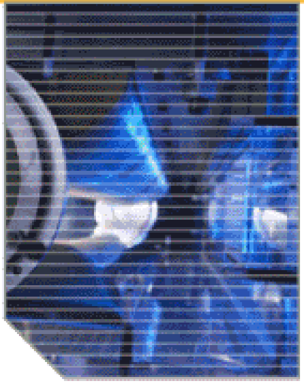
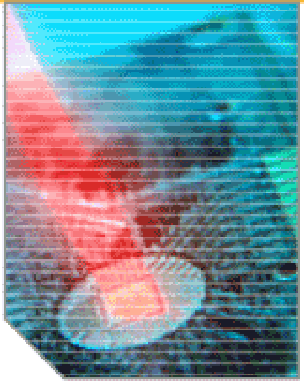
Improving Your View
The Upcoming Role of PostgreSQL in Flat Panel Display Production

CYMER®

Joe Conway

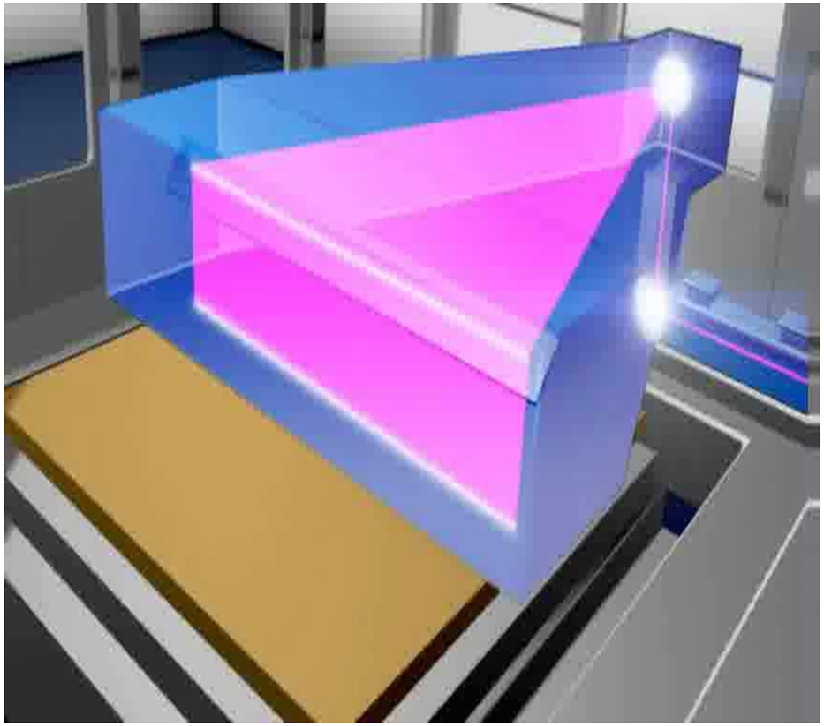
Agenda

- Introduction
  - Use Case
  - Players
- Flat Panel Display (FPD) Overview
  - Market
  - Process
- Control System Architecture
  - Hardware
  - Software
- PostgreSQL's Roles
  - What
  - How

Agenda

- Introduction
  - Use Case
  - Players
- Flat Panel Display (FPD) Overview
  - Market
  - Process
- Control System Architecture
  - Hardware
  - Software
- PostgreSQL's Roles
  - What
  - How

Introduction



- Case Study
- Complex Equipment
- Software Controlled
- PostgreSQL at the core

CYMER®

TCZ Overview

- Joint Venture of Cymer & Carl Zeiss SMT Founded July, 2005
- Corporate Headquarters: San Diego, CA
- Manufacturing Facilities:
  - San Diego, Calif. – Light Source Manufacturing
  - Oberkochen, Germany – Stage System and Optics Manufacturing
- Demonstration Facility and Integration Center:
  - Pyongtaek Korea

Cymer Manufacturing Facility San Diego, Calif.

Zeiss Manufacturing Facility Oberkochen, Germany

Demonstration Facility and Integration Center Pyongtaek, Korea

Cymer Overview

Cymer is the world's largest supplier of excimer light sources enabling deep-ultraviolet (DUV) photolithography

- 2006 Revenues: $543.9 Million
- About 975 Employees worldwide
- Founded in 1986
- Major Customers Include:
  - Canon, Nikon, ASML
  - The world's top chip-makers
- Over 3000 laser light sources installed

CYMER

Carl Zeiss SMT AG Overview

Zeiss SMT AG is representing one out of six business units from Carl Zeiss AG, a global leader in the optical and opto-electronic industries.

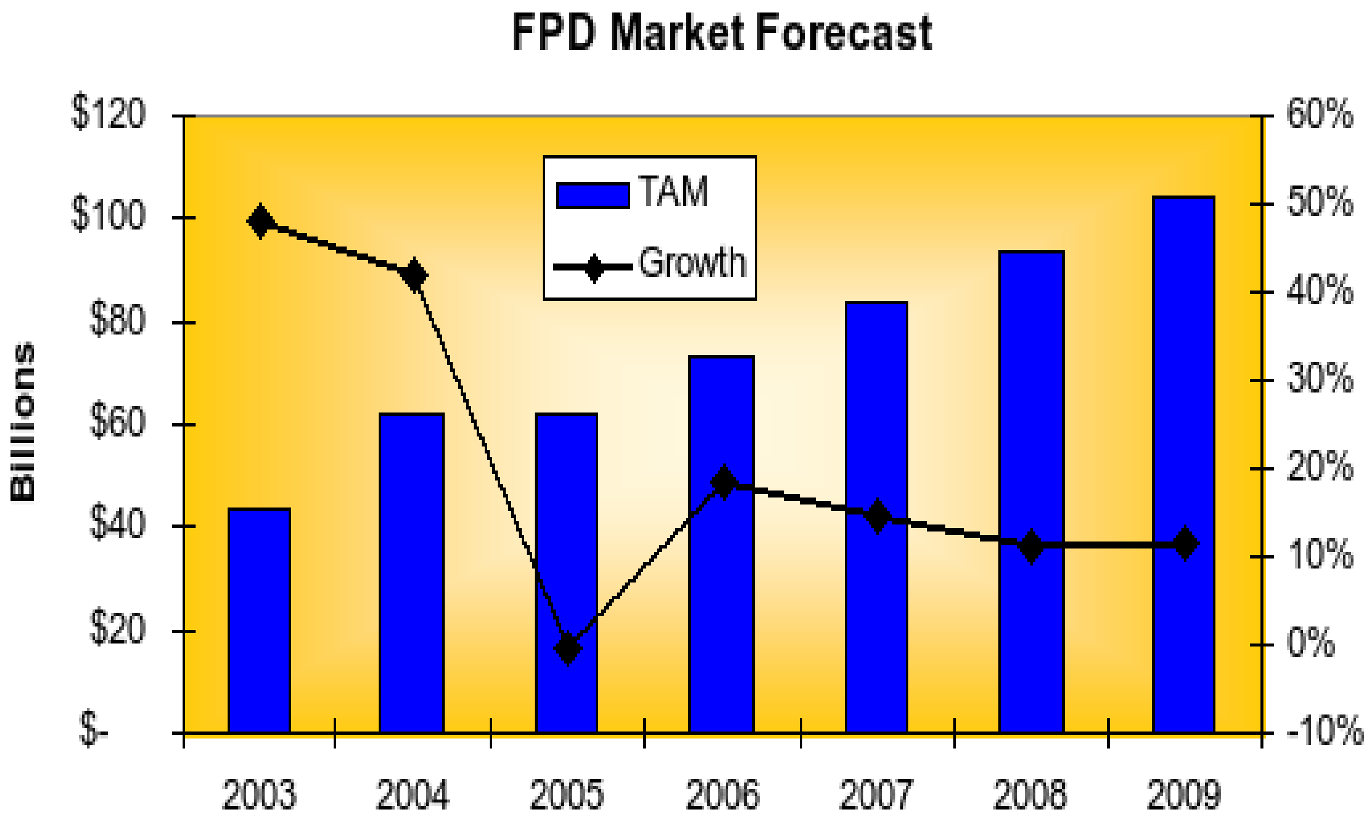
- 2005 Revenues: €656 Million
- > 1,900 employees worldwide
- Founded in 2001
- Currently four divisions:
  - Lithography Optics
  - Laser Optics
  - Semiconductor Metrology System
  - Nano Technology Systems
- Major Customers Include:
  - ASML, Cymer

CYMER®

Agenda

- Introduction
  - Use Case
  - Players
- Flat Panel Display (FPD) Overview
  - Market
  - Process
- Control System Architecture
  - Hardware
  - Software
- PostgreSQL's Roles
  - What
  - How

Why Do We Care?



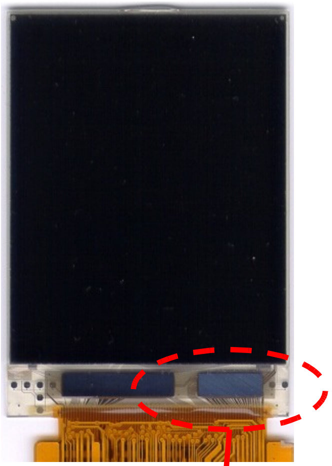
Source: DisplaySearch June 05

Trend 1 – Reduce Size of TFT

- Brighter displays for digital cameras, mobile applications
- Higher resolution for phones, portable DVD players
- Faster response to reduce “blurring” that can occur with LCD displays.

Trend 2 – Replace external IC's with "System-on-Glass"

Typical cell phone display

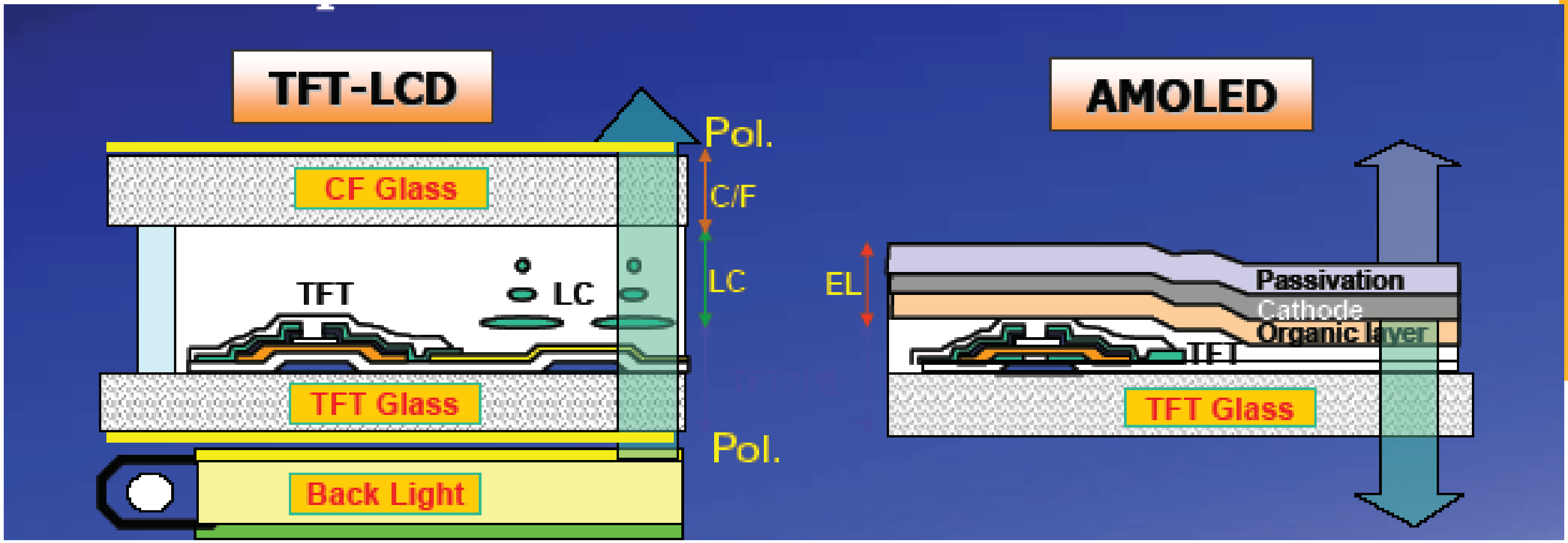


External IC's

SOG Panel

Integrated Drive Electronics

Trend 3 – Transition to OLED



Samsung Electronics

- Active emitters replace liquid crystal + color filter + back light to give improved performance, lower cost
- Volume production of OLED displays still requires further improvements:
  - OLED material lifetimes need to be extended.
  - TFT’s need to be redesigned to support higher current loads.
  - OLED yields still much lower than LCD.

Two Types of Silicon Transistors

<u>Amorphous Silicon</u>

- Used for majority of displays today.
- Well suited for LCD TV.
- Higher power requirement than p-Si.
- Not able to support high speed needed for SOG.
- Challenged to support OLED TV due to material degradation under high currents.

vs.

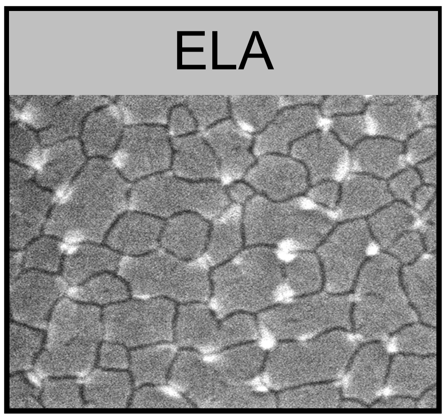
<u>Polycrystalline Silicon</u>

- Increasing use for small & medium displays.
- Smaller, faster TFTs.
- Supports trend towards System-on-Glass.
- Poly-Si TFT’s very stable under high current load needed by OLED
- Yield of poly-Si process has been lower than a-Si due to limitations in Crystallization process step

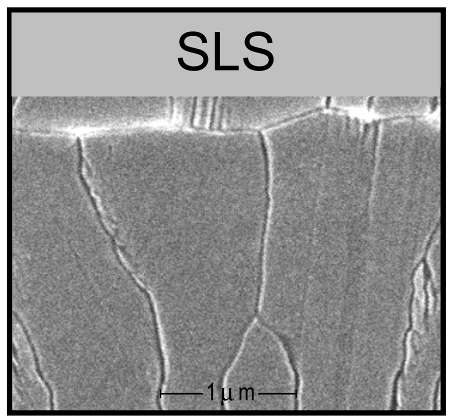
CYMER®

Laser Crystallization

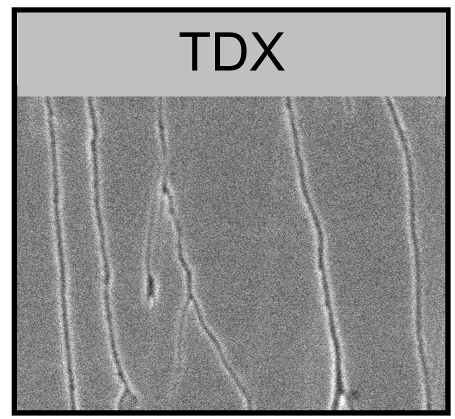
- Excimer laser annealing is the most common technique for making poly-Si
  - XeCl or XeF laser used to melt local region of a-Si.
  - Silicon crystals are formed during cooling.
  - Challenge is to create large crystals to produce high electron mobility.



mobility ~ 75 cm2/Vs

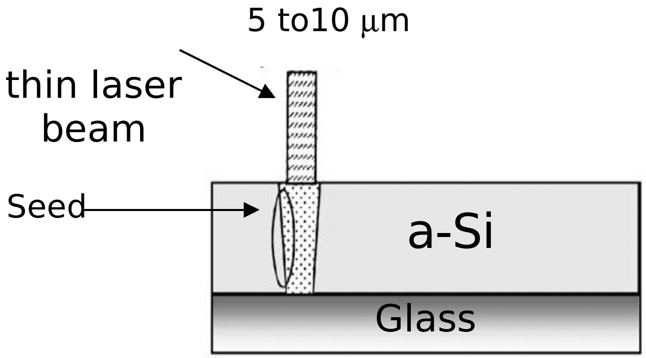


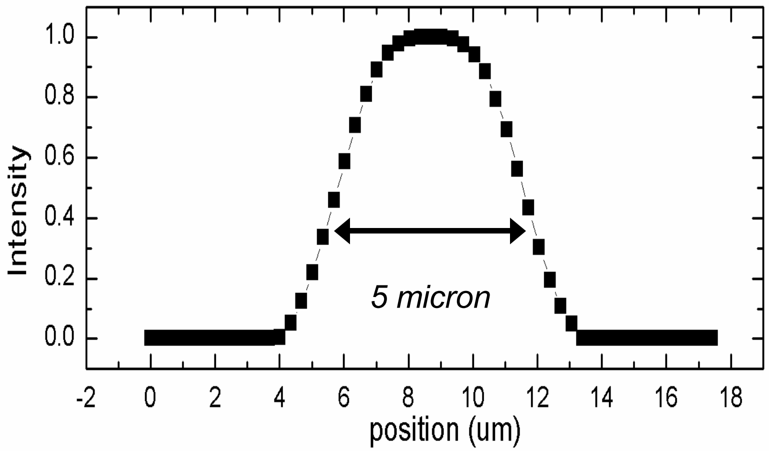
mobility ~ 150 cm2/Vs



mobility ~ 300 cm2/Vs

Crystallization Process

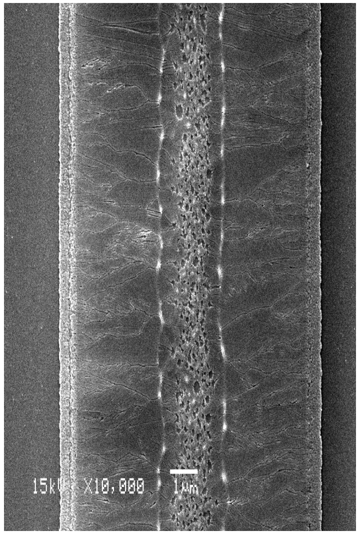




A thin beam is used to achieve "Lateral Growth" with growth initiated from seeds at liquid-solid interface

Beam is long enough to expose glass in single pass

Beam Properties: 730 mm x 5 micron.

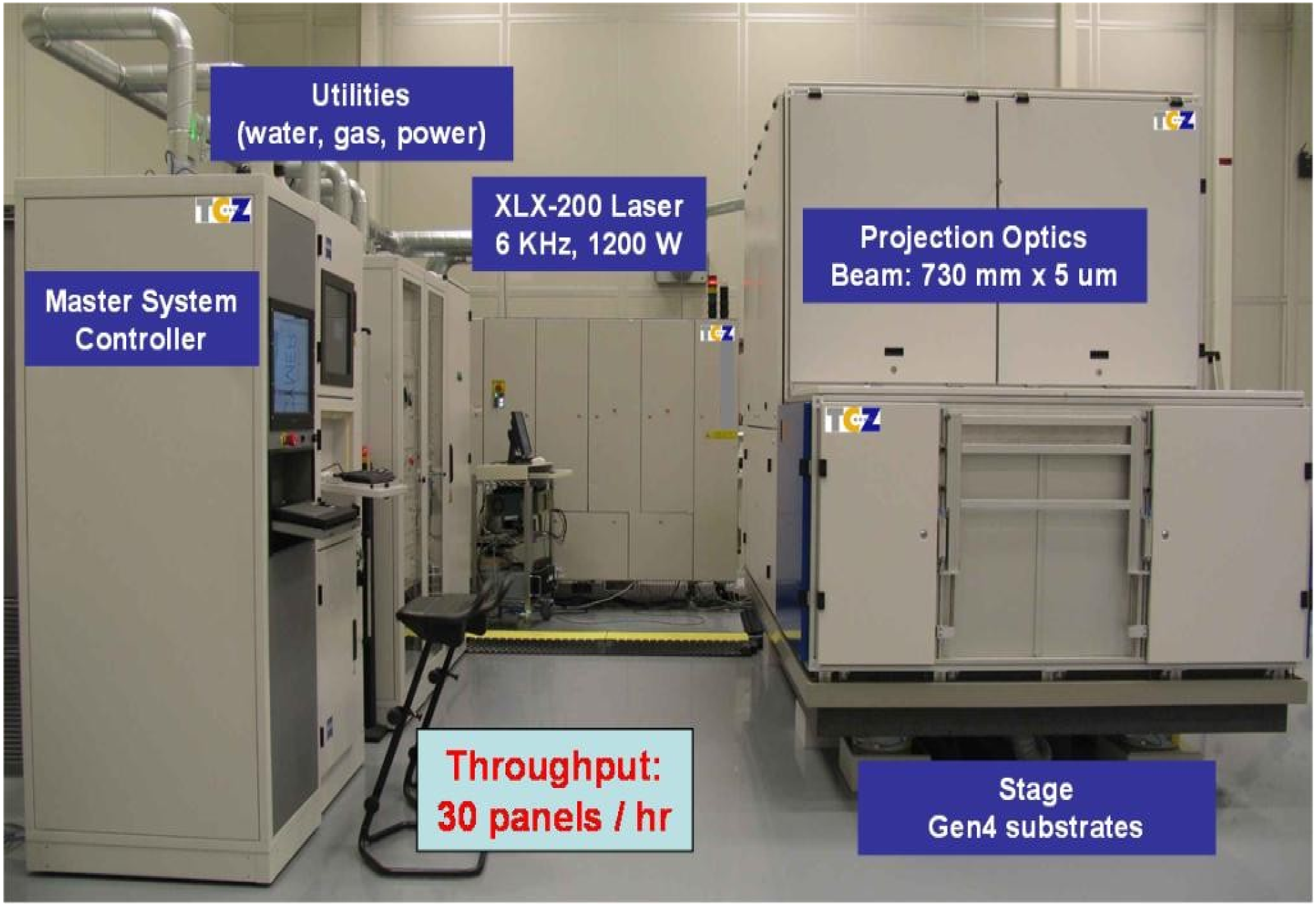


Agenda

- Introduction
  - Use Case
  - Players
- Flat Panel Display (FPD) Overview
  - Market
  - Process
- Control System Architecture
  - Hardware
  - Software
- PostgreSQL’s Roles
  - What
  - How

CYMER®

The Hardware

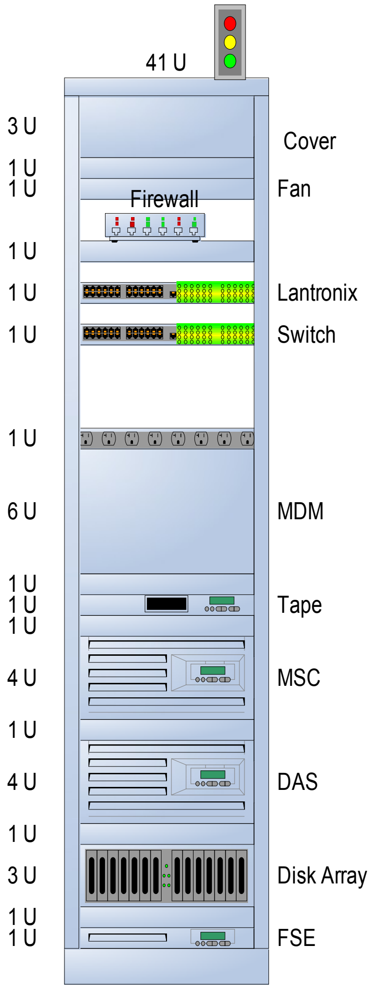


CYMER®

Master Control Cabinet (MCC) Overview

- Dual Opteron server with 8GB RAM running CentOS for control (MSC)
- Dual Opteron server with 8GB RAM running CentOS for data acquisition (DAS)
- One dual P4 server running Windows Server 2003 for diagnostic utilities
- 3.5 TB Storage Array
- Cisco Switch and PIX Firewall
- 16 Port Lantronix for serial to TCP/IP conversion

MSC – Master System Controller server
MDM – Master Distribution Module
DAS – Data Acquisition Service server
FSE – Field Service Engineer server



Hardware Devices

- MSC and/or DAS communicate with:
  - 1200 W, 6 kHz Xenon Fluoride DUV Laser
  - Beam Delivery Unit with Active Beam Steering and Stabilization
  - Active Illuminator Auto Focus Control
  - Precision Motion Control (stage)
  - Projection Optics Module (POM) with Motor Control and Metrology Sensors (temperature, O2, beam energy, beam profile)
  - Electrical and Fluid Utilities
  - Unknown (in advance) Material Handler
  - Unknown (in advance) Factory Automation Host

CYMER®

Software Design Objectives

- High Availability
  - Run reliably in production environment 7 x 24 x 365.
- Distributed
  - Hard real-time requirements handled by the devices.
  - Overall coordination, control, operator feedback, and data collection handled by MSC.
  - Allow for multiple paths of information flow (e.g. local operator control and factory host).
- Adaptable
  - Capable of controlling and monitoring an arbitrary set of intelligent devices.
  - Allow new device types to be added, or modification of existing devices, without entire rewrite/recompile.

Implementation Methods

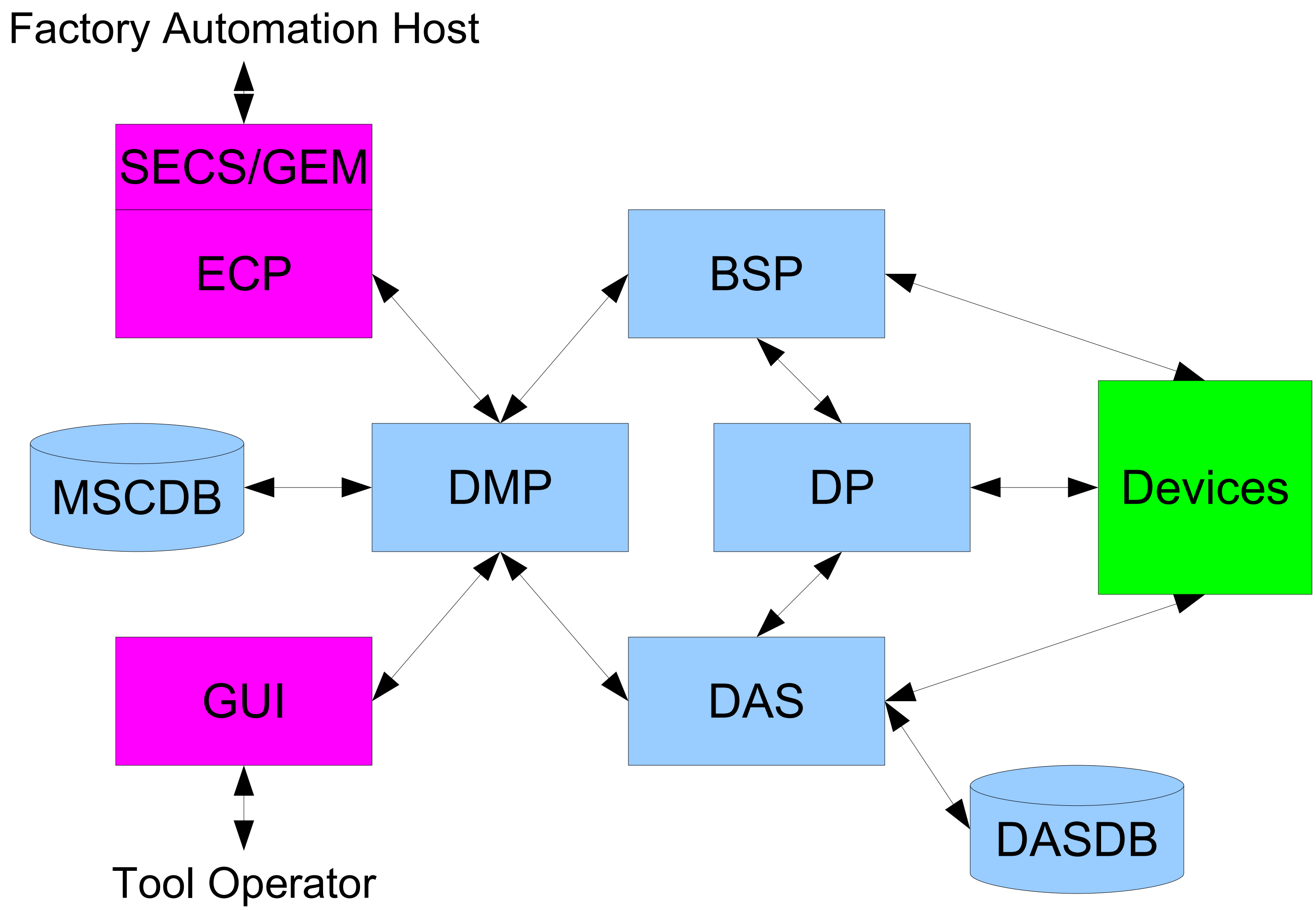
- Device commands are stored and loaded dynamically. They are not hard-wired in the code. They are be cached for performance reasons as required.
- The number of parameters for any task or system level device configuration is not fixed by the GUI design. Additional parameters added to a task do not involve coding modifications to the user interface screens.
- TCP/IP is the main type of communication to the devices. When required, use the Lantronix to convert serial communication protocols.
- The system does not block when waiting to get response from devices for timing critical commands. They are executed asynchronously

CYMER®

Implementation Methods

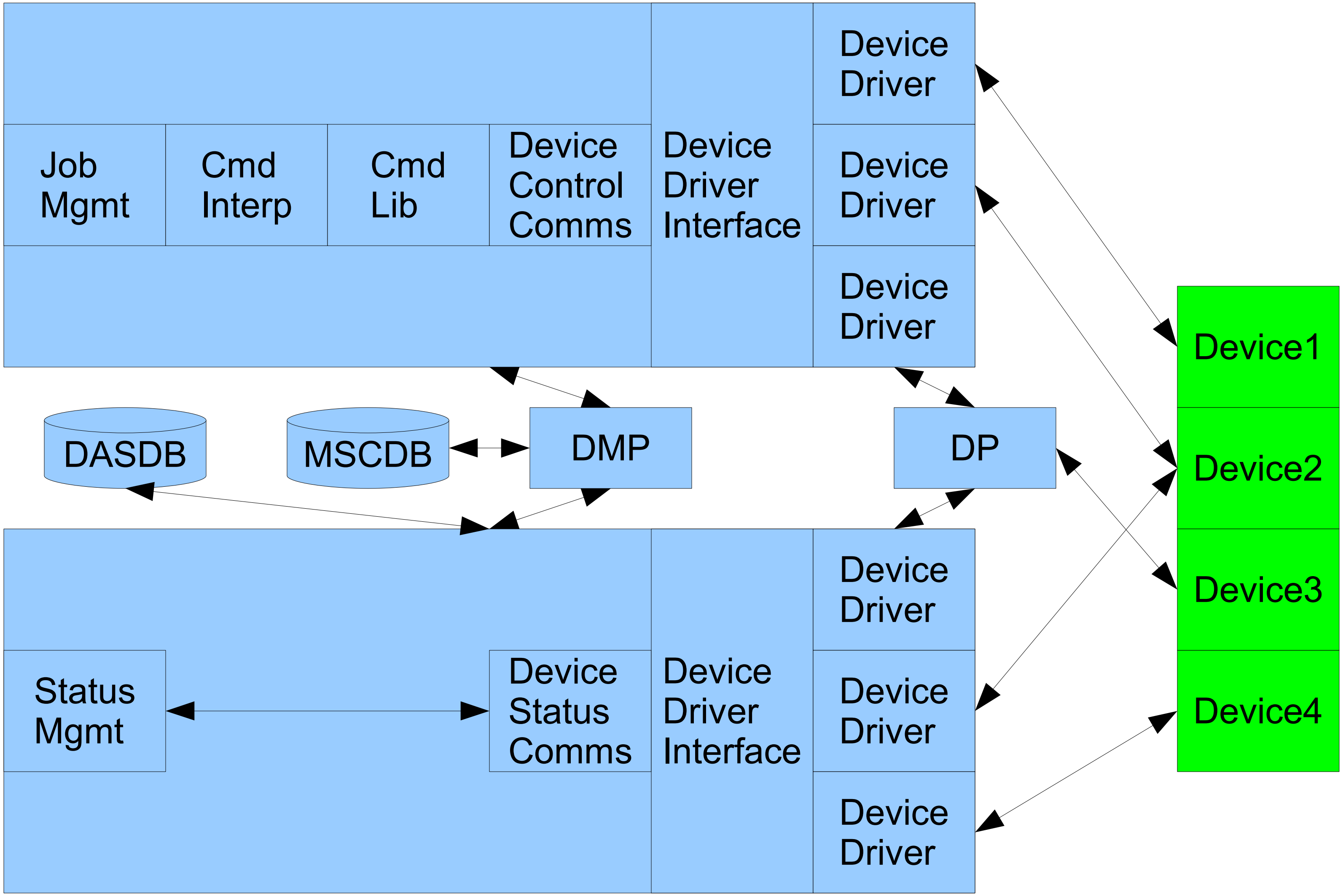
- Bulk of the software is C++. The GUI is written in Java. Control and data collection "Tasks" are stored in parameterized script form, similar to PostgreSQL procedural language functions. Tasks may access devices via Command Library calls.
- Work is done on the system by running Projects or Recipes. Projects are created by the operator using the provided Tasks. A Recipe is a special Task.
- At runtime, the Command Interpreter is used to parse Tasks, thereby executing device commands. Control logic is thus moved from compiled code to dynamic data.

MSC Software Architecture

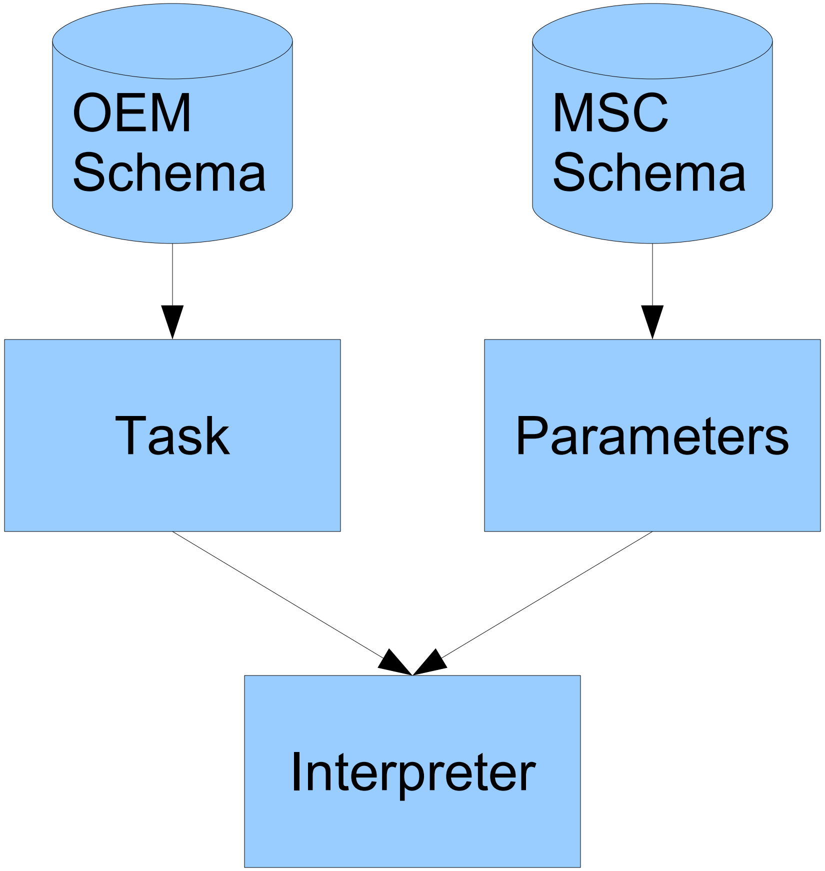


CYMER®

MSC Software Architecture



Task Execution



Device Support Example

- Laser Serial Connection
  - Device Driver – about 100 lines C++
  - Device Class – about 500 lines Python
  - Device Configuration – about 100 lines SQL

Agenda

- Introduction
  - Use Case
  - Players
- Flat Panel Display (FPD) Overview
  - Market
  - Process
- Control System Architecture
  - Hardware
  - Software
- PostgreSQL's Roles
  - What
  - How

Role of PostgreSQL

- What
  - Configuration
    - User Controlled
    - OEM Controlled
  - Data
    - Status
    - Streaming
    - Logging
- How
  - General
  - Device Abstraction
  - Data
    - Status
    - Streaming

CYMER®

Role of PostgreSQL - Configuration

- User Controlled
  - System configuration – global level variables that allow operator control of system behaviors. Examples:
    - Connection timeouts
    - Calibration factors
    - Device enable/disable
  - Defined users
  - Project and recipe
    - Project level parameters
    - Included Tasks and associated parameters
    - Permissions
  - Chart Templates

Role of PostgreSQL - Configuration

- OEM Controlled
  - Bootstrap user configuration
  - GUI “Look and Feel”: color, tabs, labels
  - Job execution: encrypted Task and Class scripts
  - Internal Setup
    - Routing Tables for DMP
    - Message Mapping
  - Device Setup and Adaptability
    - Configuration: “Gold” lists
    - NLF-2-DF translation
    - Device list – table of device driver shared objects to load dynamically

Role of PostgreSQL - Data

- Status loop data
  - Continuous polling loop, throttled to specific, relatively slow, data rate.
  - Separate thread per device, synchronized in time by “sync thread”.
  - Hundreds of attributes, flexibility required. Stored in PostgreSQL using Attribute-Value form.
  - Buffered and bulk copied. Partitioned by time/device.
  - Used with built in charting and reporting features to monitor system health
  - Examples:
    - Laser energies at various points in the system
    - Beam profiles near the exposed substrate
    - Stage and metrology positions
    - Temperatures throughout the system

Role of PostgreSQL - Data

- Streamed data
  - High data rate, up to 6 kHz
  - Many attributes
    - Laser stream ~ 24 fields
    - Metrology stream ~ 70 fields
  - Stored in PostgreSQL using Normalized form
  - Buffered and bulk copied. Partitioned by time/device.
  - Used with built in charting and reporting features to monitor system health
  - Examples:
    - Laser energies and related properties
    - Beam profile measurements

CYMER®

Role of PostgreSQL - Data

- System logging
  - All MSC components log to a central logger daemon
  - Many system Devices also log directly to the same logger
  - Logger daemon
    - buffers
    - writes to PostgreSQL database
    - sends messages in parallel to GUI via DMP
  - GUI allows
    - filtering of live log by device, severity
    - report generation by date/time, device, severity

CYMER®

General Techniques

- Schema: OEM schema is used for bootstrapping and MSC schema to overlay user settings. Application defaults to OEM schema settings in the absence of MSC schema settings.
- Partitioning: Extensive use of partitioning by timestamp and device for status and streaming data. Allows simplified and quick enforcement of data retention policies.
- dblink: The DAS and MSC servers are separated in order to ensure data collection and control do not interfere with each other. Occasionally data captured by the DAS is needed on the MSC.
- Extensive use of functions. E.g. custom C extension functions for XML and HMAC
  - XML messages used in DMP and with GUI4J
  - HMAC used for Project/Recipe locking

Device Abstraction – NLF-to-DF

- Neutral Language Format to Device Format (NLF-to-DF) conversion
  - Entity-Attribute-Value form
  - Provides mapping from generic command used by Task script to device specific command
  - Allows replacement of device with another having same function but different commands and responses
  - Examples:
    - Material handling robot
    - Newer model laser

Device Abstraction – NLF-to-DF

In PostgreSQL:

```
create table oem_device_command
(
        device_id               int not null,
        nlf_command_name        text not null,
        df_command_name         text not null,
        nlf_data_conversion     text,
        df_data_conversion      text
);

copy oem_device_command from stdin;
1       GET_DIAGNOSTIC_DATA   DI%?    INTC    HEX4
```

Device Abstraction – NLF-to-DF

In Python, Laser Class:

```
self.LaserInternalEnergyAvg=107

def Diag(self,Value):
    return int(self.Device('GET_DIAGNOSTIC_DATA',Value))
```

In Python, Task Script:

```
IntEng=[]
while time.time() < endCaptureTime:
   IntEng.append(laser.Diag(laser.LaserInternalEnergyAvg))
```

Device Abstraction – Device Variables

- Device Variables
  - A few attributes are common to all devices, but many are unique per device
  - Allows replacement of a device with another having the same function but different default behaviors
  - Simplifies addition of new devices

CYMER®

Device Abstraction – Device Variables

In PostgreSQL:

```
create table oem_device_variable
(
        device_id               int not null,
        variable_name           text not null,
        variable_label          text,
        variable_uom            text,
        variable_data_type      text not null,
        variable_type           text not null,
        variable_default        text,
        variable_required       char(1) not null,
        variable_validator      text,
        variable_order          int,
        variable_list           text,
        variable_internal       int,
        variable_public         int not null
);
```

CYMER®

Device Abstraction – Device Variables

In PostgreSQL – example device, simplified:

variable_label	variable_data_type	variable_type	variable_default	variable_validator	variable_list
Device Enabled	TEXT	SELECTION	FALSE	IS_ALPHA	TRUE;FALSE
Send Full Status to GUI	TEXT	SELECTION	TRUE	IS_ALPHA	TRUE;FALSE
Device Connection Attempts	INT	INPUT	3	INSTANTCHECK:[...]	
Device Connection Timeout	TEXT	INPUT	5	IS_ALPHANUM	
Command Placeholder	TEXT	INPUT	%	IS_ALPHANUM	
Read Terminator	TEXT	INPUT
IS_ALPHANUM	
Device Library	TEXT	INPUT	libmscomegabus.so	IS_ALPHANUM	
Log Level	TEXT	INPUT	0	IS_ALPHANUM	
IP Address	TEXT	INPUT	ice9lantronix	IS_ALPHANUM	
Port Number	TEXT	INPUT	3004	IS_ALPHANUM	
Service Mode Supported	TEXT	SELECTION	No	IS_ALPHA	No;Yes

Data – Status

- Attribute-Value form
  - List must be flexible
  - Thousands of available attributes; hundreds deemed interesting
  - Data rate is relatively slow
- Partitioned
  - by device
  - by timestamp
- Joined across devices by synchronizing thread timestamps
  - Each device has own collection thread, but synchronizing thread used to allow cross-correlation of data

CYMER®

Data – Status

In PostgreSQL:

```
create table das_status_log
(
        device_id       bigint,
        status_timestamp timestamp with time zone,
        status_name     text,
        status_value text,
        status_uom text
);

create table das_status_log_[year]_[month]_[week]_[device]
(
        CHECK
        (
                device_id = [device_id] and
                status_timestamp >= [startdate] and
                status_timestamp < [enddate]
        )
) inherits (das_status_log);
```

CYMER®

Data – Status

In PostgreSQL:

```
select status_timestamp,
       status_name,
       status_value
from das_status_log_2007_05_1_7;
```

```
       status_timestamp           |  status_name    | status_value
----------------------------------+-----------------+-------------
 2007-05-01 00:00:00.055783+02 | Stage_CX        | 99.9961
 2007-05-01 00:00:00.055783+02 | Stage_CY        | 5.0022
 2007-05-01 00:00:00.055783+02 | Stage_CZ        | 0.0006
 2007-05-01 00:00:00.055783+02 | real_timestamp  | 2007-Apr-30
                                                    23:59:59.839861
                                                    +0200
```

Data – Status

CYMER®

Data – Streaming

- Normalized form
  - Relatively fixed set of attributes
  - High data rate
- Partitioned
  - by device
  - by timestamp
- Joined across devices by synchronizing shot-count
  - data is streamed at rate of one record per laser shot
  - streamed data shot record tagged with unique shot-count number
  - data across devices correlated on synchronizing shot-count number

Questions?