CYPRESS SEMICONDUCTOR CORPORATION
INTERNAL CORRESPONDENCE

DATE : July 15, 2005
TO : Shane Denton
AUTHOR : Ken Freed
TITLE : CTI Automation VT2100 Die Retest Status. Updated Efforts & Schedule
PURPOSE : Update of June 15, 2005 memo, based on new knowledge

This memo represents an update to the June 16th plan (memo KXF-131) due to:

Good News:

1. We found out that memory constraints are NOT a problem
   - hence we do not have to breakup the dprc executable into smaller units in order to fit it (and our enhancements) into memory.
2. We found that there are some very good "user hooks", as part of the dprc code, which allow calling and passing of information to user written programs at certain events.
   - We have to write some stubs to log which of these events are called, in what order, with what available information, during wafer testing.

Bad News:

3. We found out that we cannot debug screens through the turboC debugger running on top of Desqview
   - no matter how we adjust the system, we cannot reliably switch between code/debug and screen views. The keyboard gets unavoidably confused.
   - Recoding the DESQview panels to use TurboC graphics (rather than low level DESQview panel calls) seems to work
     - after a cursory examination of several packages on the web, it was decided to use the "tcu_32a" msdos panel package.
       - A few screens were roughed out and put into the dprc code. They seem to work and debug OK using TurboC under DESQview.
       - Code for screen panels under TCU_32a are much simpler than those in DESQview. We can eliminate a lot of code.

4. We found that server-PC file transfer communications is unreliable in the desk-debug environment.
    - the present system opens files over the network, and transfers them (4K at a time) to the destination.
        - This fails frequently at various points (file opening, block transfer) in the desk debug environment.
            - File transfer is slow (due to intermitant problems?) in production as well.
    - We have to retrofit this to use (more reliable) Dos-ftp. This will also allow us to reduce code size and simplify the dprc code
        - there is lots of "netbios" code in dprc which was used in the Apple-Tops network and server. We should remove it.

Architecture Change

Original: Based on conversation with AMS, the original plan was to issue "retest" commands after each die.

AMS said that the CMI code to do this is very versatile; e.g.,
- If a certain number of retests in a row are really bad, then you can forget about further retesting (since you can conclude that defective die really are bad).
- Conversely, if so many retests in a row are really good, then you can abort further testing.

New: Create a "retest" map, and test the wafer a second time with it:

1. Let the wafer test as is presently done
2. Leave the wafer on the chuck
3. Figure out which die in the map should be retested
4. Create a map with these "retest" die as bin 1's and all others as bin 0's.
5. Retest the wafer to the new map
6. Compare the original and retest maps to see which die are really defective
7. Create a final map and allow the wafer to be removed from the chuck.

While this approach is not as versatile as the original, it will (hopefully) be simpler to implement (i.e., less VT code to change).

Unknowns / Risks

Can we change the dprc code to:
1. Leave the wafer on the chuck?
2. Update the active wafer map being tested?
3. Initiate testing of the wafer

- or does VOS have control over these functions?

VT2100 Die Retest Project Efforts

Updated Project Effort:

	15July05 Plan	16June05 Plan
Total Project Estimate:	**65 days** of full time effort	62 days of full time effort
Total spent as of 16June05	- 14 full time days	
Total spent as of 15July05	- 13 full time days	
Total to go as of 15July05	38 full time days	

Updated Schedule:

Milestone	**15July05 Plan**	16June05 Plan
Go/no go decision on outstanding question resolution	**~ August 17, 2005**	
DPRC rebuild in debug	**~ September 7, 2005**	
Die Retest in debug	**~ October 5, 2005**	~ September 15, 2005

Work Completed

	Step	Description	Full Time Effort, Status
0	Feasibility	Created a highly stubbed out build to see what we are missing. Identified and got missing source code from Agilent Got DESQVIEW panel editor from the web, ordered DESQVIEW API books.	**Done** took ~ 14 days
1a	Build DPRC.EXE under TurboC and get it to run under desqview at desk	Turned out that DESQview hooks (api1.obj and api2.obj) depend on the version of DESQview being used. Had wrong version. Turned out that panels could not be debugged via turboC. Had to come up with something else. Still have to complete the roughed out panels Turned out files transfer from server only intermittently. Still have to come up with something else.	In progress ~ 11 days spent

Additional Feasibility Work Needed

	Step	Description	Full Time Effort, Status
2a	**Architecture Questions**: Stub out user hooks to determine what we can do.	We have to write some stubs to log which of these events are called, in what order, with what available information, during wafer testing. Need source code for "lotsum.exe", which is a user exit which is presently called when the wafer is done (D_WAFER_DONE_PIF)	In progress 1 day spent ~ 3 days total
2b	How can we leave the wafer on the chuck after testing?		~ 3 days
2c	How can we update the test map and kickoff a new test?		~ 3 days
3	**File Transfer Questions**: via Dos FTP	Need to see what software works under Dos and DESQview and what we can call from the dprc code. Tried several packages, but they didn't work under Dos or network card drivers were missing Can we quickly implement via rfc ftp socket programming ourselves?	In progress 1 day spent ~ 4 days total One package found and tested (it works) but it will cost $750. for development and license s/w.
4	**Decision point - can we do this project?**	Add'l question: Should we proceed with the dprc rewrite in any case? Do we need it?	~ 1 day to review
	Additional Feasibility Effort:	previous 16June05 had this at 62 days	**~ 12 days** of full time effort

Major Milestone: Go/No Go Decision

July 15, 2005 + 12 days / 0.7 availability = ~ **August 17, 2005**

DPRC Rewrite

	Step	Description	Full Time Effort, Status
5	DPRC rewrite	This is a rewrite of the existing dprc.exe module without die retest functionality. We have to make sure we are implementing die retest on top of a stable, debuggable base.	
5a	DPRC rewrite: screens	Recode all of the dprc DESQview screens to run under TurboC (via the tcu_32a package) so we can run dprc under turbo C debugging	~ 5 days
5b	DPRC rewrite: FTP file transfer retrofit		~ 2 days
5c	DPRC rewrite: Machine test	Need this new code to be backward compatible with present DPRC functionality. Need to get the debugger to work on a real live machine	~ 5 days

Major Milestone: DPRC rebuild in debug

August 17, 2005 + 12 days / 0.7 availability = ~ **September 7, 2005**

Die Retest:

- Subject to revision based on results from "Additional Feasibility Needed"
- The below assumes we don't have to change the rdpio.exe kla prober interface

	Step	Description	Full Time Effort, Status
6	Retest Design	Put together what we know works into a design document and think it through. Review with AMS,CSD,BRT Question: what should be our first pass retest criteria? - am assuming we don't have to scrape up a sort1 map for sort2 retest decisions in this first pass Don't want to loose sight of the big picture during implementation	~ 2 days
7	Code and test die retest logic	At this point we have a "normal" CTI automation project.	~ 12 days
8	Post Install	Post install EPRC, bug track down, documentation	TDB - **not included in this plan**

Major Milestone: Simple (first pass) DPRC die retest in debug

September 7, 2005 + 14 days / 0.7 availability = ~ **October 5, 2005**

	Total Project Estimate:	previous 16June05 plan had this at 62 days	**65 days** of full time effort
	total spent as of 16June05		- 14 full time days
	total spent 17June05 - 15July05		- 13 full time days
	Total to go as of 15July05		= 38 full time days togo