Production DDU Design: Parts, Purpose and IO Usage

Connections, Front Panel

15 fibers (input) from DMB
- Differential outputs from Fiber transceivers go directly to 2 Xilinx Virtex-2-Pro FPGAs (XC2VP20-6FG676c, 404 I/O each); these "InCtrl" FPGAs have 8 Rocket I/O each. The DMB-DDU fiber transmission is based on 80 MHz clocks provided by matched on-board oscillators on the DMB and DDU boards. The fiber connections are duplex, and the primary data direction is DMB-DDU, but the board is designed to support limited communication from DDU-DMB.
- Each InCtrl FPGA has 2 associated TI FIFO pairs (2 TI 74LV396-9 FIFOs serially cascaded, 0.125 MB+ capacity each) operated with 18-bit in/36-bit out, Write clock @160MHz, Read clock @156MHz.
- LVCMOS 3.3V communication is used for the FIFOs.
- There are 2 separate Readout Control paths per InCtrl FPGA, with one Read Control path controlling its own TI FIFO pair.

Logical Fiber Input Overview:
- 4 DMB Fibers --> 1 Read Control --> 1 TI FIFO pair
- Physical Fiber Input Overview:
  - 7-8 DMB Fibers --> 1 InCtrl FPGA --> 2 TI FIFO pairs

1 fiber for Gigabit Ethernet (GbE) readout
- Driven by Rocket IO on the "DDUctrl" FPGA (XC2VP7-6FG672c, 396 I/O)
- The data for GbE packets is buffered in an on-board "Output FIFO" (IDT 7V273100, 72-in/18-out) prior to transmission. More below.
- 1 SLINK board for TF use
  - Driven by DDUctrl FPGA, in parallel with Output FIFO input bus
  - Has particular power and control requirements. More below.
- 1 FMM connector (RJ45 type)
  - 4 LVDS signals driven by "VMeCtrl" FPGA (XC2V500-5FG676c, 264 I/O)
  - Conforms to CMS TTS/FMW requirements. More below.

1 Block of 4 LEDs
- 4 green, orange, yellow and red LEDs are hard-wired to show 3.3V power, 2.5V power, FPGA Program_Done and VME DTACK respectively. The DTACK LED is driven by on-board discrete logic.

16 pairs of LEDs
- Each green/yellow LED pair is connected by light-pipes that correspond to its fiber connection. The green LED indicates a clock lock and channel (FOCut). The yellow LED flashes to show data activity (DADV on the link).
- The 15 DMB fiber LEDs are driven by InCtrl FPGA logic. The GbE LEDs are driven by DDUctrl FPGA logic.

Connections, Backplane
- 1 standard 5M VME connector on P1
- 1 HSC connector for 3.2Gbps differential communication to DCC on custom P3
- 2 differential High-Speed Connection paths per DDU, 3.12 Gb/s each --> 620 MB/sec of real data throughput (due to Bb10b)
- These connectors also carry clock & control signals from the TTCrx mounted on the DCC.

Clocks and Distribution
- 156 MHz PECL on-board; goes to DDUctrl FPGA only. Used for DDU-DCC Gigabit Transceiver (GT) links (3.12Gb/s), and an internal DCM provides divide-by-2 for 78MHz TI FIFO control and data processing in DDUctrl. Data goes out to GBE FIFO and SLINK at 78MHz on a common bus, 64-bits wide (plus control signals).
- 80.00 MHz PECL on-board; drives a 1-5 PECL clock driver. 4 outputs to the InCtrl FPGAs & 1 output to VMEctrl, 1 load each.

62.5 MHz LVTL on-board; goes only to DDUctrl for Gbe clocking
- 40.08 MHz LVTL HLC clock from TTC via DCC/backplane connection; DDUctrl FPGA serves as 1-1 clock driver with 1-output to each FPGA.
- 10 MHz LVTL SCLK (Serial/Slow Clock); derived from 80 MHz clock by DCM inside VMEctrl FPGA and fanned-out to All Other FPGAs and serial devices on a common bus (4 loads) with termination. Requires 24 mA IO drive setting.

FPGAs

- XC2VP20 "InCtrl FGPA" (need 2 for Input Control, use 254+ I/O each)
- use FG676 package (404 I/O), -8 speed grade (faster -7 costs more)
- Two linked XC18V04 FPGAs used to program both FPGAs in parallel
- Both InCtrl FPGAs use InCtrl FGPA for Differential communication to DCC on custom P3
- 7-8 DMB fibers
- JTAG is used for all PROM programming via the VME FPGA; JTAG is available on all FPGAs for slow control

- Main Clock is 80.00 MHz DMB-DDU-matched oscillator (fanout from PECL clock driver)
  - drives 2 differential input pairs on InCtrl FPGA (4 BREP IO, not BREP2) for Fiber Reference clocks (1 Top edge, 1 Bottom edge) as well as logic clocks (via BUFG) and GT USRCLKs (GT is Gigabit Transceiver logic module from Virtex2 library)

Each InCtrl FPGA is functionally split into a "top half" and a "bottom half", each responsible for processing the data from the 4 top-edge RocketIO and 4 bottom-edge RocketIO respectively. Each FPGA has the following functions (see file ddu_in.pdf for logic schematics):
- 24 Block RAMs configured as 22 FIFO MemUnits, 4 kB each
- MemUnits are assigned to the RocketIO on an as-needed basis. In this way, any RocketIO that need more memory may be allotted any combination of the 22 MemUnits as they are available.
- A MemControl Unit to control assignment of the BRAM FIFOs. There is a priority algorithm to select the assignments preferentially.
- A MemControl Unit to control assignment of the BRAM FIFOs. There is a priority algorithm to select the assignments preferentially.
- 4 InUnits, each containing one RocketIO, a buffer to record MemUnits in use, and data monitoring logic. This logic watches for Status Codes from both InCtrl FPGAs and DM, as well as flags & clears any 4-bit word boundary violations that occur. InUnits can also transmit data via fiber to any selected DMs using a common data bus from the DDUctrl FPGA. Before the MemUnit in use becomes full, the InUnit will request another MemUnit from MemCtrl, then switch to the new unit when the current one goes Full. InUnits will "free" a MemUnit for reuse when it goes Empty. Each InUnit has a 4-bit path to the MemCtrl FIFO to report status information.
- A ReadControl Unit that writes exclusively to one on-board TI FIFO pair (0.25 MB external storage). The RdCtrl gets the data from each RocketIO and checks LIA numbers for an event as the data is read full. RdCtrl forwards the Kill signal from the InCtrl towards the appropriate RocketIOs and watches for timeouts or changes in the link status of the InUnits. It also monitors the AlmostFull/Full status of the external FIFOs and the internal MemCtrl, and then sets the WriteDataStatus signal to Busy/CSV (TTS & DDM respectively) as needed. The event number and a summary of all the checks done within InCtrl are sent out to the FIFO as the last word of an event. This information will be assimilated in the DDUctrl FPGA. Each RdCtrl has a 4-bit path to the DDUctrl FPGA to report its operating conditions.
- Both halves of the InCtrl FPGA share a common JTAG control unit. This is used primarily as an interactive monitoring/debug tool.
59 configurable IO to DDUctrl FPGA
24 IO for fiber support (3 per fiber)
70 Other IO, 12 required for parallel PROM load and INIT signals
--- optional 8 input from switches, 8 output to LEDs
--- optional 42 IO for header/testpoints for InCtrl FPGA
--- optional 6 IO for testpoints for InCtrl FPGA
32 additional signals for InCtrl FPGA (4 per InCtrl)
1 signal to specify chip as InCtrl FPGA#0 or #1

>> How exactly do we use these IO in the DDU InCtrl?
-111 IO; each InCtrl FPGA uses these signals:
  - 80 MHz Main Clock (4 IO), 40 MHz LHC Clock (from DCC via DDUctrl FPGA), 10 MHz SlowClk from VMEctrl
  - 5-bits from DDUctrl: LIA, EVCntRst, LISR, SoftReset, BCO/BXR
  - 13-bits to/from DDUctrl FPGA as spare (include 1 clock I/O pin)
  - 16-bits input data from DDUctrl for TxData to DMB (via Rocket IO)
  - 1 DLL Lock/Ready (to DDUctrl), 2 FIFO_Active (1 per RdCtrl)
  - 4 for external inputs left free: INIT, CS, WRITE, DOUT (Don’t Use)
  - 2 * 18-bit DOUT to FIFOs (bits 16-17 flag Fill/LW, Hdr/Tr)
  - 2 * 11-bit FIFO control: WCLK, ~WEN, ~PAE, ~PAF, FF, ~3/4Full
  - 2 * 4-bit FIFO status bits (to DDUctrl, 1 4-bit set per RdCtrl)
  - 72 IO; need per Rocket IO used on each FPGA (multiply below by 8):
    - 2-bits for Tx control from DDUctrl FPGA: TxEn, Kill Fiber
    - 3-bits for Fiber/FIFO Status: POK (in), FOKout, DAV (to FP-LEDs)
    - 4 for DMB status: T[3:0] for FMM (to VMEctrl)
  - 58 IO; added header connectors for 2 Logic Analyzer (LA) ports (18 bits each, only InCtrl FPGA#0), 6 testpoints, 8 LEDs, 8 switches
  - 8 IO dedicated for parallel PROM load
  - InCtrl FPGA#1 “kills” RocketIO #6 (the 16th RocketIO) as it is unused

XXC2VP7 "DDUctrl FPGA" (uses 391+ I/O):
- use FF672 package (396 I/O), requires ~6 speed grade or faster (~7) with flip-chip package.
- TWO linked XC18V04 PROMs, use parallel load DIN[7:0] and CCLK lines
  --- use SelectMAP Master mode for program clock, FPGA drives CCLK
  - 2 Rocket IO to backplane (data to DCC via HSC connector, 3.2 Gb/s)
  - 1 Rocket IO to front panel output Gigabit Ethernet fiber (1.25 Gb/s)

The DDUctrl FPGA reads data from the on-board TI FIFOs, one at a time. Each event is checked for the following during readout:
- improper event formatting & data corruption
- status/error flags passed from InCtrl via TI FIFO
- event & bunch number synchronization from each source
- CFEB & wordcount & CRC from each source
- DMB CRC & LTC/DCT consistency from each source
- TMB wordcount & CRC from each source
- ALCT wordcount & CRC from each source
- missing DMB, CFEB, TMB or ALCT data from each source
- Full and Near Full condition from each CSC
- timeout conditions

DDUctrl performs the following tasks (logic schematics ddu_ctrl.pdf):
- event readout, processing and monitoring
- receives & decodes TTC Command Bus instructions from DCC (via backplane); e.g. BCO, Sync/Soft Reset, Start/Stop data taking
- receives & distributes CMS Clock & LIA from DCC to other FPGAs
- watches for changes in status of the RdCtrl units
- monitors the status feedback from the DCC/SLINK paths and Stops TI FIFO readout as needed
- sends event information in the headers and trailers, conforming to CMS Common Data Format rules
- generates a CRC word for all data transmitted for the event
- sends all data out via DCC/SLINK paths
- sends data to the external GbE FIFO on a prescalable basis
- creates & transmits GbE packets from data stored in external GbE FIFO
- reads Board ID & Killed Channel constants from Flash RAM and sends Kill Channel signals to InCtrl FPGAs
- controls a 16-bit data bus to send control or data signals out on any fiber via InCtrl FPGAs
- monitors the DDU Board Operation condition, summarizing the 4 RdCtrl and DDUctrl status on one 4-bit path to the VMEctrl FPGA for TTS

>> DDUctrl FPGA IO on-board hardware connections:
- 3 dedicated input lines from PECL/LVTTL sources (4 IO)
  - 1 LVTT, clock input (SCLK) from VMEctrl
  - 2 IO for local 78 MHz LVTT clock feedback
  - 2 global resets Out (Sync & Soft reset)
- 18 control lines from TTC (15), DCC (2) & VME (1)
- 7 dedicated lines for TI FIFOs (36 data, 36 control)
- 96 configurable IO to InCtrl FPGAs
- 116 IO for SLINK & GbE
  --- Min. 83 for SLINK; 33 more are for GbE only.
- 62 Other IO, 12 required for parallel PROM load and INIT signals
  - optional 8 input from switches, 8 PROM lines go to LEDs
  --- optional 41 IO to header/testpoints, 1 pushbutton switch
  12 additional signals for RocketIO (4 per RocketIO)

>> How exactly do we use these IO in DDUctrl?
- 7 IO used for 4 ext. CLks for DDUctrl: Main Clock (156 MHz) MUST use LVDS IO (IBUFGDS) & divide-by-2 for 78 MHz, one 62.5 MHz for GbE, 40.08 MHz LHC clock, 10 MHz serial shift clock (SCLK)
  --- 78 MHz feedback LVTTL goes out * back for DCM lock (2 IO)
- 18 IO; DDU board control
  - 1 pushbutton input (debounced, goes to DDUctrl) used for logic RST
  - 7 FIFO_Active (1 per RdCtrl from InCtrl FPGA)
  - 5-bits control output from DDUctrl common for other FPGAs:
    - LIAout, EVCntRst, LISR(TSyncRst), SoftReset, BC0/BXR
  - 4 control/IO signals left free: CS, WRITE, DOUT, INIT (Don’t Use)
- 30 IO; each InCtrl Rocket IO has individual controls (below times 15):
  - Enable (for TxEn), Kill (for Rx/Tx)
  - 16-bit output for DMB Tx data bus (parallel to both InCtrl FPGAs)
  - 16 status bits from InCtrl FPGAs (4-bits from each RdCtrl)
  - 24 IO for fiber support (3 per fiber)
  - 32 IO; each TI FIFO pair has individual controls (mult. by 4 below):
    - 24 IO for Fiber support (3 per fiber)
    - ~OE, ~REN, EF, ~HF, ~PAF, FF, TransferWEN, TransferCLK
  - 70 Other IO, 12 required for parallel PROM load and INIT signals
    - optional 8 IO from switches, 8 output to LEDs
    - optional 42 IO for header/testpoints for InCtrl FPGA#0
  - 16 IO; control input from TTC/DCC
    - cmd_bus[8:0], L1A, Link Ready/Full, plus 4 others
    - 32 additional signals for RocketIO (4 per RocketIO)
  - 66-bit output data (to GbE FIFO/SLINK)
  - 18-bit input data from GbE FIFO for (GbE packets out via RocketI0)
  - 10 IO; control/IO signals left free: CS, WRITE, DOUT, INIT (Don’t Use)
  - 57 other IO; 8 switches, 8 LEDs (share PROM DIN), 2 LA Ports (36 IO), 5 extra IO to "test points"
XC2V500-5FG456c "VMEctrl FPGA" (uses 264 IO)
- One XC18V04 PROM, uses parallel load DIN[7:0] and CCLK lines
  --> use SelectMAP Master mode for program clock
- Program VME PROM from VME FPGA, JTAG tag cable or Discrete on-board VME Logic; uses parallel load DIN[7:0] and CCLK lines to FPGA
- Main clock is 80.00 MHz DMB-DDU-matched oscillator (fanout from PECL clock driver)

VMEctrl performs the following tasks (logic schematics ddu_vme.pdf):
- Decodes VME Bus signals into on-board Parallel, Serial and JTAG paths
- Controls all Read and Write functions for the on-board 1 MB Serial Flash RAM; constants are read out and sent to the appropriate devices after every Soft Reset
- Continually receives 4-bit status summaries from DDUctrl and 15 fiber sources, uses this to determine the instantaneous status of the entire 20-degree DDU Sector, sending the result to TTS via FMM

VMEctrl FPGA IO on-board hardware connections:
- 2 dedicated clock input lines from PECL clock driver
- 2 global resets In (Sync & Soft reset from DDUctrl)
- 15 configurable IO to DDUctrl FPGA
- 68 configurable IO from InCtrl FPGAs
- 67 dedicated lines for VME Backplane IO
- 4 dedicated output lines to RJ45 for FMM/TTS; LVTL-LVDS
- Conversion done on-board
- 32 IO for 8 JTAG paths (for FPGAs, PROMs & FIFOs)
- 13 IO for other Serial controls (FIFO threshold, etc. from SRAM)
- 59 Other IO, 12 required for parallel PROM load and INIT signals
- 1 --> optional 8 input from switches, 8 PROM lines go to LEDs
- --> optional 39 IO for header/testpoints

>>> How exactly do we use these IO in VMEctrl?
- 47 VME input IO: ADR[23:1], AM[5:0], GA[5:0], IRQ1, IackIN, sysreset, syscall, clk, write, lword, berr, as, ds0, ds1
- 20 VME output IO: dtack, ToVME, doe, D[15:0], IackOUT
- 70 PROM IO: 4-bit PROM output (to DDUctrl [JTAG] and other FPGAs)
- 60-bit DMB status input (15 DMBs * 4 bits each from InCtrl FPGAs), 4-bit DDUctrl status inputs, 2 LEDs
- 9 control IO
- 80 MHz PECL clock, 2 Global Reset inputs (from DDUctrl)
- 1 DLL Lock/Ready (to DDUctrl)
- 4 control/IO signals left free: CS, WRITE, DOUT, INIT
- 14 Serial control IO (for 4 Input FIFOs, 1 Output FIFO, 1 DDUctrl)
- Common SCLK (4 loads), SData (2 loads), 1 FIFO SEN
- 6 bits for FlashMemory Control (M_SCLK, SI, SO, ~CS, ~WP, ~RESET)
- 4 bits for SerADC Control (ADCclk, ADCdin, ADCsel, ADCout)
- 1 Serial Enable bit to DDUctrl FPGA
- 9 spare signals to DDUctrl FPGA
- 8 spare signals to InCtrl FPGAs
- 32 IO for 8 JTAG paths (4 IO each):
- 1 for DDUctrl PROM
- 1 for DDUctrl FPGA
- 1 for VMEctrl PROM (by VMEctrl, VME-Recovery & JTAG cable)
- 2 for InCtrl FPGAs (1 each)
- 1 for InCtrl PROMs (in series, matched with DONE/PROGRAM order)
- 1 for S-Link
- 1 for output FIFO
- 55 other IO; 8 switches, 8 LEDs (share PROM DIN), 2 LA Ports (36 IO), 3 extra IO to "test points"

FIFOs & on-board Flash Memory

Input FIFO: use 8 TI 74LV3690-6 (128 PQFP, 32K x 36-bits)
- FIFOs are paired for a combined depth of 64K
  -> first FIFO: 18-bits In (160MHz), 36-bits Out (156MHz)
  -> second FIFO: 36-bits In (156MHz), 36-bits Out (156MHz)
Output FIFO: use IDT 72V72100 (256 PGB, 64K x 72-bits, 72/18)
- use 66-bits In (78MHz), 18-bits Out (62.5MHz)
- Serial Flash RAM: use 1 AT45DB011B (over 1 MB, 8-pin SOIC), 20 MHz max.
- 10 MHz derived from SCLK; can do 20 MHz, but the GbE FIFO serial clock is limited to 10 MHz max.
- Stores constants in Pages for DDU Board ID, Killed channels, etc.
Useful OpCodes: d2h, d7h, 82h
- All constants only use ~100 bits of memory; over 1 MB is available for additional constants if desired

Use Block0: page=KillCh (16b), page=BoardID (8b), page=free (32b), page=GbE FIFO Thresh (34b)

Other Notes

- Hard_Reset has 3 sources logical OR'ed together w/discrete on-board logic:
  - TTC via DCC/Backplane connection
  - Power-On sensor
  - PROGRAM button input (debounced)

- Hard_Reset goes to all FPGA PROGRAM pins.
The 2nd on-board pushbutton (debounced) makes a Soft_Reset in DDUctrl, which farg_it_out, and other FPGAs, and sends sync’d MRST to all FIFOs. (This could be reconfigured for another purpose as needed.)
- VMEctrl sets the LD & FMFT/SI signals in time with the MRST

The board has discrete "Emergency VME-Recovery" logic on board to allow for bootstrap VMEctrl PROM programming. There is also a 10-pin JTAG header connector for programming this PROM via laptop if desired. All other PROMs _must_ be programmed via VME, with no alternative provided.

The sections below the UCF files containing the Location and Timing Constraints for all FPGA firmware.

DDU InCtrl FPGA UCF file

# use with "$785/DDUD5in: IN5CTRL" Logic schematic
#FNLOCK_BEGIN
#Sun Sep 27 12:48:16 2005
#INPUTS
#  Clock & Control signals in:
  #  from switches (added 4)
  NET "$1116/""SOFTSTR" LOC = "AB4";
  NET "$1116/""IN_EV_RST" LOC = "AB1"; # l1arst
  NET "$1116/""IN_RL_RDY" LOC = "AA5"; #/-""LRST" LOC = "AA5";
  NET "$1116/""BXRIN" LOC = ""; # not used
  NET "$1116/""IN_BCO" LOC = "AB2"; # in from DDUctrl: "bcOut"
  NET "$1116/""MODEIN1" LOC = "H6"; # to Switch8
  NET "$1116/""MODEIN2" LOC = "H7";
  NET "$1116/""MODEIN3" LOC = "G1";
  NET "$1116/""MODEIN4" LOC = "G2";
  NET "$1116/""MODEIN5" LOC = "G3";
  NET "$1116/""MODEIN6" LOC = "G4";
  NET "$1116/""MODEIN1" LOC = "H5";
  NET "$1116/""MODEIN0" LOC = "G5"; # to Switch1
  NET "$1116/""18082/""LLAIN" LOC = "PA2"; # I/O out
  NET "$1116/""IN_FPGAID" LOC = "C25"; # fpgaid
  NET "$1116/""SCLKIN" LOC = "E14"; # aclk, not used
  NET "$1116/""CLKIN40" LOC = "AD14"; # spare_clk, gclk_s
  NET "$1116/""CLKIN40" LOC = "C13"; # clk80n0, gclk_p
  NET "$1116/""CLKOP_B" LOC = "AD13"; # clk80p1, gclk_p
  NET "$1116/""CLKOP_B" LOC = "AE13"; # clk80n1, gclk_s
  # # Spare signals lines:
  NET "$1116/""DDUSPARE0" LOC = "AD25"; # spare0/12 from DDUctrl
  NET "$1116/""DDUSPARE1" LOC = "AC24"; # spare1/13 from DDUctrl
# NET "$1I162/DDUSPARE2" LOC = "AC25"; #spare2/14 from DDUctrl
# NET "$1I162/DDUSPARE3" LOC = "AC26"; #spare3/15 from DDUctrl
# NET "$1I162/DDUSPARE4" LOC = "AB23"; #spare4/16 from DDUctrl
# NET "$1I162/DDUSPARE7" LOC = "AB26"; #spare7/19 from DDUctrl
# NET "$1I162/DDUSPARE8" LOC = "AA25"; #spare8/20 from DDUctrl
# NET "$1I162/DDUSPARE9" LOC = "AA26"; #spare9/21 from DDUctrl
NET "$1I162/IMDB_FUL" LOC = "AA22"; #spare5/17 to DDUctrl In0-3 DBM_Full
NET "$1I162/IMDB_FULL" LOC = "AB25"; #spare6/18 to DDUctrl In4-7 DBM_Full
# NET "$1I162/-FMT01IN" LOC = "Y12"; #spare0/22 from DDUctrl, "DoFake0"
# NET "$1I162/-FMT02IN" LOC = "W12"; #spare1/23 from DDUctrl, "DoFake1"
# NET "$1I162/VMESPAR0" LOC = "F22"; #spare "INPUT0/4" to/from VMEctrl
# NET "$1I162/VMESPAR0" LOC = "E25"; #spare "INPUT1/5" to/from VMEctrl
NET "$1I162/GRTST_TO_VME" LOC = "E26"; # out to VMEctrl for EvCntRst or BCO
# NET "$1I162/INFAKE1" LOC = "F21"; # in from VMEctrl for Fake_L1A_En # # "INPUT3/7"
# TestPoints:
NET "$1I162/TP_56" LOC = "E5"; #Testpoint56
NET "$1I162/TP_57" LOC = "E6"; #Testpoint57
# NET "$1I162/TP_58" LOC = "F7"; #Testpoint58
# NET "$1I162/TP_59" LOC = "E7"; #Testpoint59
NET "$1I162/TP_60" LOC = "B14"; #Testpoint60
# NET "$1I162/TP_61" LOC = "C14"; #Testpoint61
# FIFO Status Signals in:
# from output FIFO
NET "$1I162/-PAEOIN" LOC = "K22"; #not really needed, monitor only?
NET "$1I162/-HPAOIN" LOC = "M22"; #Eff0In, use for Warn flag or Release?
# NET "$1I162/-PFAOIN" LOC = "J23";
# NET "$1I162/-FAEOIN" LOC = "U23"; #not really needed, monitor only?
NET "$1I162/-PAFIN" LOC = "T22";
# NET "$1I162/-PAFIN1" LOC = "R22";
# from input FIBERs
NET "$1I162/IFOKO" LOC = "D1"; #fok0
NET "$1I162/IFOK1" LOC = "D2";
NET "$1I162/IFOK2" LOC = "C1";
NET "$1I162/IFOK3" LOC = "C2";
NET "$1I162/IFOK4" LOC = "A1A";
NET "$1I162/IFOK5" LOC = "Y5";
NET "$1I162/IFOK6" LOC = "W5";
NET "$1I162/IFOK7" LOC = "Y4"; #floating IO on 2nd FPGA
# KILL input FIBERs from DDU Control
NET "$1I162/KILL0" LOC = "AB14";
NET "$1I162/KILL1" LOC = "A015";
NET "$1I162/KILL2" LOC = "A15";
NET "$1I162/KILL3" LOC = "AA14";
NET "$1I162/KILL4" LOC = "AB15";
NET "$1I162/KILL5" LOC = "AA15";
NET "$1I162/KILL6" LOC = "AA16";
NET "$1I162/KILL7" LOC = "AD17";
# Data signals in
NET "$1I162/RXP0" LOC = "A22"; #r0+
NET "$1I162/RXN0" LOC = "A23"; #rd0-
NET "$1I162/RXP1" LOC = "A17"; #rd0+
NET "$1I162/RXN1" LOC = "A18"; #rd0-
NET "$1I162/RXP2" LOC = "A11";
NET "$1I162/RXN2" LOC = "A12";
NET "$1I162/RXP3" LOC = "A6";
NET "$1I162/RXN3" LOC = "A7";
# NET "$1I162/RXP4" LOC = "AF6";
NET "$1I162/RXN4" LOC = "AF7";
NET "$1I162/RXP5" LOC = "AF11";
NET "$1I162/RXN5" LOC = "AF12";
NET "$1I162/RXP6" LOC = "AF17"; #floating IO on 2nd FPGA
NET "$1I162/RXN6" LOC = "AF18"; #floating IO on 2nd FPGA
}
NET "$1I162/L1_5" LOC = "R2";
NET "$1I162/L1_6" LOC = "R4";
NET "$1I162/L1_7" LOC = "R5";
NET "$1I162/L1_8" LOC = "P6";
NET "$1I162/L1_9" LOC = "R6";
NET "$1I162/L1_10" LOC = "R8";
NET "$1I162/L1_11" LOC = "R9";
NET "$1I162/L1_12" LOC = "T1";
NET "$1I162/L1_13" LOC = "T2";
NET "$1I162/L1_14" LOC = "P5";
NET "$1I162/L1_15" LOC = "T6";
NET "$1I162/L1_16" LOC = "P3";  #l1 clk1
NET "$1I162/L1_17" LOC = "P2";  #l1 clk2

# Foundation 3.1 does not understand high/low 50%. but required for 4.2 & 5.1
# but required for 4.2 & 5.1

# VIRTEX 2 Geometry Summary (XC2VP20−FG676):
# 2 Slices per CLB, 2 TBUFs per CLB
# 2 LUTs per Slice, 2 FDs per Slice (share common RST/CE/CLK)
# 2 Fast Carry MUXCYs per Slice, plus 1 Fast ORCY
# 1 MUXFS and 1 MUXFx per Slice (x=6,7 or 8)

For "Normal" Coords (x,y) Slices, TBUFs, RAMBs, etc all have independent coordinate origins, so use GRID coords if you must combine different types of components.
Component coordinate ranges:
- SLICE  x0y0 to x91y111
- TBUF  x0y0 to x90y111, even x only
- RAMB16 x0y0 to x7y13 (88 total)  
  GT  x0y0 to x3y1 (8 total)  
  DCM  x0y0 to x3y1 (8 total)
- MULTI18X18 x0y0 to x7y13 (88 total)

In GRID Coords (X,Y) bottom left is X3Y4, top right is X138Y227
X value cycles through SLICE–TBUF–SPECIAL with increasing X
-X3Y4 is a SLICE, so is X138Y227
-SPECIAL is RAMB/MULT or GT or DCM
-SPECIAL only exists at X=8,26,44,62,80,98,116,134
RAMB/MULT exist at all these X locations
-GTs at X=26,62,80,116 and Y=19,227
-DCMs at X=8,44,98,134 and Y=3,231
Y values are continuous for Slices, except for PPC holes
-TBUFs are only even in Y (2 per CLB)
-RAMBS are spaced 16 apart in Y,
starting at Y=35 on GT columns, Y=19 otherwise
(GTs effectively take up a RAMB spot)
-MULTs are 1 lower in Y than RAMB (at Y_RAMB − 1)
- "Dead Zone" holes:
   No TBUFs from x14y40 to x58y71 and x62y40 to x76y71
 GRID: x23y88 to x24y143 and x95y88 to x119y143

************
INST "$1I162/$2I41683" AREA_GROUP = AG_JTAG; #Def AreaGp
INST "$1I162/$2I41683" LOC = SLICE_X8Y46:SLICE_X3Y5;LocSLS8y48x55y63
AREA_GROUP "AG_JTAG" COMPRESSION = 1;

************
** IN_UNIT0-3 RLOC **
************
0 AP_CLB_x531:
SET "$1I162/$1I4142/$1I4152/af_clb_x531rpm/mem/distmem/dist_mem/DPRAM" RLOC C_ORIGIN = X102Y204; #mAF_CLB slx_x6y100--slx_x6y102 RLOC=X0Y0 X105Y20

************
## JTAG AREA GROUP
************
SET "$1I162/$1I4142/$1I4152/af_clb_x531rpm/mem/distmem/dist_mem/DPRAM" RLOC C_ORIGIN = X102Y204; #mAF_CLB slx_x6y100--slx_x6y102 RLOC=X0Y0 X105Y20
# MEM_UNIT RLOC

## --Corner 0--

### Define Global Location for MU0:

INST "$11162/$114474/$114457" RLOC_ORIGIN = X129Y100; #WR_SEL FDCE

INST "$11162/$114474/$114448" RLOC_ORIGIN = X129Y100; #SF_BRAM, SL_X84Y48

INST "$11162/$114474/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU1:

INST "$11162/$114474/$114457" RLOC_ORIGIN = X129Y112; #WR_SEL FDCE

INST "$11162/$114474/$114448" RLOC_ORIGIN = X129Y112; #SF_BRAM, SL_X84Y64

INST "$11162/$114474/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU2:

INST "$11162/$114432/$114457" RLOC_ORIGIN = X129Y164; #WR_SEL FDCE

INST "$11162/$114432/$114448" RLOC_ORIGIN = X129Y164; #SF_BRAM, SL_X84Y80

INST "$11162/$114432/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU3:

INST "$11162/$114432/$114457" RLOC_ORIGIN = X129Y196; #WR_SEL FDCE

INST "$11162/$114432/$114448" RLOC_ORIGIN = X129Y196; #SF_BRAM, SL_X84Y96

INST "$11162/$114432/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU4:

INST "$11162/$114452/$114457" RLOC_ORIGIN = X111Y148; #WR_SEL FDCE

INST "$11162/$114452/$114448" RLOC_ORIGIN = X111Y148; #SF_BRAM, SL_X72Y72

INST "$11162/$114452/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU5:

INST "$11162/$1143496/$114457" RLOC_ORIGIN = X111Y180; #WR_SEL FDCE

INST "$11162/$1143496/$114448" RLOC_ORIGIN = X111Y180; #SF_BRAM, SL_X72Y88

INST "$11162/$1143496/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU6:

INST "$11162/$1143465/$114457" RLOC_ORIGIN = X93Y196; #WR_SEL FDCE

INST "$11162/$1143465/$114448" RLOC_ORIGIN = X93Y196; #SF_BRAM, SL_X60Y96

INST "$11162/$1143465/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU7:

INST "$11162/$1143451/$114457" RLOC_ORIGIN = X93Y164; #WR_SEL FDCE

INST "$11162/$1143451/$114448" RLOC_ORIGIN = X93Y164; #SF_BRAM, SL_X60Y80

INST "$11162/$1143451/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU8:

INST "$11162/$1144630/$114457" RLOC_ORIGIN = X75Y148; #WR_SEL FDCE

INST "$11162/$1144630/$114448" RLOC_ORIGIN = X75Y148; #SF_BRAM, SL_X48Y72

INST "$11162/$1144630/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU9:

INST "$11162/$11434488/114457" RLOC_ORIGIN = X75Y132; #WR_SEL FDCE

INST "$11162/$11434488/114448" RLOC_ORIGIN = X75Y132; #SF_BRAM, SL_X48Y64

INST "$11162/$11434488/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU10:

INST "$11162/$1144464/$114457" RLOC_ORIGIN = X75Y180; #WR_SEL FDCE

INST "$11162/$1144464/$114448" RLOC_ORIGIN = X75Y180; #SF_BRAM, SL_X48Y88

INST "$11162/$1144464/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### --Corner 1--

### Define Global Location for MU16:

INST "$11162/$1143429/$114457" RLOC_ORIGIN = X3Y132; #WR_SEL FDCE

INST "$11162/$1143429/$114448" RLOC_ORIGIN = X3Y132; #SF_BRAM, SL_X0Y64

INST "$11162/$1143429/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU7:

INST "$11162/$11434399/$114457" RLOC_ORIGIN = X3Y164; #WR_SEL FDCE

INST "$11162/$11434399/$114448" RLOC_ORIGIN = X3Y164; #SF_BRAM, SL_X0Y80

INST "$11162/$11434399/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU17:

INST "$11162/$1143465/$114457" RLOC_ORIGIN = X3Y196; #WR_SEL FDCE

INST "$11162/$1143465/$114448" RLOC_ORIGIN = X3Y196; #SF_BRAM, SL_X0Y96

INST "$11162/$1143465/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU18:

INST "$11162/$11434464/$114457" RLOC_ORIGIN = X2Y1148; #WR_SEL FDCE

INST "$11162/$11434464/$114448" RLOC_ORIGIN = X2Y1148; #SF_BRAM, SL_X12Y72

INST "$11162/$11434464/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU19:

INST "$11162/$1144426/$114457" RLOC_ORIGIN = X2Y1148; #WR_SEL FDCE

INST "$11162/$1144426/$114448" RLOC_ORIGIN = X2Y1148; #SF_BRAM, SL_X12Y88

INST "$11162/$1144426/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp

### Define Global Location for MU20:

INST "$11162/$1144475/$114457" RLOC_ORIGIN = X2Y1180; #WR_SEL FDCE

INST "$11162/$1144475/$114448" RLOC_ORIGIN = X2Y1180; #SF_BRAM, SL_X12Y88

INST "$11162/$1144475/AREA_GROUP = AG_MEMTBUF_UP; #Def AreaGrp
INSTAT $11162/$715013/$114488* RLOC_ORIGIN = X39Y52; #SF_BRAM_SL_X2Y4
INST "$11162/$715013* LOC = TBFX5245:TBFX5245; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M041:
INST "$11162/$714614/$114488* RLOC_ORIGIN = X129Y56; #WR_SELF DCCE
INST "$11162/$714616/$114488* RLOC_ORIGIN = X129Y56; #SF_BRAM_SL_X2Y4
INST "$11162/$714616* LOC = TBFX3524:TBFX3524; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M042:
INST "$11162/$714668/$114488* RLOC_ORIGIN = X129Y52; #WR_SELF DCCE
INST "$11162/$714684/$114488* RLOC_ORIGIN = X129Y52; #SF_BRAM_SL_X2Y4
INST "$11162/$714684* LOC = TBFX3524:TBFX3524; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
#--Corner 3--
# Define Global Location for M049:
INST "$11162/$714615/$114488* RLOC_ORIGIN = X129Y36; #SF_BRAM_SL_X2Y4
INST "$11162/$714615/$114488* RLOC_ORIGIN = X129Y36; #SF_BRAM_SL_X2Y4
INST "$11162/$714615* LOC = TBFX5615:TBFX5615; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M050:
INST "$11162/$714616/$114488* RLOC_ORIGIN = X129Y4; #SF_BRAM_SL_X2Y4
INST "$11162/$714616/$114488* RLOC_ORIGIN = X129Y4; #SF_BRAM_SL_X2Y4
INST "$11162/$714616* LOC = TBFX5815:TBFX5815; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M051:
INST "$11162/$714661/$114488* RLOC_ORIGIN = X111Y52; #SF_BRAM_SL_X2Y4
INST "$11162/$714661/$114488* RLOC_ORIGIN = X111Y52; #SF_BRAM_SL_X2Y4
INST "$11162/$714661* LOC = TBFX5252:TBFX5252; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M052:
INST "$11162/$714638/$114488* RLOC_ORIGIN = X111Y20; #SF_BRAM_SL_X2Y4
INST "$11162/$714638* LOC = TBFX5052:TBFX5052; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M053:
INST "$11162/$714762/$114488* RLOC_ORIGIN = X93Y4; #SF_BRAM_SL_X2Y4
INST "$11162/$714762* LOC = TBFX5656:TBFX5656; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M054:
INST "$11162/$714731/$114488* RLOC_ORIGIN = X93Y36; BWR_SELF DCCE
INST "$11162/$714731* LOC = TBFX4853:TBFX4853; # SF_BRAM_SL_X2Y4
INST "$11162/$714731* LOC = TBFX4853:TBFX4853; # SF_BRAM_SL_X2Y4
INST "$11162/$714731* LOC = TBFX4853:TBFX4853; # SF_BRAM_SL_X2Y4
INST "$11162/$714712/$114488* RLOC_ORIGIN = X75Y100; #WR_SELF DCCE
INST "$11162/$714712* LOC = TBFX4449:TBFX4449; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M056:
INST "$11162/$714974/$114488* RLOC_ORIGIN = X75Y68; #WR_SELF DCCE
INST "$11162/$714974* LOC = TBFX4848:TBFX4848; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Define Global Location for M057:
INST "$11162/$714632/$114488* RLOC_ORIGIN = X75Y20; #SF_BRAM_SL_X2Y4
INST "$11162/$714632* LOC = TBFX4048:TBFX4048; # AREA_GROUP "AG_MEMTBUF_DN" #Def AreaGrp
# Control Input from other FPGAs
NET "$1I135/MODEIN6" LOC = "AD25";
NET "$1I135/MODEIN7" LOC = "AD26"; #to Switch
# # Control Input from other FPGAs
NET "$1I135/-RODYIN0" LOC = "AE1"; #d11_ready0
NET "$1I135/-RODYIN1" LOC = "AD15"; # From DCC
NET "$1I135/-RODYIN2" LOC = "C4";
NET "$1I135/-SCLKIN" LOC = "D14"; #clck
NET "$1I135/-SENIN" LOC = "H13"; #sen5
NET "$1I135/-SDIN" LOC = "HI2"; #ii
# # Control Input from other FPGAs
NET "$1I135/-FPK0" LOC = "T3"; #ff0in, these 4 aren’t really needed...
NET "$1I135/-FPK1" LOC = "T4";
NET "$1I135/-FPK2" LOC = "R4";
NET "$1I135/-FPK3" LOC = "P4";
# # TTC/CCB Signals in:
NET "$1I135/CLKIN40" LOC = "C13"; #ccbclk, 3.3V
NET "$1I135/CLKIN41" LOC = "P3";
NET "$1I135/CLKFB_IN" LOC = "AE13"; #clk_fb
NET "$1I135/FCKIN_0" LOC = "B14"; #clk0
NET "$1I135/FCKIN_1" LOC = "C14"; #clk1
NET "$1I135/CK6251IN" LOC = "D13"; #clk625, 3.3V
NET "$1I135/CK6252IN" LOC = "E13"; #clk625, 3.3V
# # TTC Command bus from DCC:
NET "$1I135/RAWLIA" LOC = "C12"; #LIA
NET "$1I135/BU50" LOC = "E11"; #bus0, really IS EvCntrSt signal
NET "$1I135/BU51" LOC = "G12"; #bus1, really IS BXR signal
NET "$1I135/BU52" LOC = "G11"; #bus2
NET "$1I135/BU53" LOC = "D10";
NET "$1I135/BU54" LOC = "E10";
NET "$1I135/BU55" LOC = "F11";
NET "$1I135/BU56" LOC = "F10";
NET "$1I135/BU57" LOC = "H11";
NET "$1I135/BU58" LOC = "G10"; # really the CCB_CMD_STROBE signal
NET "$1I135/-STOP_DATA" LOC = "F13"; # From DCC
NET "$1I135/-LINK_READDATA" LOC = "G13"; #L23
NET "$1I135/SBDATA" LOC = "D12"; # Serail TTC data, Not Used
NET "$1I135/TDSYBR" LOC = "E12"; # Parallel TTC data, Not Used
NET "$1I135/TDSY0" LOC = "F12"; # From TTC? Not used
NET "$1I135/TDSY1" LOC = "D11"; # From TTC? Not used
# # Data signals in:
NET "$1I135/10" LOC = "M6"; #data0 *~all Data in pins changed!
NET "$1I135/11" LOC = "L6";
NET "$1I135/12" LOC = "R6";
NET "$1I135/13" LOC = "N5";
NET "$1I135/14" LOC = "P5";
NET "$1I135/15" LOC = "P6";
NET "$1I135/16" LOC = "N3";
NET "$1I135/17" LOC = "M3";
NET "$1I135/19" LOC = "L4";
NET "$1I135/10" LOC = "L3";
NET "$1I135/11" LOC = "K4";
NET "$1I135/12" LOC = "K3";
NET "$1I135/TDSY3" LOC = "P8"; #spare23 DDTest "DoFake3"
# FIFO Status Signals in:
NET "$1I135/OFULL" LOC = "C15"; # from output Alt.FIFO
NET "$1I135/-OPAF" LOC = "R1";
NET "$1I135/-OHALF" LOC = "R2";
NET "$1I135/-OBPR" LOC = "P2";
NET "$1I135/-OBEMPT" LOC = "N2";
NET "$1I135/OFIBER_OK" LOC = "E14"; #fck
# from input FPGas
NET "$1I135/RDCTRL0STAT0" LOC = "W1"; #rdstat0
NET "$1I135/RDCTRL0STAT1" LOC = "V7";
NET "$1I135/RDCTRL0STAT2" LOC = "V6";
NET "$1I135/RDCTRL0STAT3" LOC = "V5";
NET "$1I135/RDCTRL0STAT4" LOC = "V4";
NET "$1I135/RDCTRL1STAT1" LOC = "V3";
NET "$1I135/RDCTRL1STAT2" LOC = "V2";
NET "$1I135/RDCTRL1STAT3" LOC = "V1";
NET "$1I135/RDCTRL1STAT4" LOC = "V0";
NET "$1I135/RDCTRL2STAT1" LOC = "AC11"; #rdstat8
NET "$1I135/RDCTRL2STAT2" LOC = "AB11";
NET "$1I135/RDCTRL2STAT3" LOC = "AY12";
NET "$1I135/RDCTRL2STAT4" LOC = "AC10"; #rdstat12
NET "$1I135/RDCTRL3STAT1" LOC = "AB10";
NET "$1I135/RDCTRL3STAT2" LOC = "AA10"; #rdstat15
# # S-LINK Command & Control Signals:
NET "$1I135/LSLFO10" LOC = "R26"; #ls0
NET "$1I135/LSLFO11" LOC = "N24";
NET "$1I135/LSLFO12" LOC = "R25";
NET "$1I135/LSLFO13" LOC = "N22";
NET "$1I135/LSLFO14" LOC = "L24"; #l1r0
NET "$1I135/LSLFO15" LOC = "N23"; #efa0in, FIFO-A-Empty signal
NET "$1I135/LSLFO16" LOC = "N24"; #efa0in, FIFO-B-Empty signal
NET "$1I135/ILRL3" LOC = "M23";
NET "$1I135/~ILDOWN" LOC = "K24";  # idown
NET "$1I135/~ILFF" LOC = "K23";  # lff
NET "$1I135/USFO" LOC = "P24";  # usfo
NET "$1I135/USF1" LOC = "P23";
NET "$1I135/USF2" LOC = "R24";  # tx3+
NET "$1I135/USF3" LOC = "R23";
NET "$1I135/~OWNEN" LOC = "J24";  # Owen
NET "$1I135/~OUTCTRL" LOC = "H23";  # outctrl
NET "$1I135/~TEST" LOC = "H24";  # test
NET "$1I135/~RESET" LOC = "J23";  # reset

# OUTPUTS

# Error Signals out:
# to LEDs (use parallel-load-data pins).
# Set in Schematic now: LED1-8.  cdata[7:0]
NET "$1I135/KILL0" LOC = "AD1";
NET "$1I135/KILL1" LOC = "AC2";
NET "$1I135/KILL2" LOC = "AB4";
NET "$1I135/KILL3" LOC = "AB5";
NET "$1I135/KILL4" LOC = "AB3";
NET "$1I135/KILL5" LOC = "V6";
NET "$1I135/KILL6" LOC = "V5";
NET "$1I135/KILL7" LOC = "V4";
NET "$1I135/KILL8" LOC = "W14";
NET "$1I135/KILL9" LOC = "V14";
NET "$1I135/KILL10" LOC = "AA14";
NET "$1I135/KILL11" LOC = "AB14";
NET "$1I135/KILL12" LOC = "AC13";
NET "$1I135/KILL13" LOC = "AA13";
NET "$1I135/KILL14" LOC = "V3";
NET "$1I135/~TXENO0" LOC = "Y3";
NET "$1I135/~TXENO1" LOC = "AA1";
NET "$1I135/~TXENO2" LOC = "Y1";
NET "$1I135/~TXENO3" LOC = "W6";
NET "$1I135/~TXENO4" LOC = "N5";
NET "$1I135/~TXENO5" LOC = "W4";
NET "$1I135/~TXENO6" LOC = "N3";
NET "$1I135/~TXENO7" LOC = "W2";
NET "$1I135/~TXENO8" LOC = "Y13";
NET "$1I135/~TXENO9" LOC = "Y12";
NET "$1I135/~TXEN10" LOC = "W12";
NET "$1I135/~TXEN11" LOC = "AD12";
NET "$1I135/~TXEN12" LOC = "AC12";
NET "$1I135/~TXEN13" LOC = "AA12";
NET "$1I135/~TXEN14" LOC = "AA12";
NET "$1I135/~TXEN15" LOC = "Y18";
NET "$1I135/~TXEN16" LOC = "AA18";
NET "$1I135/~TXEN17" LOC = "AC18";
NET "$1I135/~TXEN18" LOC = "AD18";
NET "$1I135/~TXEN19" LOC = "Y17";
NET "$1I135/~TXEN20" LOC = "W16";
NET "$1I135/~TXEN21" LOC = "AA17";
NET "$1I135/~TXEN22" LOC = "AA16";
NET "$1I135/~TXEN23" LOC = "AC17";
NET "$1I135/~TXEN24" LOC = "Y16";
NET "$1I135/~TXEN25" LOC = "Y15";
NET "$1I135/~TXEN26" LOC = "AA16";
NET "$1I135/~TXEN27" LOC = "AC16";
NET "$1I135/~TXEN28" LOC = "AA15";

# Rocket IO: Gigabit Transceiver pins
NET "$1I135/RXPO" LOC = "A15";  # unused rx2+, Rocket 6
NET "$1I135/RXNO" LOC = "A17";  # tx2+, Rocket 6
NET "$1I135/TXPO" LOC = "A18";  # tx2-
NET "$1I135/TXNO" LOC = "A21";  # tx1-
NET "$1I135/RXN1" LOC = "A20";  # tx3-
NET "$1I135/TXP1" LOC = "A22";  # tx3+
NET "$1I135/TXN1" LOC = "A23";  # tx4-
NET "$1I135/RXN2" LOC = "AF21";  # rx4-
NET "$1I135/TXP2" LOC = "AF22";  # rx4+
NET "$1I135/TXN2" LOC = "AF23";  # rx3- 

# Send G-Bit fiber status signals to Front Panel LEDs
NET "$1I135/FORKOUT" LOC = "AB9";  # G-Bit Fiber OK
NET "$1I135/DAV" LOC = "AB8";  # G-Bit DAV (Xmit data to PC)

# Control Signals out:
# input FIFO control
NET "$1I135/OUT_EVK153/OUT_EVK159" LOC = "J7";  # fifo_prst
NET "$1I135/~MRST" LOC = "J6";  # fifo_mrst
NET "$1I135/BCOUT" LOC = "AC19";
NET "$1I135/LIOUT" LOC = "AB19";
NET "$1I135/~OSYNCRST" LOC = "AD19";  # syncrst
NET "$1I135/~OSOPTRST" LOC = "AB19";  # softrst
NET "$1I135/INIFIFO0" LOC = "G5";  # infifo_ce0
NET "$1I135/INIFIFO1" LOC = "J4";
NET "$1I135/INIFIFO2" LOC = "K5";
NET "$1I135/INIFIFO3" LOC = "C1";
NET "$1I135/INIFIFO4" LOC = "G4";  # infifo_ren0
NET "$1I135/INIFIFO5" LOC = "H3";
NET "$1I135/INIFIFO6" LOC = "J5";
NET "$1I135/INIFIFO7" LOC = "N5";

# Data signals out:
# Output FIFO control
NET "$1I135/CIFBOUT" LOC = "AD13";  # 78 MHz out from FPGA-DCM
NET "$1I135/OCCLK1" LOC = "M25";  # as above
NET "$1I135/IRCLK0" LOC = "K6";  # as above for InFIFO-B 041
NET "$1I135/IRCLK1" LOC = "N6";  # as above for InFIFO-B 243
NET "$1I135/OCCLK1" LOC = "J26";  # 62.5 MHz out, oscillator freq.
NET "$1I135/-IDMB_FUL0" LOC = "U7"; # spare5 InCtrl0 InUnit detects DMB_Full
NET "$1I135/-IDMB_FUL1" LOC = "T8"; # spare6 InCtrl1 InUnit detects DMB_Full
NET "$1I135/-IDMB_FUL2" LOC = "V9"; # spare14 InCtrl12 InUnit detects DMB_Full
NET "$1I135/-IDMB_FUL3" LOC = "AF8"; # spare15 InCtrl13 InUnit detects DMB_Full
NET "$1I135/SPARE0" LOC = "U5"; # spare0 to InFpga#0
NET "$1I135/SPARE1" LOC = "U4"; # spare1 to InFpga#0
NET "$1I135/SPARE2" LOC = "U3"; # spare2 to InFpga#0
NET "$1I135/SPARE3" LOC = "V1"; # spare3 to InFpga#0
NET "$1I135/SPARE4" LOC = "U1"; # spare4 to InFpga#0
NET "$1I135/SPARE5" LOC = "T7"; # spare5 to InFpga#0
NET "$1I135/SPARE6" LOC = "R7"; # spare6 to InFpga#0
NET "$1I135/SPARE7" LOC = "W11"; # spare7 to InFpga#0
NET "$1I135/SPARE8" LOC = "AC9"; # spare8 to InFpga#1
NET "$1I135/SPARE9" LOC = "AA9"; # spare9 to InFpga#1
NET "$1I135/SPARE10" LOC = "AE8"; # spare10 to InFpga#1
NET "$1I135/SPARE11" LOC = "AD8"; # spare11 to InFpga#1
NET "$1I135/SPARE12" LOC = "K7"; # spare12 to InFpga#1
NET "$1I135/SPARE13" LOC = "L7"; # spare13 to InFpga#1
NET "$1I135/SPARE14" LOC = "M8"; # spare14 to InFpga#1
NET "$1I135/SPARE15" LOC = "N8"; # spare15 to InFpga#1
NET "$1I135/SPARE16" LOC = "P22"; # spare16 to InFpga#1
NET "$1I135/SPARE17" LOC = "R22"; # spare17 to InFpga#1
NET "$1I135/SPARE18" LOC = "H22"; # spare18 to InFpga#1
NET "$1I135/SPARE19" LOC = "J22"; # spare19 to InFpga#1
NET "$1I135/SPARE20" LOC = "K22"; # spare20 to InFpga#1
NET "$1I135/SPARE21" LOC = "L22"; # spare21 to InFpga#1
NET "$1I135/SPARE22" LOC = "G22"; # spare22 to InFpga#1
NET "$1I135/SPARE23" LOC = "F22"; # spare23 to InFpga#1
NET "$1I135/SPARE24" LOC = "E22"; # spare24 to InFpga#1
NET "$1I135/SPARE25" LOC = "D22"; # spare25 to InFpga#1
NET "$1I135/SPARE26" LOC = "C22"; # spare26 to InFpga#1
NET "$1I135/SPARE27" LOC = "B22"; # spare27 to InFpga#1
NET "$1I135/SPARE28" LOC = "A22"; # spare28 to InFpga#1
NET "$1I135/SPARE29" LOC = "Z22"; # spare29 to InFpga#1
NET "$1I135/SPARE30" LOC = "Y22"; # spare30 to InFpga#1
NET "$1I135/SPARE31" LOC = "X22"; # spare31 to InFpga#1
NET "$1I135/SPARE32" LOC = "W22"; # spare32 to InFpga#1
NET "$1I135/SPARE33" LOC = "V22"; # spare33 to InFpga#1
NET "$1I135/SPARE34" LOC = "U22"; # spare34 to InFpga#1
NET "$1I135/SPARE35" LOC = "T22"; # spare35 to InFpga#1
NET "$1I135/SPARE36" LOC = "R22"; # spare36 to InFpga#1
NET "$1I135/SPARE37" LOC = "Q22"; # spare37 to InFpga#1
NET "$1I135/SPARE38" LOC = "P22"; # spare38 to InFpga#1
NET "$1I135/SPARE39" LOC = "O22"; # spare39 to InFpga#1
NET "$1I135/SPARE40" LOC = "N22"; # spare40 to InFpga#1
NET "$1I135/SPARE41" LOC = "M22"; # spare41 to InFpga#1
NET "$1I135/SPARE42" LOC = "L22"; # spare42 to InFpga#1
NET "$1I135/SPARE43" LOC = "K22"; # spare43 to InFpga#1
NET "$1I135/SPARE44" LOC = "J22"; # spare44 to InFpga#1
NET "$1I135/SPARE45" LOC = "I22"; # spare45 to InFpga#1
NET "$1I135/SPARE46" LOC = "H22"; # spare46 to InFpga#1
NET "$1I135/SPARE47" LOC = "G22"; # spare47 to InFpga#1
NET "$1I135/SPARE48" LOC = "F22"; # spare48 to InFpga#1
NET "$1I135/SPARE49" LOC = "E22"; # spare49 to InFpga#1
NET "$1I135/SPARE50" LOC = "D22"; # spare50 to InFpga#1
NET "$1I135/SPARE51" LOC = "C22"; # spare51 to InFpga#1
NET "$1I135/SPARE52" LOC = "B22"; # spare52 to InFpga#1
NET "$1I135/SPARE53" LOC = "A22"; # spare53 to InFpga#1
NET "$1I135/SPARE54" LOC = "Z22"; # spare54 to InFpga#1
NET "$1I135/SPARE55" LOC = "Y22"; # spare55 to InFpga#1
NET "$1I135/SPARE56" LOC = "X22"; # spare56 to InFpga#1
NET "$1I135/SPARE57" LOC = "W22"; # spare57 to InFpga#1
NET "$1I135/SPARE58" LOC = "V22"; # spare58 to InFpga#1
NET "$1I135/SPARE59" LOC = "U22"; # spare59 to InFpga#1
NET "$1I135/SPARE60" LOC = "T22"; # spare60 to InFpga#1
NET "$1I135/SPARE61" LOC = "R22"; # spare61 to InFpga#1
NET "$1I135/SPARE62" LOC = "Q22"; # spare62 to InFpga#1
NET "$1I135/SPARE63" LOC = "P22"; # spare63 to InFpga#1
NET "$1I135/SPARE64" LOC = "O22"; # spare64 to InFpga#1
NET "$1I135/SPARE65" LOC = "N22"; # spare65 to InFpga#1
NET "$1I135/SPARE66" LOC = "M22"; # spare66 to InFpga#1
NET "$1I135/SPARE67" LOC = "L22"; # spare67 to InFpga#1
NET "$1I135/SPARE68" LOC = "K22"; # spare68 to InFpga#1
NET "$1I135/SPARE69" LOC = "J22"; # spare69 to InFpga#1
NET "$1I135/SPARE70" LOC = "I22"; # spare70 to InFpga#1
NET "$1I135/SPARE71" LOC = "H22"; # spare71 to InFpga#1
NET "$1I135/SPARE72" LOC = "G22"; # spare72 to InFpga#1
# Foundation 3.1 does not understand high/low 50%...but required for 4.2 & 5.1
NET $11135/SCLKIN" period=75ns high 50%; #SCLKIN = aclk, 10Mhz
NET $11135/CLOCKOUT" period=22ns high 50%; #CLOCKOUT = ccclk, 40Mhz
# T_clk40 = 11.43ns was met
NET $11135/CKFB_IN" period=13.8ns high 50%; #CKFB_IN = -ck625, 6.25mhz
T_clk625 = 10.10ns was met T_clk125 = 6.23ns was met
NET $11135/FCKIN" period=5.9ns high 50%; #FCKIN=clk156,156mhz; max=6.2ns
NET $11135/CKFB_IN" period=12.0ns high 50%; #CKFB_CLK_fd, 78mhz; max=12.0ns
T_clk = 11.04ns was met T_fclk = 5.32ns was met
NET $11135/1113030/DRCK1" period=30ns high 50%;
NET $11135/1113030/DRCK2" period=30ns high 50%;

# FCKIN -> CLK156 comes from 156MHz oscillator, goes to FD to drive CLK78;
# these are used to drive CKFB_OUT (78mhz, 1 load to FPGA:CKFB_IN, Slow
# 8mhz and 1 load to FPGA: CKFB_Node, Fast
# CKFB_OUT is phase-delayed by ~1.7ns due to Drive settings, but IRCLks
# get delayed up to 0.5ns due to extra loads, so:
# at the pins, CKFB_IN is slower than IRCLks by 1.2-1.6ns
Roundtrip time to FIFO is 0.3-0.6ns, and this can be absorbed into the
lock delays such that
**effectively, CKFB_IN is slower than IRCLks 0.6 - 1.3ns**
This matches well with the FIFO RCLK-to-Output lag of 2.4-4.5ns.
Mean of FIFO RCLK-to-Output lag=2.5-3.6ns for Empty, 2.5-4.3ns for Data.
Note that (CKFB_IN -> CLK/FCLK) has an IBUF delay and so does FIFO Data,
so the IBUF delays CANCEL OUT: check constrains for double-counting.
Constraints need only use Effective CKFB lag & FIFO RCLK-to-Output lag.

# Specify timing for CLK_REG-to-FCLK_REG as 6ns each way:
INST $11135/11133030/812986" TNN=FFS FWRCREG;
INST $11135/11133030/813205" TNN=FFS FWRCREG;
INST $11135/11133030/813206" TNN=FFS FWRCREG;
INST $11135/11133030/813207" TNN=FWS FWRCREG;
TIMESPEC "TS_FTO0" - FROM "FCLK" TO "CLK0" 6.0ns;# < 5.09ns was met
TIMESPEC "TS_TO0" - FROM "FCLK" TO "CLK0" 6.0ns;# < 2.67ns was met

# Specify timing for Empty-to-FCLK_REG accounting for FIFO-Output lag,
with Period=6.0ns: < 2.87ns was met!
TIMESPEC "TS_MTOF-FOMM" "INMT":"TO-FWRCREG" 3.0ns;#assume ISE inc.IBUF delay
< 3.08ns was met
TIMESPEC "TS_MTOF-FROM":"INMT":"TO-FWRCREG" 2.4ns;#assume ISE inc.IBUF delay
MT & CLK IBUF delays could affect, but ISE probably double-counts: add 0.8ns?
# Period=6.0ns; < 3.4ns TS_MTofF

# Specify timing from -CLK IFDDRs to CLK REGs as 6.0ns:
TIMESPEC "TS_QTO0" - FROM "FDAT" TO "CLK0" 6.0ns;# < 5.67ns was met

# FIFO REN has 1.5ns Setup, plus 1ns clk/rclk difference: 6.4-2.5-3.9ns
Use when IRCLK driven by CLK156 via OFDDR: assume that ISE inc. CKFB_IN
TIMEGRP "FREN" OFFSET = OUT 3.9ns AFTER "$11135/CKFB_IN"; # IBUF delay
< 2.78ns was met
DPU VMEctrl FPGA UCF file
# from switches
NET $11135/SW0" loc = "B19";#to Switch1
NET $11135/SW1" loc = "A19";
NET $11135/SW2" loc = "D18";
NET $11135/SW3" loc = "C18";
NET $11135/SW4" loc = "B18";
NET $11135/SW5" loc = "A18";
NET $11135/SW6" loc = "D17";
NET $11135/SW7" loc = "C17";#to Switch8
# VME Signals in (and in/out): (all new)
# VME Data (2-way I/O)
NET $11135/VMECLKN" loc = "AB12"; #clk_16
NET $11135/VMED0" loc = "E18";
NET $11135/VMED1" loc = "F18";
NET $11135/VMED2" loc = "D21";
NET $11135/VMED3" loc = "C22";
NET $11135/VMED4" loc = "E19";
NET $11135/VMED5" loc = "E20";
NET $11135/VMED6" loc = "E21";
NET $11135/VMED7" loc = "E22";
NET $11135/VMED8" loc = "G21";
NET $11135/VMED9" loc = "G22";
NET $11135/VMED10" loc = "H19";
NET $11135/VMED11" loc = "H20";
NET $11135/VMED12" loc = "H21";
NET $11135/VMED13" loc = "H22";
NET $11135/VMED14" loc = "J17";
NET $11135/VMED15" loc = "J18";
# VME Control
NET $11135/-ISYSFAIL" loc = "J19";
NET $11135/-ISYSERR" loc = "J21";
NET $11135/-OTSYSMUTE" loc = "J22";
NET $11135/-OTSVME" loc = "C21";
NET $11135/-IDSO" loc = "K17";
NET $11135/-IDSI" loc = "J20";
NET $11135/-INA2" loc = "K19";
NET $11135/-IBERR" loc = "J20";
NET $11135/-ILWORD" loc = "K18";
NET $11135/-IACK" loc = "K20";
NET $11135/-INACK" loc = "L21";
NET $11135/-ODTACK" loc = "L22";
NET $11135/-IACKIN" loc = "AA15"; # was testpoint14
NET $11135/-ITACK" loc = "AA15"; # was testpoint14
# Ground -DOE inside FPGA:
NET $11135/-DOE" loc = "C22"; # also for DDU4;
# VME Address Mode
NET $11135/AMS5" loc = "K21";
NET $11135/AMS0" loc = "K22";
NET $11135/AMS1" loc = "L17";
NET "$1I135/AM2" LOC = "L18";
NET "$1I135/AM3" LOC = "L19";
NET "$1I135/AM4" LOC = "L20";
NET "$1I135/AM2" LOC = "J4";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J6";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/AM4" LOC = "J3";
NET "$1I135/AM2" LOC = "J2";
NET "$1I135/AM3" LOC = "J1";
NET "$1I135/OTCK6" LOC = "W2";
NET "$1I135/OTMS6" LOC = "W1";
NET "$1I135/OTDI7" LOC = "V5";  #tdi7, to Output FIFO
NET "$1I135/OTDO7" LOC = "U5";  #tdo7, from Output FIFO
NET "$1I135/OTCK7" LOC = "Y2";
NET "$1I135/OTMST7" LOC = "Y1";
NET "$1I135/OTDI8" LOC = "N4";  #tdi8, to DDU_Ctrl FPGA
NET "$1I135/OTDO8" LOC = "N3";  #tdo8, from DDU_Ctrl FPGA
NET "$1I135/OTCK8" LOC = "N2";
NET "$1I135/OTMS8" LOC = "N1";
# Serial path control
# Flash RAM:
NET "$1I135/M_WR" LOC = "AA18";
NET "$1I135/M_RD" LOC = "AB18";
NET "$1I135/M_SI" LOC = "W17";
NET "$1I135/M_SCLK" LOC = "Y17";
NET "$1I135/M_RST" LOC = "AA17";
NET "$1I135/M_CS" LOC = "AB17";
# Serial ADC (temp & volt monitors)
NET "$1I135/ADC_CLK" LOC = "U14";  #
NET "$1I135/ADC_DIN" LOC = "V14";  #
NET "$1I135/ADC_DOUT" LOC = "W14";  #
NET "$1I135/ADC_CS" LOC = "W12";  #
# Other Serial destinations:
# Foundation 3.1 does not understand high/low 50%...but required
NET "$1I135/OSCLK" LOC = "AA5";  #sclk, out to destinations
NET "$1I135/SI" LOC = "AB5";  # out to destinations
NET "$1I135/-SEN4" LOC = "U12";  #Output FIFO
NET "$1I135/-SENS" LOC = "V12";  #DDU_Ctrl FPGA
# Data signals out:
# Spare signals to DDU_Ctrl, InCtrl
NET "$1I135/VME0" LOC = "AB14";  # out for Fake_LIA_Enable to DDUctrl
NET "$1I135/VME1" LOC = "U13";  # out for GbE_PreScale[bit0]
NET "$1I135/VME2" LOC = "V13";  # out for GbE_PreScale[bit1]
NET "$1I135/VME3" LOC = "W13";  # out for GbE_PreScale[bit2]
NET "$1I135/VME4" LOC = "Y13";  # out for Kill SLink_4Enable DAVLCT
# check from DMB
NET "$1I135/VME5" LOC = "AA13";  #use as AutoSLD_EN signal to DDU_Ctrl
NET "$1I135/VME6" LOC = "AB13";  #use as ~LD6 (Board ID) signal to DDUctrl
NET "$1I135/VME7" LOC = "V14";  #use as ~LD6 signal from DDU_Ctrl
NET "$1I135/INPUT0" LOC = "M6";  #to/from InFPGA0
NET "$1I135/INPUT1" LOC = "M5";
NET "$1I135/INPUT2" LOC = "M4";  # in for EvCntRst from InFPGA0
NET "$1I135/INPUT3" LOC = "M3";  # out for Fake_LIA_Enable to InFPGA0
NET "$1I135/INPUT4" LOC = "M2";  #to/from InFPGA1
NET "$1I135/INPUT5" LOC = "M1";
NET "$1I135/INPUT6" LOC = "V7";  # in for BCO from InFPGA1
NET "$1I135/INPUT7" LOC = "V6";  # out for Fake_LIA_Enable to InFPGA1
# Testpoints:
NET "$1I135/TP_0" LOC = "AA12";  # testpoint7
NET "$1I135/TP_1" LOC = "W10";  # testpoint12
NET "$1I135/TP_2" LOC = "W10";  # testpoint13
NET "$1I135/TP_3" LOC = "AA15";  # testpoint14
NET "$1I135/TP_4" LOC = "AA11";  # testpoint76
NET "$1I135/TP_5" LOC = "AA11";  # testpoint77
NET "$1I135/L0_0" LOC = "E17";  #la0_0
NET "$1I135/L0_1" LOC = "E16";
NET "$1I135/L0_2" LOC = "D15";
NET "$1I135/L0_3" LOC = "C15";
NET "$1I135/L0_4" LOC = "B15";
NET "$1I135/L0_5" LOC = "A15";
NET "$1I135/L0_6" LOC = "D14";
NET "$1I135/L0_7" LOC = "C14";
NET "$1I135/L0_8" LOC = "B14";
NET "$1I135/L0_9" LOC = "A14";
NET "$1I135/L0_10" LOC = "E14";
NET "$1I135/L0_11" LOC = "E13";
NET "$1I135/L0_12" LOC = "D13";
NET "$1I135/L0_13" LOC = "C13";
NET "$1I135/L0_14" LOC = "B13";
NET "$1I135/L0_15" LOC = "A13";
NET "$1I135/L0_16" LOC = "A17";  #la0_clk1
NET "$1I135/L0_17" LOC = "B17";  #la0_clk2
NET "$1I135/L1_0" LOC = "D12";  #la1_0
NET "$1I135/L1_1" LOC = "E12";
NET "$1I135/L1_2" LOC = "F13";
NET "$1I135/L1_3" LOC = "D11";
NET "$1I135/L1_4" LOC = "F11";
NET "$1I135/L1_5" LOC = "E11";
NET "$1I135/L1_6" LOC = "A10";
NET "$1I135/L1_7" LOC = "B10";
NET "$1I135/L1_8" LOC = "C10";
NET "$1I135/L1_9" LOC = "D10";
NET "$1I135/L1_10" LOC = "F10";
NET "$1I135/L1_11" LOC = "E10";
NET "$1I135/L1_12" LOC = "A0";
NET "$1I135/L1_13" LOC = "B9";
NET "$1I135/L1_14" LOC = "C9";
NET "$1I135/L1_15" LOC = "D9";
NET "$1I135/L1_16" LOC = "C12";  #la1_clk1
NET "$1I135/L1_17" LOC = "B12";  #la1_clk2
NET "$1I135/L0_11" LOC = "E13";
NET "$1I135/L0_12" LOC = "D13";
NET "$1I135/L0_13" LOC = "C13";
NET "$1I135/L0_14" LOC = "B13";