methodics™

ProjectIC Based SoC Integration Workflow

SoC Integration is a challenging and important task that all hardware teams undertake on a regular basis.

A SoC is typically made up of series of IPs, each of which could also be a sub-system in its own right. The key role of integration is to accept new releases of the component IPs and/or sub-systems, verify whether these newer releases work in the context of the SoC.

The releases that are available for integration should have passed some quality control of their own - i.e. basic checks that ensure that the release that is available for integration satisfies a minimum quality level, and moves the SoC forward.

Integration Roles

Integration roles can be defined as follows:

Integrator

The key role in the integration flow is that of the integrator. The integrator decides how and when to pick up new releases of component IPs and sub-systems that are available.

As part of the integrator's role, she needs the following capabilities:

- Indication of when a new release of each of the component IPs is available for integration. This information should be easily available.
- Quickly and seamlessly introduce any subset of the available releases into an existing workspace - the new SoC configuration under test
- Quickly and seamlessly revert any subset of the newly introduced releases from an existing workspace - tweak the new SoC configuration
- Launch the required tests in the workspace to qualify the configuration with new releases
- Easily capture the tested configuration as the new top-level configuration for the SoC - release the SoC with new IPs after test

Contributor

Contributors are IP and sub-system owners that provide new releases to the integrator for integration. These new releases can be generated to fix bugs and/or add new features as required by the SoC.

As part of the Contributor's role, he needs the following capabilities:

- Generate a new release of an IP or sub-system after making the modifications required
- Guarantee the minimum quality level of these releases based on the commonly agreed terms for the team or SoC. This is done by testing the release in its context before notifying the integrator of the availability of a new release.
- An easy way to indicate the availability of a new release for integration

Consumer

Consumers of the SoC use qualified, integrator blessed versions for other flows - for example to run downstream flows like power analysis or synthesis. These users need the following capabilities:

- An SoC configuration that is known to have been qualified by the integrator
- An indication when a new version of the configuration has been qualified by the integrator.

Integration Flow

ProjectIC uses the notion of 'Moving Aliases' to implement the integration flow. An IP or subsystem release can be aliased with a string that has some meaning - for example, 'SOC_READY' for a release that is SOC_READY for integration, 'GOLD' for a integration qualified release - as a communication channel to accomplish integration.

In this example, we will use the 'SOC_READY' alias for component IPs when they are SOC_READY for integration, and a 'GOLD' alias when the integrator has blessed an IP as qualified after running her tests.

Additionally, we will use two top level SoC definitions for this flow:

1. the 'lib.soc_integ' top level configuration (a container in ProjectIC terms) is used by the integrator to run her integration step. This top level configuration uses IPs at the 'SOC_READY' alias, as shown:

```
[integrator] pi ip list lib.soc_integ -v
 IP lib.soc_integ@4.TRUNK (LATEST):
     Description          - Top level container for Integration
     DM type              - CONT
     Resources            - coco.CPU_Cache@SOC_READY.TRUNK
                            coco.CPU_Core@SOC_READY.TRUNK
                            coco.CPU_Logic@6.TRUNK
 ...
```

2. the 'lib.soc_top' top level configuration is used by consumers to pick up integrator qualified configurations. This top level configuration is shown below:

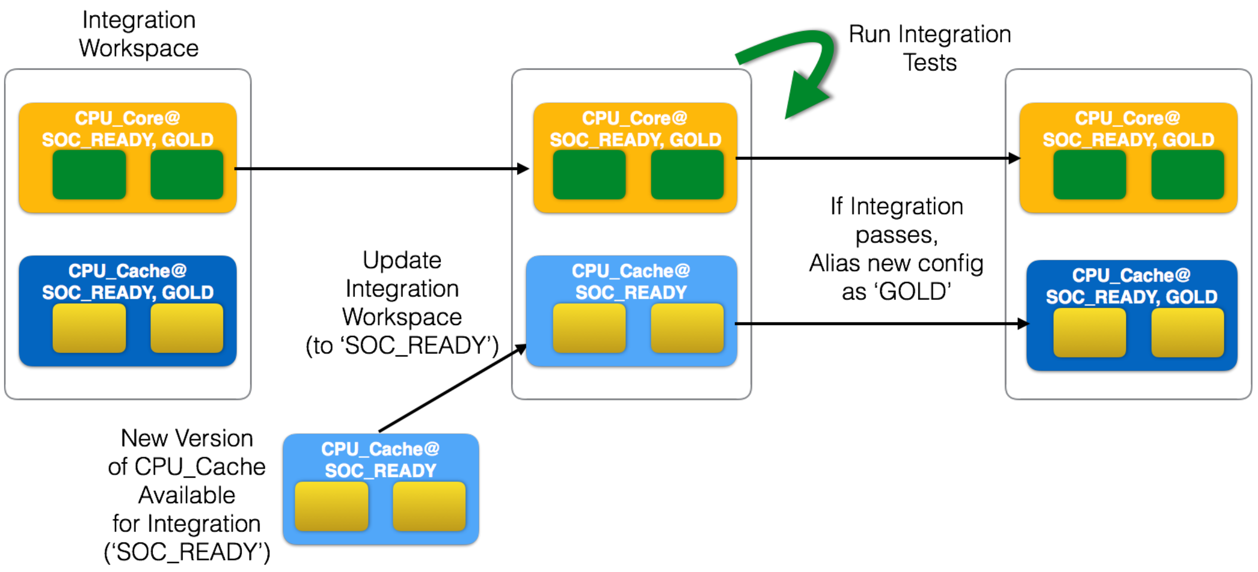
```
[integrator] pi ip list lib.soc_top -v

 IP lib.soc_integ@4.TRUNK (LATEST):
     Description          - Top level container for Downstream Consumers
     DM type              - CONT
     Resources            - coco.CPU_Cache@GOLD.TRUNK
                            coco.CPU_Core@GOLD.TRUNK
                            coco.CPU_Logic@6.TRUNK
```

Some things to note in the above definitions:

- The coco.CPU_Logic IP is not expected to change during this project. Hence, that particular IP is used at a fixed release (coco.CPU_Logic@6). A container can include resource IPs and sub-systems at both fixed releases or at moving aliases.
- When workspaces are built with these top levels, all the IPs dependencies of these sub-systems are also automatically brought into that workspace. Thus workspace status generally contain several more IPs than the ones included as direct resources in the top level.

- The top level includes 3 IPs, but each of these IPs is really a subsystem, which includes IPs of its own. ProjectIC understands hierarchy across all its functions, so this is never a limitation.

The integration flow is indicated in the diagram below.


Integration Workspace
CPU_Core@ SOC_READY, GOLD
CPU_Cache@ SOC_READY, GOLD
Update Integration Workspace (to 'SOC_READY')
New Version of CPU_Cache Available for Integration ('SOC_READY')
CPU_Cache@ SOC_READY
CPU_Core@ SOC_READY, GOLD
CPU_Cache@ SOC_READY
Run Integration Tests
If Integration passes, Alias new config as 'GOLD'
CPU_Core@ SOC_READY, GOLD
CPU_Cache@ SOC_READY, GOLD

Integration starts with the integrator creating a workspace with the 'soc_integ' project.

As shown in the tree display above, the SoC uses two of its component IPs - coco.CPU_Core and coco.CPU_Cache - on an alias called 'SOC_READY' . What this implies is that instead of fixing the versions of these IPs, the SoC will use the latest versions that are marked 'SOC_READY'. Using these moving aliases is a critical communication mechanism between the contributors and the integrator.

The integrator can load this SoC into a workspace using the ProjectIC IP load command:

```
[integrator]% pi ip load lib.soc_integ integ1
```

The 'integ1' directory now contains all the IPs and sub-systems of the SoC. These IPs and sub-systems are created as links to a common 'IP Cache' for speed and disk-space optimization. Please refer to ProjectIC documentation for more on the IP Cache. The snippet below shows the composition of this integration workspace as a set of links. Each directory is an IP in the listing.

```
IEC_ARM_coco.SY750 -> /mdx_local/mdx_data/bic_shares/IEC_ARM_coco/SY750/TRUNK/5/IEC_ARM_coco.SY750
 coco.CPU_Cache -> /mdx_local/mdx_data/bic_shares/coco/CPU_Cache/TRUNK/GOLD/coco.CPU_Cache
 coco.CPU_Core -> /mdx_local/mdx_data/bic_shares/coco/CPU_Logic/TRUNK/6/coco.CPU_Core
 coco.CPU_Logic -> /mdx_local/mdx_data/bic_shares/coco/CPU_Logic/TRUNK/6/coco.CPU_Logic
 Memories.CS-TS28N-GS-1PSRM -> /mdx_local/mdx_data/bic_shares/Memories/CS-TS28N-GS-1PSRM/TRUNK/5/Memories.CS-TS28N-GS-
1PSRM
```

The integrator can now query the workspace to find out if there are any newer releases available for integration. At this time, there are no new releases that are marked 'SOC_READY' for integration.

```
[integrator]% pi workspace status -v
 Workspace ID: 16
 Directory: /home/vishal/workspaces/integ1
 IP: lib.soc_integ@4.TRUNK
 Resources:

| Name                          | Expected Version   | Local Version  | WS Status  | Server Status  | Mode         |
| lib.soc_integ                 |     4.TRUNK        |    4.TRUNK     |    OK      |      OK        | Container    |
| IEC_ARM_coco.SY750            |     5.TRUNK        |    5.TRUNK     |    OK      |      OK        | Refer        |
| Memories.CS-TS28N-GS-1PSRM    |     5.TRUNK        |    5.TRUNK     |    OK      |      OK        | Refer        |
| coco.CPU_Cache                |  SOC_READY.TRUNK   |    8.TRUNK     |    OK      |      OK        | Refer        |
| coco.CPU_Core                 |  SOC_READY.TRUNK   |    8.TRUNK     |    OK      |      OK        | Refer        |
| coco.CPU_Logic                |     6.TRUNK        |    6.TRUNK     |    OK      |      OK        | Refer        |
```

A contributor can now generate a new release of an IP, and mark it 'SOC_READY'. As part of the release, the contributor runs a quality check to make sure that the release meets the required quality criteria. These quality checks can be run automatically in the user's workspace as part of the release process by using ProjectIC's pre- and post-release hooks. In the example below, the pre-release hook runs a compilation check to make sure that the released files compile.

```
[contributor]% pi release coco.CPU_Core -d "Added feature for SoC"
 INFO:pi:Calling pre-release hook for coco.CPU_Core@8.TRUNK
 INFO:pi:    /mdx/scripts/hooks/project_pre_release $IP_DIR
 Running pre-release checks
 ~~~~ Verilog Compile Check ~~~~
 ./coco.CPU_Core/rtl/test.v
 ./coco.CPU_Core/rtl/mt48lc32m16a2.v
 ./coco.CPU_Core/test/test.v
 ./coco.CPU_Core/formal/test.v
 ./coco.CPU_Core/verif/test.v
 ./coco.CPU_Core/netlist/test.v
 ./coco.CPU_Core/Cmodel/test.v
 ~~~~~~~~~~~~~~~
 Compilation Successful!
 ~~~~~~~~~~~~~~~
 INFO:pi:Successfully created coco.CPU_Core@9.TRUNK
```

Once the release is created, the contributor can then mark this release as 'SOC_READY' for integration. Note that this release was 'coco.CPU_Core@9.TRUNK'.

```
[contributor] pi ip aliasset coco.CPU_Core@9.TRUNK SOC_READY
 Success, alias 'SOC_READY' set on 'coco.CPU_Core@9.TRUNK'.
```

Now, the integrator can query his workspace to find out if there are any new releases available for integration.

```
[integrator] pi workspace status -v
 Workspace ID: 20
 Directory: /home/mdx/workspaces/s1
 IP: lib.soc_integ@4.TRUNK
 Resources:

 | Name                          | Expected Version   | Local Version  | WS Status  | Server Status  | Mode         |
 | lib.soc_integ                 |     4.TRUNK        |    4.TRUNK     |    OK      | New Resources  | Container    |
 | IEC_ARM_coco.SY750            |     5.TRUNK        |    5.TRUNK     |    OK      |      OK        | Refer        |
 | Memories.CS-TS28N-GS-1PSRM    |     5.TRUNK        |    5.TRUNK     |    OK      |      OK        | Refer        |
 | coco.CPU_Cache                |  SOC_READY.TRUNK   |    8.TRUNK     |    OK      |      OK        | Refer        |
 | coco.CPU_Core                 |  SOC_READY.TRUNK   |    8.TRUNK     |    OK      | New @SOC_READY | Refer        |
 | coco.CPU_Logic                |     6.TRUNK        |    6.TRUNK     |    OK      |      OK        | Refer        |
 Recommendation: Update spark.soc
```

As can be seen from the output of the workspace status command, the integrator has the indication that a new component IP is available for integration. This workspace can now be updated using the projectIC update commands to bring in this new release into the workspace.

Either as part of the update process (using the post-update hook), or as a separate step, the integrator can run her integration checks. These checks are typically run as tests at the top level. Depending on the results of these tests, the integrator can mark this new configuration as 'GOLD', or leave the previous configuration as 'GOLD'.

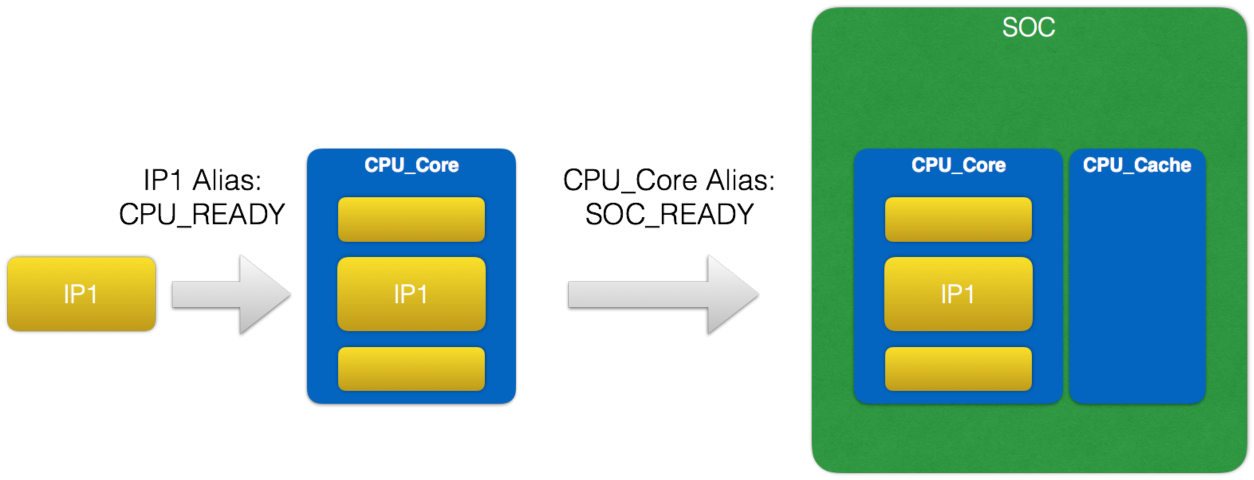
In the example below, some tests are run as part of the update step (using the post-update hook on lib.soc_integ) and as a result, the IP is automatically checked for integration issues and qualified.

```
[integrator] pi update
 INFO:pi:Updating workspace to LATEST. For large IPs this could take a while.
 INFO     : Preparing update ...
 INFO     : Reading BOM: /home/mdx/workspaces/s1/.methodics/ws.bom
 INFO     : Setting up jobs for 2 IPs
 INFO     : Submitted 2 server jobs
 INFO     : Waiting for 2 server jobs
 INFO     :  - use Ctrl-C to stop waiting: the jobs will keep running
 INFO     : Server jobs done

 -----------------------------  SUMMARY  ---------------------------------
 BLOCK           RELEASE  SERVER JOB  LOCAL JOB  MODE   RELPATH
 -----           -------  ----------  ---------  ----   -------
 coco.CPU_Cache  SOC_READY    success     None       refer  coco.CPU_Cache
 coco.CPU_Core   SOC_READY    success     None       refer  coco.CPU_Core

 2 server jobs succeeded
 -------------------------------------------------------------------------
 INFO:pi:Calling post-update hook for spark.soc@4.TRUNK
 INFO:pi:    /mdx_local/tools/scripts/integ_check
 Running Integration Tests...
 Tests complete!
 Results:
 --------
 Total tests: 10
 Tests Run: 9
 Tests Passed: 9
 Tests Failed: 0
 Integration Passed!
 Setting 'GOLD' on coco.CPU_Cache@8.TRUNK...
 Setting 'GOLD' on coco.CPU_Core@9.TRUNK...
  -
 Update summary:
     IP                              Old                    New
     coco.CPU_Core                   SOC_READY.TRUNK [@8]   SOC_READY.TRUNK [@9]
```

Hierarchical Integration

Integration can be performed at any level. For example, if coco.CPU_Core has IPs of its own, then contributors can release IPs to be integrated into lib.CPU_Core using a similar scheme. Once new IPs are integrated into lib.CPU_Core, the resulting release can be marked for integration at the SoC level and so on.

Each IP and sub-system can define its own 'ready for integration' aliases. An example is shown below:



As shown, an IP (IP1) can be integrated by the CPU_Core, which in turn can be integrated into the SoC - all of this is done using the moving aliases flow described above.

For more information on this topic, or on any of Methodics products and solutions, please email contact@methodics.com.