Henry Samueli School of Engineering

UCI Satellite (UCISAT)

Mission: Launch UCISAT-1 into LEO to capture Earth images with CMOS payload

Int'l Collaborator: Aoyama Gakuin University

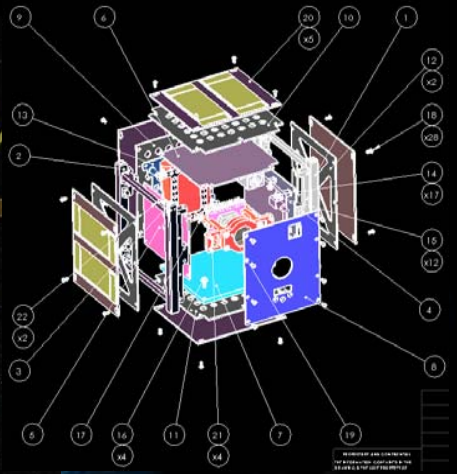
UC Irvine Team

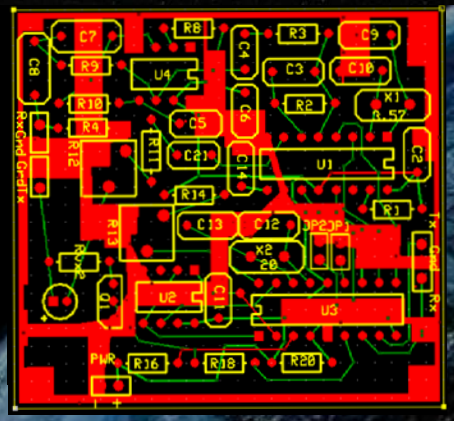
Aoyama Gakuin University Team

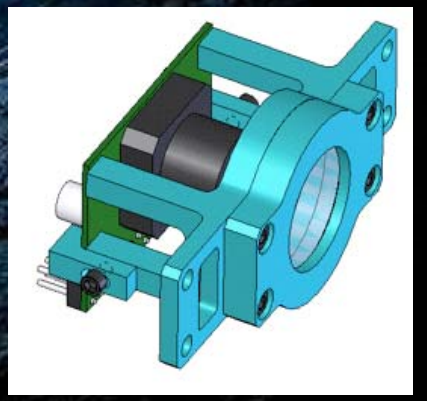
UCI Satellite (UCISAT)

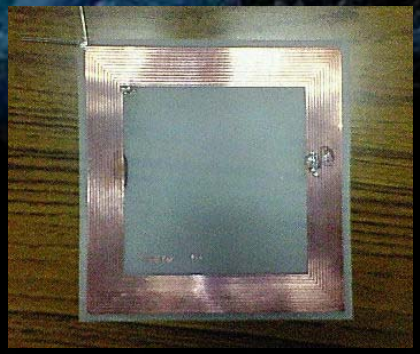
Subsystems:

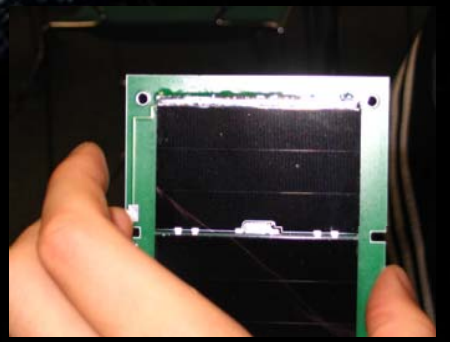
- *Attitude Control (ADCS)
- Communications
- C&DH
- Payload (CMOS)
- Power
- Structures









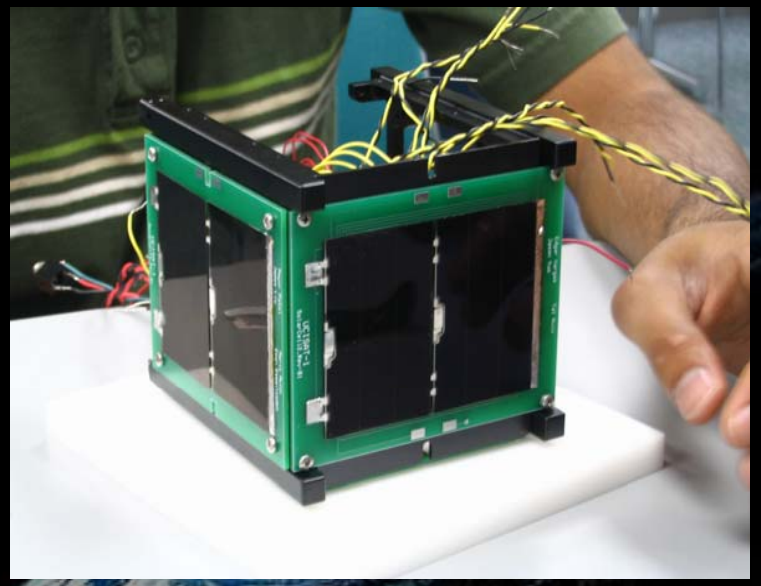


Henry Samueli School of Engineering

UCI Satellite (UCISAT)

Status:

- On-board communications manufactured & tested
- Ground station assembled and operational
- CMOS capturing and storing images
- Power PCB layout on next rev
- ADCS and C&DH hardware undergoing simulations/testing before design is finalized



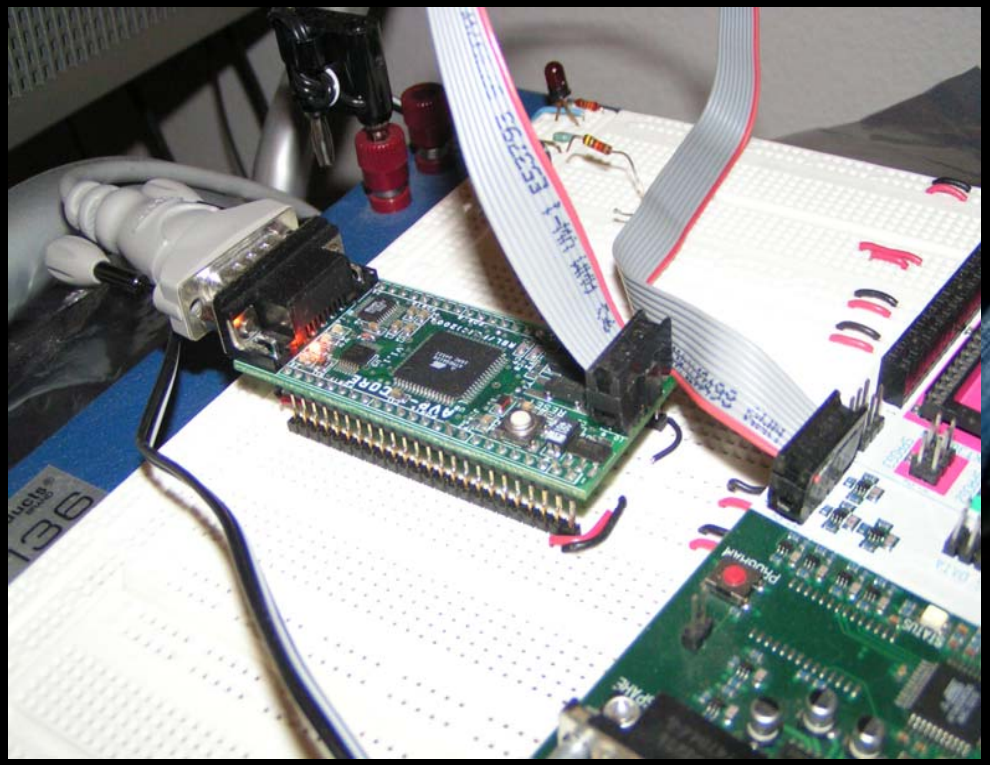
Launch:

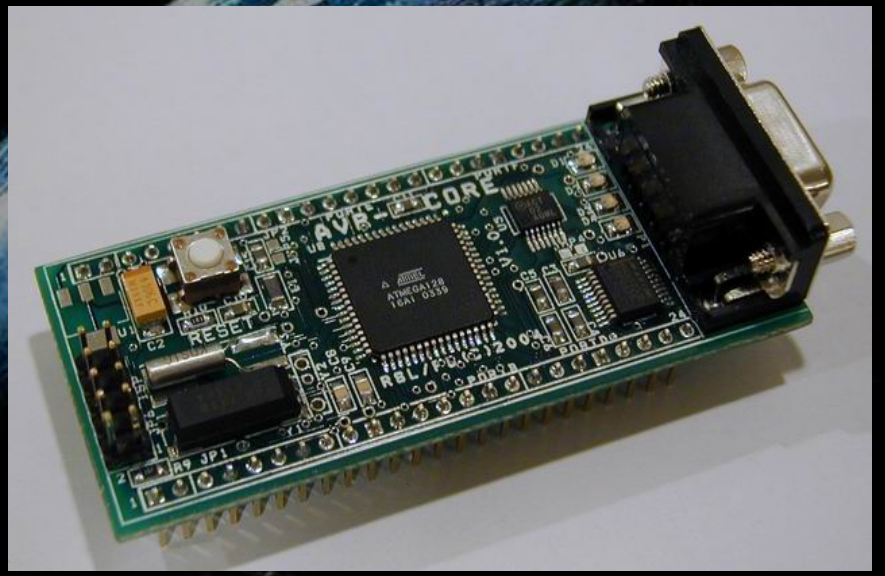
Seeking launch aboard Dnepr III in late 2008

UCISAT-1

Failure Recovery System

Presented by Juan Osuna





Henry Samueli School of Engineering

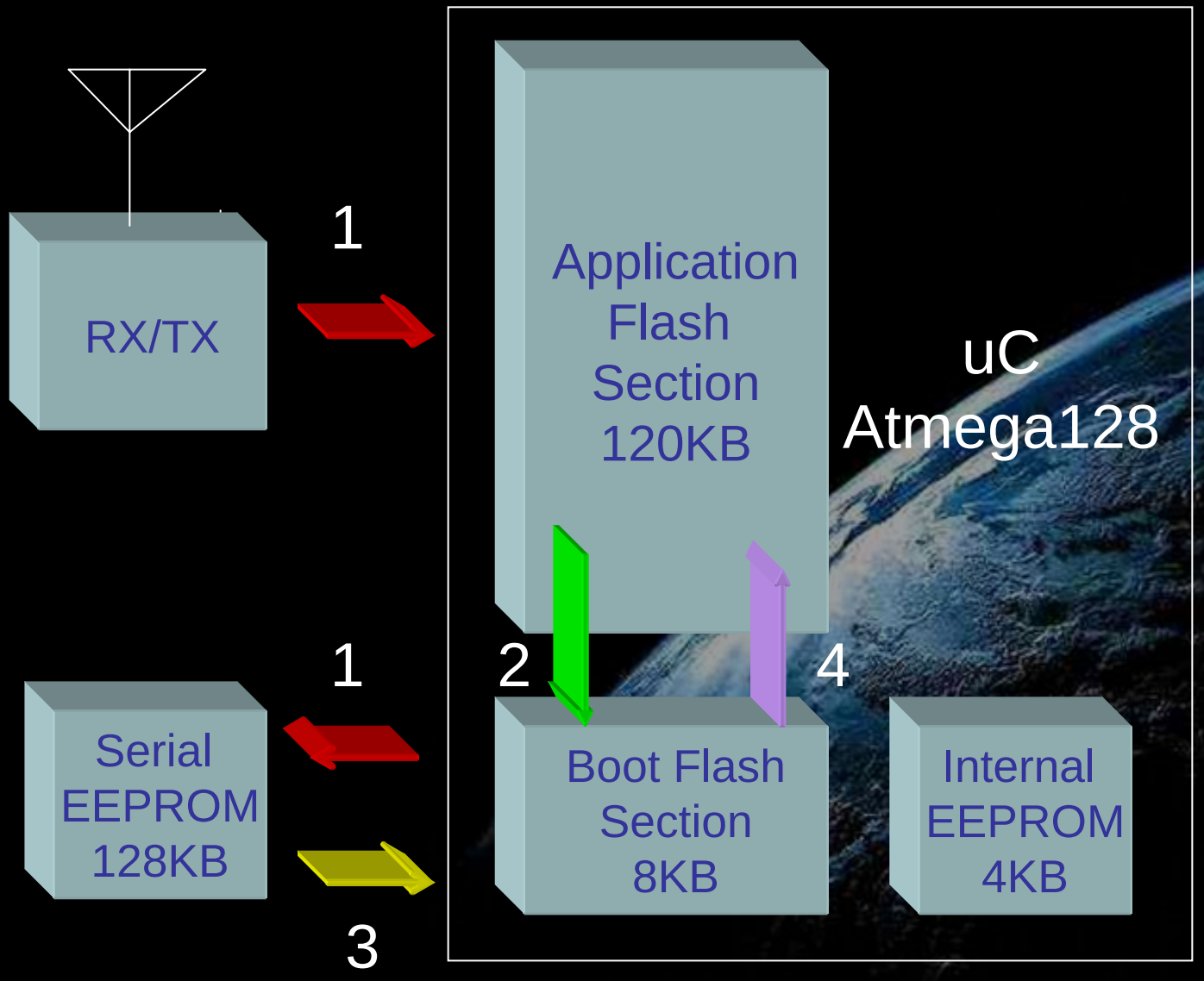
Recovery-Oriented Computing

- Murphy's Law: "Whatever can go wrong, will go wrong"
- Software has not evolved to deal with our world: computers crash, servers go down, data is lost…that is reality.
- Recovery design space:
  - Availability = MTTF / (MTTF + MTTR)
  - Decreasing MTTR increases system availability
- Building systems that recover effectively from failures may be more viable than aiming for systems that never fail.
- Recovery is not immune to failures either!

UCISAT-1 FAILURE RECOVERY SUBSYSTEM

- DIAGNOSTICS
  - DIAGNOSTICS WILL CHECK FOR FUNCTIONALITY OF SUBSYSTEMS (SENSORS, POWER SUBSYSTEM, CAMERA ETC)
- CRASH RECOVERY AND FIRMWARE CRASH DETECTION
  - WATCHDOG TIMER RESETS UC IN CASE OF A FIRMWARE LOCK UP
  - SOFTWARE CRASH INCIDENCE STATISTICS KEPT IN MICROCONTROLLER'S INTERNAL EEPROM
- ON-ORBIT FIRMWARE UPDATES
  - SATELLITE CAPABLE OF RECEIVING FIRMWARE UPDATES
  - FIRMWARE UPDATE STORED TEMPORARILY IN EXTERNAL MEMORY

ON-ORBIT FIRMWARE UPDATES



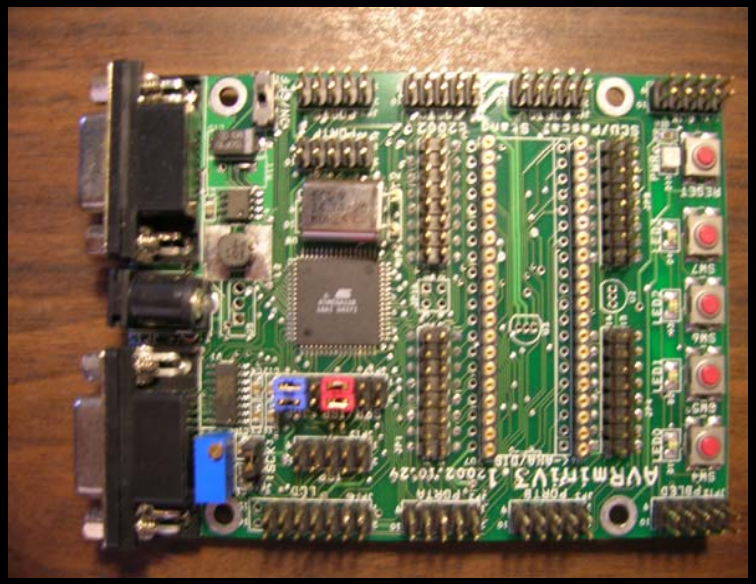
ON-ORBIT FIRMWARE UPDATES

- STATUS
  - SUCCESSFULLY ABLE TO UPDATE MICROCONTROLLER FIRMWARE VIA SERIAL PORT
  - NEW FIRMWARE LOADED USING X-MODEM PROTOCOL (TEMPORARY IMPLEMENTATION FOR TESTING PURPOSES ONLY)
- TESTING PERFORMED
  - SIMULATED POWER FAILURES/GLITCHES DURING FIRMWARE DOWNLOAD INTO MICROCONTROLLER'S INTERNAL FLASH
- WORK TO BE DONE
  - FIRMWARE SIGNATURE/CHECKSUM VERIFICATION
  - IMPLEMENT IMMUNITY TO POWER FAILURES/GLITCHES AND COMMUNICATION DISRUPTIONS DURING FIRMWARE UPLOAD INTO EXTERNAL EEPROM
  - IMPLEMENT 2ND LINE OF DEFENSE

UCISAT-1

Development Using Programmable Logic Devices

Presented by Tom Wypych



Henry Samueli School of Engineering

PROGRAMMABLE LOGIC DEVICES

OVERVIEW

- PROGRAMMABLE LOGIC DEVICES ARE DISCRETE COMPONENTS WHICH CONTAIN A COLLECTION OF GATE ELEMENTS
- GATE ELEMENTS ARE ARRANGED ALONG NETWORK GRID LINES, AND CAN BE DYNAMICALLY CONNECTED TO CREATE A USER-SPECIFIED CIRCUIT SYSTEM UPON STARTUP AT THE TIME OF DEVICE START UP
- CIRCUIT CONNECTION MAP IS STORED, LOADED, AND RUN BY WAY OF A SYSTEM-PROGRAMMED FIRMWARE

APPLICATION

- COMPLEXITY OF THE LOGIC DESIGN IS LIMITED ONLY BY THE ABILITIES OF THE PROGRAMMER
- FOR NEARLY EVERY DESIGN THERE EXISTS A DEVICE WELL SUITED TO PERFORM THE INTENDED TASKS
- THE MOST HELPFUL DESIGN USING PLD CAPABILITIES IS THE ASYNCHRONOUS PARALLEL EVENT PROCESSOR

PROGRAMMABLE LOGIC APPLICATIONS

PLDS CAN BE USED FOR THREE CATEGORIES OF TASKS:

1. *EASE OF DEVELOPMENT:* A SINGLE PLD CAN BE SUBSTITUTED FOR MANY INDIVIDUAL HARDWIRED LOGIC COMPONENTS. FEWER SYSTEM COMPONENTS MEANS A SIMPLER HARDWARE DESIGN.

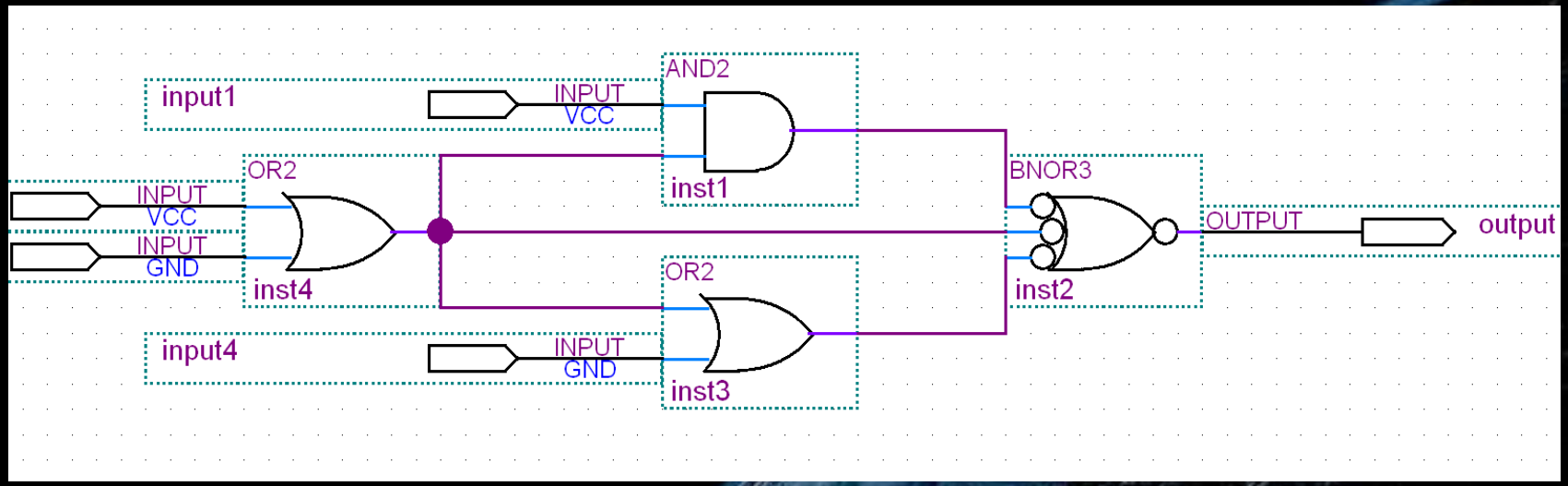
2. *HIGH SPEED CO-PROCESSING:* A PLD CAN BE USED TO TO INTERFACE WITH SIMPLER MICROCONTROLLERS TO PERFORM INTENSIVE AND REPETITIVE DSP TASKS. THIS PROVIDES AN EFFICIENT USE OF RESOURCES IN RESTRICTIVE OPERATING ENVIRONMENTS (SUCH AS A CUBESAT)

3. *CUSTOM END-TO-END DEVICE SOLUTIONS:* PLDS CAN BE USED TO CREATE CUSTOM HARDWARE DEVICES WITH A MINIMUM USAGE OF MICROCONTROLLER RESOURCES AND A MINIMUM OF SYSTEM COMPONENTS

…OR ALL THREE SIMULTANEOUSLY!

PLD CODING

ENTIRE FIRMWARE CAN BE SYNTHESIZED FROM LOGIC DIAGRAMS



PLD CODING

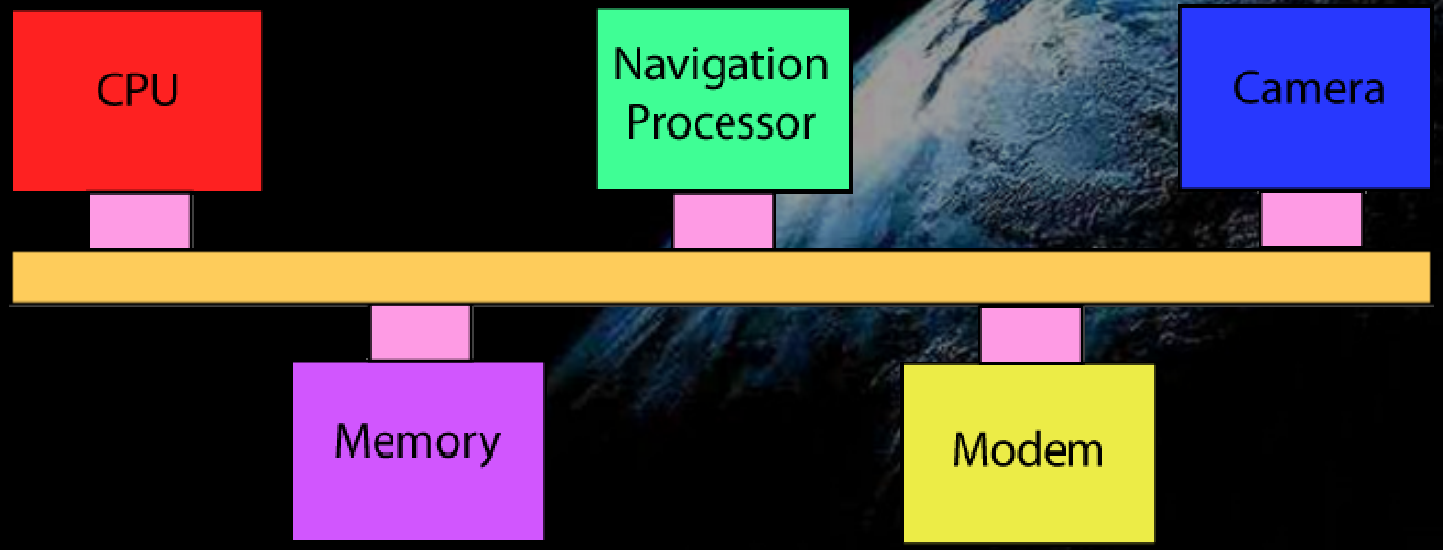
- SYMBOLIC CODING IS EASY FOR BEGINNERS, THERE ARE LIMITS TO COMPLEXITY FROM HUMAN ERROR
- VHDL OR VERILOG CAN BE USED TO TAP THE ABILITIES OF A MODERN PROGRAMMING LANGUAGE (FUNCTION CALLS, ELEMENT REUSABILITY, SHARED STATE VARIABLES, ITERATIVE PROCEDURES)

```
OUTPUT WIRE [7:0] C,
INPUT WIRE  [7:0] B,
INPUT WIRE  [7:0] A,
INPUT WIRE CLOCK

ALWAYS @(POSEDGE CLOCK)
BEGIN    C = A + B;
END
```

PLD CASE EXAMPLES

- Conventional design techniques requires many devices statically configured at the time of board and back plane design
- This process limits flexibility from a completed design, and can stress the limited resources of a small sat environment



PLD CASE EXAMPLES

CASE EXAMPLE: BACK PLANE

- MANY CUBESAT DESIGNS EMPLOY COMMUNICATIONS BETWEEN MULTIPLE DESIGN SUBSYSTEMS. USING AN INTERFACING BACK PLANE CAN ALLOW FOR SYSTEM-TO-SYSTEM COMMUNICATIONS
- CONVENTIONAL BACK PLANES REQUIRE EXTENSIVE ARBITRATION PROTOCOLS OR DEVICE INTERFACES HARDWIRED INTO THE BACK PLANE BOARD
- USING A PROGRAMMABLE LOGIC DEVICE NEW DATA INTERFACES CAN BE DYNAMICALLY CREATED DURING DEVELOPMENT UPDATES
- DYNAMIC INTERFACE RECONFIGURATION ALLOWS FOR ERRORS IN HARDWARE DESIGN TO BE CORRECTED IN SOFTWARE

PLD CASE EXAMPLES

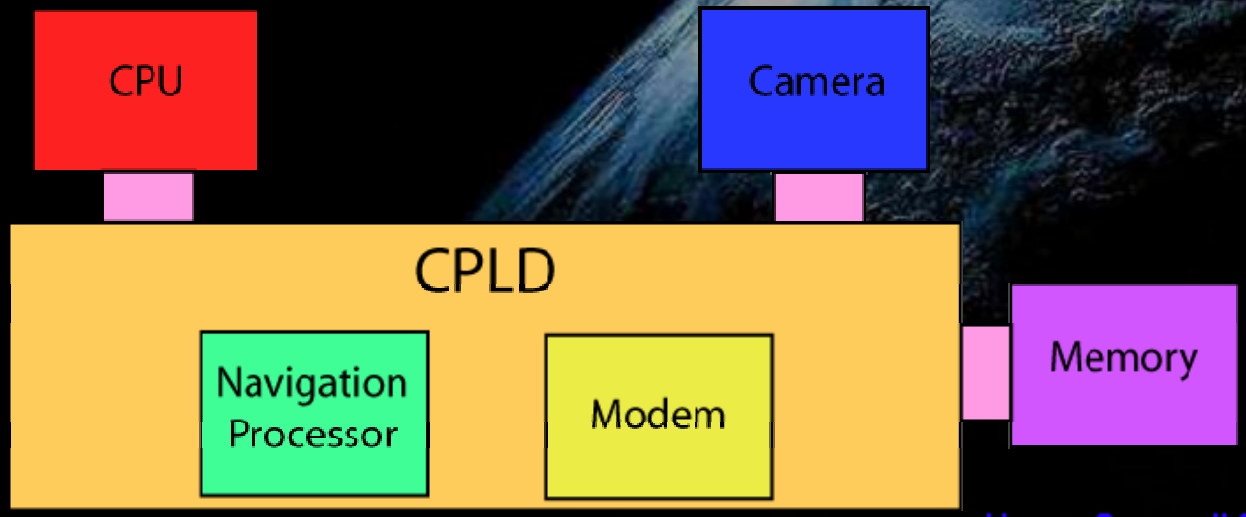
UCI CubeSat

CASE EXAMPLE: SOFTMODEM

- PROGRAMMABLE LOGIC DEVICES CAN BE USED TO IMPLEMENT THE DESIGN OF HIGH BANDWIDTH DEVICES OTHERWISE INFEASIBLE TO IMPLEMENT WITH CONVENTIONAL PROCESSORS BECAUSE OF RESOURCE LIMITATIONS.
- A SOFTMODEM IS JUST SUCH AN EXAMPLE; MODEM DEVICES REQUIRE CONSTANT AND INTENSIVE PROCESSING ATTENTION INFEASIBLE WITH A SIMPLE MICRO CONTROLLER
- USING A PLD TYPE DEVICE, THE ENTIRE MODEM CAN BE IMPLEMENTED USING NOTHING MORE THAN THE PLD AND SOME ANALOG CONVERSION HARDWARE

PLD CASE EXAMPLES

- NEW CPLD BASED DESIGN REQUIRES NO KNOWLEDGE OF INTERFACING PROTOCOLS OR MODES OF OPERATION AT THE TIME OF HARDWARE DESIGN
- HARDWARE COMPONENTS ARE CONSOLIDATED AND SYNTHESIZED WITHIN THE CPLD (AND GENERALLY PERFORM BETTER THAN THEIR DISCRETE COUNTERPARTS)
- THE ENTIRE MODE OF OPERATION OF THE CPLD CAN BE REPROGRAMMED DYNAMICALLY IN SECONDS



Programmable Logic Device Usage

What does it take to use a programmable logic device?

- Many offerings are available in the form of PLDs, CPLDs, and FPGAs from Xilinx and Altera
- Devices can be programmed graphically or using higher level code (Verilog or VHDL)
- From a hardware perspective, most small devices are single chip solutions, minimally requiring only power and data lines to your other devices to operate
- Optionally, interfaces can be added to program the device on the bench, or in-orbit as a slave device to a microcontroller (like Juan's firmware updates example)

REFERENCES / ACKNOWLEDGEMENTS

REFERENCES

The Berkeley/Stanford Recovery-Oriented Computing Project
http://roc.cs.berkeley.edu/

Procyon AVRlib
C-Language Function Library for Atmel AVR Processors
Written by Pascal Stang
http://hubbard.engr.scu.edu/embedded/avr/avrlib/

Open Cores
http://www.opencores.org

ACKNOWLEDGEMENTS

ADVISORS: PROFESSORS BENJAMIN VILLAC, DEREK DUNN-RANKIN,
AND A.K. HAYASHI

QUESTIONS?

Henry Samueli School of Engineering