RCC Upgrade - Implementation Description
16th November 2010
E.G. Judd

This document describes the implementation of the upgrade to the RHIC Clock and Control (RCC) system for the STAR Trigger. The aim of the upgrade project is to replace the old electronics modules with new ones, for which parts are readily available, and simultaneously fix some architectural problems.

System Architecture

The existing VME-based system contains one RCC module which receives the RHIC clock from RHIC and uses it to generate four, synchronous, STAR-specific control signals: Run/Stop, Latch, Halt and Spare. The module fans out the clock and control signals, and adjusts the phase of each channel to produce all the different signals needed by the STAR Trigger Level 0 electronics. Those clock and control signals are then passed through a VME backplane to multiple RHIC Clock Fanout boards (RCF), which provide further fanout of the signals and distribute them to every trigger module (QT crate, DSM and TCU boards) individually. The TCU passes its clock on to the Trigger Clock Distribution system (TCD). Some TCD modules are then used solely to phase-adjust that clock and feed it back to the TAC boards within the Trigger system. Over the years, as the Trigger has grown, this system architecture has resulted in unnecessary complexity and a large mass of long, relatively fragile cables that is hard to manage. The system also makes inefficient use of the available physical resources. The RCC and 6 RCF boards are housed in one specially modified 21-slot 9U VME crate. The RCC itself is a 9U VME board that is very sparsely populated, while the 6 RCF boards are tightly-packed, densely populated 6U back-of-crate boards. Finally, the current RCC and RCF modules are more than 10 years old and obtaining replacement parts for them is becoming impossible.

In the new system architecture there will still be one RCC module which receives the RHIC clock from RHIC and uses it to generate the four STAR-specific control signals. This module will be the system master. As before it will fan out and then phase-adjust the clock and control signals. However, in a change from the past, it will distribute them to slave RCC modules of which there will be just one in each trigger crate. That slave module will fan out its input signals, and adjust their phase to produce the clock and control signals needed by all the trigger modules in its crate, including TAC boards. Since the master will only need to distribute signals to each crate, instead of to each board, the large mass of long cables stretching across the system will be greatly reduced. Furthermore, since the slave RCC modules will be able to generate the clocks needed by the TAC boards, the unnecessary complexity of using TCD modules to generate trigger clocks will be eliminated.

The master and slave RCC modules will be physically identical, with the difference in functionality achieved through different firmware settings. The functionality of the new RCC module will be split between two boards: the front-of-crate board (RCC2) and the back-of-crate board (RCF2). All of the processing will be implemented on the RCC2, while the RCF2 will be kept to the simplest possible cable driver. Since the STAR trigger electronics is housed in 21-slot 9U VME crates, and the RCC2 will occupy one slot in each crate, the RCF2 needs to be able to drive up to 20 cables. 20 connectors can easily fit on one RCF2; this is less than the 24 connectors on the current RCF boards. This means that each RCC2 will need to connect to just one RCF2, which can be done through a standard VME P2 backplane. One consequence of this change is to significantly reduce the problems of dealing with tightly-packed, densely populated cable driver cards that exist in the current system. Also, the special modification to the current RCC crate will no longer be needed. The new master RCC2 can be housed in any STAR standard VME crate, where it will occupy just one slot. Furthermore, experience with the existing RCC has shown that the circuitry can be implemented in a much smaller space than is available on a 9U VME board. The new RCC2 will therefore be reduced to a 6U VME board, which gives us more flexibility in deciding where it should be located.

RCC2 Functionality

All of the processing capability of the new RCC module will be implemented on the RCC2 board, as shown in the functional block diagram in Figure 1.

The RCC2 will be a 6U VME module that can fit into the existing STAR trigger system. As well as a VME interface, the RCC2 will also have an Ethernet interface. This should make it easier to monitor the status of each board during a run, and also make it possible to use these boards in standalone applications without any supporting VME infrastructure. Control of functions on the RCC2 will be managed by a Xilinx FPGA using a 66 MHz oscillator as its primary clock source.

As shown in Figure 1, the RCC2 will have 5 possible sources of the STAR experiment clock and 2 possible front panel sources of control signals. The RCC2 will also be able to accept control commands over VME or Ethernet. On receipt of those commands the master RCC2 can then generate the appropriate control signals. The 5 clock sources are:

a) 10 MHz Local oscillator. This is the power-on default
b) RHIC clock. This will be provided by RHIC on a differential BNC twin-ax connector from the V124 module.
c) TTL clock via lemo connector, for testing purposes.
d) PECL clock via 10 pin IDC connector. This is the same form factor that the RCF2 board will use to distribute clock and control signals to the STAR trigger modules (QT, DSM, etc…). Putting an identical connector on the RCC2 input allows the master RCC to distribute clock and control signals to the slave RCC2 boards.
e) Multimode fiber clock. This is the same format that some RCF2 cards will use to distribute clock and control signals to STAR trigger VME crates in remote locations. As with the PECL signals, putting an identical connector on the RCC2 input allows the master RCC2 to distribute clock and control signals to the remote slave RCC2 boards. The fiber transmission is expected to be more robust over long distances than PECL.

A user-settable register will allow the user to select one of the five clock sources, and, where appropriate, it's accompanying control signals. The default selection, on power-up, will be the 10MHz local oscillator. The selected clock will be monitored by the FPGA. The FPGA will use its faster primary clock to check that the selected clock is still toggling. If the selected clock stops toggling for a user-settable length of time, then the FPGA will set an error flag in a register that the user can monitor. The user will also be able to choose whether, when an error occurs, the RCC2 should continue to use that selected clock or fall back to the local oscillator.

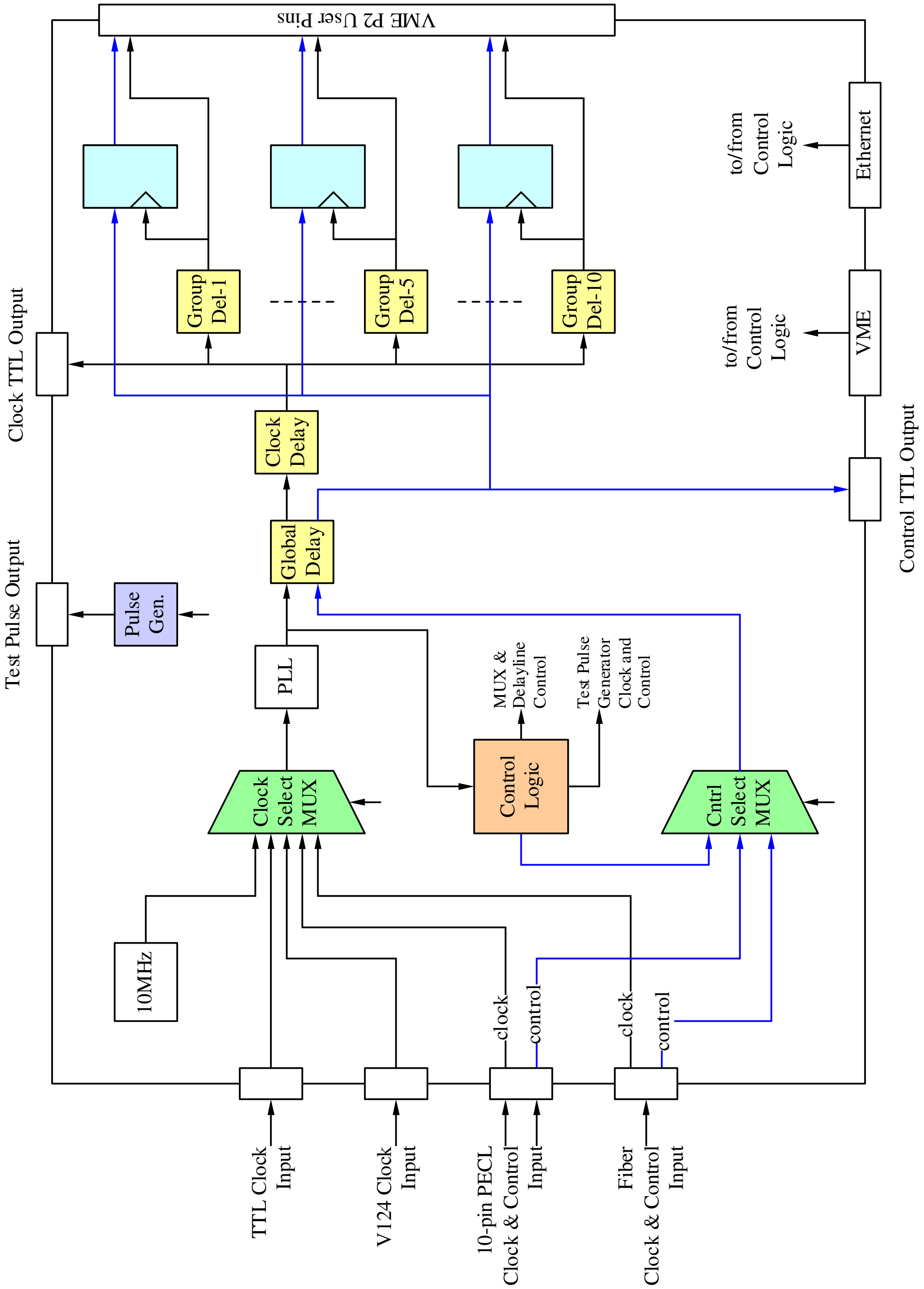

Figure 1: Block Diagram of the RCC2

The selected clock will next provide the input to a phase locked loop (PLL). The PLL will smooth out any glitches caused when switching from one clock source to another, and ensure that the selected clock has the 50% duty cycle needed by the DSMS and TCU. The output of the PLL will go back to the FPGA, where it will be used as the clock for STAR-synchronous logic. That logic will generate the four STAR-specific control signals when the RCC2 is acting as a master. It will also produce a scaled-down clock and control signals for a test pulse generator that produces PMT-like pulses. These pulses will be driven out on a front panel connector, and can then be used for testing STAR trigger electronics.

In parallel, the PLL output and the four STAR control signals will both go through a common global delay. This delay will make it possible to synchronize clock and control signals in different VME crates that receive their inputs from the master on cables of very different lengths. A crate where the clock and control signals are received early, because the cable from the master RCC is short, could have its clock and control signals delayed to align them with the signals in a remote crate that receives its signals later because of a much longer cable run. After the common global delay, the clock will also go through an additional clock delay. This is necessary because the clock will be used to latch the control signals, which must ultimately be synchronous with the clock. The additional clock delay will be used to ensure that there are no setup and hold timing violations at those latches.

The next stage is for the clock and control signals to be fanned out, phase adjusted and re-synchronized. A separate analysis of the needs of the existing STAR trigger system indicates that even though each RCF2 needs to drive up to 20 channels, it is not necessary for every one of those channels to have its own phase adjustment. The needs of the current system are driven by the BBQ crate, which currently needs just four different phases. If each TAC in that crate is to receive its own set of clock and control signals (instead of being grouped together by detector as they always have been in the past) then the number of different phases would rise to nine. The RCC2 could comfortably meet all these needs by producing ten individually-adjustable channels, each of which is then fanned out (1:2) on the RCF2 to make twenty output channels. All five signals will actually be fanned out 11 times. One copy of the clock, and one of the control signals, will be driven off the board as TTL signals for monitoring purposes. The remaining signals will be split into groups: one clock and one copy of each control signal per group. The clock will be delayed again; this is the channel-specific delay needed for the individual DSMS, TACS and TCU. The control signals will be latched on the rising edge of that delayed clock, and then all the signals will be driven through the user pins of the VME P2 backplane to the RCF2.

RCC Slave Timing

When the RCC2 is configured to be the slave (the simplest case) the timing through the board will be as shown in Figure 2. The slave RCC2 will receive a 50% duty cycle clock from the master, along with the four control signals that were latched out of the master using that clock. Only one of the four control signals (the Run/Stop signal) is shown here. Both the clock and control signals go through the same global delay. The clock then goes through the extra clock delay to shift its leading edge after the edges of the control signals. The clock is then fanned out, and each copy is delayed individually. Two cases are shown here: a minimal channel delay and a large channel delay. Each delayed clock is then used to re-latch its copy of the control signals, to ensure that the relative timing on the output channels is the same as on the input.

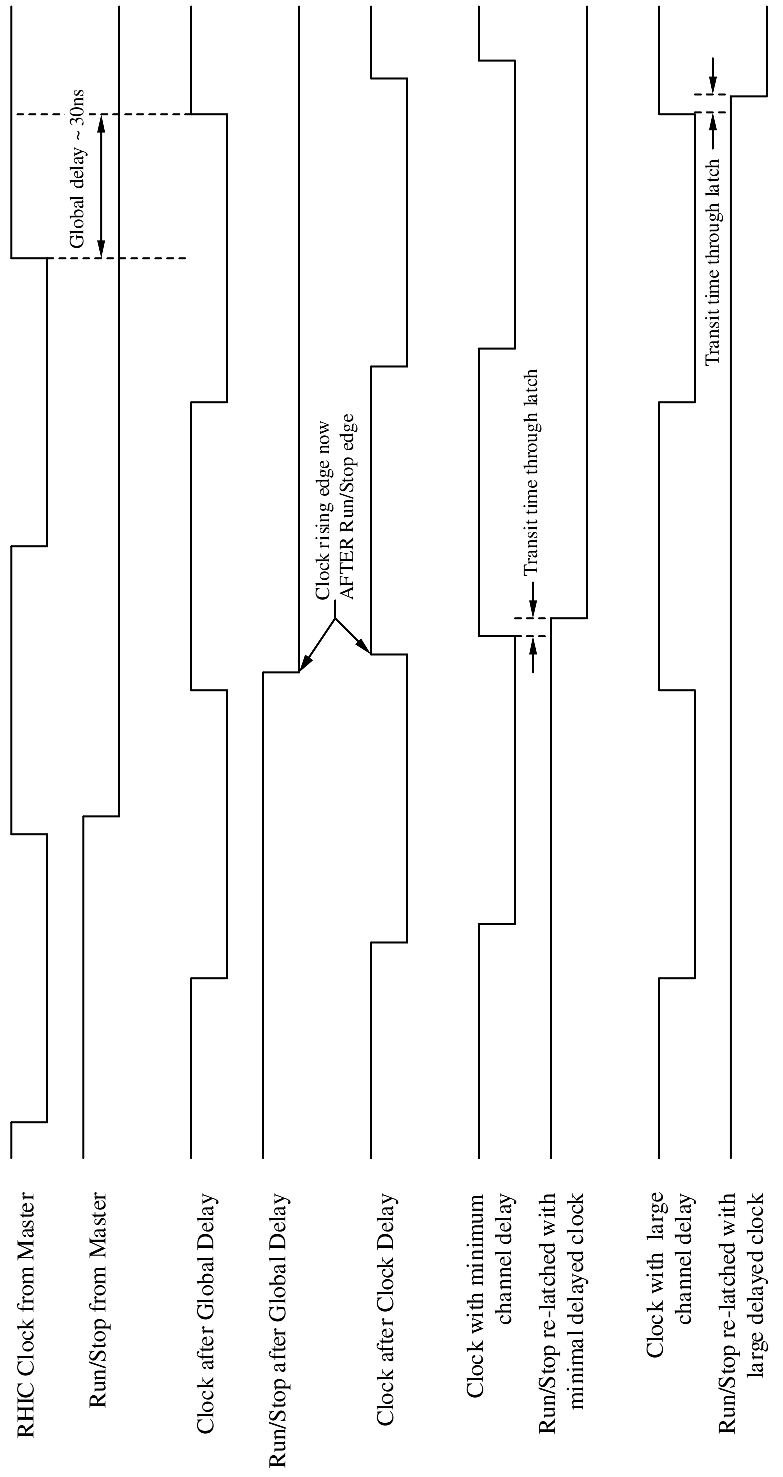
Global delay ~ 30ns
Clock rising edge now AFTER Run/Stop edge
Transit time through latch
Transit time through latch
RHIC Clock from Master
Run/Stop from Master
Clock after Global Delay
Run/Stop after Global Delay
Clock after Clock Delay
Clock with minimum channel delay
Run/Stop re-latched with minimal delayed clock
Clock with large channel delay
Run/Stop re-latched with large delayed clock

Figure 2: RCC2 Timing in Slave Mode

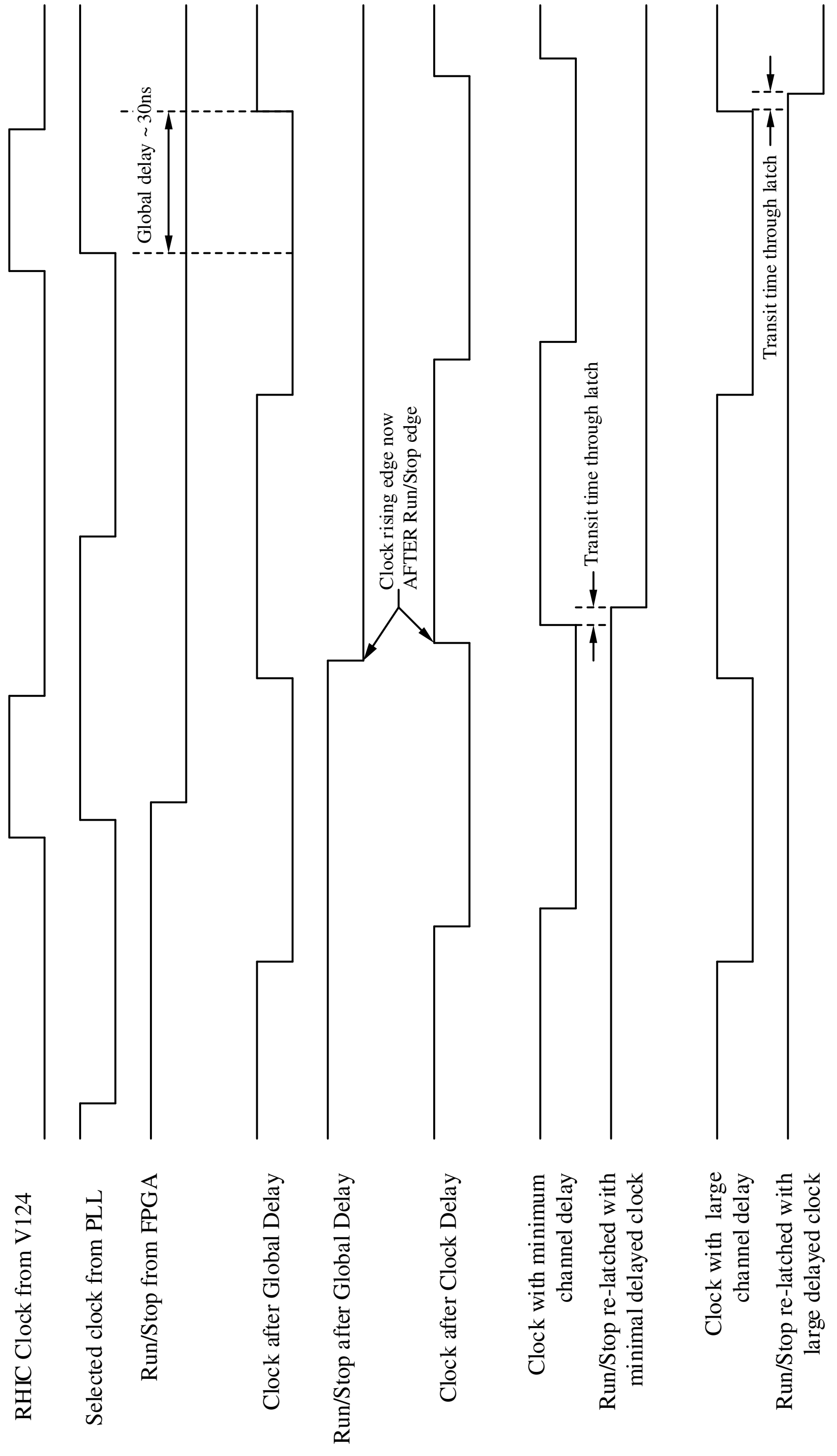
RHIC Clock from V124
Selected clock from PLL
Run/Stop from FPGA
Clock after Global Delay
Run/Stop after Global Delay
Clock after Clock Delay
Clock with minimum channel delay
Run/Stop re-latched with minimal delayed clock
Clock with large channel delay
Run/Stop re-latched with large delayed clock
Global delay ~ 30ns
Clock rising edge now AFTER Run/Stop edge
Transit time through latch
Transit time through latch

Figure 3: RCC2 Timing in Master Mode

When the RCC2 is configured to be the master the timing through the board will be slightly different, as shown in Figure 3. The master RCC2 will receive a 33% duty cycle clock (approximately) from RHIC. The PLL will convert it to a 50% duty cycle clock which the FPGA can then use to generate synchronous control signals. From that point on the timing through the delays, fanouts and latches will be the same as in the slave configuration.

Communication Circuit

The RCC2 will have two communications channels, VME and Ethernet, as shown in Figure 4. The VME communications circuitry, exclusive of the data bus, will connect directly to one small FPGA. That FPGA will be responsible for decoding the address and control bits to determine if the board is being addressed, and if the transaction is of a type that this board responds to. If the board is being addressed appropriately, the VME FPGA will generate a board select signal, and a read/write signal that will be passed on to a second, larger, FPGA: the Control FPGA. The Control FPGA is where all the control logic for the RCC2 will be implemented. It will process the VME transaction and send an acknowledge signal back to the VME FPGA, which will then generate the VME acknowledge (DTACK) signal. The VME FPGA will also be responsible for enabling and setting the direction of the bi-directional transceivers connecting the RCC2 to the VME data bus. The communications logic will be implemented in the VME FPGA as synchronous logic using a 66MHz local oscillator as the clock source. The VME FPGA will pass a copy of that clock to the Control FPGA along with the board select and read/write signals. The control FPGA can then use that clock to implement all of its communications logic that does not need to be synchronous to the RHIC clock (e.g. setting delay values etc…). The Control FPGA will have direct access to the local address and data buses so it can properly decode and respond to each VME transaction.

Overall, the circuit is typical for a VME module that responds to 32-bit transactions, but not 64-bit block transactions, and can generate an interrupt, but not function as an interrupt handler. The connection between the VME address bus and the local address bus will be implemented using simple bus drivers that always drive the VME address bus on to the local address bus. There will be no ability for data on the local address bus to be driven back onto the VME address bus, so 64-bit block transactions will not be possible. Also, when prompted by the Control FPGA, the VME FPGA will be able to generate a VME interrupt and respond appropriately when an interrupt handler drives the IACKIN/IACKOUT daisy chain. However, since it will not connect to the IACK line it will not be able to function as an interrupt handler itself.

The Ethernet core could also be implemented on the Control FPGA, but this would mean choosing a much larger FPGA than is otherwise needed, and most of it would be devoted to the Ethernet connection. It has therefore been decided to separate the Ethernet interface from the rest of the functionality of the RCC2. A ConnectCore 9P 9215 Network-Enabled Core Module will be used for the Ethernet interface. The Control FPGA and the ConnectCore module will be connected using a simple control and data bus. The ConnectCore module will also have access to the header that is used to define the VME base address of the RCC2, and a flag from the VME FPGA that will indicate when a VME transaction is in progress. In this way it will be able to determine the VME address of the board it is connected to, and when that board is being accessed over VME.

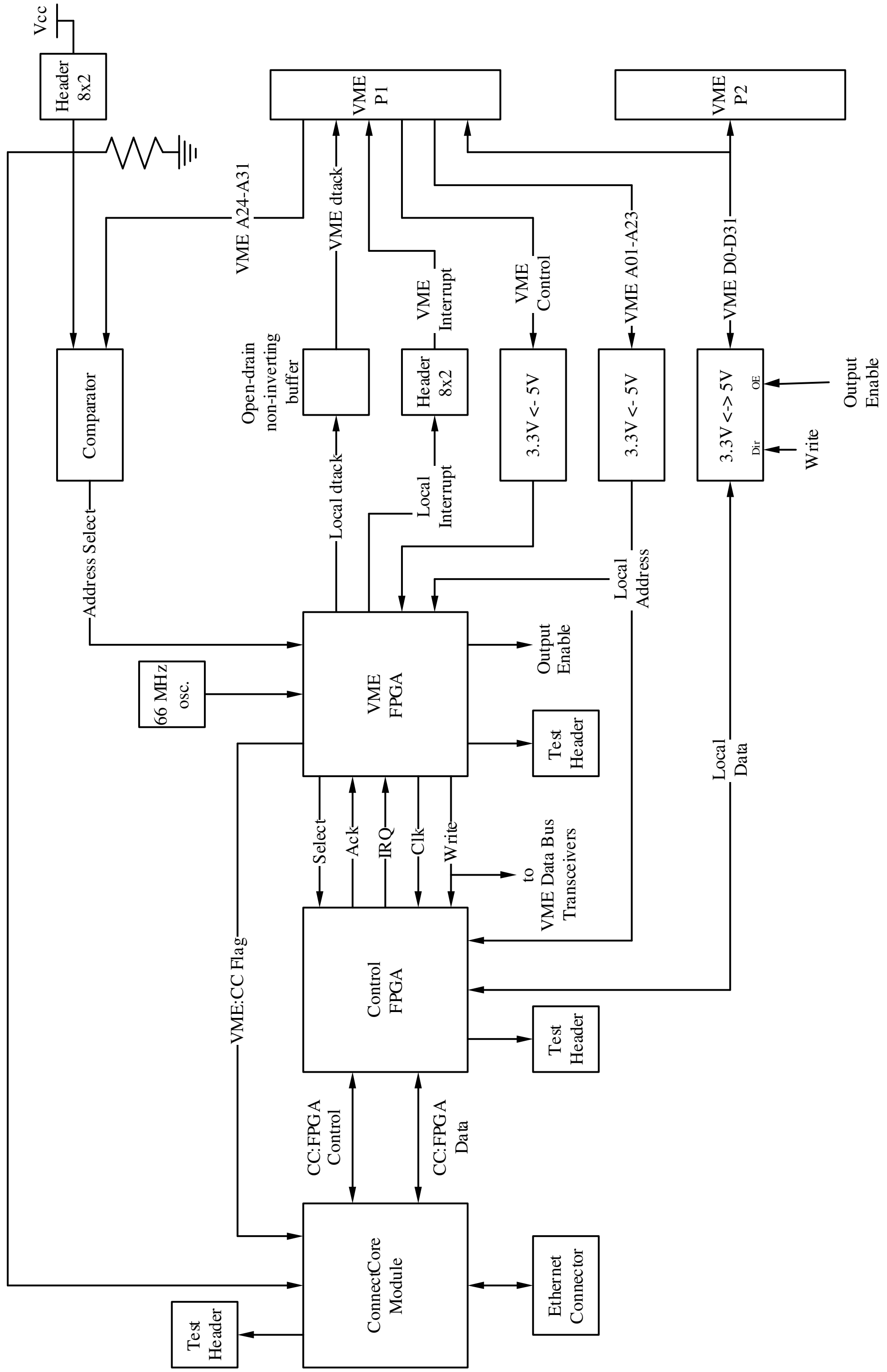

Figure 4: RCC2 VME and Ethernet Communications Circuit

Separating the communications functions in this way has several advantages:

- Two relatively small, inexpensive FPGAS can be selected because the logic that needs to be implemented on each is reduced to something simple.
- The number of connections that need to be made to each FPGA is in fact small enough that the parts do not need to be ball-grid arrays. This will greatly simplify the processes of laying out the board, loading the parts, debugging the logic and subsequently maintaining the boards.
- The VHDL development time for the FPGAS is minimized
- Since this ConnectCore module is being used on other STAR modules (the new Scalar Board and the new TCD module) development of the Ethernet interface can be shared with those groups.

Configuration Circuit

Both the FPGAS, and the Ethernet link, will need to be configured. The configuration circuit is shown in Figure 5. The Ethernet link will be set up using an RS232 connection via a console port. Based on past experience at STAR, it has been determined that it is necessary to be able to download new configuration data to at least the Control FPGA remotely, i.e. when there is no access to connect a laptop PC directly to a JTAG connection. In the past this has been accomplished by providing a VME connection to a PROM, and attempting to mock-up the JTAG connection to the PROM (e.g. on the QT boards and the latest version of the TCU). This scheme was hard to implement and has met with mixed success. A new configuration scheme is now available for Xilinx FPGAS, where they load their configuration data from a serial flash memory: Serial Peripheral Interface mode. Since the memories are widely available, with an industry-standard, published, interface it is also relatively simple to write new configuration data into them. The Control FPGA will therefore be configured in SPI mode. Ideally, the two FPGAS could be daisy-chained together, and both would be configured from the same serial flash memory. Unfortunately the Xilinx datasheets indicate that daisy-chaining from a single SPI serial flash memory is only available for certain revisions (Steps) of the silicon. Since the VME FPGA will have just the fixed logic for implementing the VME handshake, which is not going to change, it has been decided to avoid daisy-chaining complications and separate out their configuration processes. The VME FPGA will therefore be configured directly from a traditional PROM in Master Serial Mode. There will be three ways of configuring the Control FPGA:

- The FPGA can be configured directly from a remote host using a download cable and the JTAG interface.
- This can also be done using the ConnectCore module to mimic that JTAG connection.
- The FPGA can load its configuration from a serial flash memory in Serial Peripheral Interface (SPI) mode. This configuration mode was introduced by Xilinx in the Spartan-3E and Virtex-5 families. A Virtex FPGA is not necessary for this application, so a Spartan-3E FPGA will be used. A memory chip in the Forte M25P series from Numonyx (now Micron) will be used, because that is the solution that Xilinx uses as the example in its Application Note titled “Configuring Xilinx FPGAs with SPI Serial Flash”.

The serial flash memory can itself be loaded with the configuration data in any one of three ways:

- Data can be downloaded directly from a remote host via a download cable (JTAG-like connection).
- This can also be done using the ConnectCore module to mimic the remote host
- After the FPGA has been configured at least once using its JTAG connection, the FPGA can receive new data via the VME interface, or over Ethernet, and download that into the serial flash memory.

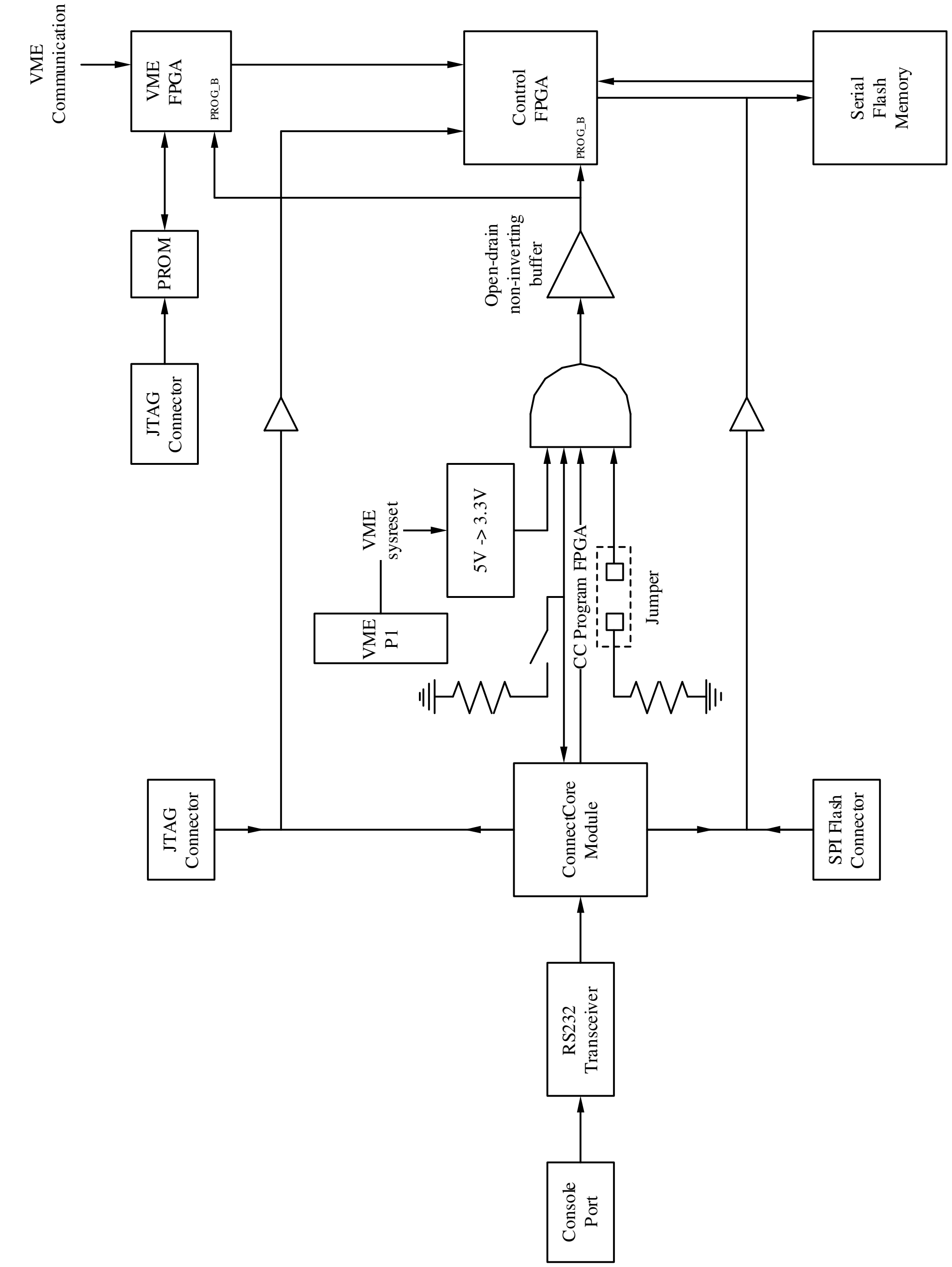

Figure 5: RCC2 FPGA and Ethernet Configuration Circuit

Once the new configuration data has been loaded into the serial flash memory the Control FPGA needs to be prompted to re-configure itself from that data. This is done by pulling the PROG_B pin on the FPGA low and then allowing it to return high. Pulling the PROG_B pin low can happen in one of four ways:

- The VME SYSRESET line is pulsed
- A manual pushbutton switch on the RCC2 is temporarily held closed. Note that this will also send a reset signal to the ConnectCore module
- The ConnectCore module can generate a PROGRAM-FPGA signal
- A jumper can be installed to hold the PROG_B pin of the FPGA low indefinitely. This last option is used to force the FPGA I/O pins into a high impedance state, which is necessary to allow direct programming access to the serial flash memory.

Fiber Connection Issues

The RCC module is essentially a clock fan-out board; it needs at most one fiber input, but potentially multiple fiber outputs. The input will be on the RCC2 and the outputs on the RCF2. This has some consequences for the design of the fiber connections.

Fiber transmitters and receivers have some circuitry to convert between light pulses and electrical signals. The stream of light pulses is a serial stream, but the electrical signals eventually need to be used in parallel, so some serialization/de-serialization circuitry is needed. The serial stream needs to have enough transitions in it to enable the receiver to recover the embedded clock and synchronize (lock) to the incoming data stream. It also needs to be DC-balanced (equal numbers of 0's and 1's) since fiber receivers are AC-coupled. For fiber connections, typically 8b/10b encoding is used, which guarantees both the number of transitions, and DC balance. The RCC module therefore also needs some circuitry to encode the clock and control information that we want to send into an appropriate form. There are several ways of doing this. Many FPGAS these days include built-in hardware for doing some/all of it (e.g. ALTERA) or provide ready-made software modules (e.g. XILINX) to do the same thing. The FPGAS can of course also provide lots of other functionality in parallel with the fiber connection. Alternatively there are dedicated chips available that have just this functionality built-in (e.g. Cypress Hotlink chipset). Finally, there are of course individual PLL, FIFO, register and memory chips that can be combined to make stand-alone circuits that can do the same thing.

Since the input and outputs will be physically separated on different boards we cannot use duplex transceivers but instead need to use separate transmitters and receivers. These components typically have some restrictions in physical placement and routing. The transmitters and receivers usually have differential inputs and outputs, and require that the differential signal paths to those ports be kept short and equal in length. If the transmitters on the RCF2 are to be driven by an FPGA, and the paths are to be kept short, then that FPGA would need to be on the RCF2. This makes the RCF2 much more complicated and that is not desired: the RCF2 is supposed to be just a simple driver board that can be easily re-designed and re-manufactured as the needs for different output form factors change. Also, the desire for simplicity on the RCF2 makes it undesirable to have a stand-alone circuit with lots of separate chips for each output channel. For the RCF2, the best solution seems to be to use a dedicated SERDES-Encoder chip that can receive slow parallel data from the RCC2 through the backplane and then drive the fiber optic transmitter directly. It would be located physically close to the transmitter to keep the trace lengths short. This minimizes both the number of chips on the RCF2 and the complexity of the circuit. If the RCF2 is going to use a dedicated SERDES-Encoder chip then it makes sense for the RCC2 to use the same scheme. The plan therefore is for the RCC2 to have a fiber receiver connected to a dedicated SERDES-Decoder chip feeding the FPGA. The FPGA will provide synchronous and asynchronous control signals to both the RCC2 and RCF2 SERDES chips, and receive status information back from them.

Ideally, the fiber optic link between the RCC2 master and slave modules would embed the selected clock, and carry just the four STAR specific control signals. The RHIC clock frequency is 9.383MHz, and the encoding procedure will increase the number of bits from four to at least five. The serial data rate would therefore be a minimum of 47MBaud (9.383 x 5). Finding a compatible set of fiber transmitters, receivers and SERDES chips that will operate at this relatively low baud rate has not been possible. Instead, the decision has been made to operate the link using a clock running at twice the selected clock frequency (18.766MHz minimum), and transferring ten bits per clock cycle instead of just five. This increases the baud rate to 188MBaud, which can be accommodated by many parts. The ten serial bits per clock cycle will be made by encoding eight parallel input bits using the standard 8b/10b encoding procedure. Only four of those eight bits will be needed for the STAR specific control signals. One of the remaining bits will therefore be used to transmit the original, slower, selected clock. On the receiving end, the slave RCC2 will have access to both the faster recovered clock and the slower selected clock.

Based on these design considerations, the HFBR-1119TZ/2119RZ Data Links from Avago and the Cypress Hotlink Transmitter/Receiver chip set (CY7B923 and CY7B933) have been chosen. In order for the link to operate reliably, the Hotlink receiver chip needs access to a reference clock. This clock is the frequency reference for the clock/data synchronizing PLL. It must be provided by a crystal-controlled time base, and the frequency must be within 0.1% of the link's embedded clock. There are two possible embedded clock frequencies: 20MHz (if the master RCC2 is operating on its 10MHz local oscillator) and 18.766MHz (if the master is using the 9.383MHz RHIC clock). The RCC2 will therefore have two crystal oscillators: 20 MHz and 18.766MHz. A MUX IC chip, controlled from the FPGA, will be used to select one of those two clocks as the fiber receiver reference clock. The output of the receiver chip includes both the recovered clock, and the 8 received data bits appropriately synchronized to that recovered clock. The FPGA will use that clock both to process the received data and to generate the synchronous control signals for the Hotlink receiver. The fiber transmission circuit on the RCF2 is described in the "RCF2 Functionality" section at the end of this document.

Input Circuit

The design of the input circuit depends both on the design of the fiber receiver (see above) and also the MUX design. In theory the clock MUX could go inside or outside the FPGA. There are some factors that could push it outside the FPGA, but those are not relevant in this design, so it can safely go inside the FPGA, which simplifies the board design.

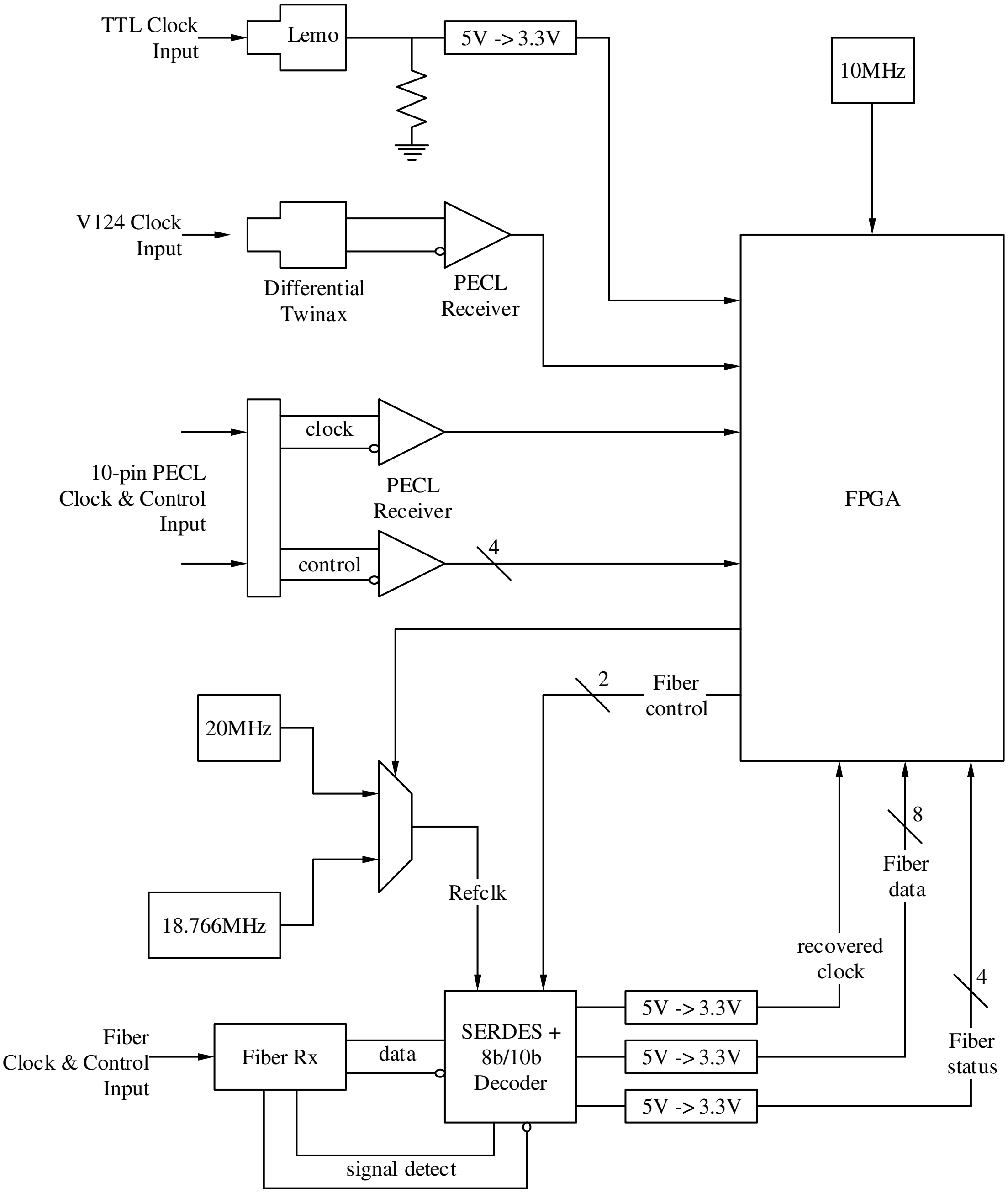

Figure 6: RCC2 Input Circuit

One such consideration is that the clock delay through the FPGA will probably be slightly different for each of the five possible clocks because they will each have a slightly different routing from their input pins to the common output pin. If the MUX was in an external chip then the transit times could be guaranteed to be equal. Careful choice of FPGA input pins can make this effect very small. Also, since we have delay downstream of the MUX that is set for each output channel independently, any extra little delays in transit time through the FPGA can be dealt with there.

Another factor is that if the MUX is inside the FPGA, and the board switches from one clock to another, there will be a glitch in the selected clock. External chips are available that accomplish this switch without a glitch. However, we do not need this feature. Downstream of the MUX, the selected clock goes through a PLL which will automatically smooth out any glitches in the distributed clock. PLL chips are also available that will produce a 50% duty cycle output clock, irrespective of the duty-cycle of the input reference clock. This feature is needed somewhere in the RCC2 because the existing STAR trigger modules need a 50% duty cycle clock, but the clock received from RHIC will have a different duty cycle. Xilinx FPGAs are not optimized for this type of duty cycle adjustment. Since smoothing the clock and adjusting the duty cycle can be accomplished by the PLL, the MUX circuit does not need to include these functions, and therefore the MUX can reasonably be implemented inside the FPGA.

Finally, if the MUX is external to the FPGA then it can be providing a clock to the downstream circuits even when the FPGA is not configured. This is not important to STAR. The RCC system has to be configured and providing appropriate control signals, as well as clocks, to all the clients before the clients can themselves be configured. Setting the control signals involves configuring the FPGA and allowing the user to set registers, so it doesn’t matter if the clock isn’t available when the FPGA is not configured. So again, the MUX can safely be put inside the FPGA.

The same basic reasoning about the clock MUX can also be applied to the control MUX. Based on these considerations, the input circuit for the RCC2 will be implemented as shown in Figure 6. All five possible clock inputs, and the two associated sources of control signals, will be received and routed to the FPGA. Both the clock and control MUXes will be implemented internally within the FPGA.

Output Circuit

Once the source of clock and control signals has been selected by the FPGA, the signals need to be fanned out, phase adjusted and re-synchronized. This circuit is shown in Figure 7.

- The PLL will be a Cypress CY7B991V 3.3V RoboClock part. This part can be configured to operate comfortably with a 10 MHz reference input clock, and it is guaranteed to produce 50% duty cycle outputs. In this configuration it also, automatically, produces output clocks with a frequency that is twice the input clock frequency. This is exactly what is needed for the fiber optic Tx/Rx circuits, so this part will be used there too.
- All of the delay line chips will be the same. The 3D3428-0.5 Monolithic 8-bit programmable delay line from Data Delay Devices has been selected. This device has a step size of 0.5ns and a delay range of 127ns. Each chip has an enable signal, and a set of 8 pins for parallel data input. The FPGA will produce one 8-bit data bus that is shared by all the delay line chips, and then a set of enable signals so the user can program each delay line individually. The exception to this will be the global delay chips, which will

share an enable signal. This will guarantee that the global delays are all set to the same value.

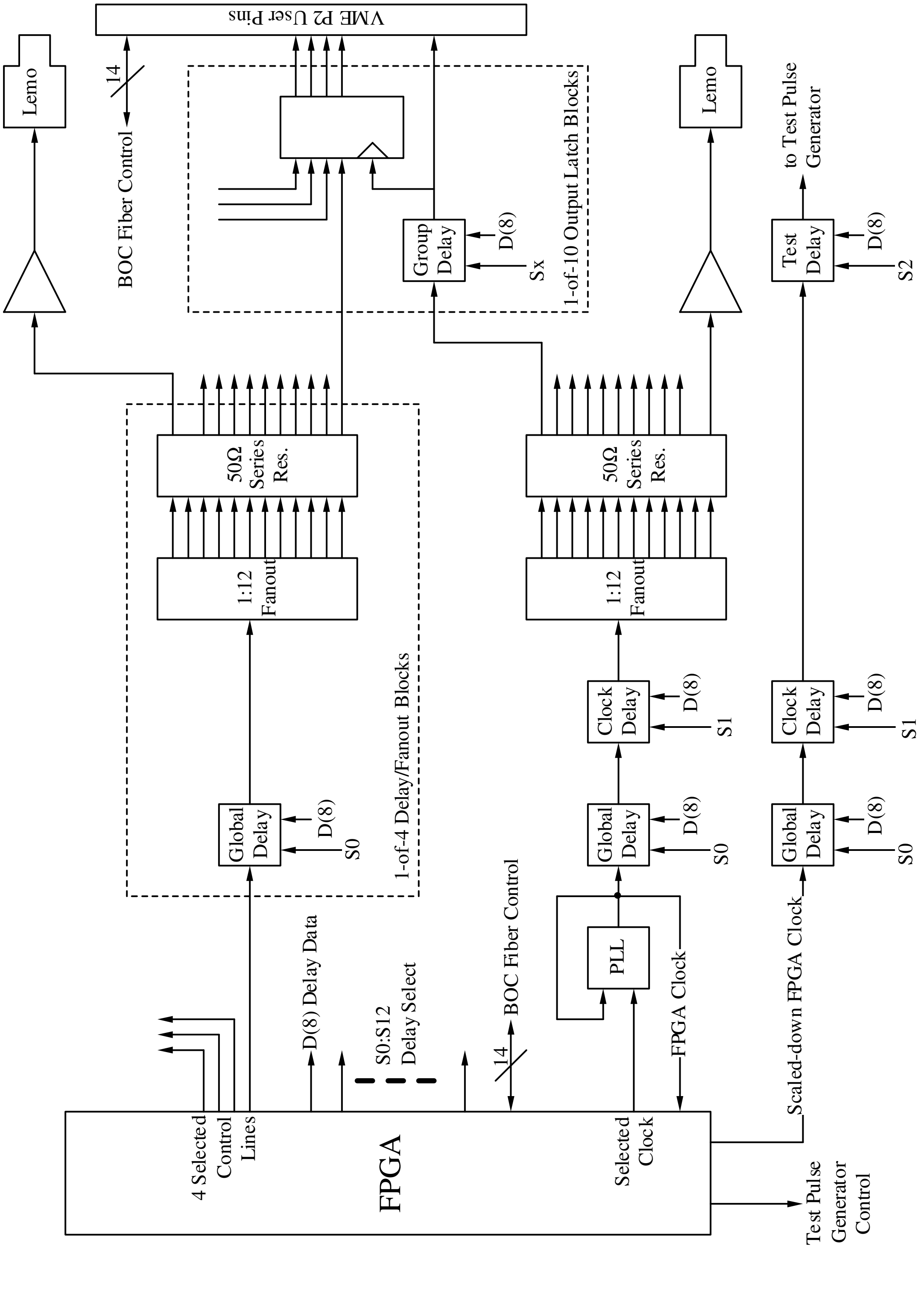

Figure 7: RCC2 Output Circuit

- The fanout buffers will be implemented using the MPC9448 3.3V LVCMOS 1:12 Clock Fanout Buffer from IDT. Cypress also makes an equivalent part, the CY29948, that is pin compatible. The outputs from both chips drive 50Ω terminated transmission lines, so a resistor pack will be needed.
- In the re-synchronization stage, the control signals will be latched using a 3.3V CMOS 16-Bit Register. These are from the same basic family of parts used in the VME interface circuit. Each register can be operated as two 8-bit registers so only five will be needed to implement all 10 output channels.

Test Pulse Generator Circuit

In order to assist in the testing of the STAR trigger electronics system it has been decided to add a feature to the RCC2 that does not exist in the current RCC system. The RCC2 will generate a PMT-like pulse at a user-settable rate that is synchronous with the selected clock. The pulse will have a leading edge time of no more than 2ns. The user can set the amplitude in the range from 0 to -2.5V. This circuit will be controlled from the FPGA, which will produce a scaled-down copy of the clock from the PLL to govern the pulse rate. This circuit is still being designed.

RCF2 Functionality

The RCF2 board will be a 6U VME back-of-crate card, designed to cover the P2 and P3 backplanes. This form factor will enable it to fit into the existing card cages at the back of the STAR trigger VME crates. It will connect to just the P2 backplane.

It is expected that there will be at least two versions of the RCF2. The first version will produce purely PECL outputs, as shown in Figure 8. It will provide all the signals that are needed by the current set of STAR trigger modules: TAC, QT, DSM and TCU. The second version will produce a mix of PECL and fiber optic outputs, as shown in Figure 9.

It was explained in the “Fiber Connection Issues” section (pages 10 and 11) that the fiber optic link will operate at a frequency that is twice the selected clock frequency. Therefore, in Figure 9 in the fiber optic output block, the clock frequency is first multiplied by two by a PLL. That fast clock is then used as the write-clock input to the Hotlink transmitter chip. The chip latches its input data, and the synchronous control signals, on the rising edge of its write-clock. However, because of the way the RCC2 output circuit is constructed, the edges of the data signals could be very close to that rising edge. An inverter may be used to move the clock edges away from the data edges to avoid setup and hold timing conflicts. In parallel, the fast clock will also be used to latch any control signals coming from the FPGA that need to be synchronous with the write-clock. The data input to the transmitter chip will consist of the four STAR specific control signals, and a copy of the original (slow) clock that is also provided by the PLL. The Hotlink transmitter chip produces two identical output channels, so there is no need for any further fanout circuitry.

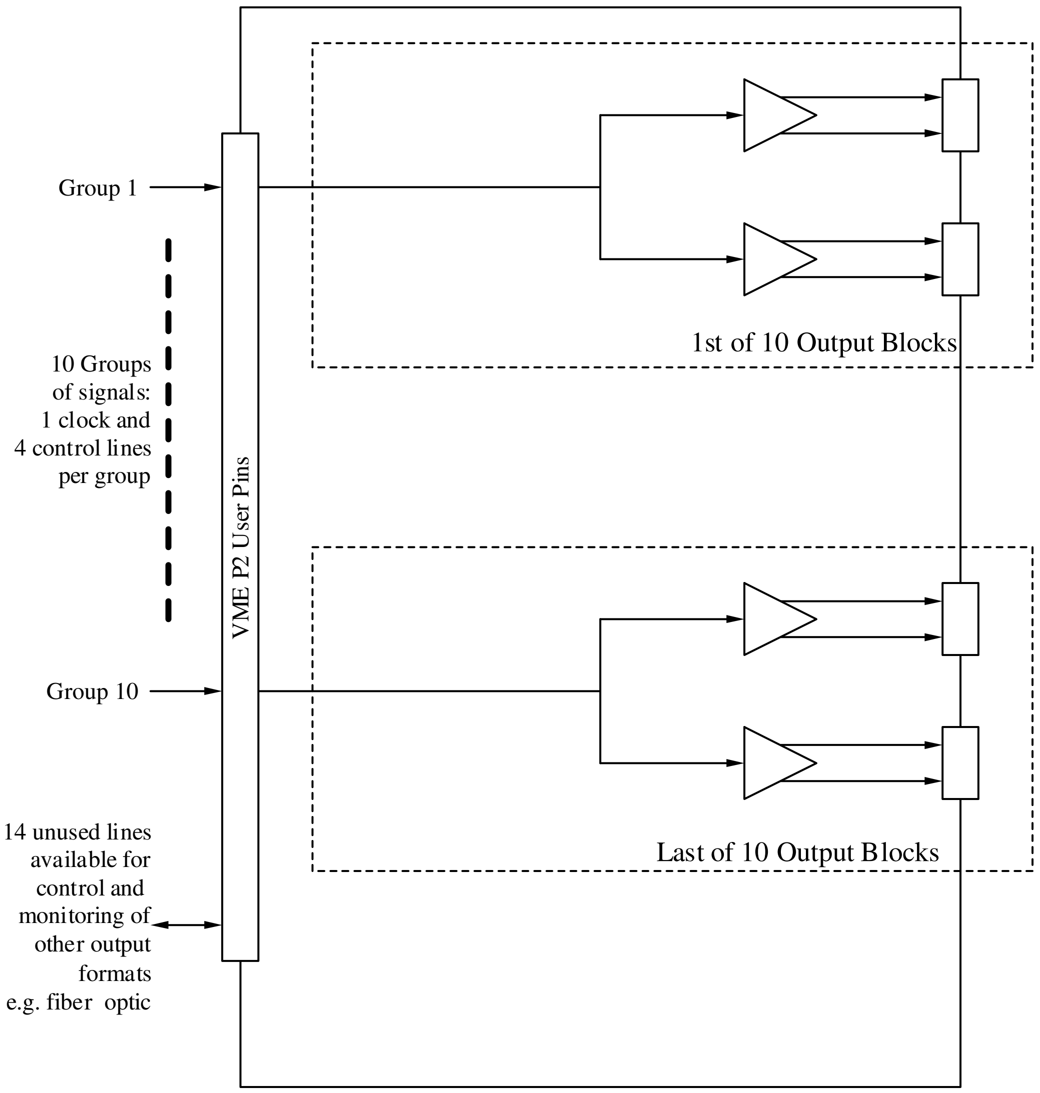
Group 1
10 Groups of signals: 1 clock and 4 control lines per group
Group 10
14 unused lines available for control and monitoring of other output formats e.g. fiber optic
VME P2 User Pins
1st of 10 Output Blocks
Last of 10 Output Blocks

Figure 8: Block Diagram of the PECL-only RCF2

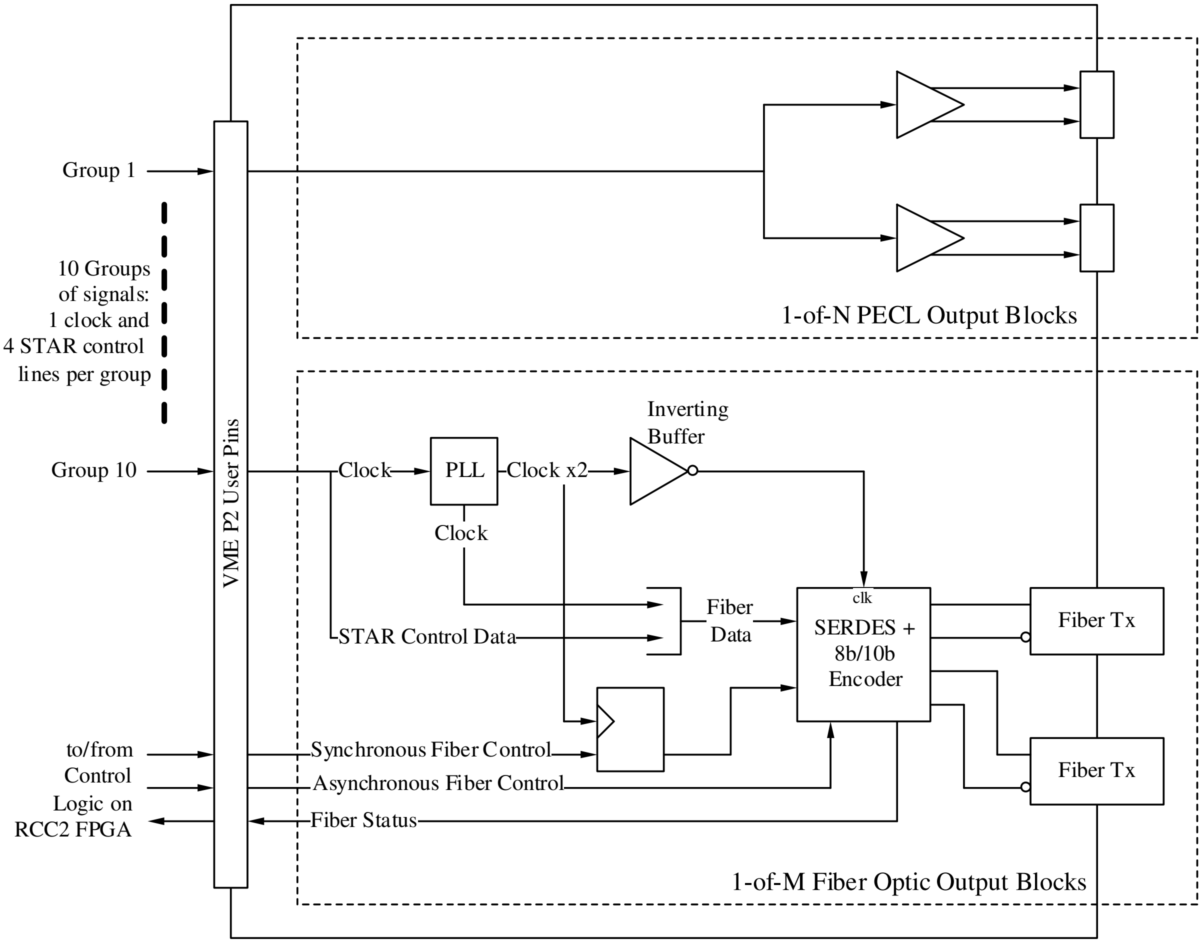
Group 1
10 Groups of signals: 1 clock and 4 STAR control lines per group
Group 10
to/from Control Logic on RCC2 FPGA
VME P2 User Pins
1-of-N PECL Output Blocks
Inverting Buffer
Clock
PLL
Clock x2
Clock
STAR Control Data
Fiber Data
clk
SERDES + 8b/10b Encoder
Fiber Tx
Fiber Tx
Synchronous Fiber Control
Asynchronous Fiber Control
Fiber Status
1-of-M Fiber Optic Output Blocks

Figure 9: Block Diagram of the mixed PECL-Fiber RCF2