VISION SDK
Use-Case Auto-Generation Tool Overview

July 2016

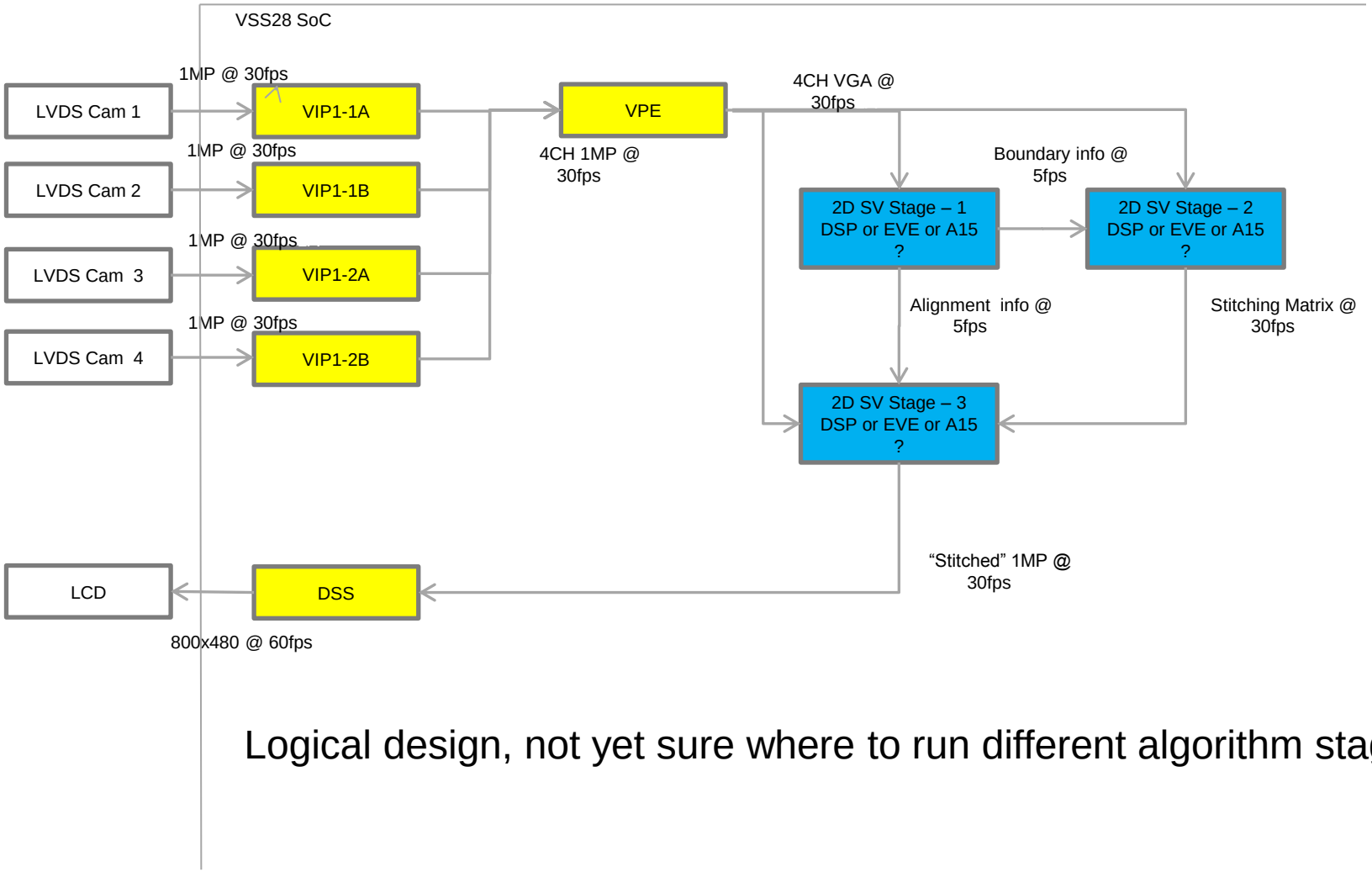
Agenda

- Introduction
- Motivation
- Use-case Generation process
- Error Handling cases
- Extending the tool - Ease for Developer

What is Vision SDK ?

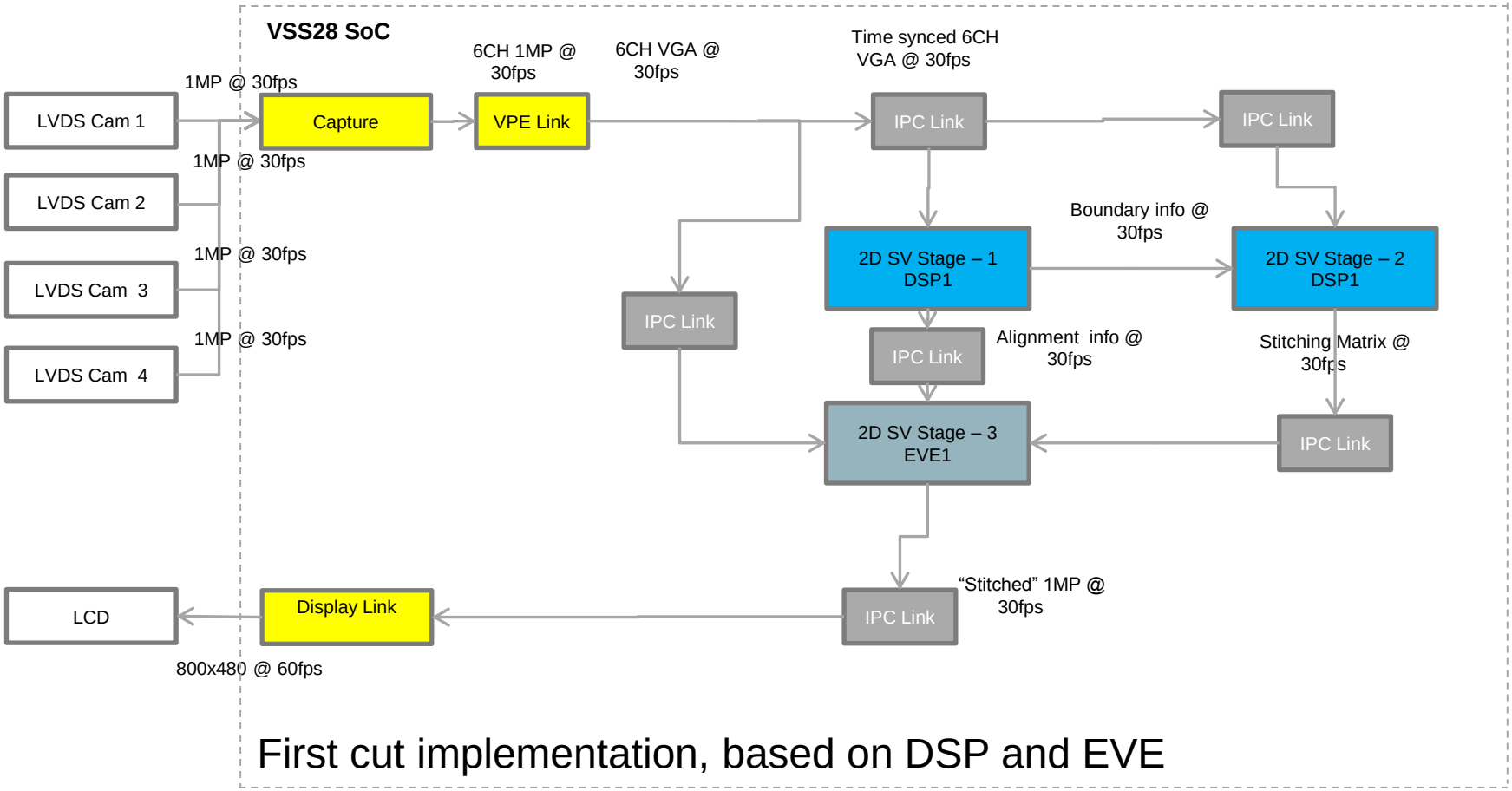
- VISION SDK is multi processor software development platform for TI family of ADAS SoCs.
- The software framework allows users to create different ADAS application data flows involving video capture, video pre-processing, video analytics algorithms, and video display.
- The SDK has sample ADAS data flows which exercises different CPUs and HW accelerators in the ADAS SoC and shows customers how to effectively use different SoC sub-systems.
- VISION SDK will be based on a framework named as the “Links and Chains” framework and user API to this framework is called “Link API”
- The SDK installer package includes all tools and components necessary to build such applications, including code gen tools, BIOS, IPC, starterware, BSP drivers, networking stacks, codecs, algorithm kernels

Example ADAS data flow – LVDS surround view – system diagram



Logical design, not yet sure where to run different algorithm stages

Example ADAS data flow – LVDS surround view in Vision SDK



IPU1 Core0 | IPU1 Core1 | DSP1 | DSP2 | EVE1 | EVE2 | A15

Normal Use-Case Generation Process

- User has to determine the flow of connections
- C code has to be written for the use-case which
  - Set LinkID and Parameters
  - Insert IPC links when buffers are exchanged across CPUs
  - Connect Links
  - Create links in sequence of source to sink
  - Write functions for use-case Create, Start, Stop, Delete (PrintBufferStatistics, Print Link Statistics)

Normal Use-Case Generation Process: Drawbacks

- Typically usecase files have around 1000+ lines
- Writing a given usecase requires a lot of time
  - It took 1 week to write the surround view use-case file
- This method is more error prone
  - It took one week to make sure surround view use-case create, start, stop, delete passes
  - Common mistakes include – this would take 1 week to resolve on HW board
    - Wrong connection of previous link ID to next link ID
    - Wrong link ID setting
    - Wrong insertion of IPC link

Motivation

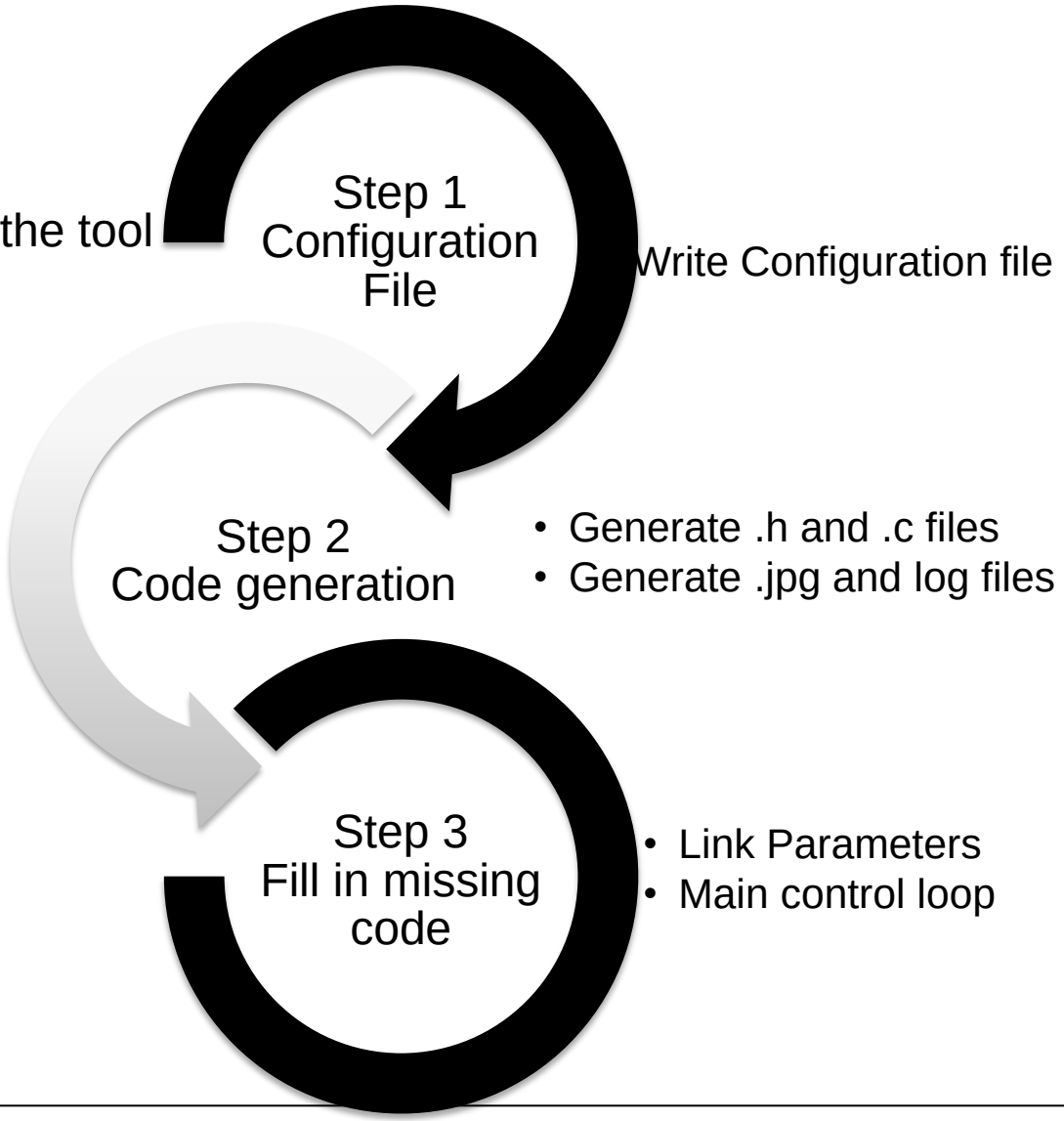
- Help end user generate use-case in minimum time
- Give overall summary of use-case code using visual representation of data flow via images and logs
- Help user focus to main problem, i.e. determine flow of connections
- Detect and report errors during use-case generation stage itself and therefore reduce time required to debug on actual HW board
- Takes very less effort for changes in data flow. For e.g. change in Algorithm, or adding a new algorithm in a existing data flow

Example Results

- Single Capture Display Use-case:
  - Generated 210 lines
  - Using 2 lines of configuration file
- Dense Optical Flow Use-Case
  - Generated 895 lines
  - Using 14 lines of configuration file
- Single Camera Frame copy to Single Camera Edge Detection
  - Required change of algorithm name in configuration file
  - Changing algorithm parameters in C code, which in this case 4 lines

Use-Case Coding Process using the tool

- Process :
  - Write Configuration file
  - Code generation using the tool
  - Fill in missing code
    - Link parameters
    - Main control loop
- Ready for use



Use-Case Coding Process: Config File

- A Link is named as [Link name] or [Link name]_[user readable suffix].
  - For e.g Capture, Capture_lvds, Capture_ov10640
- "-help" option displays all supported link names. All links in Vision SDK 2.3 are supported in the tool
  - Different suffix for different instances, i.e. Display_Video, Display_Grpx
  - Suffix can be any sequence of char's or number's readable to end user.
- Example: Single camera display

```
1 UseCase: chains_vipSingleCam_Display
2
3 Capture -> Display_Video
4 GrpxSrc -> Display_Grpx
```

- Use-case name should be mentioned
  - Files are generated with the use-case name and used for struct name and prefix of other function names
- IPC links, if required, are auto-detected and code for IPC links are autogenerated. So, no need to mention IPC IN/OUT links explicitly in config file
- Link Instance ID is auto-generated, Ex, in example shown above Display_Video will be Display Instance 0 and Display_Grpx will be Display Instance 1. No need to specify explicit instance number.

Configuration file: Grammar

- Grammar:
  - Connection: ID (optional CPU name) | ID (optional CPU name) -> Connection;
  - When CPU name is not mentioned IPU1_0 CPU is assumed
- Example: Dense optical flow view

```
1  UseCase: chains_vipSingleCameraDenseOpticalFlow
2  Capture -> Dup_capture
3
4  Dup_capture -> Alg_DenseOptFlow_1 (EVE1) -> Merge_Dof (DSP1)
5  Dup_capture -> Alg_DenseOptFlow_2 (EVE2) -> Merge_Dof
6  Dup_capture -> Alg_DenseOptFlow_3 (EVE3) -> Merge_Dof
7  Dup_capture -> Alg_DenseOptFlow_4 (EVE4) -> Merge_Dof
8
9  Merge_Dof -> Alg_VectorToImage (DSP1) -> Display_VideoDof
10 Dup_capture -> VPE_capture -> Display_VideoOriginal
11 GrpxSrc -> Display_Grpx
```

Code generation

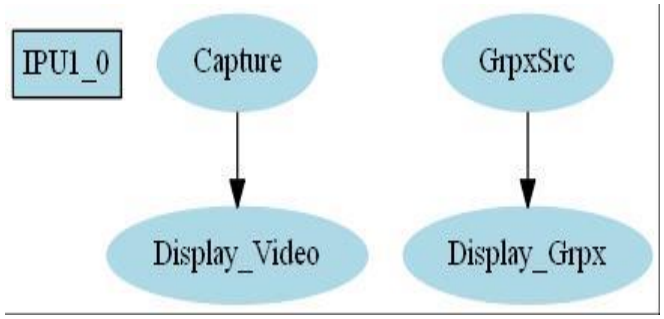
- Demo of how to generate code:
  - Options available:
    - -help Show help and supported basename and processors
    - -file Create .c and .h file
    - -img Create in .jpg image (or out.jpg)
    - -log Creates a .txt log file with debugging info
    - -debug Prints file name(in source code) and line no. in error statement
    - -path takes the next argument as output path
    - -v Verbose
- Examples:
  - ./vsdk_win32 –file test
  - ./vsdk_win32 –file –img test –path output

Generated Code and Files

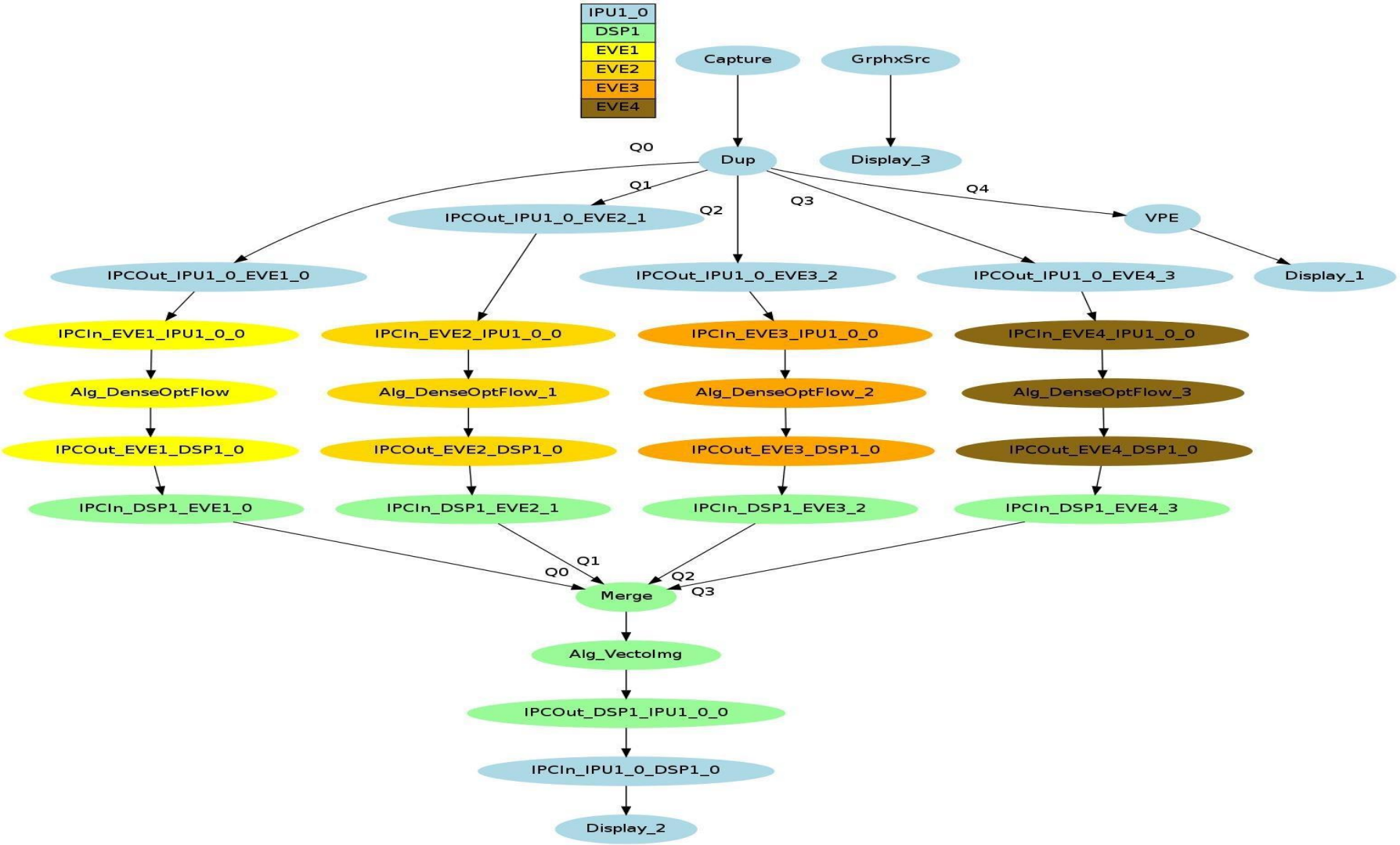
- Auto-Generated code includes
  - Usecase object have link ID variables and create structure for all links used in the use case
  - Assignment of link ID including instance numnber and CPU on which it runs
  - Reseting of all link create parameters to default values
  - Detecting and inserting IPC OUT/IN links as required
  - Setting of all parameters of DUP, MERGE, IPC links
  - Setting of inQueParams and outQueParams for all links - this defines the usecase graph connection in C code
  - Calling link create for all links in source to sink order
  - Calling start / stop / delete / print statistics of all links in the use-case
- Generated Files:
  - [use case name]_priv.h and [use case name]_priv.c file
  - Image file (optional) showing visual representation of the use-case graph
  - log file (optional) for advanced debug

Example Generated Image

- Single Camera Capture Display



Example Generated Image: Dense Optical Flow



Manually Written Code

- Following code needs to manually written to complete the use-case and make it run on the HW board
  - Set Create Parameters of all links except inQueParams and outQueParams.
    - DUP, Merge, IPC IN, IPC OUT parameters are completely setup by the generated code
    - Algorithm Plugin baseClassCreate.size, baseClassCreate.algId are setup by the generated code
    - numOutQueue of Select, numInQueue of Null are setup by the generated code
  - Create DisplayCtrl link and configure it
  - Written main loop for use case and call generated API to create, start, stop, delete a use-case

Define custom link

- User can define a custom link which is not supported by the tool
- It is named as DefLink_[Link name] in configuration file
- Here additionally edit DefLink_CreateParams struct name in [use case]_priv.h file and replace with actually parmaeter structure name of the custom link
- Set LinkID in [use case]_priv.c file and call reset create params function in [use case]_priv.c file

Define custom Algorithm

- User can define a custom link which is not supported by the tool
- It is named as Alg_[alg name] in configuration file
- Here additionally edit “[alg name]Link_CreateParams” struct name in [use case]_priv.h file and replace with actually parameter structure name of the custom link
- Set baseClassCreate.size, baseClassCreate.algId in [use case]_priv.c file and call reset create params function in [use case]_priv.c file

Error Handling cases

- Wrong number of input or output to a link
  - Ex, previous link specified for capture link
- Invalid CPU name
- Same link instance is assigned two different CPUs
- Naming of Link does not follow the rules, i.e. Linkname, Linkname_suffix
- Warnings for unsupported links and algorithm plugin’s

Extending the tool - Ease for developer

- In case extra link or alg plugin is added just include extra class in link.h
- Each class has it's separate functionalities which makes easier for user to determine where to change to include additional functionalities, etc.
  - UseCase class handles overall usecase information
  - Link class handles information regarding a particular link
- In debug mode (-debug option) line number and file name is also given in case of errors
- See VisionSDK_UsecaseGen_UserGuide for step by steps details on extending the tool to support more link types and alg plugin types

Thank You