End to End Modeling

Hiro Yamamoto

LIGO / Caltech

March 12, 1998 LSC Meeting

- Motivation
- Framework
- Interface
- Current Status
- Documentations

B. Bhawal, M. Evans, E. Maros, S. Mohanty,
M. Rahman, H. Yamamoto

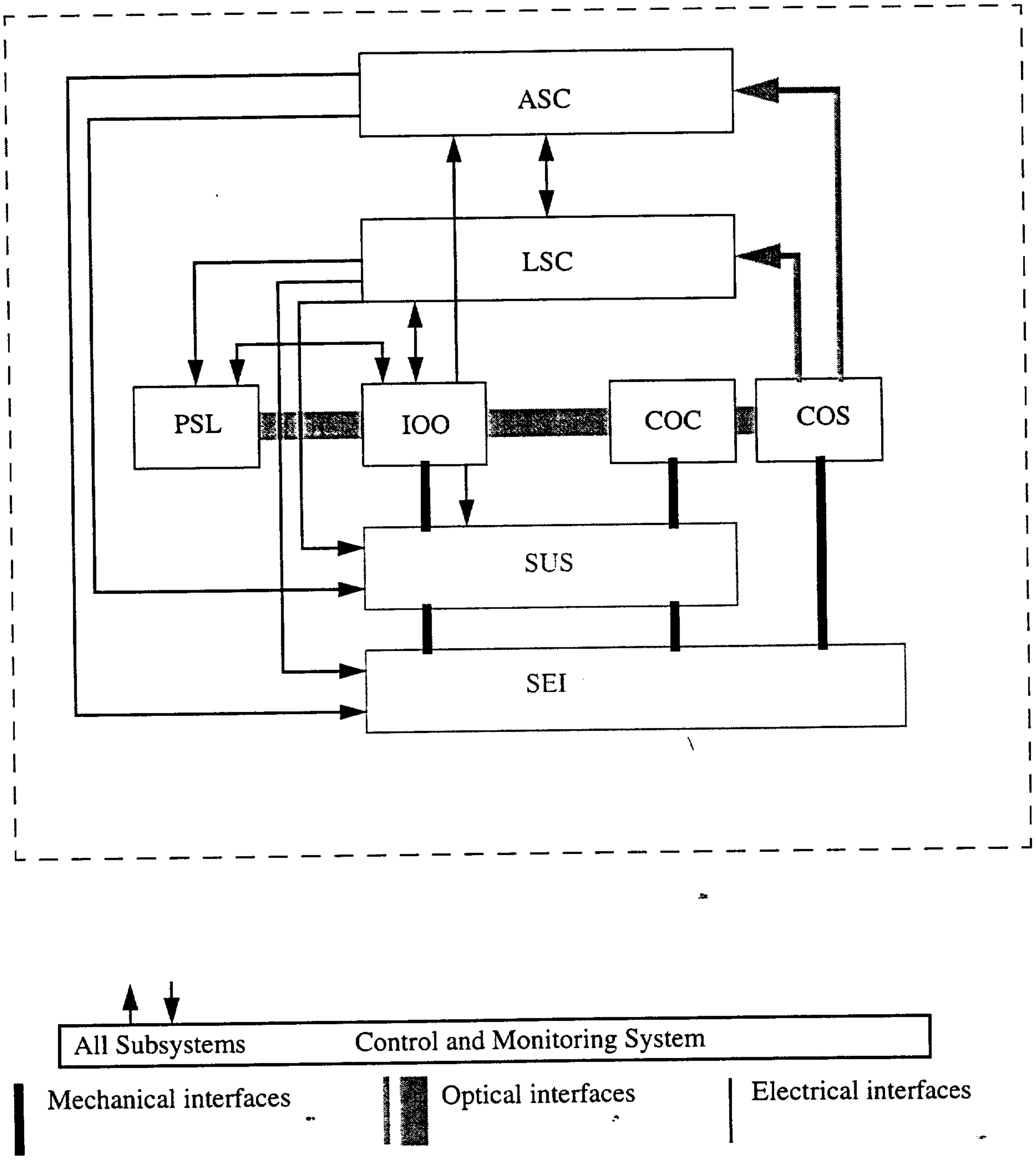
Motivation

- Traditional LIGO modeling activities
  - Models developed for each task when needed
  - There are some completed, and are used for design and diagnostics
  - No code compatibility between different models
  - Hard to expand
  - Hard to simulate new components
  - Suffer overhead penalty using high level language

- End to End model
  - A model which includes all important subsystems, so same codes are used for all simulations
  - Time domain model to simulate LIGO as close as possible
  - Easy to expand and to simulate new components by using object oriented design
  - In-house developed code for optimization for LIGO
  - Easy to use by adopting two-layer programming - no C++ knowledge needed to use the program
  - Too later for the design of LIGO
  - Diagnostics of LIGO
  - Design tool and trade study of future design

Substances

-

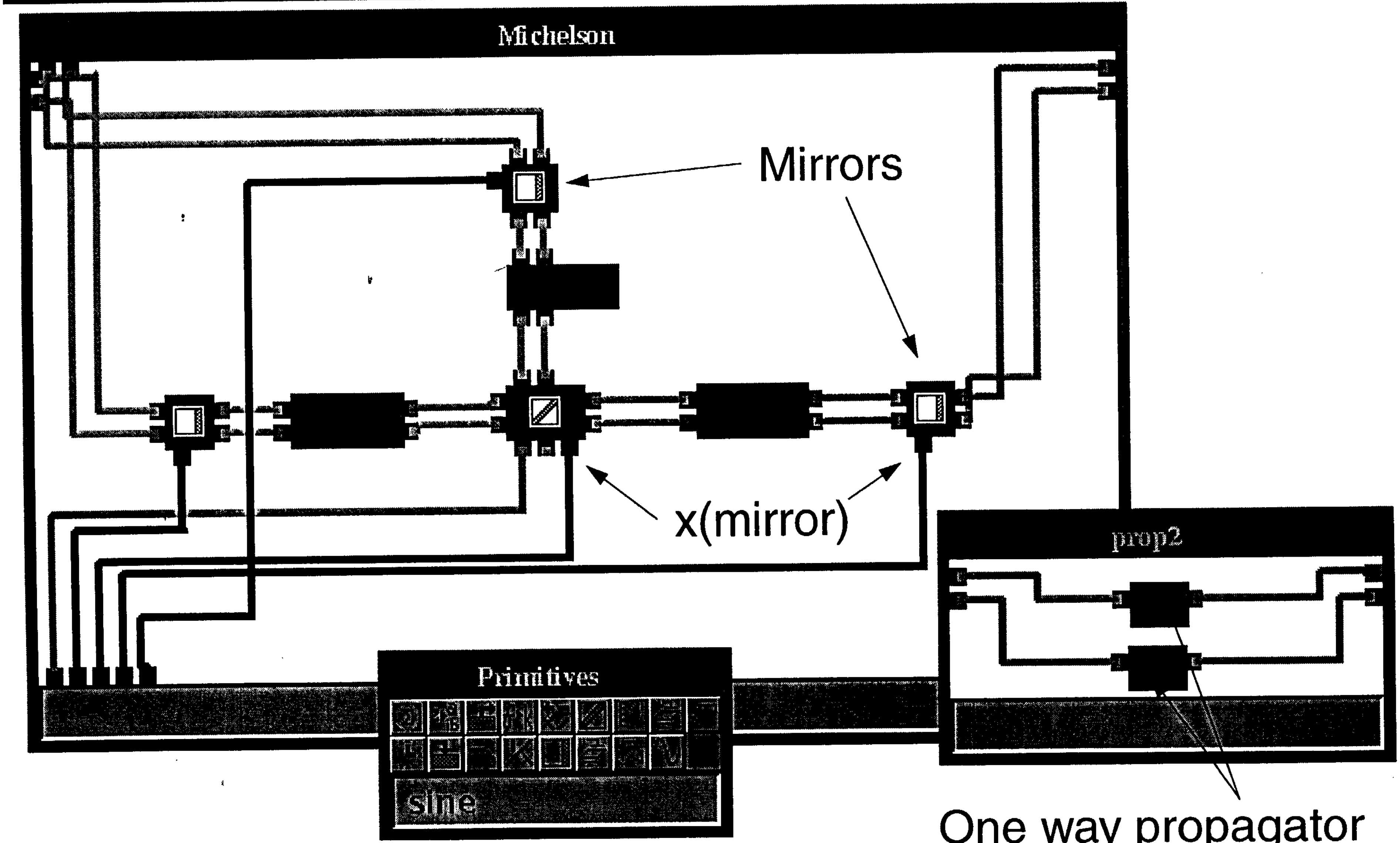

ASC
LSC
PSL
IOO
COC
COS
SUS
SEI
All Subsystems
Control and Monitoring System
Mechanical interfaces
Optical interfaces
Electrical interfaces

End to End model
simulation engine

- Adlib - “Adlib, digital instrument builder” by M. Evans of Caltech
- Time domain - digitized time evolution
- Written in C++ (g++ & motif)
- Object Oriented, modular and expandable
- Primitive modules
  - mirrors
  - propagators
  - short cavities for fast simulation
  - field source
  - sideband generator and phase modulator
  - demodulator
  - digital filter
  - math routines
- Two layer structure - no C++ knowledge needed
- Speed optimization - no penalty using upper layer
- Alfi - GUI to build the (sub)systems to simulate

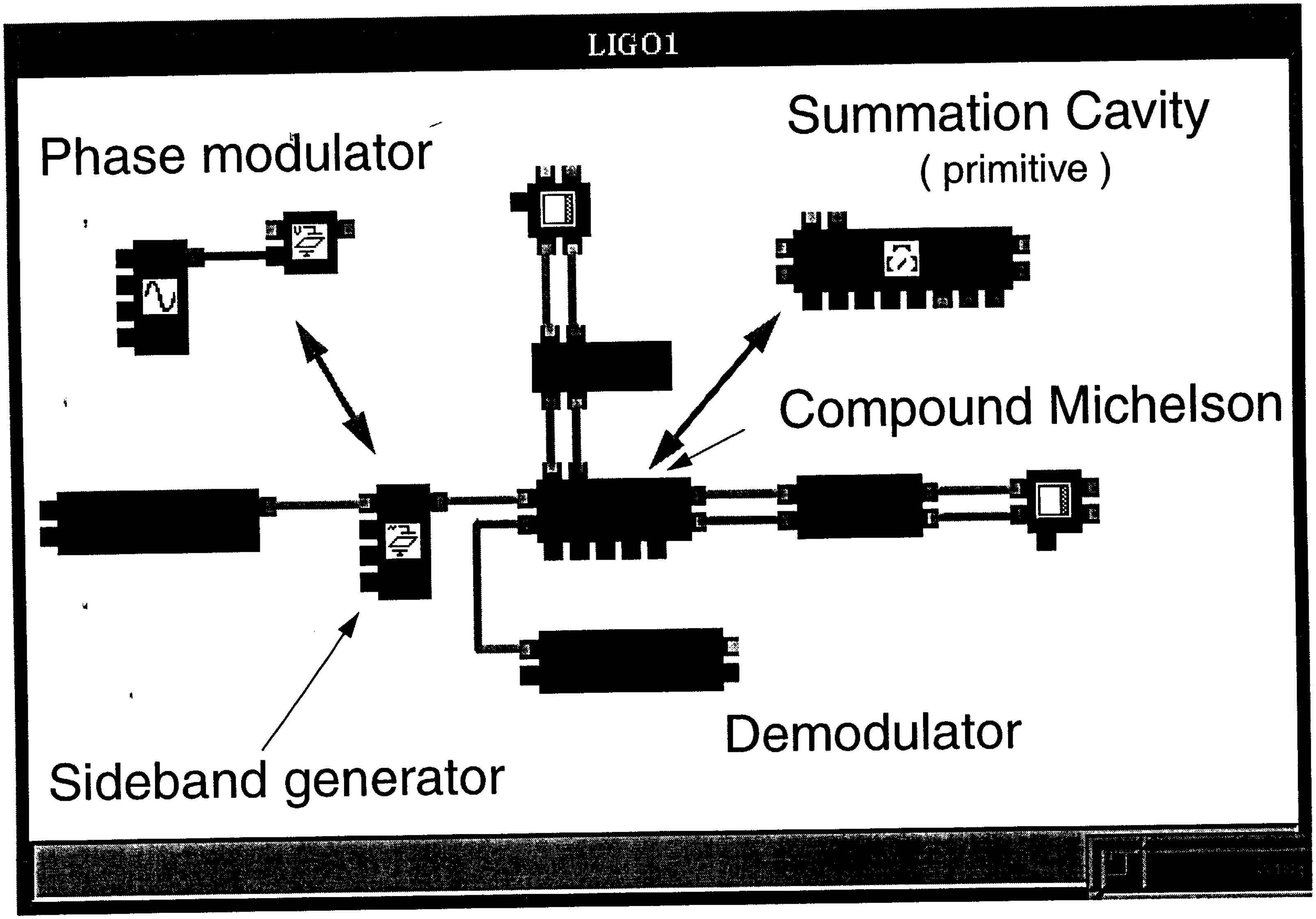
Examples

how to build (sub)systems using Alfi - GUI of E2E



Examples

OOP / Modularity



Phase modulator

Summation Cavity

(primitive)

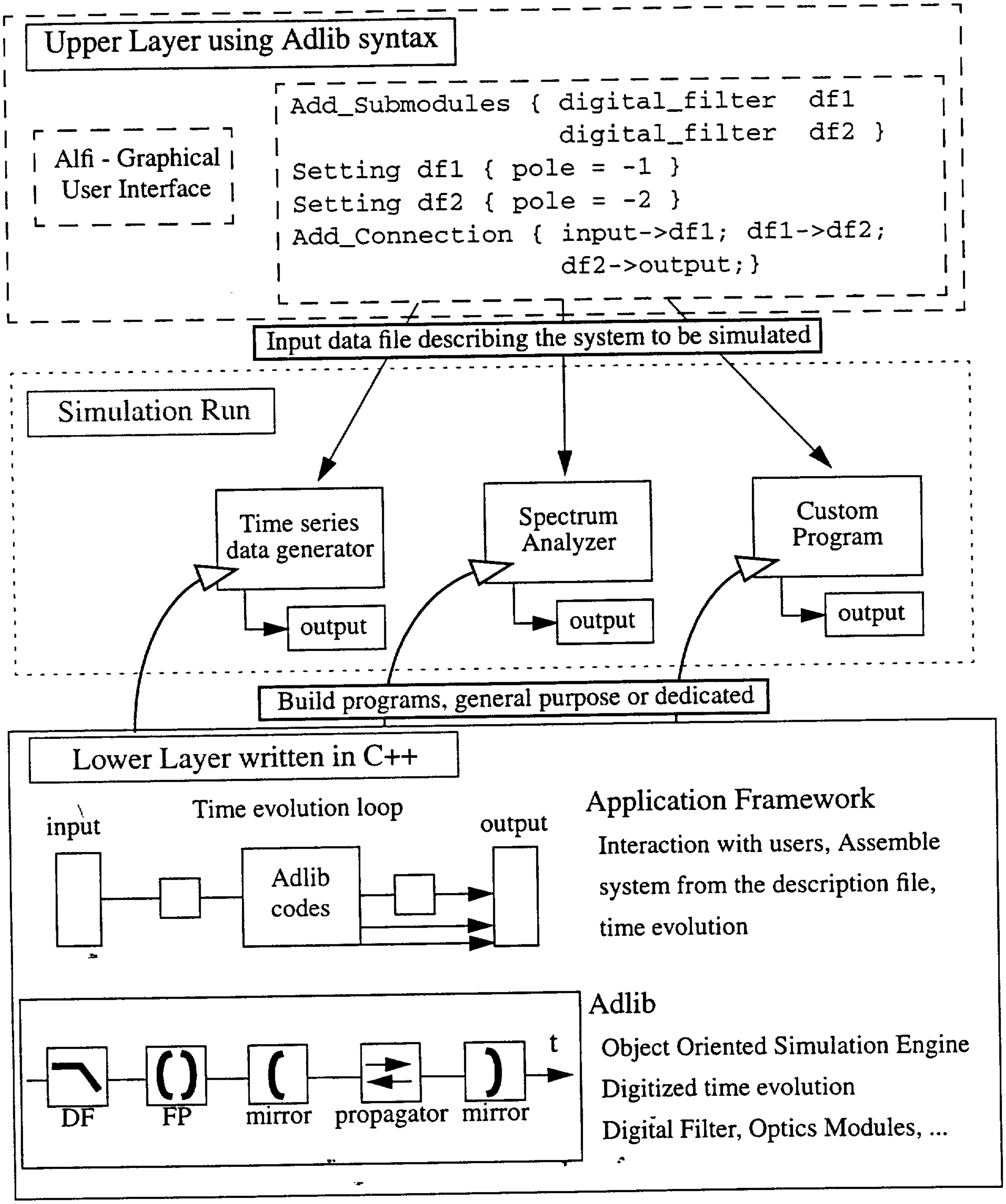
Compound Michelson

Sideband generator

Demodulator

Program structure

a.la. Matlab dedicated for LIGO



Upper Layer using Adlib syntax

Alfi - Graphical User Interface

```
Add_Submodules { digital_filter  df1
                 digital_filter  df2 }
Setting df1 { pole = -1 }
Setting df2 { pole = -2 }
Add_Connection { input->df1; df1->df2;
                 df2->output;}
```

Input data file describing the system to be simulated

Simulation Run

Build programs, general purpose or dedicated

Lower Layer written in C++

Application Framework

Interaction with users, Assemble system from the description file, time evolution

Adlib

Object Oriented Simulation Engine

Digitized time evolution

Digital Filter, Optics Modules, ...

Modeling Works

- Adlib - simulation engine development
  - ›› Improvement of the speed
  - ›› New capabilities
  - ›› Improve GUI
- Primitive module development - C++
  - ›› Optics with alignment
  - ›› Very fast simulation of steady state optics system
  - ›› Non linear systems
  - ›› Time series of seismic motion from data file
- Compound module development - GUI
  - ›› Construction of subsystems
    - SUS/SEI
    - PSL
    - ASC / LSC
    - etc. etc. etc.
  - ›› Data Analysis probe
- Analysis
  - ›› a. la. SMAC / MMAC
  - ›› Trade study
  - ›› Data analysis using the pseudo data

Current Status

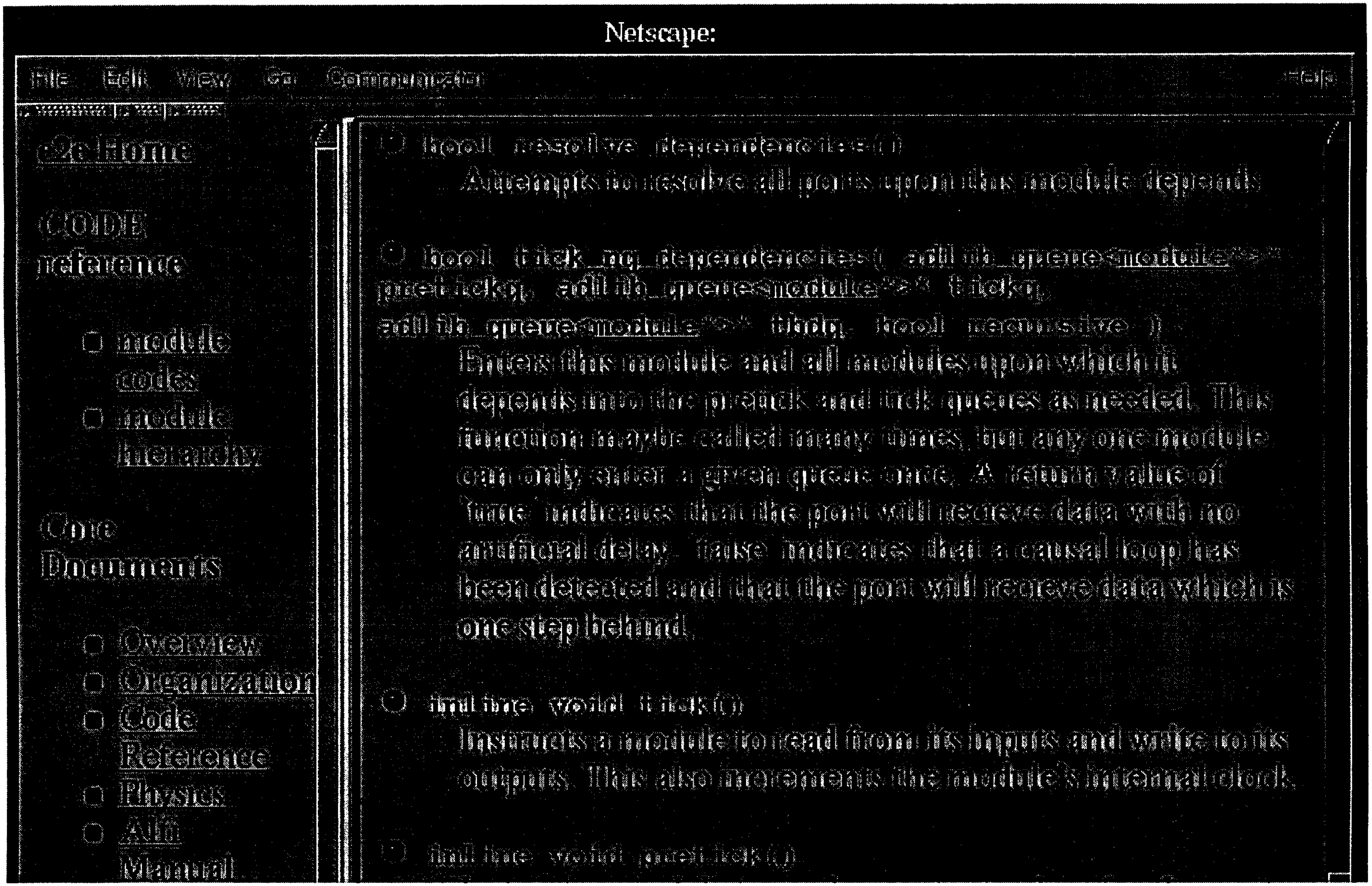
- Simulation Engine close to release
  - Program and syntax design almost done
  - Implementation almost done
  - Many primitive modules almost done
  - Extensive module validation needed
  - Alignment in optics now being implemented
- GUI to construct the system to simulate
  - Alfi first version released
  - Updated to improve the usability and stability
  - Future - runtime control and data visualization
- LIGO module development
  - Just starting
    - Lots of work to create the system description files
    - Create primitive modules when needed
    - Validations
  - Core optics will be ready by summer
  - Seismic Isolation and Suspension modeling started
- Documentation
  - see next

Documentation

- Documentation just started
- Core documents
  - Overview of End-to-End Model
    - outline, roadmap, examples, quick reference
    - ready in a month
  - Organization of End-to-End model
  - Code Reference for End-to-End Model
  - Physics of End-to-End Model
- Supplements
  - Alfi - the GUI of the End-to-End model
- Code reference
  - DOC++
  - Embed comments in the header file
  - HTML or LaTex documents
- On-line documents
  - e2e web home page
  - pdf version of core documents
  - core reference by DOC++
  - relevant manuals - e.g., DOC++

Web home page

not yet ready



e2e Home

CODE reference

- ○ module codes
- ○ module hierarchy

Core Documents

- ○ Overview
- ○ Organization
- ○ Code Reference
- ○ Physics
- ○ Alf Manual

○ bool resolve_dependencies()

Attempts to resolve all ports upon this module depends

○ bool tick_mq_dependencies(adlib_queue<module*>* pretickq, adlib_queue<module*>* tickq, adlib_queue<module*>* thrq, bool recursive)

Enters this module and all modules upon which it depends into the pretick and tick queues as needed. This function maybe called many times, but any one module can only enter a given queue once. A return value of 'true' indicates that the port will recieve data with no artificial delay. 'false' indicates that a causal loop has been detected and that the port will recieve data which is one step behind.

○ inline void tick()

Instructs a module to read from its inputs and write to its outputs. This also increments the module's internal clock.

○ inline void pretick()

Note 1, Linda Turner, 04/20/98 04:20:40 PM
LIGO-G980049-14-M