1.1 CSL Register file get methods summary

```
int get_width();
int get_depth();
string register_file_hid.get_reset_name();
string register_file_hid.get_clock_name();
string register_file_hid.get_wr_data_name();
string register_file_hid.get_rd_data_name();
string register_file_hid.get_wr_addr_name();
string register_file_hid.get_rd_addr_name();
string register_file_hid.get_wr_en_name();
string register_file_hid.get_rd_en_name();
string register_file_hid.get_valid_name();
```

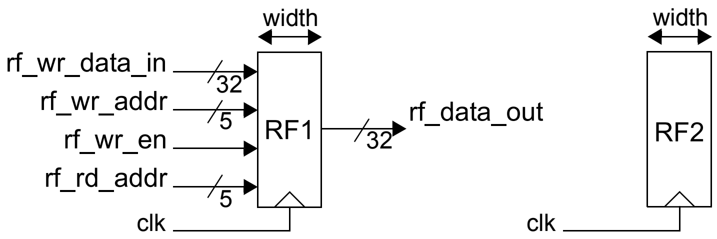
1.2 Get methods

int **get_width();**

DESCRIPTION :

It gets the width of the register file words.

FIGURE 1.1



[CSL Register File Command Summary]

EXAMPLE :

In this example we simply create two instances of a register file called *RF1* and *RF2*.

CSL CODE

```
csl_register_file RF1{
RF1(){
set_width(32);
set_depth(4);
}
};
csl_register_file RF2{
RF2(){
set_width(                 );
set_depth(4);
}
};
```

VERILOG CODE

```
int get_depth();
```

DESCRIPTION :

It gets the depth of the register file.

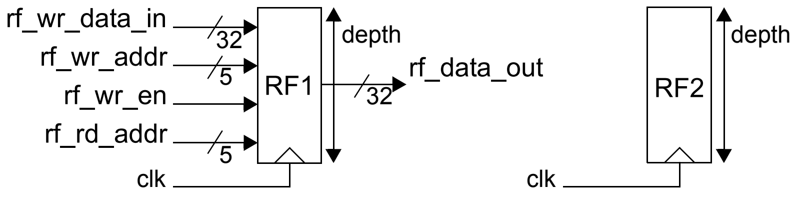


FIGURE 1.2 Register file with no special options

[CSL Register File Command Summary]

EXAMPLE :

In this example we simply create two instances of a register file called *RF1* and *RF2*.

CSL CODE

```
csl_register_file RF1{
RF1(){
set_width(32);
set_depth(4);
}
};
csl_register_file RF2{
RF2(){
set_width(RF1.get_width());
set_depth(                 );
}
};
```

VERILOG CODE

```
string register_file_hid.get_reset_name();
```

DESCRIPTION :

It gets the *name* for the reset port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of reset.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_reset_name("rst");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

```
string register_file_hid.get_clock_name();
```

DESCRIPTION :

It gets the *name* for the clock port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of the clock port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_clock_name("clk");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

```
string register_file_hid.get_wr_data_name();
```

DESCRIPTION :

It gets the *name* for the wr_data port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of wr_data port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_wr_data_name("write_data");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

`string` *`register_file_hid`*`.`**`get_rd_data_name`**`();`

DESCRIPTION :

It get the *name* for the rd_data port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of rd_data port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_rd_data_name("read_data");
 }
};
csl_unit u{
  RF RF;
  u(){}
};
```

VERILOG CODE

```
string register_file_hid.get_wr_addr_name();
```

DESCRIPTION :

It get the *name* for the wr_addr port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of wr_addr port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_wr_addr_name("write_address");
}
};
csl_unit u{
RF RF;
u(){}
};
                                ;
```

VERILOG CODE

```
string register_file_hid.get_rd_addr_name();
```

DESCRIPTION :

It gets the *name* for the rd_addr port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of rd_addr port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_rd_addr_name("read_address");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

`string` *`register_file_hid`*`.`**`get_wr_en_name`**`();`

DESCRIPTION :

It gets the *name* for the wr_en port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name wr_en port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_wr_en_name("write_enable");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

```
string register_file_hid.get_rd_en_name();
```

DESCRIPTION :

It gets the *name* for the rd_en port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of rd_en port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_rd_en_name("read_enable");
}
};
csl_unit u{
RF RF;
u(){}
};
```

VERILOG CODE

`string` *`register_file_hid`*`.`**`get_valid_name`**`();`

DESCRIPTION :

It gets the *name* for the valid port of register_file.

[CSL Register File Command Summary]

EXAMPLE :

Gets the name of valid port of register file RF.

CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);
set_valid_name("valid");
}
};
csl_unit u{
RF RF;
u(){ }
};
```

VERILOG CODE

Register file constant registers:

```
csl_reigster_file rf {
  rf() {
    set_width(32);
    set_depth(16);
    set_constant_reg(4, 345); // set rf[4] = 345
  }
};
```

Register file: associating fields and creating named registers:

Fields can be associated with a register address from the register file allowing for part select on that specific location. Also named registers can be associated with register file locations.

```
csl_register rext {
rext() {
set_width(32);
set_type(counter);
}

csl_register_file rf {
 rf(){
  set_width(32);
  set_depth(16);
  set_external_reg( 4); // rf location 4 is now driven by the external
                       // register creates an input port for rf[4] and
                        // connects the input to a mux - see AncaO's
                        // notebook
  }
};

csl_unit u {
 rext r;
 rf rf0;
 u() {
  rf0[4].connect(r);
 }
};
```

set_const_value(register_number,numeric_expression);

DESCRIPTION :

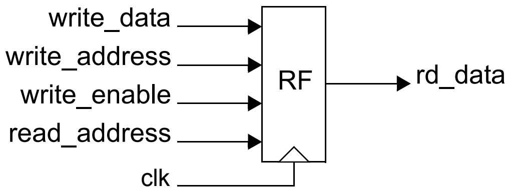
It sets the register field name within a register file to a constant value specified by the *numeric_expression.*

[CSL Register File Command Summary]

EXAMPLE :

Sets a constant value for the field *f_rf.*

FIGURE 1.3 Register file



CSL CODE:

```
csl_field f_rf(8);
csl_register_file RF{
RF(){
set_width(32);
set_depth(16);
set_const_value(4,32);

}
};
```

VERILOG CODE:

```
//
```

set_field(*field_name*,*numeric_expression***);**

DESCRIPTION :

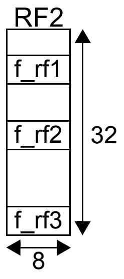
Associates the field *field_name* with the register file address specified by *numeric_expression.*

[CSL Register File Command Summary]

EXAMPLE :

Sets the fields *f_rf1, f_rf2, f_rf3* for the register file named *RF* by specify the address for each field.

FIGURE 1.4



CSL CODE

```
     csl_field f_rf1(8);
     csl_field f_rf2(8);
     csl_field f_rf3(8);

   csl_register_file rf {
      rf(){
       set_width(8);
       set_depth(32);




     }
   };
```

VERILOG CODE

```
   //verilog code
```

set_external(*numeric_expression*,*external_register_port***);**

DESCRIPTION :

Sets the register file location at *numeric_expression* as external. The second parameter specifies the external register port name that will be used to connect the outside register instead of the register file internal location.

[CSL Register File Command Summary]

EXAMPLE :

Sets the register file rf as external and specify the name of registe port.

CSL CODE

```
csl_register regx {
regx(){
set_witdth(8);
}
};
csl_register_file rf {
rf(){
set_width(8);
set_depth(32);

}
};
```

VERILOG CODE

```
//verilog code
```

`add_logic(bypass);`

DESCRIPTION :

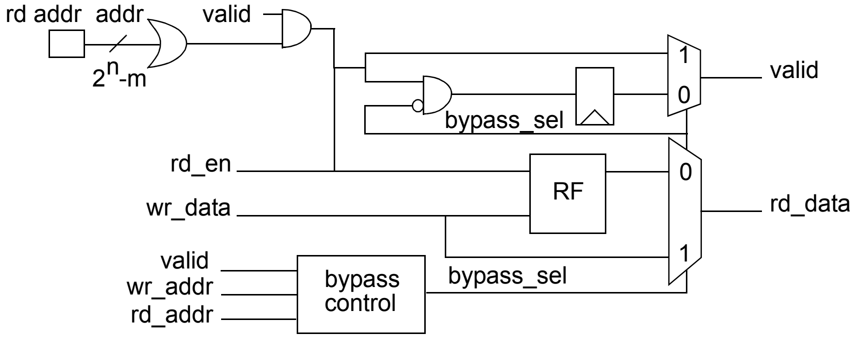
Creates the bypass using the *read_addres_signal.*

[CSL Register File Command Summary]

EXAMPLE :

When the user performs simultanously a read and a write operation at the same address into a register file,the *read_data signal* is bypassed.

FIGURE 1.5



CSL CODE:

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);

}
};
```

VERILOG CODE:

```
module
register_file_with_bypass(clk,reset,wr_en,rd_en,wr_data,rd_data,rd_add
r,wr_addr);

//list of signals and ports
input clk,reset,wr_en,rd_en;
input [1:0] rd_addr,wr_addr;
input [7:0] wr_data;
output [7:0] rd_data;
reg [7:0] rd_data;
reg [7:0] reg_file [3:0];
integer i;
```

```
//behaviour
always@(posedge clk or negedge reset or rd_en)
begin
if(!reset)
begin
for(i=0;i<4;i=i+1)
begin
    reg_file[i]=8'b0;
end
end
else
begin
    if(wr_en)
    begin
        reg_file[wr_addr]=wr_data;
    end
end
if(!rd_en)
begin
    rd_data=8'b z;
end
else
begin
    if(rd_addr!=wr_addr)
    begin
    rd_data=reg_file[rd_addr];
end
else
begin
    rd_data=wr_data;
end
end
end
 endmodule
```

set_prefix(*string*, **output|input**, **[*all*** | *register_name*.field**]);**

DESCRIPTION :

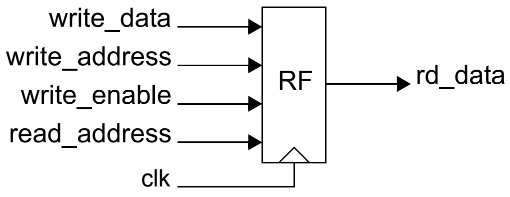
Prefixes the output names with the specified prefix.

[CSL Register File Command Summary]

EXAMPLE :

Sets the prefix “RF_out” for all the fields of the register file named RF;

FIGURE 1.6



CSL CODE

```
csl_register_file RF{
RF(){
set_width(32);
set_depth(4);

}
};
```

VERILOG CODE

```
//verilog code goes here
```

directive(field_group,_group_name_**,[all |** _register_field_name_**]);**

DESCRIPTION :

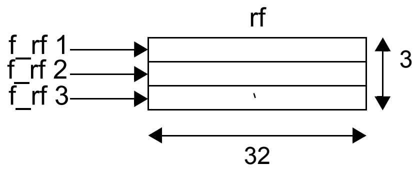
Groups the registers_fields in the argument list and name the group.You can perform different actions on the resulted group.

[CSL Register File Command Summary]

EXAMPLE :

Creates a fields group named *fld_group* which contains all the fields from register file *rf.*

FIGURE 1.7



CSL CODE

```
     csl_field f_rf1(32);
     csl_field f_rf2(32);
     csl_field f_rf3(32);

   csl_register_file rf {
   rf(){
   set_width(8);
   set_depth(32);
   set_field(f_rf1,0);
   set_field(f_rf2,1);
   set_field(f_rf3,2)
   }
   };
```

VERILOG CODE

```
   //verilog code goes here
```

directive(disconnect_register_fields_from_ios,[all | *register_field_name*]**);**

DESCRIPTION :

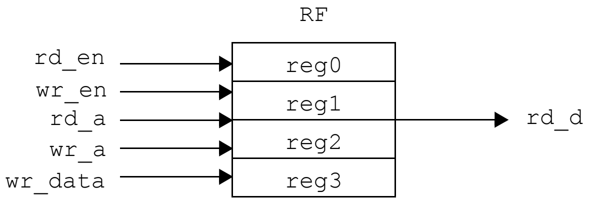
This directive is the default for the register file. All registers_fields are written by the data_in, address and wr_en signals. All registers_fields outputs are connected to the data_out, address, and are optionally read when the rd_en signal is asserted. This directive can be overriden by the connect directives below.
The connect_register flag is set and all or some register fields are specified which should not be connected to inputs and outputs.

[CSL Register File Command Summary]

EXAMPLE :

Add registers to the Register File, connect the even registers to the input-output signals (**ios**) and disconnect the old registers from ios.

FIGURE 1.8



CSL CODE

```
csl_field  reg0(1);
csl_field  reg1(1);
csl_field  reg2(1);
csl_field  reg3(1);
csl_bitrange addr_width( 2 ),data_width( 32 );
csl_signal wr_a( addr_width ),wr_d( data_width ),rd_a( addr_width
),rd_d(data_width),wr_en , rd_en;
csl_register_file RF{
RF(){
set_width( data_width );
set_depth(4);
set_field(reg0, 00);
set_field(reg1, 01);
set_field(reg2, 10);
set_field(reg3, 11);

 }};
```

VERILOG CODE

*directive(***connect_register_field_to_ios***,*[*inputs*|*outputs*|*inouts*]*,*
*[all()|reg_name][.field_name]***)***;*

DESCRIPTION :

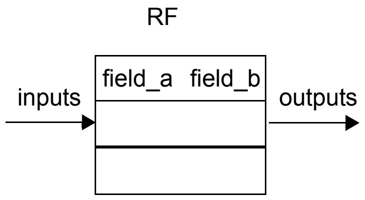
Connect the specified register_field to the input and/or the output of the register_file.

[CSL Register File Command Summary]

EXAMPLE :

In this example we illustrate how a field within a register file is connected to the register file's output.

FIGURE 1.9



CSL CODE

```
csl_field  field_a(0,3);
csl_field  field_b(4,7);
csl_signal inputs, outputs;
csl_register_file RF{
RF(){
set_width(8);
set_depth(3);
set_field(field_a);
set_field(field_b);
                                }
};
```

VERILOG CODE

```
//verilog code goes here
```

directive(name_register,*register_address,register_name***);**

DESCRIPTION :

FIX

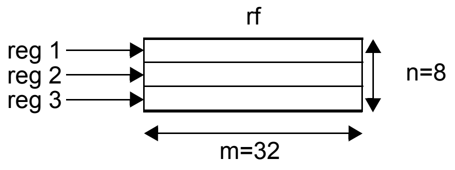
The name_register function will set the *register_name* to the read address in the register file. This operation associates the *register_field_name* with the address *address*. Note that more than one logical register can be added to the same physical address. More than one register name can be associated with the same address.
CSL generic/alternative may be used for adding a register to the memory map:

[CSL Register File Command Summary]

EXAMPLE :

Populate a Register File called *register_file_name* with registers.

FIGURE 1.10 Add registers to the Register File



CSL CODE

```
    csl_field reg1(32), reg2(32), reg3(32);
    csl_register_file rf{
    rf(){
    set_width(3);
    set_depth(32);
    name_register(register1,00);
    name_register(register2,01);
    name_register(register3,10);
    }
    };
```

VERILOG CODE