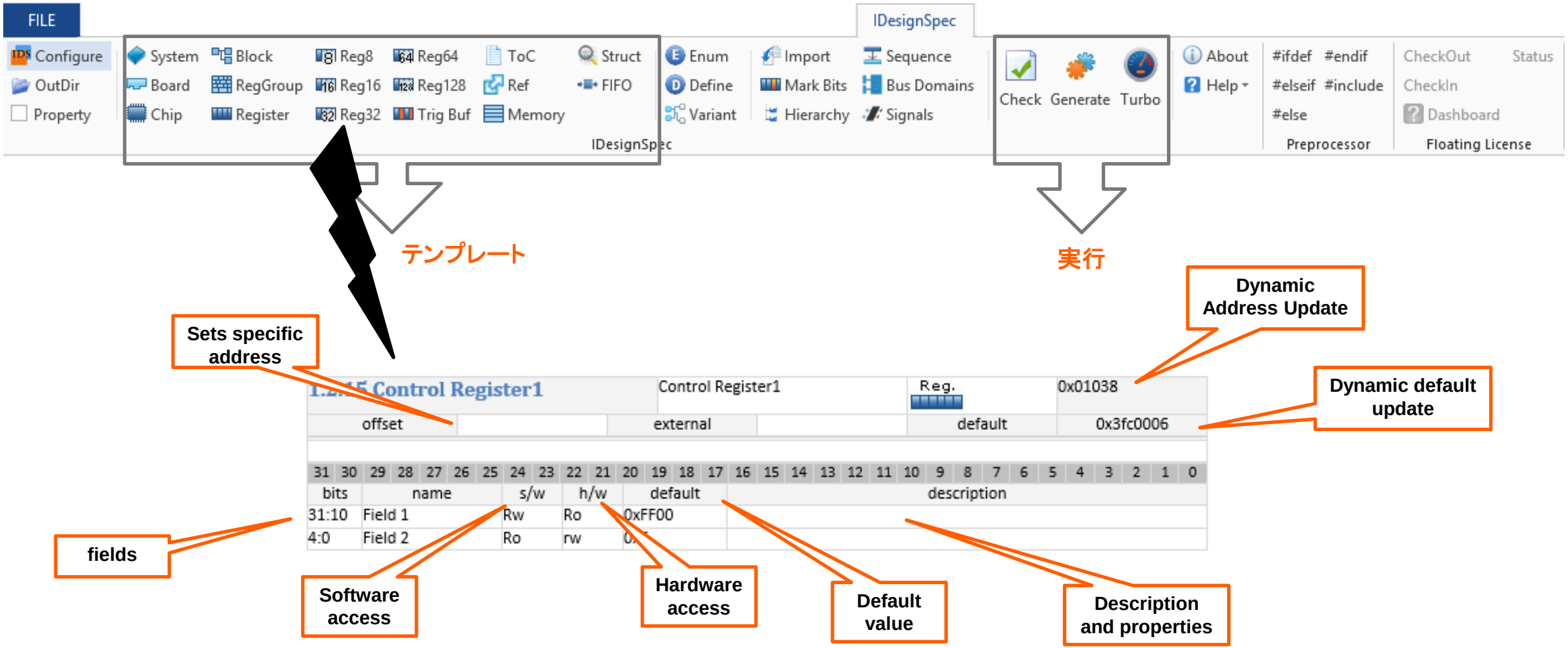


Architect	Software/ Firmware	SoC designer	Verification	Prototyping	Tech pubs	IP developer
C Header System C	C Header C/C++ API	Synthesizable Verilog, VHDL,	UVM register model, System Verilog	Perl, Python, Tcl/Velocity based custom output	PDF, Word, HTML, SoC Datasheet	IP-XACT, System RDL, Datasheet

開発チーム間の同期を維持することで、エラーを発生させない設計サイクルを実現

AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

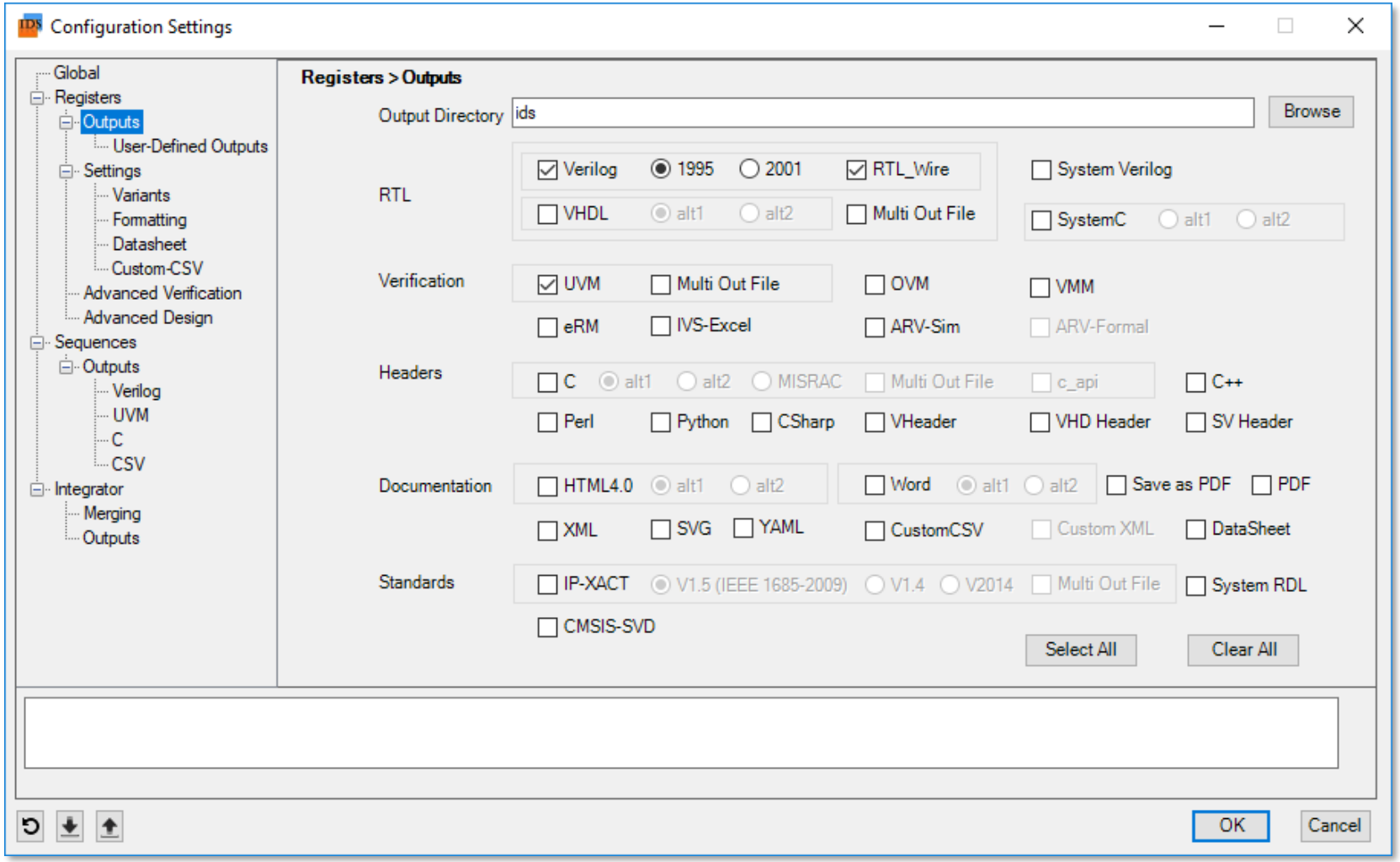
IDesignSpec：Word向けアドオン画面



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

Link to Overview

IDesignSpec：出力データ設定と生成されたコード



AGNISYS
SYSTEM DEVELOPMENT WITH CERTAINTY

自動生成されたVerilogとUVMのサンプル

```
module i2cm_ids(

    // REGISTER : ADDR PORT SIGNAL
    addr_enb,
    addr_rnw_r,

    addr_address_r,
   ...................
   ..............


    //-----------------------------------------------------------------------------
    // FIELD   : RNW
    // HW ACCESS   :  READ-ONLY                          WIDTH   :  1
    // SW ACCESS   :  READ-WRITE                         OFFSET  :  0
    //-------------------------------------------------------------
    // DESCRIPTION    :  Select bit for read and write operation.0 : Write 1 : Read
    //

    always @(posedge clk)

        begin
        if (!reset_l)
            begin


                addr_rnw_q      <=1'd0;
            end
        else
            begin

            if (addr_wr_valid)    // RNW : SW Write
                begin
                    addr_rnw_q <=  ( wr_data[0] & reg_enb[0] ) | (addr_rnw_q & (~reg_enb[0]));
                end

            end
    end // always clk
```

Verilog

```
`ifndef CLASS_i2cm_block
`define CLASS_i2cm_block
class i2cm_block extends uvm_reg_block;
    `uvm_object_utils(i2cm_block)

    rand i2cm_addr addr;
    rand i2cm_slaveregs slaveregs;
    rand i2cm_tx tx;
    ..............
    ............

    // Function : new
    function new(string name = "i2cm_block");
        super.new(name, UVM_NO_COVERAGE);
    endfunction

    // Function : build
    virtual function void build();
        //ADDR
        addr    =   i2cm_addr::type_id::create("addr");
        addr.configure(this, null, "addr");
        addr.build();

        ..............
        ..........

        //define default map and add reg/regfiles
        default_map= create_map("default_map", 'h0, 4, UVM_BIG_ENDIAN, 1);
        default_map.add_reg(addr, 'h0, "RW");
        default_map.add_reg(slaveregs, 'h4, "RW");
        default_map.add_reg(tx, 'h8, "RW");
        ............
        .....

        lock_model();
    endfunction

endclass : i2cm_block
```

UVM

自動生成されたHTMLとCヘッダファイルのサンプル

Block : i2cm

Table of Content

S.No.	Names	Default	Address
1.1	reg : addr	0x00000000	0x00
1.2	reg : slaveregs	0x00000000	0x04
1.3	reg : tx	0x00000000	0x08
1.4	reg : rx	0x00000000	0x0C
1.5	reg : ctrl	0x00000000	0x10
1.6	reg : inten	0x00000000	0x14
1.7	reg : intstat	0x00000000	0x18
1.8	reg : status	0x00000000	0x1C
1.9	reg : count_txn	0x00000001	0x20
1.10	reg : avg	0x00000000	0x24
1.11	reg : flag	0x00000000	0x28

1 : Block: i2cm 0x00

Description :

i2cm_signals:

Name	Type	Description
sda_in	In	direction : in id : 2:6 export : true width : 1 oldSignalName : sda_in

1.5 : Reg : ctrl 0x10

Description :

1.5: Reg : ctrl

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[4] singleslave
[3] srenb
[0] en
[2:1] freq

Bits	Field name	sw	hw	default	Description
0	en	rw	ro	0x0	I2c bus enable 0 : i2c OFF 1 : i2c On
2:1	freq	rw	ro	0x0	
3	srenb	rw	ro	0x0	Repeated start enable bit 0 : Disable 1 : Enable
4	singleslave	rw	ro	0x0	

1.6 : Reg : inten 0x14

Description :

1.6: Reg : inten

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

[4] ovf
[3] rs
[0] tx
[1] rx
[2] ack

Bits	Field name	sw	hw	default	Description
0	tx	rw	ro	0x0	Write complete interrupt enable editable : false
1	rx	rw	ro	0x0	Read complete interrupt enable

HTML

```c
typedef union {
    #ifdef IDS_BIG_ENDIAN
        struct {

            hwint resv8 : 24;
            hwint address : 7;          /* 7:1 SW=rw HW=ro 0x0 */
            hwint rnw : 1;          /* 0 SW=rw HW=ro 0x0 */
        } bf;

    #else     /* IDS_LITTLE_ENDIAN */
        struct {
            hwint rnw : 1;          /* 0 SW=rw HW=ro 0x0 */
            hwint address : 7;          /* 1:7 SW=rw HW=ro 0x0 */
            hwint resv8 : 24;
        } bf;

    #endif
    hwint  dw;
} i2cm_addr;

#ifdef IDS_LITTLE_ENDIAN
    #define i2cm_addr_READMASK 0xFF
    #define i2cm_addr_WRITEMASK 0xFF
    #define i2cm_addr_VOLATILEMASK 0x0
    #define i2cm_addr_RESETMASK 0xFF
    #define i2cm_addr_DEFAULT 0x00000000
#else     /* IDS_BIG_ENDIAN */

    #define i2cm_addr_READMASK 0xFF000000
    #define i2cm_addr_WRITEMASK 0xFF000000
    #define i2cm_addr_VOLATILEMASK 0x0
    #define i2cm_addr_RESETMASK 0xFF000000
    #define i2cm_addr_DEFAULT 0x00000000
#endif
```

C header