

# Lessons Learned: The Use of VHDL in the Development of Seven ASICs for the Operation of Displays on the F/A-18E/F Aircraft<sup>1</sup>

W.A. Hanna, J.M. Gladden, M.W. Macke  
McDonnell Douglas Aerospace  
St. Louis, MO 63166-0516

M.W. Cofer, Y. Kubba  
Kaiser Electronics  
San Jose, CA 95134

## Abstract

The development of seven ASICs required for the operation of two display-types on the F/A-18E/F was successfully completed, but the estimates for design and development based on advertised automation tools capabilities were extremely optimistic. The success of this development effort required the addition of more resources in terms of cost, time, and manpower. This paper provides lessons learned emphasizing different areas of improvement rather than accounting for every detail.

## 1. Introduction:

The F/A-18E/F is the US Navy's latest Attack Fighter development which is intended to serve the needs of the US Navy and the Free World for carrier-based attack-fighters for the first quarter of the 21st Century. McDonnell Douglas Aerospace (MDA) was selected as prime contractor for the F/A-18E/F development. Two major F/A-18E/F display development efforts, which are the subject of this report, are: 1.) Multi-Purpose Color Display (MPCD); and 2.) Up-Front Control Display (UFCD).

These displays are critical to increased pilot situational awareness, for both the success of the mission and the safety of the pilot. New technologies were selected for the display heads, namely, Active Matrix-Liquid Crystal Displays (AM-LCD), which in turn required the development of new display processing & LCD controllers to replace the previous CRT-based displays and controllers. Kaiser Electronics, San Jose, CA is the developer of the display processing & display systems. OIS of Troy, MI. is the developer of the AM-LCDs under subcontract to Kaiser Electronics.

The MPCD and UFCD display processing hardware share ASIC types which perform common functions for cost reduction through sharing of development and production among multiple subsystems. SRAs (Shop Replaceable Assemblies) are Printed Wiring Assemblies (PWAs). The main SRAs in use on the MPCD and UFCD are: 1) Graphics Video Processor (GVP), 2) General Purpose Processor (GPP), 3) Primary Input/Output (PIO), 4) Liquid Crystal Display Controller (LCDC), and 5) Video Processor (VP).

These highly integrated SRAs use processors, e.g. i960, large memories; and, key to advanced development, are the following ASICs: 1) Graphics Video Interface Processor (GVIP), 2) Graphic Video Mixer (GVM), 3) Graphics Generator (GG), 4) Video Processor (VP), 5) General Purpose Processor (GPP), 6) Primary Input/Output (PIO), and 7) Liquid Crystal Display Controller (LCDC)

---

<sup>1</sup> This Paper Is Cleared For Public Release by Naval Air Warfare Center, Patuxent River, MD 20670-5049

From the beginning of the MPCD/UFCD development, the ASICs were known to be larger and more complex than those developed for previous controls and displays programs. The SRAs were made smaller and more reliable by integrating major functions on the seven ASIC types. Advanced components from LSI Logic Corporation, using LSI Logic 300K family of ASICs (300K+ gates) including LCA300K, LCB300K, and LEA300K components. The 300K parts selected for this development have the following characteristics:

- 1) 100K-300K gate count; 2) 256-525 Pins for pin count; 3) 25MHz-100MHz Operating Frequency;
- 4) > 4K Bytes SRAM on Board; 5) The possibility of using Mega-Functions; e.g. 80C31 embedded controller and 6) 4-12 Watts Power dissipation. The GVIP ASIC has a user designed core processor which is a subset of MIL-STD-1750A ISA (Instruction Set Architecture).

KE formed seven teams to develop the ASICs. Each team consisted of 1 or 2 design engineers knowledgeable in ASIC development. Initially, small allocation was made for development test. The ASIC teams were to develop their ASIC designs using VHDL-based Register Transfer Logic (RTL) descriptions on state-of-the-art Engineering Workstations (100 MIPS+, 168+ MByte Memory, 1+ GByte Disks) all connected on an ethernet, using industry standard Electronic Computer Aided Engineering (CAE) tools.

In the ASIC industry, it is assumed that for the most part simulation will fall out of design and that all the steps beyond RTL description are highly automated (1-2 weeks effort per design step per ASIC). Six calendar months were allocated for RTL design and three calendar months were allocated for synthesis, mapping to fab libraries, auto place & route, scan insertion. It was also assumed that common modules will be shared between ASICs and that system design was complete prior to the start of the ASIC design. It took 12-18 months to complete ASICs design at significantly greater development costs.

## 2. Controls and Displays ASICs Summary:

Descriptions of the seven ASICs is beyond the scope of this paper. Table 1. Is a summary of features of the seven Controls and Displays ASICs.

**Table 1. Brief Description of Controls and Displays ASICs**

ASIC Title	Program	Vendor	Device	Pins	Pkg Type	Technology	Logic Gates / Mega Functions	Clks	Max Clk (MHz)	Complexity	Typ. Power	Remarks
Graphic Video Processor I/F	F18EF	LSI	LCA300317	340	COFP	0.6u CMOS	94.5K	2	25	M	1.5 w	1960 I/F and 1750A Based Core Processor
Graphic Video Mixer	F18EF	LSI	LEA300236	340	COFP	0.6u CMOS	22.4K	1	25	L	1.3 w	
Graphic Gen./AntiAlias	F18EF	LSI	LCB310K	340	COFP	0.6u CMOS	144K	1	25	H	4.1 w	ROM-1kx12 RAM-1kx36 FIFO-64x24 RAM-256x6(6) Graphics Engine
Gen. Purpose Processor	F18EF	LSI	LCA300415	340	COFP	0.6u CMOS	130K	5	16	L	2.1 w	
Peripheral I/O RE	F18EF	LSI	LCA300415	340	COFP	0.6u CMOS	115K	7	16	H	3.2 w	Complete. For Risk Mitigation
Peripheral I/O FE	F18EF	LSI	LCA300415	340	COFP	0.6u CMOS	141K	7	16	H	3.6 w	Complete. For Risk Mitigation
LCD Controller	F18EF	LSI	LCB310K	340	COFP	0.6u CMOS	40K	3	50	H	2.4 w	8031 Controller RAM-128x12(20) Because of Embedded Control Programming and
Video Processor	F18EF	LSI	LEA300415	340	COFP	0.6u CMOS	104K	4	66	H	3.3 w	RAM-128x13 RAM-256x8 (2) FIFO-64x36

Notes:

- /1 Includes Internal Scan and JTAG
- /2 Complexity - H = High  
M = Medium  
L = Low

### 3. ASIC Design Tools Utilized:

The ASICs development program had the following tools available at program start:

1. HP9000/700 workstations equipped with 168+ MByte RAM, 1 GByte Internal Disks, Mentor Graphics Tools for Schematic Capture (Design Architect), System-1076 (VHDL Compiler/Simulator), QuickVHDL, and QuickSim II. For synthesis Auto Logic and Synopsys tools were in place. Also, three Sun Workstations were equipped with LSI Logic Foundry Tools (LSI Logic Macrocell Libraries, Floor Planning, Placement & Routing). The LM1000 Hardware Modeler, Smart Models Library. The Quickturn Emulation System was later on obtained for this program to reduce risk related to co-design of ASIC hardware and the displays processing software.

2. As the program progressed more workstations were made available with a goal of at least one workstation per designer (some designers can be using more than one workstation at a time, running long simulations and/or long synthesis tasks, usually in the background). Workstations count reached 22. More memory was added to some of the workstations, and more licenses were obtained for all tools.

3. At the peak, usage exceeded a workstation per designer, and 2 or 3 shift operations were mandated on most ASICs. Also, additional workstations were brought in as leased equipment to meet design activity needs. Also, the single ring network was converted into two ring network because of some catastrophic network failures due to heavy network traffic. The disk server for this program reached 14 Bytes of disk space at the peak.

### 4. ASIC Development Lessons Learned:

The F/A-18E/F Displays program was a major Avionics development. ASICs were a major development item for this program. Both MDA and KE had important lessons learned. They can be summarized in issues related to different phases of design development: Requirements specifications, detailed design, simulation (Testbench and Test Development), synthesis, and ICs foundry interface (Layout, Automatic Placement & Routing, and Scan Insertion) as follows:

#### 4.1. Specifications:

Although detailed specifications for the MPCD and UFCD existed, there were written specifications for the SRAs. There were no detailed written specifications or requirements for the ASICs which in most cases implemented the major functions of the SRAs. Engineers proceeded to ASIC design based on individual interpretations of high level specifications. As expected, there were conflicts regarding these interpretations among ASIC and SRA teams, as well as within Systems and Software teams.

#### Lessons Learned:

ASIC development processes should address requirements, specifications, and development for ASIC designs, testbenches, and associated tests. Written specifications for the designs, testbenches, and tests can be developed concurrently, and, should overlap with detailed design. Although this approach appears to be less efficient in terms of hours spent, the improved system understanding, and requirements documentation will result in a more rapid and affordable product development cycle. One can not over emphasize the importance of detailed requirements documentation. It carries the design through the different phases of design to a successful and cost effective and timely completion.

#### 4.2. Detailed Design:

Some ASIC designs were scheduled at 1 person for 9 month manpower loading, based in part on the advantages possible when making use of high-level design versus schematic entry. It is possible to achieve these levels of productivity if individual designers are extremely experienced in all aspects of design and great care is taken to reuse designs and methodologies across multiple ASICs.

##### Lessons Learned:

Most ASIC designs required 12-18 months to develop and in some cases another engineer was added to help in design and to assist in resolving difficult design issues. Design and development of a major ASIC (100K+ gates) takes longer than 9 months to complete, Also, regardless of complexity, it is advisable to have more than one person involved in design as stated previously.

VHDL and Logic Synthesis are now the preferred approach to design. Modeling (or 'style') guidelines should be agreed to. The proper use of guidelines ensures model consistency and interoperability. This is especially important for VHDL modeling. Most HDLs use a preprogrammed or "built-in" logic value system to evaluate system behavior. VHDL allows users to define a logic value system appropriate for the level of abstraction required for the model. IEEE standard packages; e.g. std\_logic\_1164 should be utilized instead of vendor specific logic types. This will improve transportability and interoperability. The same holds true for math and text I/O type packages.

#### 4.3. Testbench and Test Development:

There is an ongoing assumption that simulation falls out of proper HDL based design. This notion is valid to some extent, although there are additional efforts required to take place in parallel with design description generation. A testbench has to be planned and developed, and test cases developed, to test the different requirements identified for the particular design. The Testbench design and test development times were planned to be at approximately the same level of effort as the design (1 person, 9 months). Test Development, we found out, can be significant effort up to 3 times the size of the design effort in extreme cases (e.g. programmable embedded designs). While the ASIC designers should participate in testbench and test case development, it is highly recommended that the bulk of the development test effort be done by other Specialists with similar qualifications for checks and balances. People tend to repeat their mistakes.

**Assumption:** Testbench and test development is on the order of design description generation.

**Actuals:** 1-3 months for Testbench Development (testbench, drivers and checkers)  
2-6 months for Test Development and matching Tests to Requirements (Compliance Matrix)  
1 week-2 months for Regression Testing as the design goes through final changes

##### Lessons Learned:

On the average, the number of engineers required to perform testing can be 1-3 times the number of designers. Test engineers developed testbenches (drivers and checkers for major functions of the ASICs), and then developed test cases which are combinations of VHDL code, force and trace files for the simulator, and assembly language code when mega-functions, such as 80C31 micro-controller are embedded in the ASIC and/or SRA. This is also the case for ASICs which interface to microprocessors.

One ASIC had interfaces to an i960 and a built-in core MIL-STD-1750A based processor. The development of assembly language brought in software specialists which added three people for periods ranging from 2 months to the duration of development when software was an integral part of ASIC operation.

Testbenches should be cross-referenced to hardware test plans, specifications, and drawings.

Decisions on the appropriate level to test are heavily influenced by the type of design. Stand-alone testing of an ASIC module or partition minimizes the required machine resources and the amount of code being tested while basic checkout of the module function is performed. This approach also allows the ASIC debug process to begin prior to the completion of the entire ASIC design (divide and conquer). Extensive module checking is more economical at the module level than at the ASIC level. The goal of this approach is to have a high degree of confidence that the module will work correctly when interconnected with other ASIC modules.

Large ASICs simulation activities require large amounts of processing power, memory, and disk storage. Deliberate decisions should be made to choose the correct tool set for the application. Benchmarks of real designs indicate a 10 to 15 times performance differential between VHDL simulators for the same RTL code (interpreted vs. compiled VHDL code, as well as code optimization related). The pay back of increased performance is realized every time a simulation is executed. Faster simulations and decreased test times translate directly to reduced hours and schedule as well as contributing to greatly increased test coverage.

#### **4.4. Hardware (Logic) Synthesis:**

“Synthesis is a process of converting RTL description into generic digital macrocells, which are mapped into foundry-specific macrocell libraries.” Synthesis tools are essential due to ASIC size and complexity, as indicated by high gate and pin counts, which reflect the wide data path (32-bit+). The synthesis tools available reflect the best thinking in computer science today. However, the tools are not mature enough to guide the user through the process. Some of our more experienced counterparts developed methodologies to counteract the limitations of the tools. Our experience here was limited. RTL designs were not trimmed to work with the limitations of the tools. In essence, we needed to limit the size of modules, and the types of VHDL constructs to meet the limitations of synthesis tools. Some of the modules sizes caused the tools to thrash (hundreds of CPU hours of speed/size optimizations which were counter productive). The synthesizer was asking for more memory and more iterations when in reality, it needed to say module too big for synthesis. This resulted in significant loss of time doing unnecessary iterations.

**Assumptions:** 1-2 weeks for design synthesis.

**Actuals:** 1-2 months for design synthesis if synthesis is performed as a single step. 1-2 weeks for ASIC top level synthesis if module synthesis is performed incrementally during the RTL design description phase.

#### **Lessons Learned:**

Two automated synthesizers were used for this program. One of the synthesizers was not up to the job. The other synthesizer performed well, if constructs and module sizes were constrained.

Detailed logic synthesis should not start in earnest until RTL simulations are completed. It is necessary to verify that the RTL description is synthesizable, as well as to determine if the synthesized design will fit in the target hardware. Logic synthesis, like logic simulation, can be resource intensive. If the same machine is to be used for both design functions, it can be important not to have these design tools executing concurrently. We had a situation where automated synthesis scripts were kicking in while design simulations were being performed. This was taking away a lot of processing power, thus slowing up the whole design process. A premature “rush to gates” can cause a significant negative impact on computer resource availability.

Logic synthesis should be modular. The appropriate size and coding style for modules can vary significantly, depending upon both the function description and the computer configuration on which the automated synthesis tool is executing. Tools and computers are changing rapidly. Modules to be synthesized should probably be limited to the order of 10,000 gates (with today's technology). Increases in computer resources, tool efficiencies, and improved user knowledge will make it possible to increase synthesizable module size. If the module to be synthesized becomes too large, machine resource requirements can grow beyond the machine's capabilities, and can result in unpredictable tool behavior. One of the designs on this program was not partitioned into smaller modules. As a result, the synthesizer was thrashing while trying to meet size and timing constraints. Unfortunately, the tool did not provide feedback that this was the case.

Data type and construct selection have a significant impact on a synthesizer's ability to function correctly. Vendor-specific packages, often promoted as an efficiency enhancements, should not be used. They lead to a loss of portability and consistency, and can lock a user into an inadequate tool suite. For VHDL, we recommend the use of the IEEE-Standard-Logic Library. Some of the tools used proprietary logic libraries.

Automated tools help significantly, but logic synthesis is still a complex task. Care should be taken to ensure that results are consistent and repeatable. Most logic synthesis tools contain many types of settings for controlling the synthesis process. These settings should be captured, and documented using scripts which are shared among different modules. For efficiency and repeatability, consistent settings (scripts) should be used from module to module, and from design to design.

#### **4.5. Layout, Automatic Placement & Routing, Scan Insertion and Gate Level Simulation:**

This function is the interface to the foundry. There is another implied function which is gate level simulation using the foundries' golden simulator. For this program we used the simulator, used previously for VHDL simulation, with more detailed timing parameters, instead of using golden simulator. This approach was more efficient than using golden simulator which is much slower than the one we used. The tools for this phase are controlled by the foundry. We could have done better in this area if the tools were more up-to-date. The particular foundry is planning on introducing improved tools. For example, the gate level golden simulator is extremely slow.

For scan insertion, KE retained a consultant. The consultant developed some testability and scan insertion guidelines. Somehow, the designs were always found to be lacking for good testability measure, which meant design revisions which added to the schedule. Again, with better documentation of design guidelines for testability, design can be made testable the first time. On a new Avionics Upgrade Program testability and scan insertion is being done using third party tools, by the designer during the detailed design phase, with milestones for testability at the module level.

**Assumptions:** 10 weeks scheduled.

**Actuals:** 10 weeks to complete after pathfinder (GG).

#### **Lessons Learned:**

Foundry Interface tools need to be improved. Testability guidelines need to be detailed and made known during RTL design phase. Testability factors should always be taken into account in the design. They should not be added after the fact. Also, there should be no golden simulators. The same simulator used in RTL design should be used to accept detailed gate level timing to reduce design cycle time by eliminating another activity and all the associated costs of tools and experts. This is possible if foundry interface tools are commercially available Off-The-Shelf, not the home grown type. We also

recommend close working relation with lcs foundry, on this program we had a weekly meeting scheduled with LSI.

#### **4.6. The "Computer-Aided-Engineering" Factor:**

ASIC development processes are highly complex and make extensive use of automation. A philosophy for a Computer-Aided-Engineering (CAE) infrastructure becomes important for the guidance it provides when sorting through the myriad of choices. The infrastructure should address applications, computing platforms, operating systems, networking, graphical user interfaces, peripherals, system support, and product data. CAE requirements should be derived from product development processes.

The organization should have a good understanding of the development process before attempting to automate. Commercially available, "Best-in-Class" tools should be acquired when possible to support the functions of the development process. National and International standards should serve as a foundation for CAE integration. Data standardization rather than tool standardization, should be attempted. Every effort should be made to use a unified design database, so that design and product information is only entered once.

"Best-in-Class" tools are required when pushing technology limits, Rarely does a single CAE vendor supply "Best-in-Class" tools across all major development functions. Electronic CAE tools, in general, are not robust, due to their complexity and also in part to the limited market they serve. Tools and data interfacing, even within a single vendor's tool set or environment, can be difficult. Off-the-shelf CAE interfaces are desirable, but CAE tool interfaces are notoriously weak. The use of "standard's-based" data and tool interfaces offer increased flexibility, but usually at the cost of more machine resources to "translate" design data. ASIC development data typically covers multiple levels of abstraction and multiple design views. Large amounts of data are created, generated, and processed. It is critical that the data be organized in a manner so that both people and tools can easily view and manipulate it.

"Best-in-Class" tool approach can be more difficult to implement than a "turnkey" CAE system, but it minimizes risk associated with tool usage because each tool is the best for the particular function. This also requires tool environment's inherent "plug and play" philosophy. This approach dictates the use of in-house tool specialists experienced in interfacing tools and interacting with CAE tool vendors. CAE specialists should be integrated into the development team for the duration on the program. Their ability to off-load the design and test engineers from trouble-shooting various tool and network problems is a significant factor in meeting schedule. For this program, one lost day resulted in more than a month of lost effort, not counting the costs associated with the overall program schedule.

#### **4.7. The "Management" Factor:**

The original schedule and at least two major improvisations were milestone driven. Adaptive scheduling and contingency planning needed to be applied, and the situation continuously monitored. We lost some key technical specialists on this program, which is common in today's work environment. Due to poor or non-existent documentation in some areas, the work was not easy to continue when the individual left. Also, when tools do not work, important decisions have to be made to either repair the tool, or look for an alternate tool. For example, the main simulator was not able to handle large descriptions. However, we continued to add memory, when the proper solution was to switch to the more efficient simulator. We also, had a similar problem with the synthesis tools. In the case of the simulator, we kept the simulator because of management perceived risk of switching simulators. In the case of the synthesis tools, we switched from vendor A to vendor B synthesis tools. The writer believes that we should have done the same for the simulation tool.

## Lessons Learned:

Managing development programs requires sharp management skills which cover the gamut of PLOCC: **Planning, Leading, Organizing, Coordinating, and Controlling**. A certain amount of cheer leading is needed as well. Scheduling should be initially done based on most recent history and good rules of thumb. This program tried to fit schedule to program mandated milestones. None were met. As work proceeds, schedule has to be updated based on the reality of the situation. A good team will perform for you, a bad team will obviously not meet its schedule, in which case management intervention is needed to get the team on track by applying:

1) Team Building through better communications: formal (memos and news bulletins), and informal (one on one sessions, and quick meetings). Also, you need to improve team spirit through socials (company supported work lunches, pizza parties, Saturday donuts and sandwiches, picnics, soft ball games, ... etc.) and incentives (raises, promotions, bonuses) based on measurable performance improvements.

2) Performance Evaluation, and decisions related to changing assignments, need to be done only when it is absolutely necessary. Anytime, you loose team members, there is a setback. This is valid also for personnel in support areas not directly related to the program; e.g. computers operations, .. etc.

3) Continuous monitoring of schedule performance, and applying management science tools; e.g. forecasting methods, to predict future milestones achievements and completion dates based on team performance rather than common practice.

4) Good Rules of Personnel Assignment, adding personnel as needed to cover peaks and re-assigning personnel when no longer needed. Conditions existed such that personnel were reassigned prematurely, and it was not possible to bring them back.

5) Requirements for adequate documentation of designs, tests, and methods used. A documented design activity can substitute other personnel at a much lower schedule risk than situations in which documentation is sloppy or worse yet, non-existent. Methods used for S/W documentation are applicable and can be tailored to advanced electronic design; e.g. MIL-STD-2167 methodology.

## 4.8 Other Lessons Learned:

1. Minimize the use of subcontractors; i.e build up the in house design capability. If the design task & schedule requires the use of subcontractors, special management techniques are required to achieve the best use of subcontractors effort. Detailed documentation of effort and agreed to metrics for measuring performance and task completion are to be developed, explained, and used to manage the subcontractor's effort.

2. Use consultants where it makes sense; e.g. Synthesis, Layout, and ASIC emulation.

3. Maximize the use of off-the-shelf libraries; e.g. Smart Models, Hardware Models, etc.

## 5. Verified Rules of Thumb: ASIC Design Schedule Estimation

An industry rule of thumb was to estimate 50 gates/person/day hardware design productivity using a schematic approach (mid '80s common practice). With the use of synthesis, an improvement factor of 1-3 can be used; i.e. 100-150 gates/person/day, is now a valid estimate. This improvement will be negated if methodology is not well established, or team is not well experienced. These rules are based on actual cost data, not included because it is extremely competition sensitive. For ASIC designs on MPCD/UFCD:

Average Design Size = 100,000 gates (30,000 lines of VHDL code)  
 Gates Per Line of VHDL Code = 3 Gates/Line of Code  
 Calendar Development Time = 9 Months (~ 200 Days)  
 Designer Effort = 100,000 gates / 200 Days = 500 Gates/Person/Day  
 [Often quoted number. Not valid because it does not account for required design verification effort.]  
 Design & Test Development Effort = 75,000 Hours (For 5 ASICs) = 15,000 Hours/ASIC  
     ~ 1875 Work Days  
 Effective Design Productivity = 100,000 gates / 1875 Days =  
     ~50 Gates/Person/Day (Includes Systems Engineering and Development Tests)

## 6. Recommended Design Process Flow:

Based on this experience, Figure 1. is a highly recommended design process flow.

## 7. Summary and Conclusions:

The ASIC development for the MPCD/UFGD subsystems was estimated to be a 9 calendar months effort based on 500 gates/person/day level of achievement. It became a 12 to 18 calendar months effort. Also, 50 gates/person/day was only achieved. The cost of the program was also significantly greater than what was estimated originally. A more realistic estimate should have been 14-18 calendar months as follows:

### Conservative (Worst Case Schedule):

4 Months Requirements Specification  
 8 Months RTL development including testbench and test cases,  
 2 Months synthesis,  
 1 Month scan insertion,  
 1 Month for Gate Level (pre-scan, post-scan, pre-layout, post-layout) Simulation  
 2 Months foundry I/F activities which includes auto-place & route, gate  
level simulation, and testability reviews and mods.

### 18 Months

### Progressive (Typical/Expected Schedule):

3 Months Requirements Specification  
 6 Months RTL development including testbench and test cases,  
 1 Months synthesis,  
 1 Month scan insertion,  
 1 Month for Gate Level (pre-scan, post-scan, pre-layout, post-layout) Simulation  
 2 Months foundry I/F activities which includes auto-place & route, gate  
level simulation, and testability reviews and mods.

### 14 Months

### Aggressive (Best Case/Optimistic Schedule):

3 Months Requirements Specification  
 6 Months RTL development including testbench and test cases,  
 1/2 Month final synthesis (step is done incrementally per module completion),  
 1/2 Month final scan insertion (step is done incrementally per module completion),  
 1 Month for Gate Level (pre-scan, post-scan, pre-layout, post-layout) Simulation  
 1 Months foundry I/F activities which includes auto-place & route, gate  
level simulation, and testability reviews and mods.

### 12 Months

Because of turn-over, an ASIC design team should consist of at least two designers for the ASIC regardless of size of effort to be able to carry the design through. 1-3 ASIC test development engineer, and systems engineering should work closely with ASIC designers to assure correct interpretation of design requirements. ASIC test development should start very early with a good size development test team. The so called automated processes, namely synthesis, auto-place & route, and scan insertion should be integrated into the design activity with experts in the areas of synthesis, testability, and chip layout participating in the design process and not isolated from it!

Electronic CAE tools' maturity will always be in question. For this reason, the best tools should be selected up front. Thereon, the design team should work closely with tools' support personnel on maintaining tools and be willing to take some risk in changing tools when necessary. An example from our experience is compiled VHDL tools which became available after the development started and tools were committed. The compiled VHDL tools were an order of magnitude better than the tools used for the majority of the effort. The parts of this development that took advantage of the new tools were able to run in 1/10th of memory at a much higher level of performance (2-15 times CPU performance increase). Some Video Frame simulations were reduced from 28 hours to less than 8 hours, which is a phenomenal design productivity improvement. This was only possible by taking the risk of adding a major HDL tool after the program started.

The Electronic Design and Development Process needs more continuity. It is important to retain more technical leads by better rewards and increased recognition of their contributions. On this program, more technical management was needed.

Finally, all ASICs in this development reached completion and are working as specified. On the PIO, a conscious decision was to produce a REV-A part with reduced function to insure schedule success. Both the REV A and follow on part were produced and are fully functional. The MPCD and UFCD were major additions to the new F18 aircraft and were part of first flight which was successfully completed on December First, 1995.

### Acknowledgments

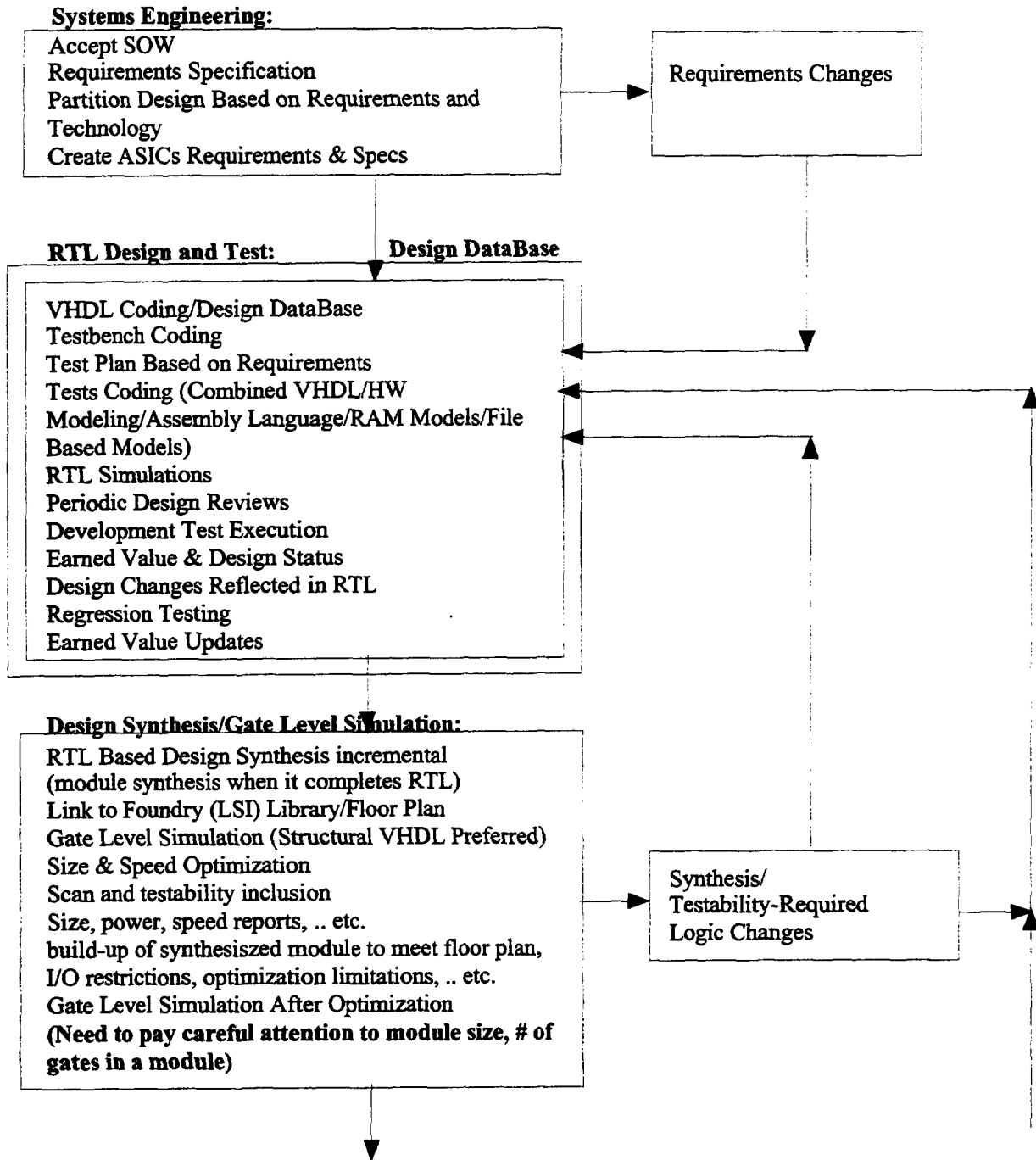
The authors are indebted to the F/A-18E/F management both at MDA and KE for the unqualified support to improve the design team in terms of manpower and design capability. In particular, we would like to recognize the relentless efforts of Gerry Daniels (F-18 General Manager), Norm Martens (Avionics Manager), Dallas Worrell (Displays Manager), Allen Gates (KE President), and Alan Brown (KE Executive Director of Engineering). At least 18 design engineers at KE and 8 design and test development engineers at MDA produced the results we are reporting in this paper. To everyone of them, our many thanks and eternal gratitude.

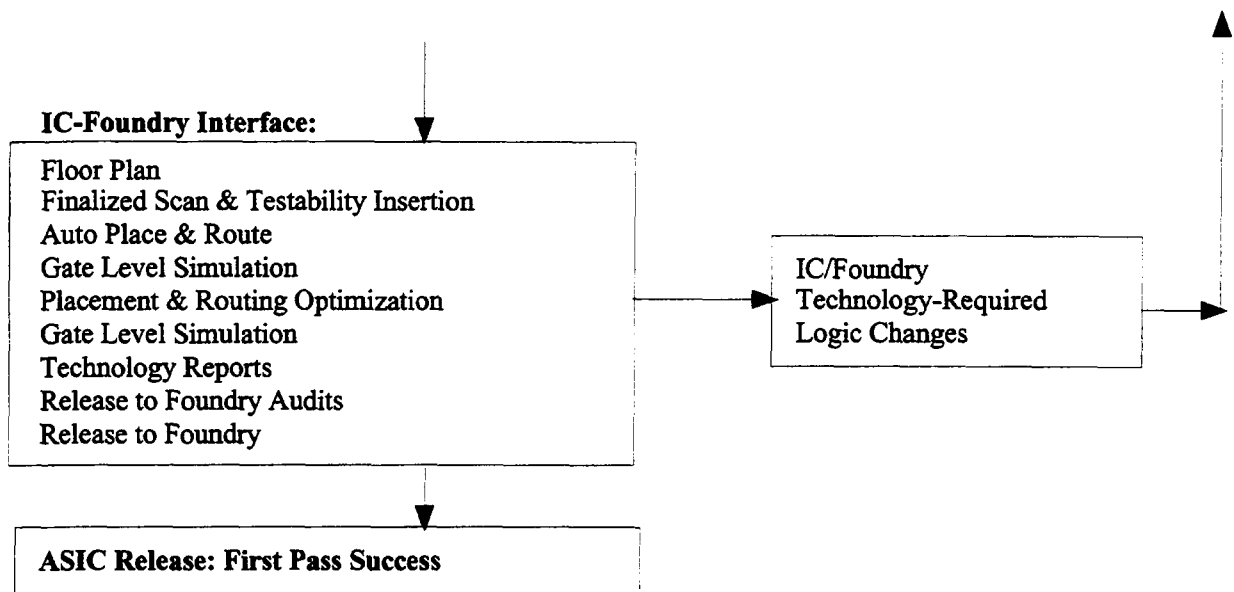
### References

1. F/A-18E/F Schedules and Program Activities: private communications
2. "VHDL: Hardware Description and Design;" Roger Lipsett, Carl Schaefer, and Carry Ussery, Kluwer Academic Publishers, Boston 1991
3. Introduction to HDL-Based Design Using VHDL;" Steve Carlson, Synopsys, Inc. 1991
4. "System 1076;" Mentor Graphics, Willsonville, Oregon, 1993.
5. "The Use of VHDL as the Data Base for the Complete Electronic Hardware Design Cycle;" W.A. Hanna; Simulation Multi-Conference: 26th Annual Simulation Symposium; Arlington, VA 29 March - 1 April 1993.

**Figure 1. Recommended Design Process Flow**

**ASIC Design Methodology Flow Chart-  
Based on Most Recent Experience in Developing  
7 ASICs for Controls of F/A18-E/F DISPLAYS (1994)**





**RAMARKS:**

- (1) Systems Engineering Accepts SOW, Customer Requirements & Specification Documents. Produces Partitioning of Subsystem to PWA, MCMs, ASICs, Microprocessors, I/O, Memory, ...etc.
- (2) Systems Engineering Produces SPECS & REQUIREMENTS for each ASIC
- (3) Systems Engineering Verifies Consistent Interfaces (I/O) between ASICs, Microprocessors, I/O, and Memory.
- (4) RTL Design Team Accepts SPECS & REQUIREMENTS and Produce Top-Down Partitioning and Test Plan cross-referenced to system requirements document(s)
- (5) Testbench planning and implementation in parallel with RTL coding.  
Design Tests are produced by RTL team while coding RTL.
- (6) Synthesis of major ASIC modules as soon as they complete coding & Verification of RTL model
- (7) Initial Floor Plan using IC-Foundry floor planning tool should be used to do synthesis phase partitioning (tight coupling of synthesis and auto-place& route)
- (8) Feedback to MCM and PWA team-mates early on to resolve pin-out issues
- (9) Use Test Compiler/Scan Insertion/Testability tools early on. Reflect scan in RTL if possible.
- (10) Synthesis dictated changes to logic have to be reflected in RTL code.
- (11) IC\_Foundry tools should also reflect back logic changes in RTL
- (12) RTL remains the DataBase. Incremental synthesis, floor planning, auto place&route as necessary with regression testing of RTL design base for all changes. Final regression is a complete RTL execution of all tests in the test matrix.
- (13) Earned Value is based on running RTL tests
  - 100 points divided among the different tests (100% coverage of all requirements)
  - full point is earned if the test is developed, run successfully
  - a weight  $0 < w < 1$  used if test was developed, did not run successfully
  - q weight of Zero if no test is developed to cover a requirement
- (14) Regression is the process of repeating some or all tests if design changes.  
Each regression is accounted for separately. For One major ASIC in 1994, we ran six complete regressions. Each regression consisted of close to 100 tests. Another ASIC ran 3 regressions only, each regression had close to 300 major tests. Both were first pass success when released to IC fab.