CHAPTER 1 CSL Pipeline

TABLE 1.1 Chapter Overview

1.1 CSL Pipeline Command Summary
1.2 CSL Pipeline Commands

1.1 CSL Pipeline Command Summary

CSL Pipeline

```
csl_pipeline pipeline_name(number_of_stages);
```

CSL Pipestage

```
pipestage_naming_convention(NAMING_CONV);
set_prefix(PREFIX_TYPE);
set_prefix(prefix_name);
set_suffix(suffix_TYPE);
set_suffix(suffix_name);
pipeline_name.set_number_of_pipestages(numeric_expression);
pipeline_name.set_attribute(PIPELINE_ATTR);
pipeline_name.set_type(PIPELINE_TYPE);
pipeline_name.associate_pipeline(pipeline_name1);
pipeline_name.replicate(new_pipeline_name);
csl_pipestage pipestage_object_name;
add_pipestage(pipestage_object_name);
set_previous_pipestage(pipestage_object_name,pipestage_object_name);
set_next_pipestage(pipestage_object_name,pipestage_object_name);
set_pipestage_number(n);
set_pipestage_name(name);
connect_stall(stall_signal_name0);
connect_enable(enable_signal_name0);
set_pipestage_valid_input(signal_object_name);
set_pipestage_valid_output(signal_object_name);
branch(list_of_pipestage_names);
merge(list_of_pipestage_names);
inline_file(pipestage_name, file_name);
inline_code(code_statements);
reset_init_value(init_value);
state_element.add_pipeline_delay(direction,expression);
```

1.2 CSL Pipeline Commands

The csl_pipline command creates a new pipeline object. State elements which are part of the processor pipeline are assigned the chip_pipeline object. The first pipestage number is intialized. Each subsequent pipestage is connected to the previous pipestage using the set_previous_pipestage method. Previous pipestage can be connected to subsequent pipestages using the set_next_pipestage method. Pipestages are either automtaically assigned a pipestage number based on the previous pipestage number incremented by one or are explicitly assigned a new pipestage number. Pipestages can be named. The pipestage number and/or the pipestage name may be used in the generated Verilog variable names. The use of the pipestage name and number is controlled by the use_pipestage_name and use_pipe_stage_number methods.
Create a new pipestage. The pipestage is attached to a pipeline. The previous and next piestages are set. The number of the pipestage is automatically based on the value of the previous pipestage. The default number for the first pipestage int he pipeline is 0. The pipestage number of a pipestage can be set explicitly.
All output signals connected to state elementss in each pipestage are optionally prefixed or suffixed with the name _<*pipeline_name*><*stage_name*><*pipestagenumber*>. The sufffix naming convention can be overridden.
If a stall signal or enable signal is used to control the state elements in the pipeline then all of the state elements in the pipeline must have an enable signal.

Pipelines which fork and join can be constructed with the **`set_next_pipestage (branch) and the set_previous_pipestage (merge) methods.`**

CSL Pipeline:

csl_pipeline *pipeline_name(number_of_stages)*;

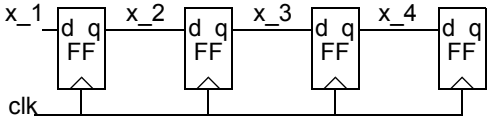
DESCRIPTION :

Create a new pipeline. All signals connected to flip flops in each pipestage are prefixed with the name Each set of pipe stage signals is suffixed with"*_number*" where number is the number of the pipe stage. Pipe stages are added to the pipeline object. The pipe line object can then be checked for various correctness conditions such as correct stall logic and correct signal connections.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.1 A pipeline with four pipestages



CSL CODE

```
csl_pipeline pipeline_name(4);
```

VERILOG CODE

CSL Pipestage:

`pipestage_naming_convention(NAMING_CONV);`

DESCRIPTION :

Use for naming convention one of the enums in NAMING_CONV.

[***CSL Pipeline Command Summary***]

EXAMPLE :

//

CSL CODE

//

VERILOG CODE

//

```
set_prefix(PREFIX_TYPE);
```

DESCRIPTION :

Override the prefix naming convention using one of the enums in **PREFIX_TYPE**:

```
enum {NO_PREFIX, PIPELINE_NAME, PIPESTAGE_NAME,
PIPELINE_PIPESTAGE_NAME} PREFIX_TYPE;
```

[***CSL Pipeline Command Summary***]

EXAMPLE :

TABLE 1.2

enum	Description	Example
NO_PREFIX		
PIPELINE_NAME		
PIPESTAGE_NAME		
PIPELINE_PIPESTAGE_NAME		

Show the different examples:

- pipeline_name
- pipestage_name
- pipeline n

CSL CODE

```
//csl code goes here
```

set_prefix(*prefix_name***);**

DESCRIPTION :

Override the prefix naming convention using the string *prefix_name*.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
    //csl code goes here
```

set_suffix(suffix_TYPE);

DESCRIPTION :

Override the suffix naming convention using one of the enums in **suffix_TYPE**:

```
enum {NO_SUFFIX, PIPELINE_NAME, PIPESTAGE_NAME,
PIPELINE_PIPESTAGE_NAME} SUFFIX_TYPE;
```

[***CSL Pipeline Command Summary***]

EXAMPLE :

TABLE 1.3

enum	Description	Example
NO_SUFFIX		
PIPELINE_NAME		
PIPESTAGE_NAME		
PIPELINE_PIPESTAGE_NAME		

Show the different examples:

- pipeline_name
- pipestage_name
- pipeline n

CSL CODE

```
//csl code goes here
```

set_suffix(*suffix_name***);**

DESCRIPTION :

Override the suffix naming convention using the string *suffix_name*.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
//csl code goes here
```

pipeline_name.**set_number_of_pipestages(***numeric_expression***);**

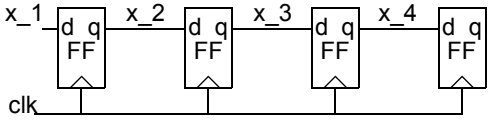
DESCRIPTION :

Set the total number of pipe stages in the pipeline named This method is used by the pipeiine checker to verify that the pipieline is less than or equal to *numeric_expression* pipestages in length.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.2 A pipeline with four pipestages



CSL CODE

```
csl_pipeline pipe;
pipe.set_number_of_pipestages(4);
```

VERILOG CODE

pipeline_name.**set_attribute(PIPELINE_ATTR);**

DESCRIPTION :

Set the pipeline attributes using one of the enums in **PIPELINE_ATTR**:

```
enum {NO_STALL, STALL } PIPELINE_ATTR;
```

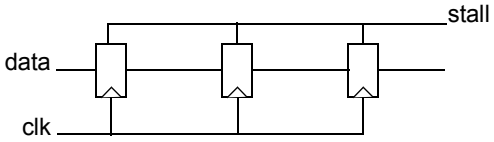
[***CSL Pipeline Command Summary***]

TABLE 1.4

PIPELINE_ATTR	Description
NO_STALL	
STALL	

EXAMPLE :

FIGURE 1.3 A pipeline with stall signal



CSL CODE

```
csl_pipeline pipe(3);
pipe.set_attribute(STALL);
```

VERILOG CODE

pipeline_name.**set_type(PIPELINE_TYPE);**

DESCRIPTION :

Set the pipe stage type using one of the enums in **PIPELINE_TYPE**:

```
enum {VALID, DATA, CONTROL, OTHER} PIPELINE_TYPE;
```

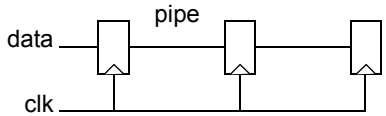
[***CSL Pipeline Command Summary***]

TABLE 1.5

PIPELINE_TYPE	Description
VALID	
DATA	
CONTROL	
OTHER	

EXAMPLE :

FIGURE 1.4 A data pipeline



CSL CODE

```
csl_pipeline pipe(3);
pipe.set_type(DATA);
```

VERILOG CODE

pipeline_name.**associate_pipeline(***pipeline_name1***);**

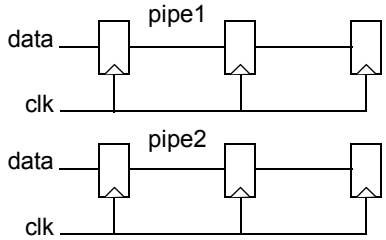
DESCRIPTION :

Associate the pipeline object named *pipeline_name* with the pipeline named *pipeline_name1*. The two pipelines should have the same number of pipe stages, control signals, muliplexers with the same relative pipe stage inputs driving the same relative pipe stages and. Differences between the two pipe stages are flagged by the cslc pipeline checker.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.5



CSL CODE

```
csl_pipeline pipe1(3);
csl_pipeline pipe2(3);
pipe1.associate_pipeline(pipe2);
```

VERILOG CODE

pipeline_name.**replicate(***new_pipeline_name***);**

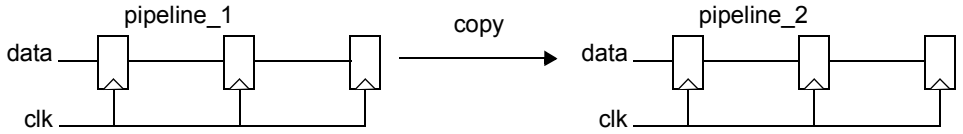
DESCRIPTION :

Create a copy of the pipeline. The new pipeline name *new_pipeline_name* is an object which can have its pipeline type modified. For example, a pipeline for the valid bits can be created and then replicated. The replicated pipeline can be set to a control pipeline. Then the control pipeline can be replicated into a new pipeline and the new pipeline can have its type set to data.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.6



CSL CODE

```
csl_pipeline pipeline_1(3);
pipeline_1.replicate(pipeline_2);
```

VERILOG CODE

csl_pipestage *pipestage_object_name*;

DESCRIPTION :

Create a new pipestage. The pipestage is attached to a pipeline. The previous and next pipestages are set. The number of the pipestage is automatically based on the value of the previous pipestage. The default number for the first pipestage int he pipeline is 0. The pipestage number of a pipestage can be set explicitly.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipestage p3;
```

VERILOG CODE

add_pipestage(*pipestage_object_name*);

DESCRIPTION :

Create a new pipestage.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe;
csl_pipestage p0;
mp_pipe.add_pipestage(p0);
csl_pipestage p1;
mp_pipe.add_pipestage(p1);
csl_pipestage p2;
mp_pipe.add_pipestage(p2);
```

VERILOG CODE

set_previous_pipestage(*pipestage_object_name,pipestage_object_name*);

DESCRIPTION :

Set the previous pipestage of the current pipestage.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe;
csl_pipestage p0;
mp_pipe.add_pipestage(p0);
csl_pipestage p1;
mp_pipe.add_pipestage(p1);
csl_pipestage p2;
mp_pipe.add_pipestage(p2);
p3.set_previous_pipestage(p2);
p2.set_previous_pipestage(p1);
p1.set_previous_pipestage(p0);
```

VERILOG CODE

set_next_pipestage(*pipestage_object_name,pipestage_object_name*);

DESCRIPTION :

Set the next pipestage of the current pipestage.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe;
csl_pipestage p0;
mp_pipe.add_pipestage(p0);
csl_pipestage p1;
mp_pipe.add_pipestage(p1);
csl_pipestage p2;
mp_pipe.add_pipestage(p2);
p3.set_next_pipestage(p2);
p2.set_next_pipestage(p1);
p1.set_next_pipestage(p0);
```

VERILOG CODE

set_pipestage_number(*n*);

DESCRIPTION :

Set the pipestage number belonging to This will be the pipe stage number for the pipe stage. If this method is not used then the default for the pipestage is 0 ("_0") if this is the first piestage and each subsequent pipestage is assigned the auto-incremented pipestage number(n + 1) **. The pipeline checker verifies that all pipestage numbers are unique.**

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe;
mpo_pipe.set_pipestage_number(2);
```

set_pipestage_name(*name*);

DESCRIPTION :

The name of the pipestage belonging to Each subsequent pipestage uses the autoincremented pipestage name(n + 1) unless the pipestage name is set using the **set_pipestage_name** method. The pipeline checker verifies that all pipestage names are unique.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe();
mpo_pipe.set_pipestage_name("stage1");
```

connect_stall(*stall_signal_name0***);**

DESCRIPTION :

Connect a stall signal to a pipeline. The stall is inverted before being connected to the FF enable. The stall signal controls the enable fo the state elements in the pipeline.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe();
csl_signal stall;
mpo_pipe.connect_stall(stall);
```

connect_enable(*enable_signal_name0***);**

DESCRIPTION :

Connect a enable signal to a pipeline. The enable is inverted before being connected to the FF enable. The enable signal controls the enable fo the state elements in the pipeline.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
csl_pipeline mp_pipe();
csl_signal enable;
mpo_pipe.connect_enable(enable);
```

set_pipestage_valid_input(signal_object_name);

DESCRIPTION :

//

[***CSL Pipeline Command Summary***]

EXAMPLE :

//

CSL CODE:

//

VERILOG CODE:

set_pipestage_valid_output(signal_object_name);

DESCRIPTION :

//

[***CSL Pipeline Command Summary***]

EXAMPLE :

//

CSL CODE:

VERILOG CODE:

branch(*list_of_pipestage_names***);**

DESCRIPTION :

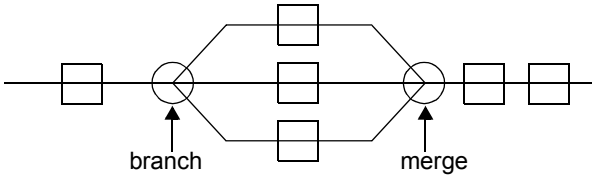
Pipeline branch specifies that the pipe stages named *list_of_pipestage_names* are all part of a pipeline branch. The valid pipe that is assocated with the pipeline has the valid bits qualified by the pipeline mux selects.

[***CSL Pipeline Command Summary***]

USE set_preivous_pipestage and set_next_pipestage instead of branch and merge
Use the example below with set_preivous_pipestage and set_next_pipestage methods

EXAMPLE :

FIGURE 1.7



CSL CODE

```
//csl code goes here
```

merge(*list_of_pipestage_names***);**

DESCRIPTION :

Pipeline merge specifies that the pipe stages named *list_of_pipestage_names* are all part of a pipeline merge. Each branch which is part of the merge has previously been specified to be part of a pipeline branch. The merge mechanism is a mulitplexer which is controlled by a mux select line. The mux select line is used to create the valid bit qualifiers for the entry point into each pipe stage in the branch.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
//csl code goes here
```

inline_file(*pipestage_name, file_name***);**

DESCRIPTION :

Insert the contents of the named file *file_name* after the pipestage named *pipestage_name* in the generated verolog code.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
//csl code goes here
inline_file(fn)
```

VERILOG CODE

```
module
  generated pipestage code
  `include "fn
```

inline_code(*code_statements***);**

DESCRIPTION :

Insert the code after the pipestage.

[***CSL Pipeline Command Summary***]

EXAMPLE :

CSL CODE

```
//csl code goes here
```

VERILOG CODE

```
`include "fn
```

reset_init_value(*init_value***);**

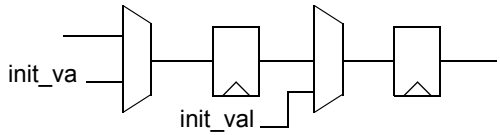
DESCRIPTION :

At reset initialize all flip flops in the pipeline with the value *init_value*.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.8



CSL CODE

```
//csl code goes here
```

state_element.**add_pipeline_delay**(direction,expression);

DESCRIPTION :

//

[***CSL Pipeline Command Summary***]

EXAMPLE :

We add a pipeline to the output of the unit with the delay of 1

CSL CODE:

```
csl_register_file rf;
rf.add_pipeline_delay(output,1);
```

VERILOG CODE:

match_pipeline_latency(in0, outn, new_pipeline_name, in10, out1n);

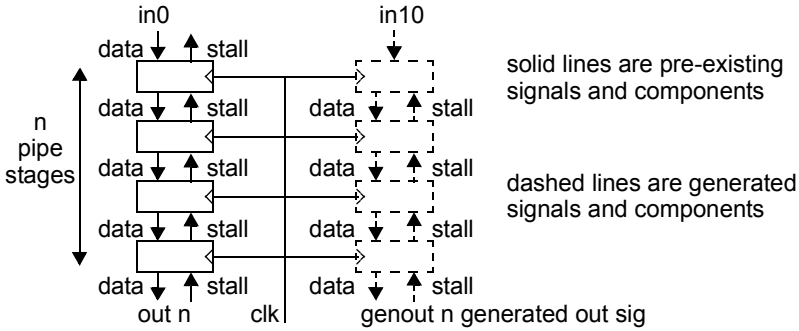
DESCRIPTION :

Pipeline latency match generator
Pipeline latency checker
The pipeline will measure the number (**n**) of pipestages between two specified signals in an existing pipeline and generate **n** pipe stages in a new pipeline. The import signal to the **n** pipestages is specified. The clk signal and enable signals for each stage are extracted from the original set of pipestages.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.9



Checks that the two pipelines are the same length. If the pipelines branch and merge the checker will follow all paths & check all paths

```
get_pipeline_latency(in0, outn);
```

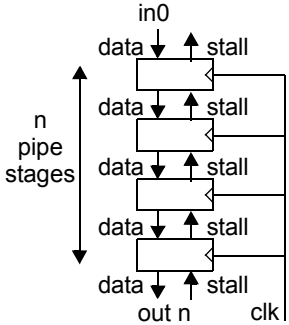
DESCRIPTION :

Return the number of pipestages between the 2 signals.

[***CSL Pipeline Command Summary***]

EXAMPLE :

FIGURE 1.10



If the pipelines branch and merge the checker will follow all paths & check all paths.