Oslo - Stockholm - Utrecht - Brussels - Copenhagen - Helsinki

ni.com/nidays

NATIONAL INSTRUMENTS

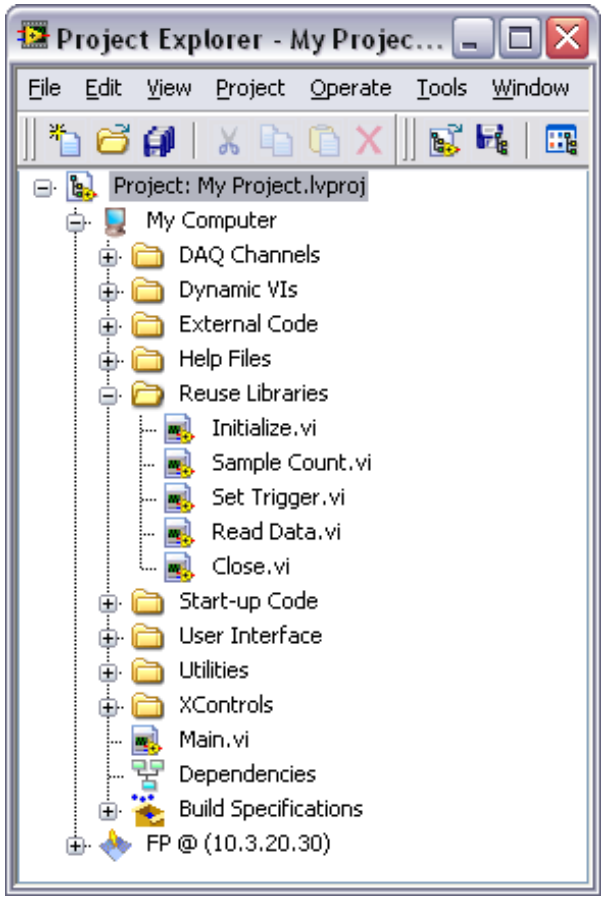
WHAT'S NEW IN LABVIEW 8.5

What's New in LabVIEW 8.5

- LabVIEW Project 2.0
- VI Merge
- Memory control tools
- LabVIEW MathScript
- Customer-requested features
- Multicore, multithreading

NI LabVIEW 8 Project in Review

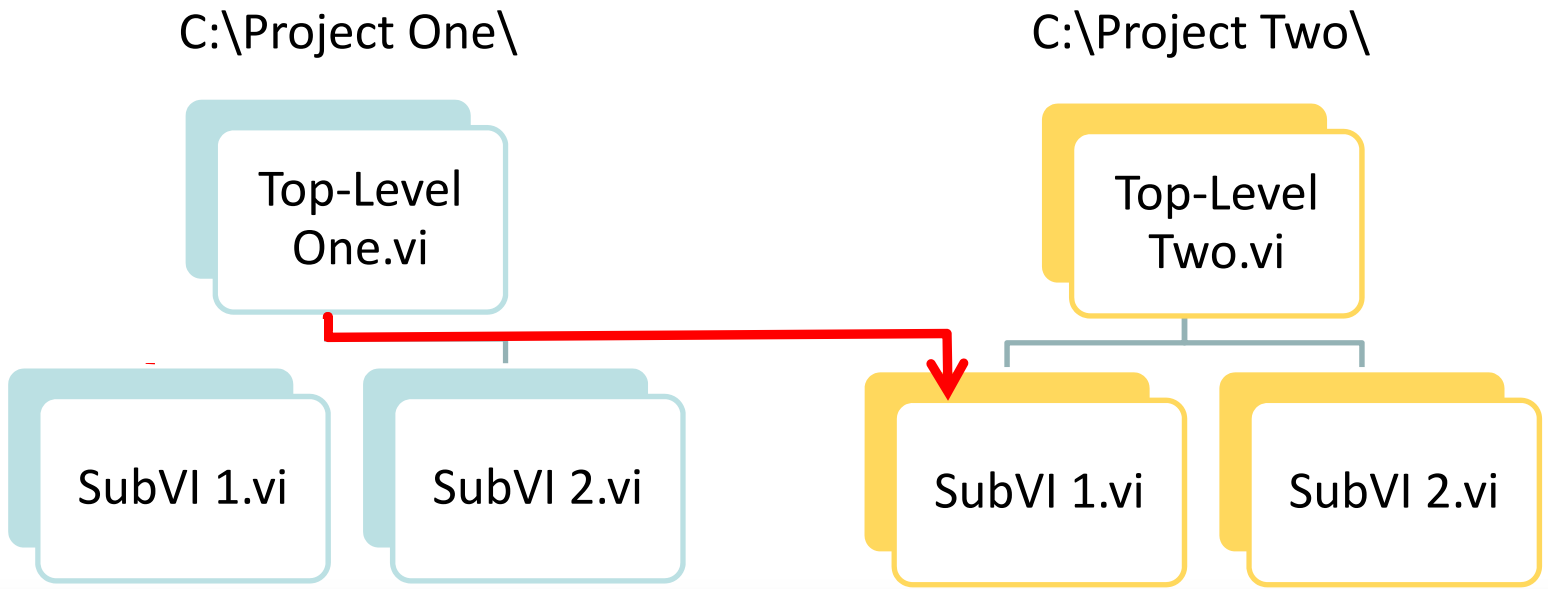
- Positive feedback
  - Target management
  - Software deployment
- User requests
  - Help with cross-linking
  - Simpler file management



Cross-Linking Defined

A VI references a subVI that you didn't intend.

- Older version
- Different branch of same code

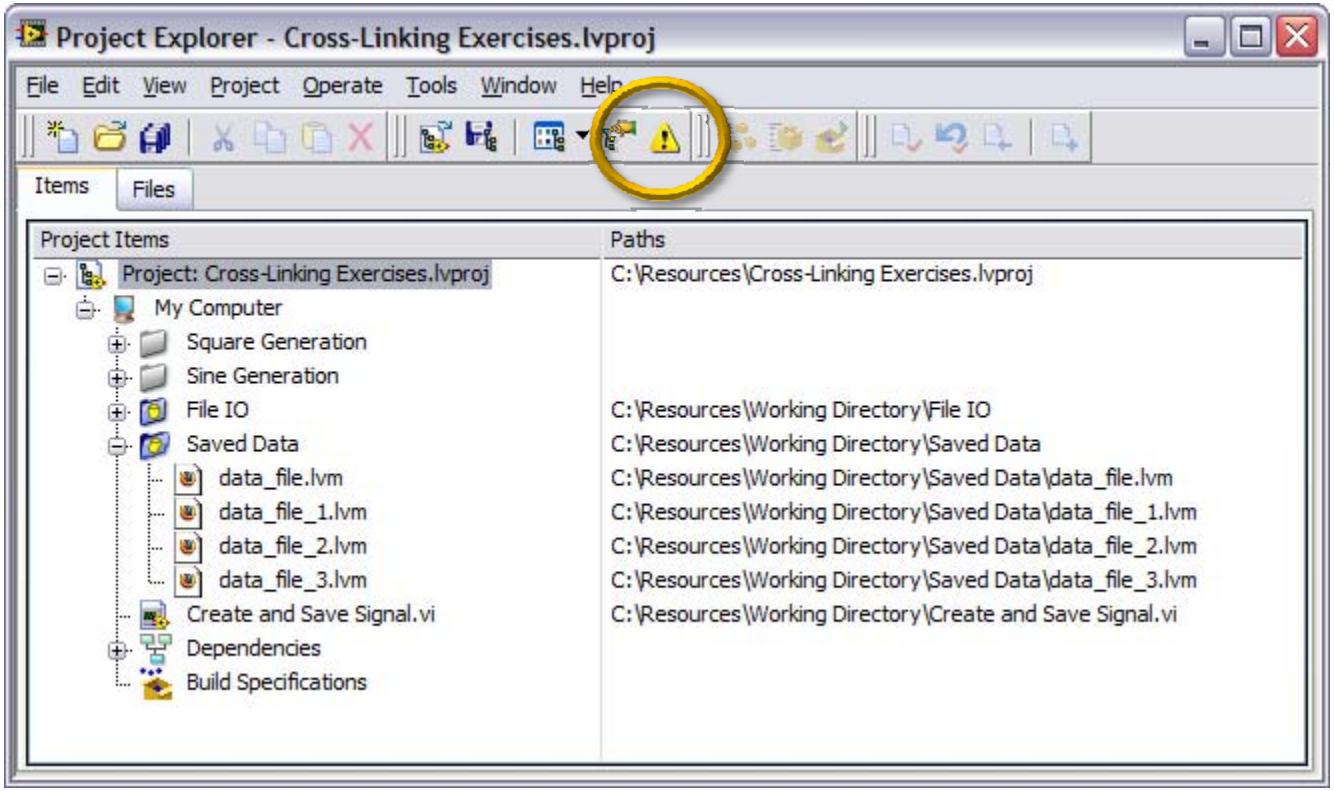


Cross-Linking Background

- Some causes of cross-linking
  - Loading subVI has same name as VI in memory
  - Search for missing VI results in incorrect linking
- Effects of cross-linking
  - Accidental modifications
  - Reduced productivity
  - Unexpected system behavior

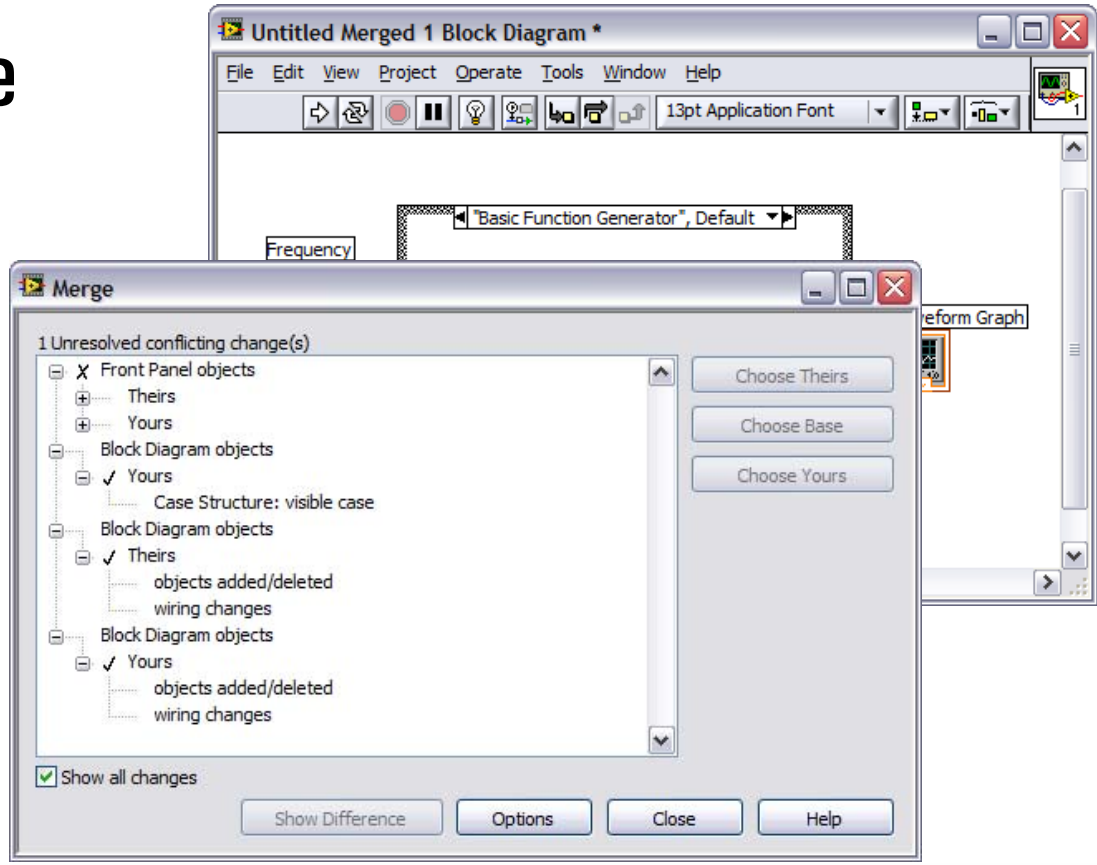
LabVIEW 8.5 Project

- Predict, detect, and repair VI cross-linking



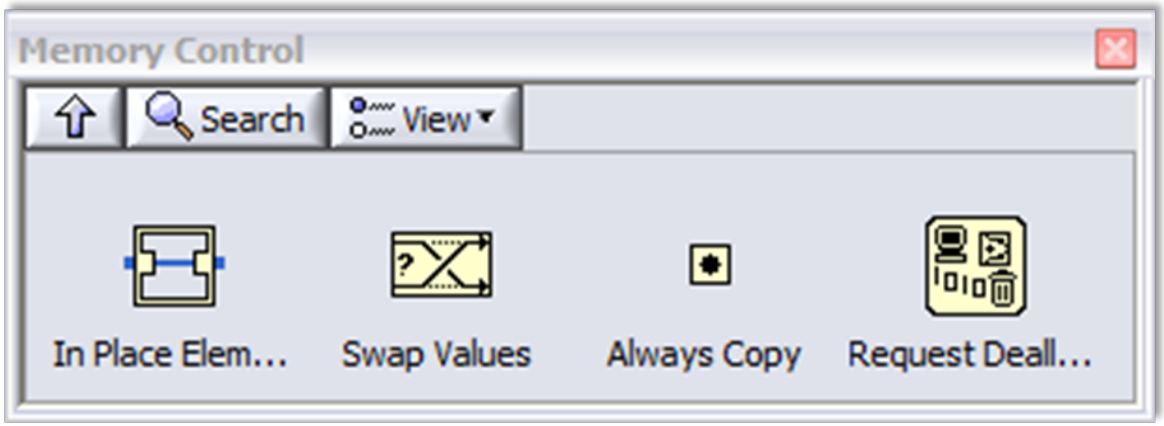
VI Merge for Team-Based Development

- Automatically merge separate edits to a VI introduced by two developers



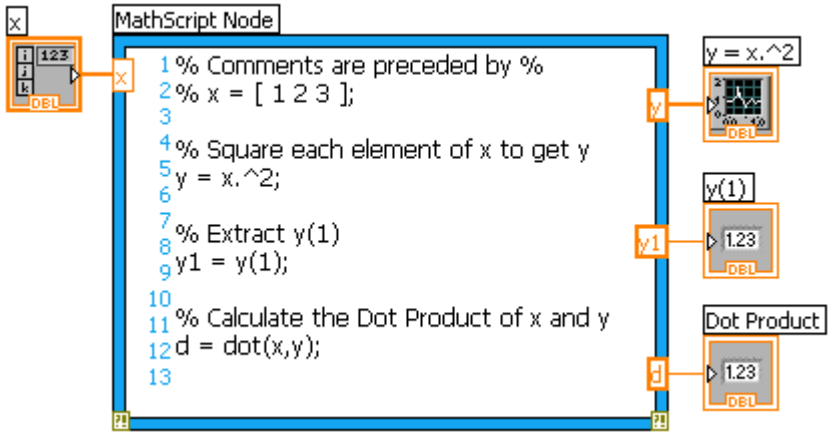
Optimize Memory Usage

- Control LabVIEW memory usage with block diagram VI and structures
- Available on the PC and LabVIEW Real-Time



LabVIEW MathScript Improvements

- Automatic error checking
- 70+ new functions
- New plot types
- Faster performance

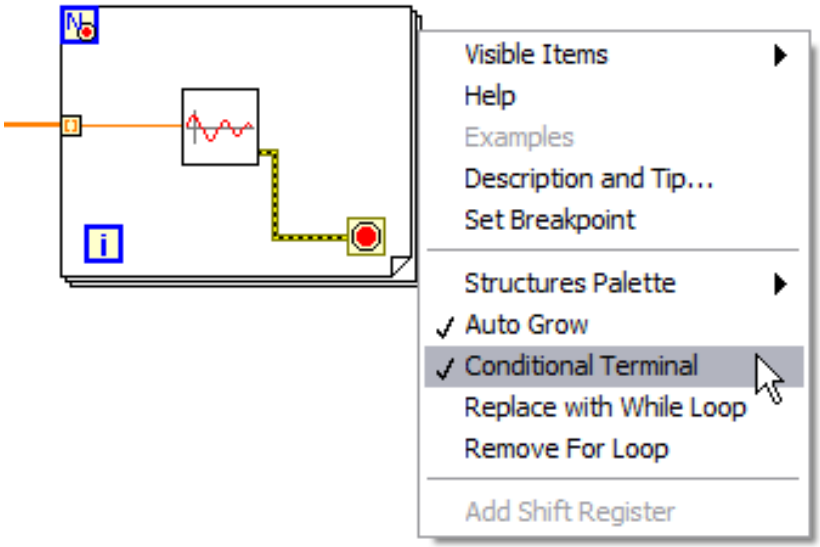


Performance Improvements: Matrix Operations

Matrix Size	Iterations	Time in 8.20	Time in 8.5	Speed-up
10x10	100	2.08	0.201	10x
100x100	10	1.38	0.077	18x
500x500	1	4.12	0.211	20x
1000x1000	1	Out of memory	0.89	N/A

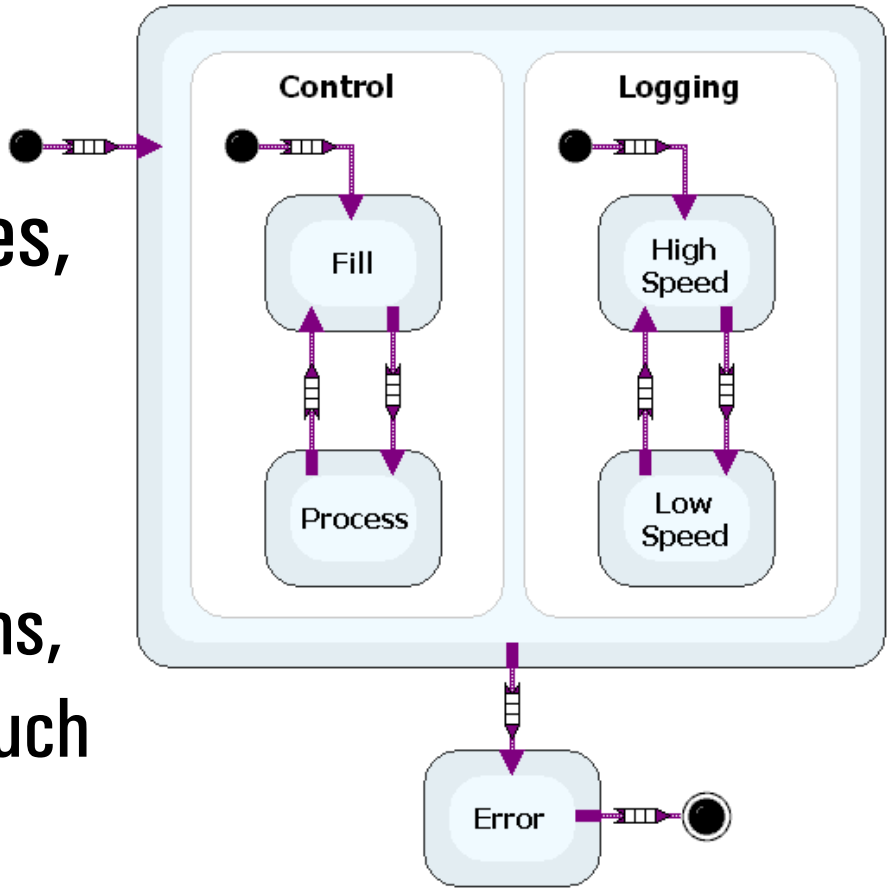
Additional Requested Features

- Lifetime serial number
- DVD installation
- Save to multiple versions
- For loop with break



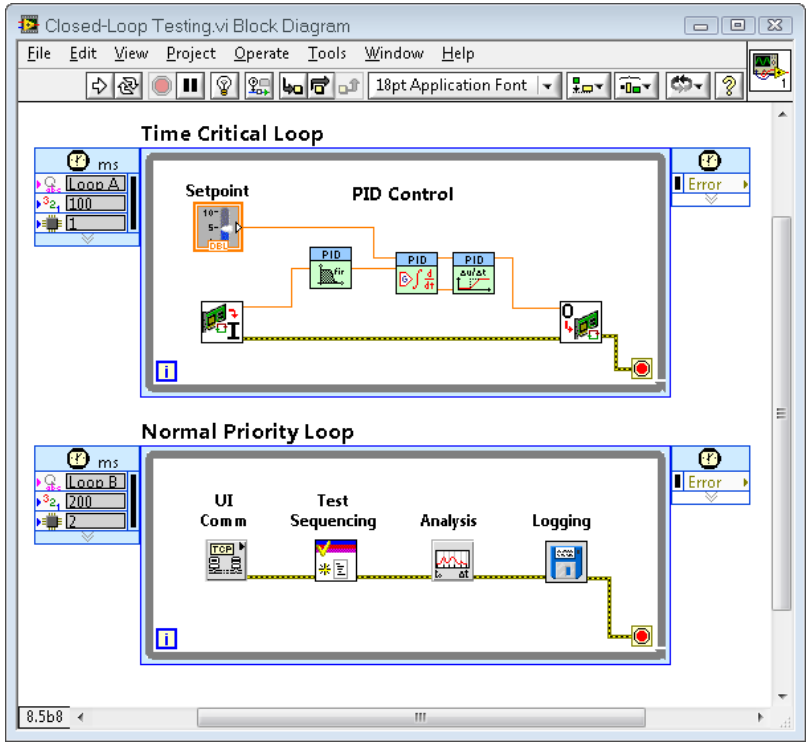
LabVIEW Statechart Module

- Combine with I/O to implement user interfaces, state machines, etc.
- Deploy to:
  - Desktop PCs, RT systems, FPGAs, μProcessors, touch panels



Multithreading Features in LV 8.5

- Scale # of execution system threads based on available cores
- Improved thread scheduling for LabVIEW timed loops
- Set Processor Affinity with Timed Structures



Will My Application Run Faster?

- Key Considerations
  - Is the code **sequential or parallel**?
  - How are **shared resources** being utilized within the application?
- Conclusions
  - Code speed-up depends on application
  - Many applications will require minor modifications to fully optimize for multicore

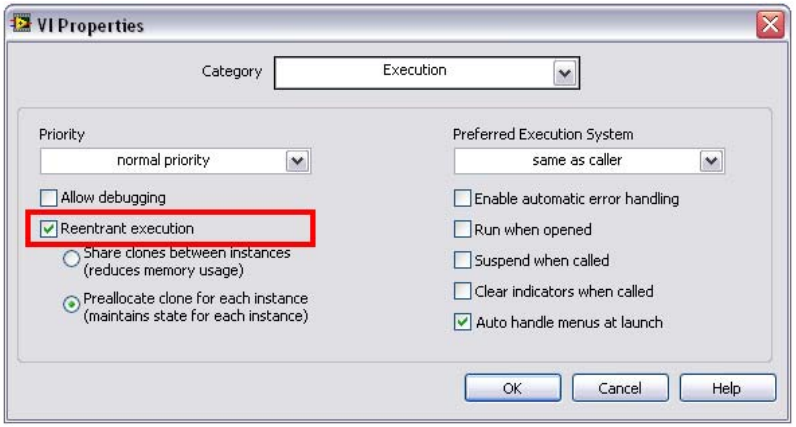
Potential Bottlenecks:

Shared Resources

1. Data Dependencies
   - Example: Data stored in global variables that need to be accessed by different VIs would be a shared resource
2. Hard Disk
   - Example: Computers can only read or write to the hard disk one item at a time
3. Non-reentrant VIs

Re-entrancy

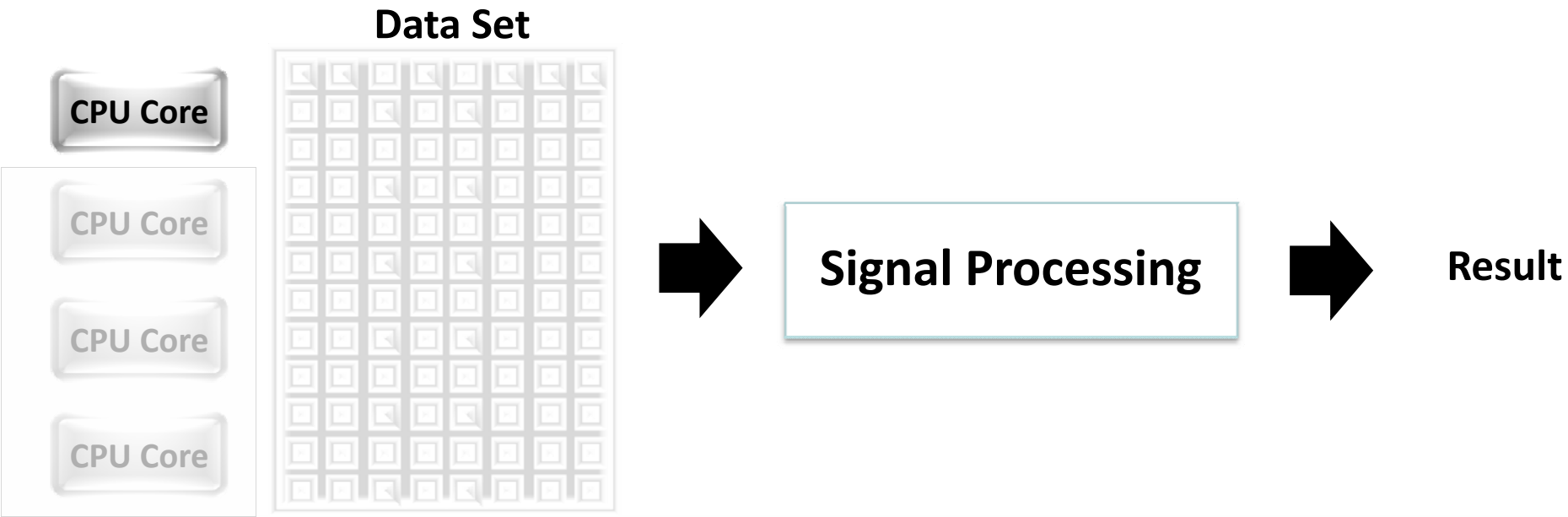
- VIs that are non-reentrant cannot be called simultaneously:
  - one call runs and the other call waits for the first to finish before running



File»VI Properties select the Execution category and choose Reentrant execution.

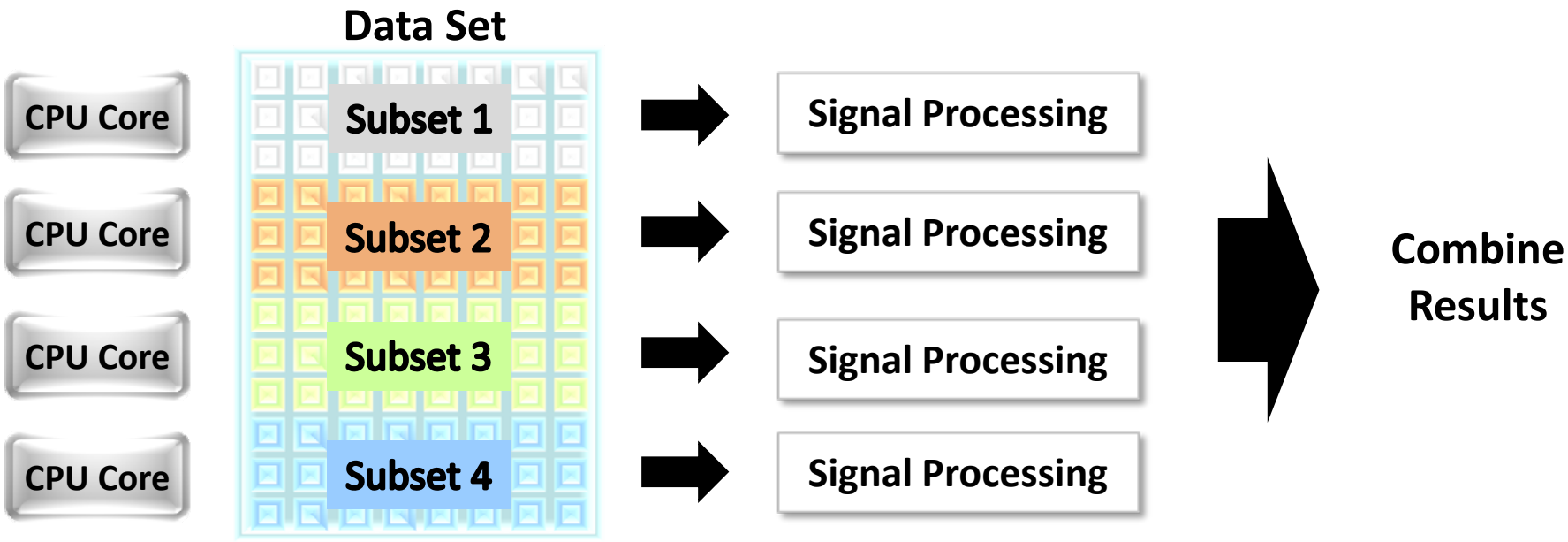
Data Parallelism

- You can speed up processor-intensive operations on large data sets on multicore systems.

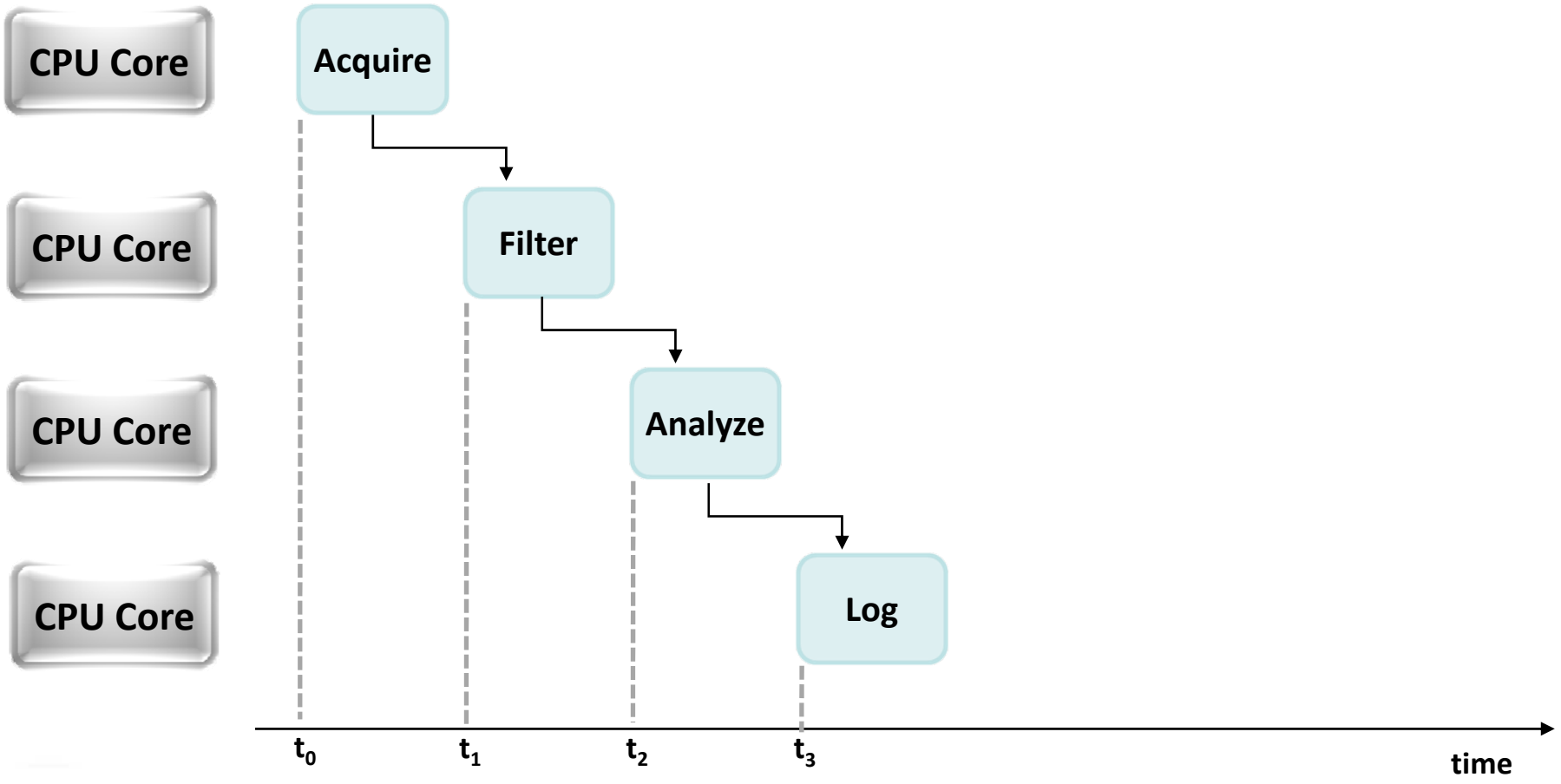


Data Parallelism

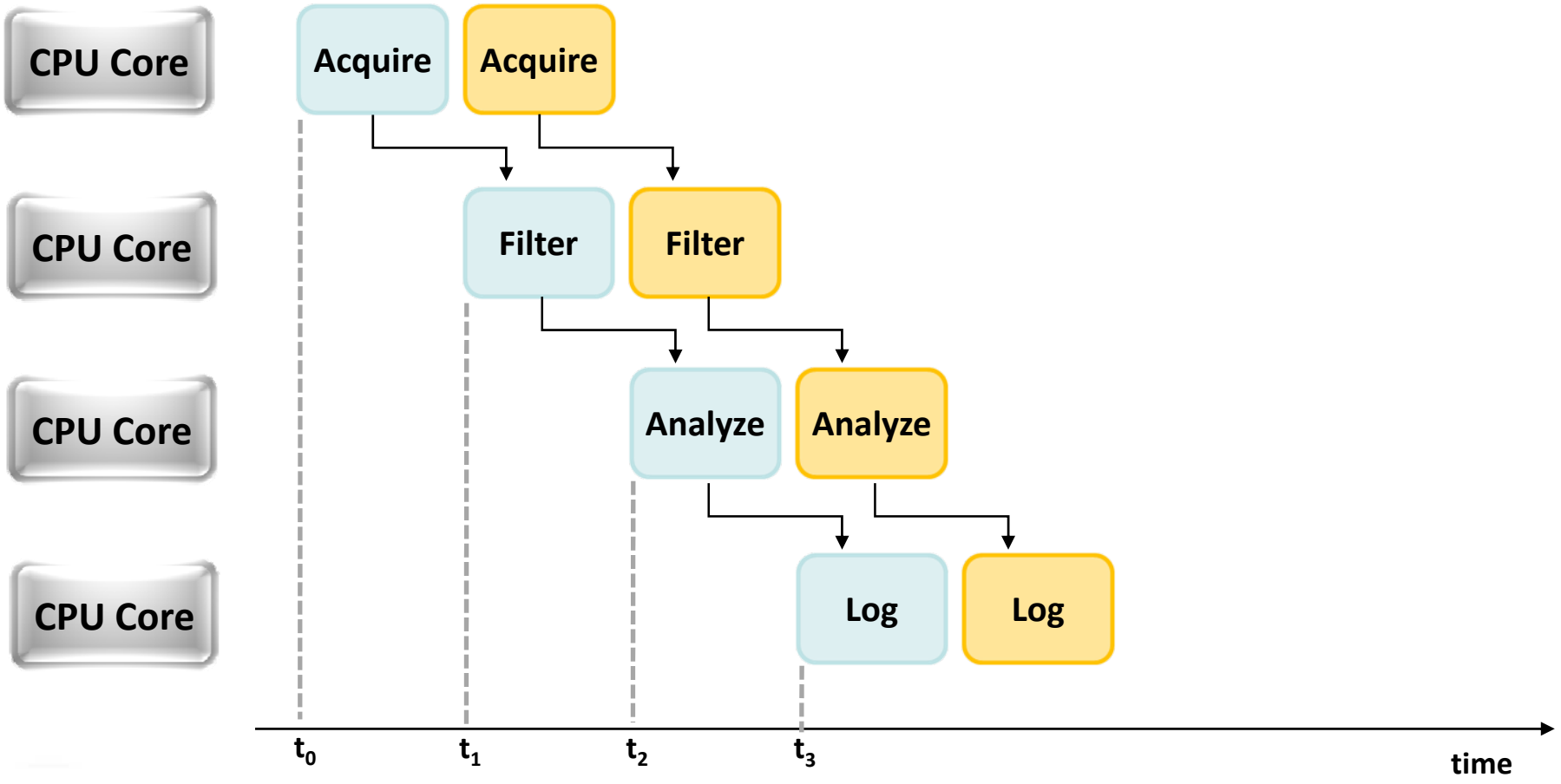
- You can speed up processor-intensive operations on large data sets on multicore systems.



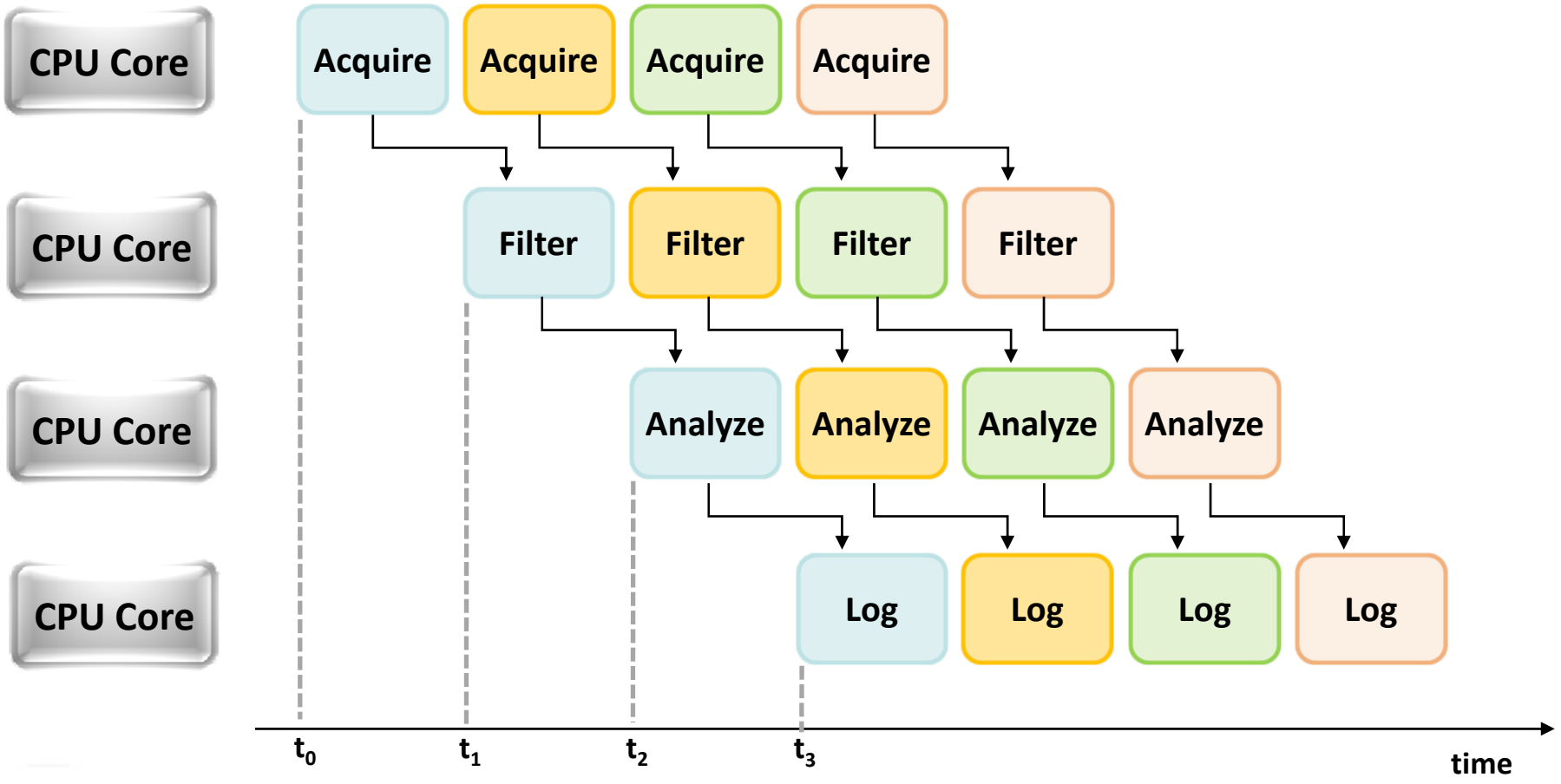
Pipelining Strategy



Pipelining Strategy

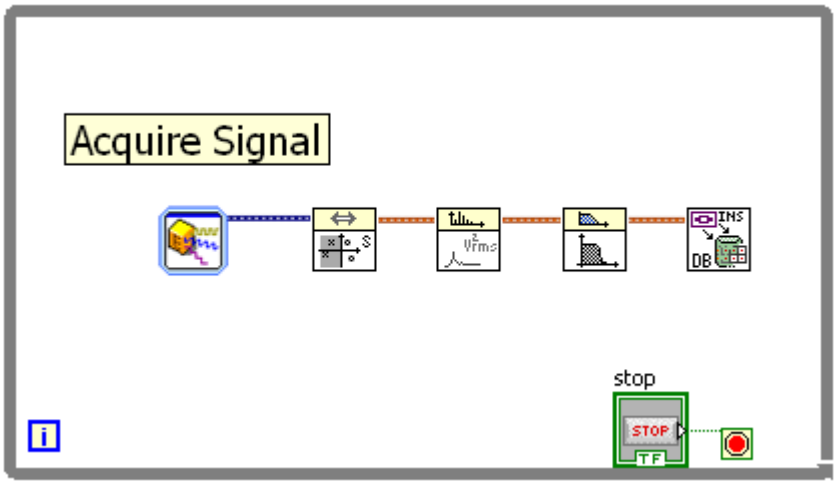


Pipelining Strategy

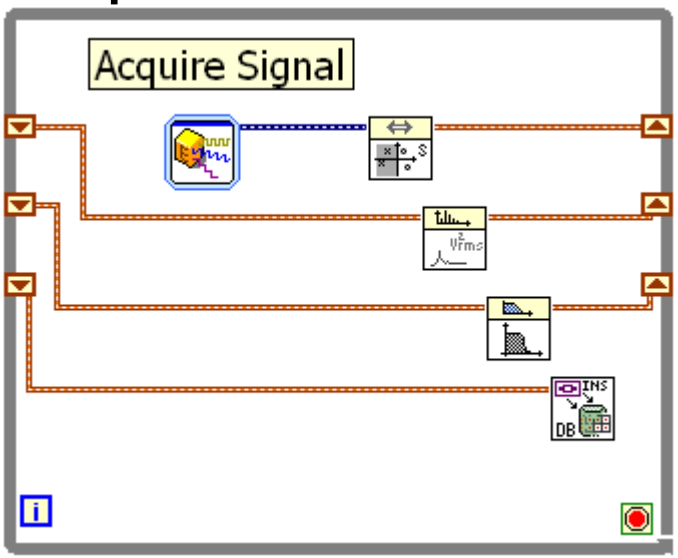


Pipelining in LabVIEW

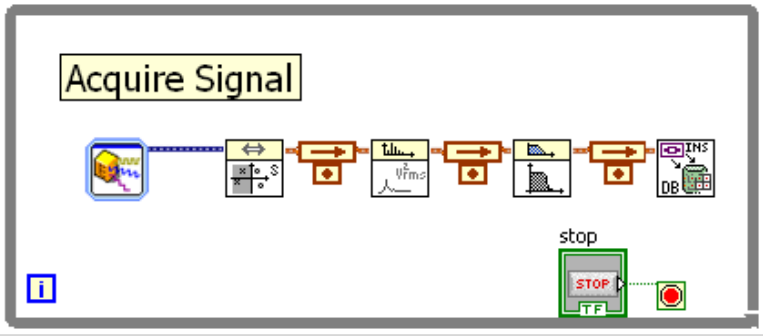
Sequential



Pipelined



Or:



Note: Queues may also be used to pipeline data between different loops

Useful Links

- http://www.ni.com/labview/
- LabVIEW 8.5
- Upgrade your software now
- LabVIEW 8.5 Upgrade Notes [attachment]