Driving ICAP Resource

This material exempt per Department of Commerce license exception TSU

Objectives

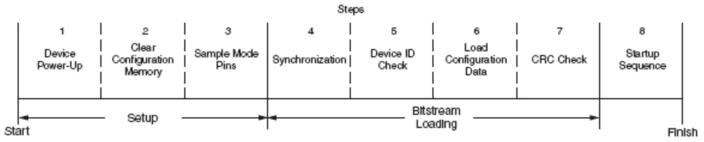
- **After completing this module, you will be able to:**
  - Describe a configuration process
  - List available ports on ICAP resource
  - State when custom ICAP is needed

Outline

- Configuration Sequence
- ICAP Resource
- Custom ICAP Processor
- Summary

Configuration Sequence

Steps are the same for all devices and modes



Steps

1 Device Power-Up

2 Clear Configuration Memory

3 Sample Mode Pins

4 Synchronization

5 Device ID Check

6 Load Configuration Data

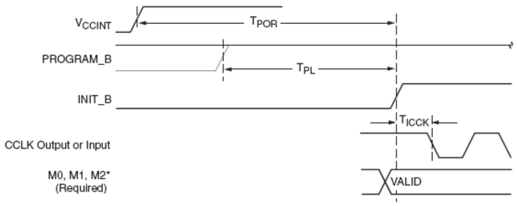
7 CRC Check

8 Startup Sequence

Start — Setup — Bitstream Loading — Finish

1) Device power up

- Check that your system powers up the FPGA quickly enough
- INIT_B is a bidirectional, open-drain pin (external pull-up is required)



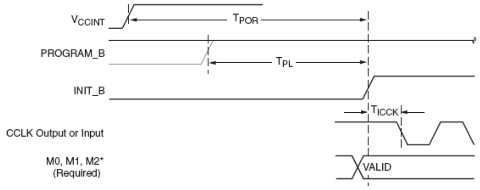
Configuration Sequence

2) Clear configuration memory

- Configuration memory is cleared any time the device is powered up or after the PROGRAM_B pin is pulsed Low
  - There is a minimum length of time PROGRAM_B must be Low
  - Can be held Low as long as you want
  - After it is released, the configuration memory is cleared twice
- During this time, I/Os are placed in a High-Z state
- INIT_B is internally driven Low during initialization, then released after Tpor
  - If INIT_B pin is held Low externally, the device waits at this point in the initialization process until the pin is released

3) Samples Mode Pins

- This is done when INIT_B goes High
- Then CCLK starts running



Configuration Sequence

4) Synchronization

- For BPI-Up, BPI-Down, Slave SelectMAP, and Master SelectMAP modes, the bus width must be first detected
- A special 32-bit synchronization word (0xAA995566) is sent to the configuration logic
  - This alerts the upcoming configuration data and aligns the configuration data with the internal configuration logic

5) Check Device ID

- ID check must pass before the configuration data frames can be loaded
  - This prevents configuration with a bitstream that is formatted for a different device
  - If an ID error occurs during configuration, the device attempts to do a fallback reconfiguration

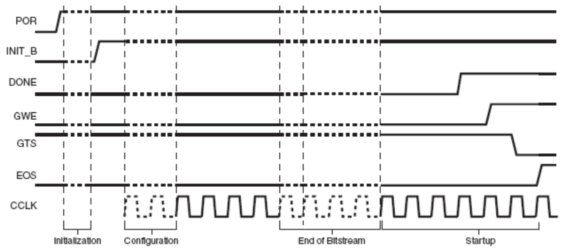
6) CRC (Cyclic Redundancy Check)

- After the configuration data is loaded the configuration bitstream can issue a *Check CRC* instruction to the device, followed by an expected CRC value
- If the value does not match, the device pulls INIT_B Low and aborts configuration
  - The CRC check is included in the configuration bitstream by default, but can be disabled
  - Intended to catch errors in transmitting the configuration bitstream

Configuration Sequence

7) **Start Up**

- The startup sequence is controlled by an eight-phase sequential state machine
- The startup sequencer performs the following tasks (user selectable)
  - Waits for DCMs to Lock (optional)
  - Waits for DCI to Match (optional)
  - Negates Global 3-state (GTS) (which activates I/O)
  - Releases DONE pin (open-drain output requiring an external pull-up)
  - Asserts Global Write Enable (GWE) (allows RAMs and FFs to change state)
  - Asserts End of Startup (EOS)
    - Note that the last four steps are default



Full Bitstream Integrity

- If a complete bitstream is corrupted, the FPGA never enters user mode
  - Design never becomes active and prevents damage to the FPGA*
  - User can monitor success or failure of configuration
    - DONE signal
    - System behavior
  - FPGA status register can be read via SelectMAP or JTAG
    - Provides information as to why the configuration failed
- User can decide how to proceed
  - Fix the bitstream
  - Download different bitstream

Partial Bitstream Integrity

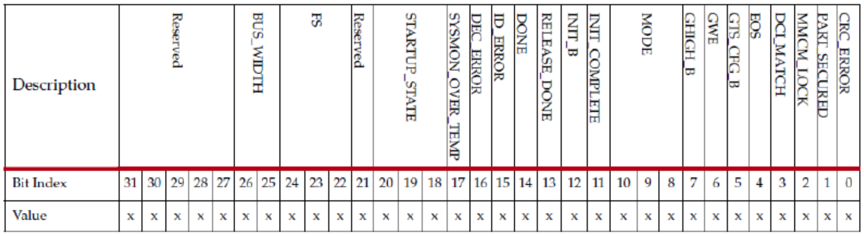
- Because the FPGA is already in user mode, partial bitstreams containing errors can be loaded
  - Contamination of a portion of the static design
  - Damage to the FPGA after a short period of time
- Errors must be detected manually
  - Detectable via the ICAP peripheral
  - The ICAP can be sent a series of "commands" that force the contents of the status register to the output of the ICAP peripheral
  - Select bits within the status register can then checked to detect a proper partial reconfiguration
    - In the Virtex 6 FPGA: Status Register:Bit 0 – CRC error
- If an error is detected
  - The partial bitstream can be reloaded (in the case of a communications error)
  - Another operational partial bitstream can be loaded
  - A black-box/null bitstream can be loaded
- Error detection and recovery is the designer's responsibility!

Types of Bitstream Errors

- Data errors
  - Simple recovery: reload bitstream
- Address errors
  - Problematic
    - The corrupted address might have been outside the PR
      - This implies that the static logic may have been affected
    - Best solution: reload the entire FPGA
- No way to distinguish a data error from an address error!
- Likelihood of errors – low
  - Most frequently, bit files are co-located with the FPGA; little opportunity for corruption
  - Even remote programming via Ethernet or the PCIe® protocol is relatively low
    - Protocols available for maintaining data integrity

Virtex-6 FPGA Status Register

- Contains information regarding global signals
- Readable through SelectMAP, JTAG interfaces, and the ICAP peripheral
  - Simple state machine writes a sequence of instructions to the ICAP peripheral
  - Data is then presented on the output of the ICAP primitive



Description	Reserved					BUS_WIDTH		FS			Reserved	STARTUP_STATE			SYSMON_OVER_TEMP	DEC_ERROR	ID_ERROR	DONE	RELEASE_DONE	INIT_B	INIT_COMPLETE	MODE			GHIGH_B	GWE	GTS_CFG_B	EOS	DCI_MATCH	MMCM_LOCK	PART_SECURED	CRC_ERROR
Bit Index	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Outline

- Configuration Sequence
- **ICAP Resource**
- Custom ICAP Processor
- Summary

Internal Configuration Access Port (ICAP)

- ICAP provides one of the mechanisms to configure the FPGA
  - Typically used for dynamic partial reconfiguration
- ICAP allows partial reconfiguration under software or hardware control
  - Software control requires some sort of processor system to drive the ICAP input with configuration data and control system
  - Hardware control requires some sort of finite state machine to fetch the configuration data and then drive it into the ICAP

ICAP Primitive

- ICAP – Internal Configuration Access Port (generic)
  - ICAP_VIRTEX6
  - Cannot be inferred and must be instantiated
    - No macro nor CORE Generator™ tool support
- The ICAP peripheral enables partial reconfiguration
  - Enables access to the configuration functions from within the FPGA fabric
  - Similar to SelectMAP

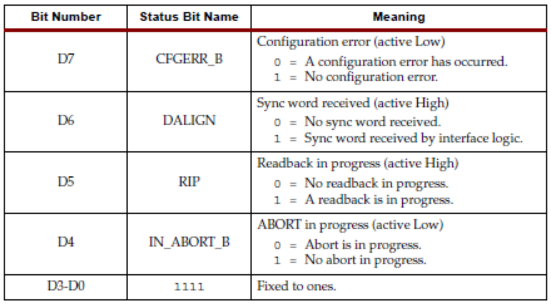
```
ICAP_VIRTEX6_inst : ICAP_VIRTEX6
generic map (ICAP_WIDTH => "X32“) -- Specifies I/O width
port map (
   BUSY => BUSY,  -- 1-bit Busy(=0)/Ready(=1) output
   O => O,        -- 32-bit Configuration data output bus
   CLK => CLK,    -- 1-bit Clock Input
   CSB => CE,     -- 1-bit Active-Low ICAP Enable
   I => I,        -- 32-bit Configuration data input bus
   RDWRB => WE    -- 1-bit Read(=1)/Write(=0) Select
);
```

ICAP Interface Description

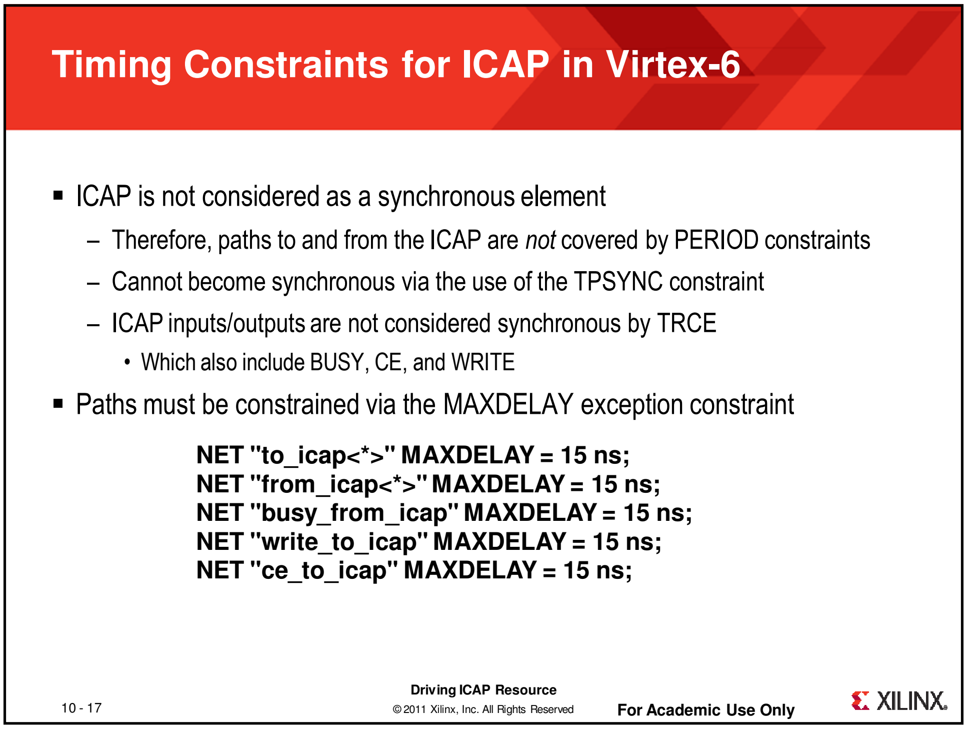
- I(31:0) – data to be loaded into configuration memory
  - ICAP_WIDTH attribute indicates a bus width of 8, 16, or 32 bits
- O(31:0) – data read from configuration memory
  - ICAP_WIDTH applies to the output as well
- CLK – clock input
  - Data must arrive synchronized with this clock
  - Output data is synchronized to this clock as well
- CSB – active Low ICAP enable
  - Same as CS_B in the SelectMAP interface
- RDWRB – read/write select
  - Low value indicates write
- BUSY – active High indicates if ICAP is currently being read
  - Always Low when writing to ICAP

ICAP Output Port

- ICAP Output port provides the status during the reconfiguration
- The status which can be monitored are
  - Configuration error due to CRC error, IDCODE error
    - Identified by Bit 7
  - Sync error
    - Identified by Bit 6
- Abort sequence is generated and indicated by Bit 4



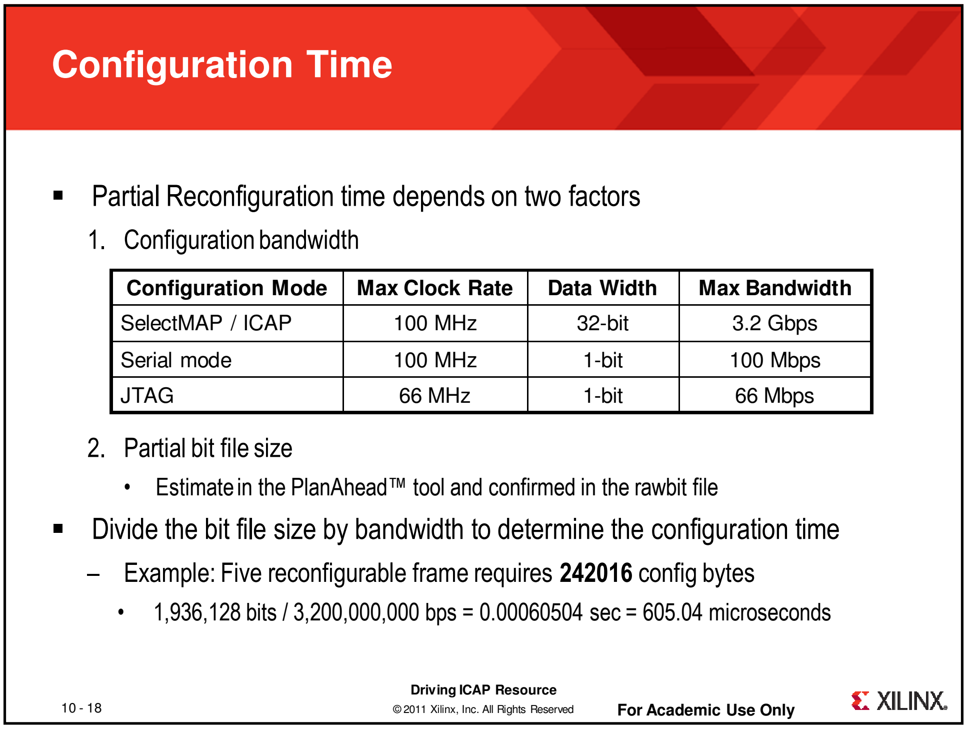
Bit Number	Status Bit Name	Meaning
D7	CFGERR_B	Configuration error (active Low) 0 = A configuration error has occurred. 1 = No configuration error.
D6	DALIGN	Sync word received (active High) 0 = No sync word received. 1 = Sync word received by interface logic.
D5	RIP	Readback in progress (active High) 0 = No readback in progress. 1 = A readback is in progress.
D4	IN_ABORT_B	ABORT in progress (active Low) 0 = Abort is in progress. 1 = No abort in progress.
D3-D0	1111	Fixed to ones.



Timing Constraints for ICAP in Virtex-6

- ICAP is not considered as a synchronous element
  - Therefore, paths to and from the ICAP are *not* covered by PERIOD constraints
  - Cannot become synchronous via the use of the TPSYNC constraint
  - ICAP inputs/outputs are not considered synchronous by TRCE
    - Which also include BUSY, CE, and WRITE
- Paths must be constrained via the MAXDELAY exception constraint

```
NET "to_icap<*>" MAXDELAY = 15 ns;
NET "from_icap<*>" MAXDELAY = 15 ns;
NET "busy_from_icap" MAXDELAY = 15 ns;
NET "write_to_icap" MAXDELAY = 15 ns;
NET "ce_to_icap" MAXDELAY = 15 ns;
```



Configuration Time

- Partial Reconfiguration time depends on two factors

1. Configuration bandwidth

Configuration Mode	Max Clock Rate	Data Width	Max Bandwidth
SelectMAP / ICAP	100 MHz	32-bit	3.2 Gbps
Serial mode	100 MHz	1-bit	100 Mbps
JTAG	66 MHz	1-bit	66 Mbps

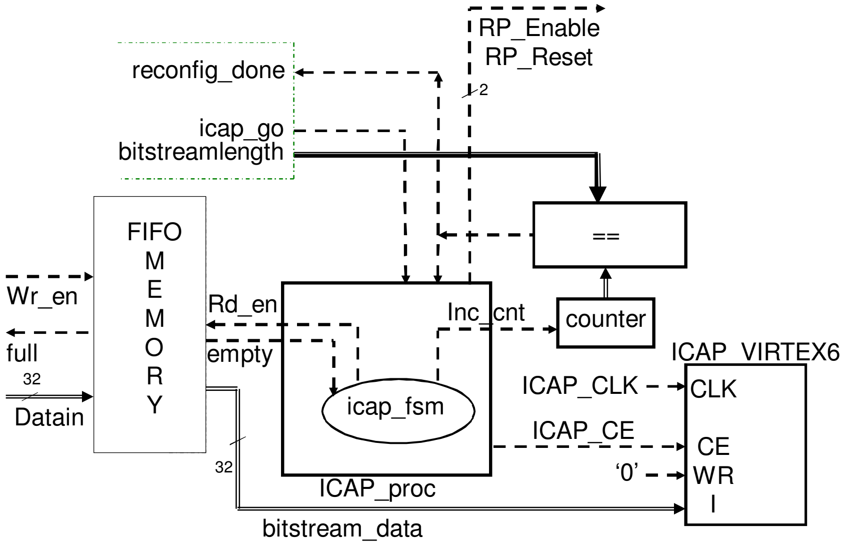
2. Partial bit file size
   - Estimate in the PlanAhead™ tool and confirmed in the rawbit file

- Divide the bit file size by bandwidth to determine the configuration time
  - Example: Five reconfigurable frame requires **242016** config bytes
    - 1,936,128 bits / 3,200,000,000 bps = 0.00060504 sec = 605.04 microseconds

Outline

- Configuration Sequence
- ICAP Resource
- **Custom ICAP Processor**
- Summary

Custom ICAP Processor

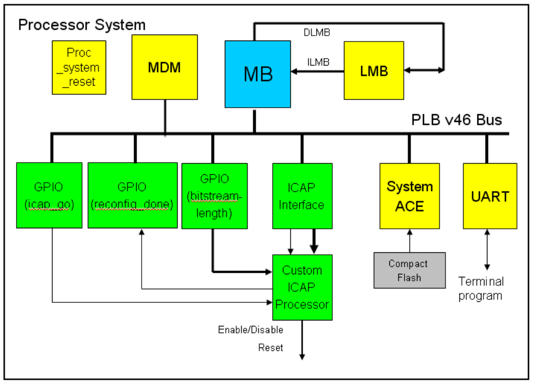


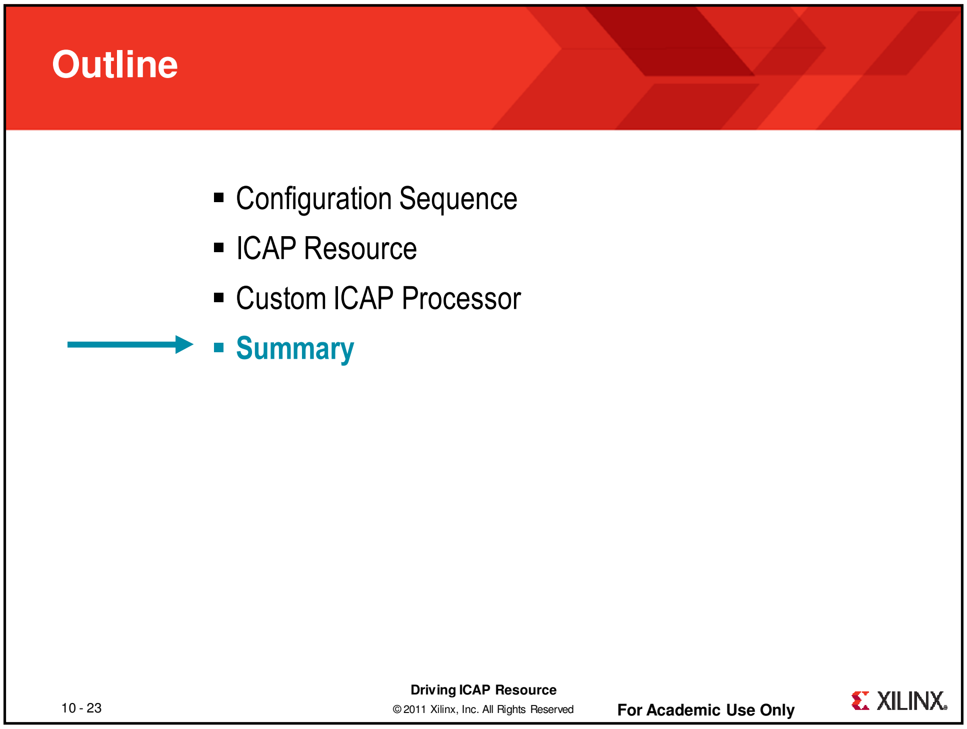
Custom ICAP Processor

- FIFO Memory is 1024x32
  - Big enough to hold streaming data
  - The input width can be changed to 64 if source system can handle
- *icap_go* signal is a pulse asserted by the source system after writing bitstream length information at the port
- *bitstreamlength* provides number of configuration data bytes
- *reconfig_done* is asserted when last word was written to ICAP
- *RP_Enable* is de-asserted when *icap_go* is received and re-asserted when last word is written into ICAP
- *RP_Reset* is asserted for one clock cycle when *RP_Enable* is re-asserted
- No readback functionality implemented

Custom Access to ICAP

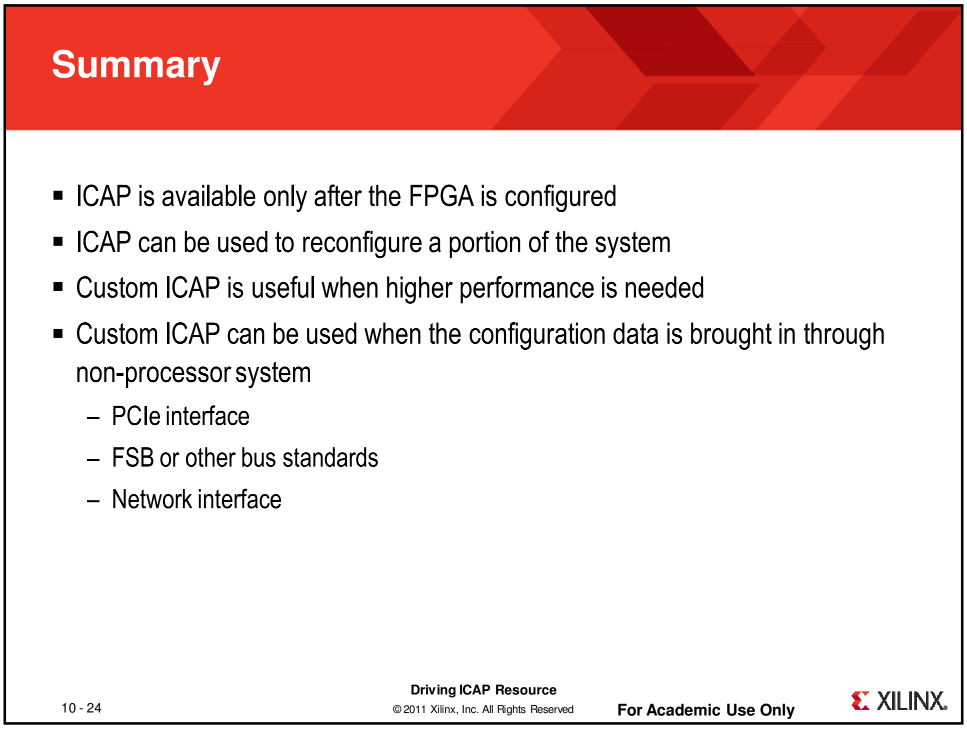
- Uses GPIOs to monitor and initiate partial reconfiguration
- Uses the ICAP Interface custom peripheral to load a bitstream into the ICAP device
  - Controlled via a state machine
- Use when
  - Maximum performance required
  - Custom bitstream encryption is applied
  - Special control is required





Outline

- Configuration Sequence
- ICAP Resource
- Custom ICAP Processor
- **Summary**



Summary

- ICAP is available only after the FPGA is configured
- ICAP can be used to reconfigure a portion of the system
- Custom ICAP is useful when higher performance is needed
- Custom ICAP can be used when the configuration data is brought in through non-processor system
  - PCIe interface
  - FSB or other bus standards
  - Network interface