Fully redundant clock generation and distribution with dynamic oscillator switchover

M. J. Mueller
U. Weiss
T. Webel
L. C. Alves
W. J. Clarke
M. Strasser
E. Engler
G. Cautillo
H. Osterndorf
J. Schulze

This paper describes a fully redundant clock generation and distribution approach with a fully dynamic switchover capability and concurrent repair. It highlights the challenges as the design evolved from a single source, to a "cold" standby backup, and finally to a fully redundant transparent switchover with no interruption of the workloads running on an IBM System z9™. The function split between hardware and the various levels of firmware is described, including the methods to determine the defect component in the clock distribution paths. Finally, we describe the joint effort with a major chip technology vendor to design and develop the necessary circuitry, according to the z9™ requirements, for clock synchronization and switching.

Introduction

Each zSeries* generation is evaluated to determine the most significant sources of outages, and design changes are then made to remove these sources [1]. The traditional single-source clock generation design was inherently a single point of failure, which was addressed in the IBM zSeries 990 by providing two independent oscillator cards, each housed in a separate field-replaceable unit (FRU) [2]. This design does not prevent the system from going down when the oscillator fails, but it does enable a restart on the other oscillator card and a subsequent concurrent repair of the failed card. The detection of failures in the oscillator signal distribution was basically limited to a total loss of pulse or signal. The new design for the IBM z9* improves the detection of many clock signal failure modes, it provides a dynamic switchover to the alternate oscillator without any disruption to the application programs that are running, and it enables full repair concurrently.

Clock generation and distribution design evolution

The conventional clock signal is usually generated by a crystal oscillator with a relatively low frequency. Depending on the system requirements, this frequency is multiplied by means of a phase-locked loop (PLL) to a higher frequency, which allows distribution over different package levels—modules, cards, and boards—to the processor chips. The upper limit of this frequency is defined by the card and board material and the length of the oscillator distribution wires. A buffer circuit sends individual oscillator signals to all processors. Finally, an additional PLL, located on the processor chip, multiplies the oscillator signal to the frequency required by the processor. New system requirements, such as programmable oscillator frequency, result in additional components and reduce the reliability of the oscillator clock signal generation and distribution. They also increase the complexity of this function and the risk for design flaws.

Adding a second source of clock generation is a first step toward a failsafe, redundant oscillator generation and distribution design. Together with a multiplexer on the processor cards, it makes it possible to switch from a failing oscillator to a backup oscillator clock source. However, such a design is not capable of automatic switchover and is unable to detect certain classes of failure modes.

The next step forward is the introduction of intelligent monitoring and switchover. In such a design it is critical that all potential failure modes be addressed early in the design phase to ensure optimum effectiveness. If the design covers only a few classes of defects, the complexity and additional failure rates of the added components may offset the availability benefits. In addition, all support logic and firmware must be adequately robust to ensure

0018-8646/07/$5.00 © 2007 IBM

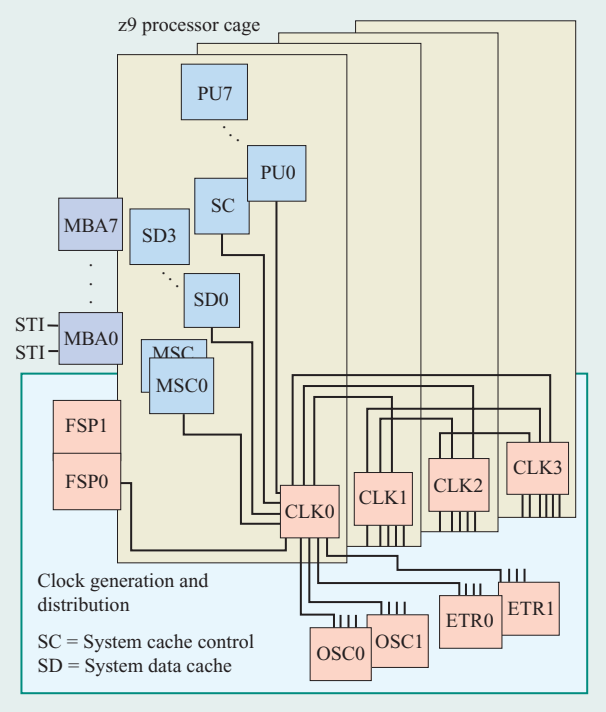


Figure 1

System z9 clock signal generation and distribution system structure.

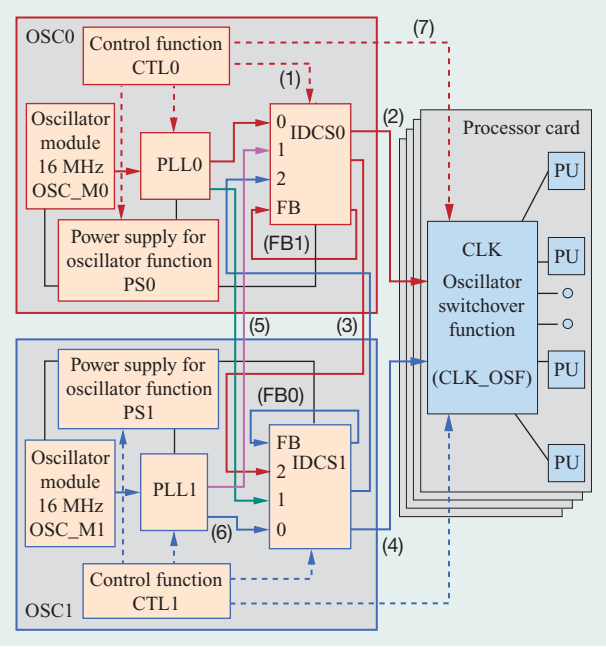


Figure 2

System z9 redundant oscillator signal generation and distribution with automatic switchover function. (FB = feedback.)

that there are no single points of failure left in the overall design. The z9 clock signal generation and distribution design addresses all of these issues.

System z9 clock signal generation and distribution

The z9 design is based on a superscalar microprocessor architecture. The central electronic complex (CEC) uses the same processor book concept introduced with the IBM zSeries 990. The processor book contains processor units (PUs), cache, memory, and the cross-books cache coherency fabric (**Figure 1**). The memory bus adapter (MBA) chips are now on the MBA fan-out cards that are plugged into the processor book. In addition, each processor book contains a clock chip (CLK). Each CLK receives both clock signals from each of the oscillator cards and external timer reference (ETR) signals from each of the ETR cards. A Sysplex Timer* [3] provides a common reference time to each zSeries system clustered together to form a Parallel Sysplex*. The firmware required for the CLK initialization and run-time clock distribution runs on redundant controllers, called *flexible service processors* (FSPs) [4]. This redundancy ensures that firmware control is based on the latest status of the clock distribution structure.

There were several significant challenges to the overall design. First, because the z9 is a synchronous multiprocessor system, the interfaces between processor chips require low-skew[1] oscillator signals. Therefore, even during an oscillator switchover, the skew has to be maintained within a small, tolerable range. Second, no missing pulses were allowed to occur during switchover. This requires that the first missing oscillator pulse must be detected, and the switchover to the backup oscillator completed within one cycle. Also, the detection design and switchover was required to handle many different types of failures modes, such as power supplies, crystal oscillator, PLL, monitoring and switching circuits, frequency drifts, and clock signal wire. Finally, the oscillator repair had to be performed concurrently with system operation.

Design overview

The System z9 redundant clock generation and distribution with dynamic switchover is based on the Intelligent Dynamic Clock Switch (IDCS) [5] module from Freescale Inc.[2] and the customized z9 CLK (**Figure 2**). Because the z9 design engineers were involved early in the definition and design stages of the IDCS, they

[1] *Skew* is the difference in the arrival time of a clock signal at two or more circuits. Ideally the clock skew should be zero. Skew is introduced by wire length differences, manufacturing variations, and other physical characteristics.
[2] Freescale Semiconductor, Inc. sold its Timing Solutions business to Integrated Device Technology, Inc.

were able to influence the specification of the circuit to meet the rigorous IBM z9 requirements.

Two design options were initially considered. First, the IDCS module would be mounted directly on each of the processor boards with two external separate oscillator sources. However, this option was dropped because an IDCS failure would cause the entire processor card to fail as the output frequency decreased slightly during a switchover (all four processor cards are required to be fully frequency-synchronized). The second option, mounting the IDCS on each oscillator card and cross-feeding the output of each card into the IDCS, resolved this issue. However, the design was still exposed to failures of the voltage regulator modules or the IDCS itself. Finally, a combination of the two options was selected. In the final design, the IDCS is placed on the oscillator card, and the dynamic oscillator switchover function is implemented on the CLK oscillator switchover function (CLK_OSF).

With this design approach, the switchover from one oscillator signal to a secondary oscillator signal can occur in the IDCS on the oscillator card or in the CLK_OSF on the CLKs. The IDCS automatic switchover on the oscillator card occurs when one of its asynchronous inputs fails [(1) in Figure 2]. During the IDCS switchover, the frequency at the output is slightly reduced until the circuit is relocked to the secondary input.

The CLK_OSF automatic switchover occurs when the primary input signal (2) fails. During switchover to the secondary input signal (4), a phase jump occurs that causes a single oscillator cycle to be slightly longer than all other cycles. In order to achieve CLK_OSF automatic switchover, both input signals (2) and (4) must be synchronized, and the secondary input signals must be early with respect to the primary input signal. The synchronization is achieved by feeding the output from IDCS0 on OSC0 to the input of IDCS1 on OSC1. Therefore, both CLK input signals are generated from PLL0. The delay between the primary and secondary input signal is generated by adding a delay line in the feedback path of the IDCS on the oscillator card (labeled FB in Figure 2). The delay takes into account the tolerances of the IDCS and wiring. The delay value has to be greater than zero, typically 500 ps, to allow shifting the secondary signal by means of delay lines in the CLK_OSF until the rising edge of the secondary signal occurs approximately 50 ps later than the rising edge of the preliminary signal at the inputs of the input select block. The timing of oscillator signals (2) and (4) sent to the CLK and the internal time of the CLK_OSF are shown in **Figure 3**.

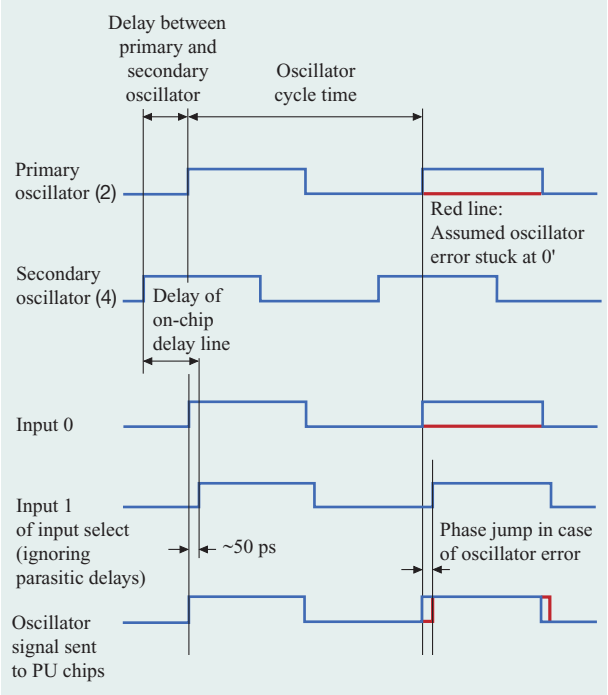


Figure 3

Timing diagram.

High-level functional description

The oscillator generation and distribution consists of a 16-MHz voltage-controlled crystal oscillator, a PLL, an IDCS, a control chip, and a power supply. These components are located on an oscillator card. The 16-MHz oscillator is used to generate a stable frequency. This signal is fed to a PLL, where it is multiplied to the final clock distribution frequency. From the PLL, one output signal is sent to the CLK0 input of the IDCS on the same card, and a copy of this signal is connected to the CLK1 input of the IDCS on the other card.

The IDCS monitors up to four differential oscillator inputs continuously (only three are actually used in the design). In the case of a failing signal, the IDCS automatically switches over to the next input in a round-robin fashion. The selected input, which initially is defined by the I^2C (Inter-Integrated Circuit bus protocol), is repowered and sent out as up to eight differential outputs. In addition to this basic function, the status of the input signals can be observed and reported to a control system.

The oscillator cards and the corresponding components are set up and controlled by the control chips and the associated firmware running on the FSPs. The setup and control functions are the following:

- Power-on and power-off of oscillator card components.
- Check power status (over current, under voltage).

Table 1 Oscillator setup conditions.

	Primary card	*Secondary card*	*Single card*	*Disabled card*
OSC_SELECT0	1	0	1	0
OSC_SELECT1	0	1	1	0
IDCS input 0	Primary oscillator in	Secondary oscillator in	Oscillator in	Oscillator in
IDCS input 1	Secondary oscillator in	Disabled	Disabled	Disabled
IDCS input 2	Disabled	Primary oscillator in	Disabled	Disabled

- Set up PLL (frequency).
- Check PLL status (PLL locked).
- Set up IDCS (frequency range, IDCS mode, primary and secondary input).
- Check IDCS status (inputs running or failing, which input used, lock indicator).
- Set up OSC_SELECT signals to CLK.

Each of the two oscillator cards (Figure 2) can be set up as a primary card, secondary card, single card, or disabled card. Setup is performed using the OSC_SELECT signals (7) and the setup of the IDCS (**Table 1**). Assuming that oscillator card 0 (OSC0) is set up as the primary card and OSC1 is set up as the secondary, signal (1) is selected from the PLL0 by IDCS0 and is driven as the primary clock signal (2) to all of the CLKs. A second signal (5) from OSC1 is connected to input 1 of IDCS0 (defined as the secondary clock input, based on Table 1). This signal (5) has the same frequency as (1). However, the two signals do not have a fixed phase relation and therefore are not synchronous. In case of a failure of the oscillator or the PLL0 on the OSC0 card, the IDCS0 switches automatically from input 0 to input 1 while continuously providing output signals to the CLKs. However, to address certain types of failures—such as the power supply (PS0), IDCS0, or wiring between the IDCS0 and the CLKs—the switchover function (CLK_OSF) was implemented on the CLK.

CLK switchover structure

The REC 0 and REC 1 functions on the CLK receive the oscillator signal from oscillator card 0 and card 1, respectively (**Figure 4**). Directly attached to each receiver is a wire test block. The wire test function continuously observes the positive and the negative leg of the differential oscillator signal. If one leg of the differential signal is broken, the wire test sends an error signal to the CNTL block. If a secondary oscillator is available, a switchover is triggered. This function is required because a differential receiver may show a good signal at its output—even with only one input working correctly and the second input broken. In this state, the system would be exposed to higher clock jitter, which might cause system failure. Otherwise, the system would be exposed to failure due to jitter induced by a failure of the differential pair.

The output of the differential receiver is also connected to a programmable delay line. This delay line has 128 steps, typically with 20 ps per step. The delay setting associated with the delay line in the primary path is always set to 0; hence, there is virtually no delay from the output of the primary oscillator to the receiver of the input select block. The delay setting associated with the secondary path is set such that the input select block always receives the secondary oscillator signal later than the primary oscillator signal. This is accomplished by the phase-compare function that continuously measures the two signals to the input select and adjusts the delay line accordingly. The amount of delay difference between the primary and the secondary signals to the input select are adjusted to maintain the difference as small as possible. Otherwise, during a switchover, the signal appears as a phase jump, which is not acceptable. The typical phase difference achieved in System z9 is 50 ps.

The input select consists of an edge-triggered flip-flop with two inputs. A rising edge of either input sets the flip-flop and sends a 1 to the REPOWER block; a falling edge resets this flip-flop and sends a 0. The input signal received first is always sent to the output. The input select sends the proper oscillator signal as soon as one of the inputs works properly, even if the other input is stuck at 1 or 0.

The input select block provides other significant functions. It provides a lock mechanism which ensures that a switching back cannot occur after a switchover has taken place. This prevents bouncing back and forth due to intermittent types of failures (e.g., a poor contact or solder joint). In addition, it contains a filtering function that prevents erroneous switchover due to minimal noise or short-term jitter.

The REPOWER function serves to redrive the output signal of the input select block to the processor chips. The signal paths are also implemented with differential point-

to-point connections to minimize any potential signal noise.

The delay decrement CNTL function guarantees low skew between processor chips after a switchover occurs. This is achieved by setting the delay line greater than 0 in the active oscillator path. Once the defective oscillator card is concurrently replaced, the delay decrement CNTL function incrementally resets the delay line back to 0. This ensures that the oscillator card replacement is prepared to resume the primary path if necessary and also reduces jitter due to the additional circuits within the delay line path.

The CNTL block contains all combinatorial logic required to set up the oscillator switch function correctly. The CNTL function decodes incoming OSC_SELECT signals and sends control and status signals to the other functions. In addition, it gathers and interprets the status of the wire test, phase compare, and input select blocks, and generates the two OSC_STATUS output signals that are sent back to each oscillator card. The OSC_STATUS provides information on the CLK condition, oscillator operation, and secondary oscillator, and indicates whether the CLK has switched to the secondary card. The oscillator switch status register also maintains important status information on the oscillator operation, the wire test, the active oscillator, and the condition of the secondary oscillator.

Delay decrement CNTL

Each CLK in the system receives the same logical representation of the OSC_SELECT signals. The CLKs select between the two redundant incoming differential oscillator signals from OSC0 or OSC1. A change of the OSC_SELECT signal indicates a switchover from the primary oscillator card to the secondary oscillator card. An OSC_SELECT change is asserted to all CLKs in the same cycle, ensuring a dynamic and synchronous switchover for all CLKs. When a CLK itself is the root cause of the switchover, the erroneous CLK triggers an immediate local switchover. However, the assertion of changed OSC_SELECT signals still occurs at the same cycle to all CLKs in the system within a short time period after the occurrence of the error.

The delay decrement CNTL logic reduces the value of the delay line (which is in the active oscillator path) from the given value after the switchover occurred, down to 0. Each delay line contains 128 delay circuits, with a typical delay value of 20 ps per circuit. Because of CLK process variations, the delay value per circuit varies from chip to chip. After a switchover, the absolute delay value of each DELAY line D_i on all CLKs in the system is the same. However, the number of delay circuits x_i varies from chip to chip because of CLK circuit delay $d_{\text{cir},i}$ variations:

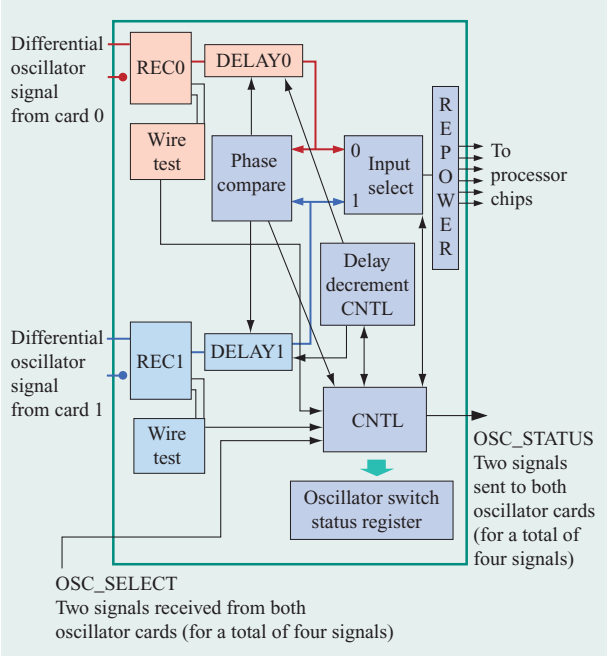


Figure 4

Clock chip oscillator signal switchover function.

$D_i = x_i \cdot d_{\text{cir},i}$ (Value of delay line on CLKi).
$D_0 = x_0 \cdot d_{\text{cir},0}$ (Value of delay line on CLK0).
$D_1 = x_1 \cdot d_{\text{cir},1}$ (Value of delay line on CLK1).
$D_0 = D_1$ (Value of delay line is the same across all CLKs).

Therefore, in order for each CLK to achieve the same delay line value, the number of delay circuits x_i must be individually selected for each CLK.

The removal of delay circuits is performed in equidistant steps over time. The equidistant step rate R_i for each individual CLK is calculated prior to doing the removal. The calculation is performed such that the overall time of removal T_R is the same for every CLK in the system. Because the number of circuits that have to be removed can be different on every CLK, the step rate varies from CLK to CLK. This approach guarantees minimum skew between chips during the time of removal. The overall time of circuit removal expressed in CLK cycles is defined as

$$T_R = R_i \cdot x_i + r_i, \quad (1)$$

where r_i is an integer remainder on CLK i and $0 \leq r_i < x_i$.

Equation (1) represents the correlation of two positive integers T_R and x_i. Given a dividend T_R and a divisor x_i, the integer division consists in obtaining an integer quotient R_i and an integer remainder r_i. The hardware

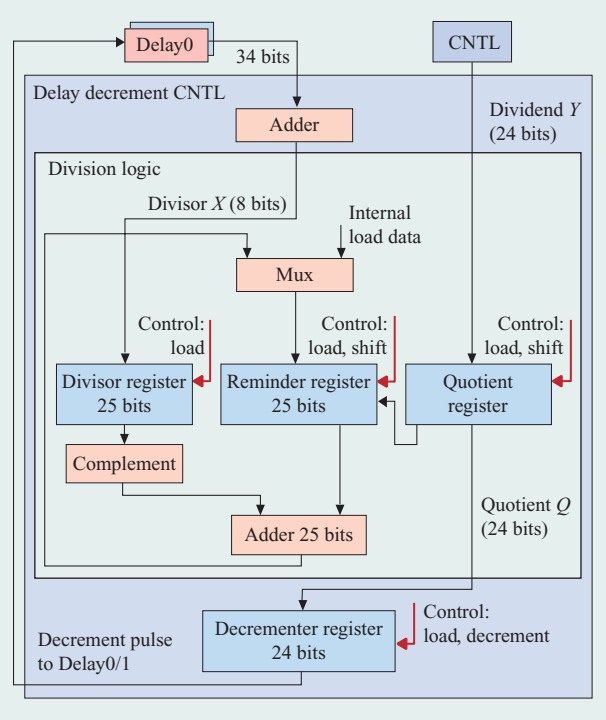


Figure 5

Data flow of the delay decrement CNTL logic.

implementation of a nonrestoring division algorithm can be found in [6].

The data flow of the delay decrement CNTL logic (**Figure 5**) contains a non-restoring division algorithm for a dividend Y of 24 bits and a divisor X of 8 bits. The first adder logic converts a 34-bit delay representation provided by the delay line into an 8-bit divisor. The 8-bit divisor represents the 128 different states of the delay line. The CNTL logic provides a configurable 24-bit dividend value. The 24-bit dividend value could be configured between $2^{24} \leq \text{dividend} \leq 2^8$. This gives firmware a wide range for the overall time of circuit removal after a switchover—between 16,777,216 and 512 CLK cycles. As mentioned earlier, this value has to be the same on every CLK in the system. The calculation of the quotient register is accomplished using a multiple of 24 CLK cycles. The quotient register is then loaded into a decrement register, which asserts a single CLK cycle decrement pulse when 0 is reached. The decrementer register is reloaded $(x_i - 1)$ times until all delay circuits of the delay line are removed.

The decrement pulse, the value of the delay lines, and the skew between two CLKs with and without the quotient calculation design approach are shown in **Figure 6**. With the division logic, CLK1_W_DIV receives, over a given time value, ten more decrement pulses than CLK0_W_DIV [Figure 6(a)]. The values of the delay lines on the two chips track tightly together [Figure 6(b)]. Because the removal rate is slightly shorter (CLK1), minimal skew between the two CLKs is achieved. The skew between the CLKs (CLK0/1_W_DIV) is limited to 20 ps over the entire removal time [Figure 6(c)]. However, without the quotient calculation method, both CLK0_W/O_DIV and CLK1_W/O_DIV use the same removal rate, and values of the delay line vary significantly. The skew value between CLK0_W/O_DIV and CLK1_W/O_DIV increases to 180 ps, resulting in a malfunction of the system.

Initializing and configuring oscillator cards

The initialization and configuration of oscillator cards, their drift error detection, and their recovery are under firmware control. Initialization is done in three steps: standby power on, logic power on, and power-on self-test (POST). Firmware is enabled for error interrupts from the oscillator cards after initialization and configuration. Drift error detection is also started.

The z9 can have only one active clock source. The firmware reads the status of the oscillator cards and selects the active clock source by setting the operating state (mode) for each oscillator card. The oscillator cards can operate in four different modes: primary card, secondary card, only available card, or disabled. Only certain combinations of oscillator card modes are allowed. Valid mode combinations are shown in **Table 2**.

Initialization functions

Standby power-on

Standby power, when switched on, provides power to the system control structure. (Logic power is separate.) Firmware senses which oscillator cards are plugged in and determines the hardware levels of these cards as soon as the controllers have booted. If two cards are plugged in, the compatibility is checked and mismatches reported. The available cards are then set to the initial mode.

Logic power-on

Logic power-on switches on all voltage levels needed by the system. Power to the oscillator cards is switched on, and the cards go through a temporary setup before the logic power for the CLKs is switched on. This is done to ensure that the CLKs always receive a valid clock signal from the oscillators. Oscillator card 0 is configured as the primary card by default, and card 1 is configured as the secondary card. Only one card is configured under the following conditions: 1) the other card is not plugged, 2) card 1 is not usable due to incompatible hardware

Table 2 Valid mode combinations for oscillator cards.

Oscillator card 0	*Oscillator card 1*
Primary card	Secondary card
Secondary card	Primary card
Single card	Disabled card
Disabled card	Single card
Initial mode	Initial mode (only in system power-off state)

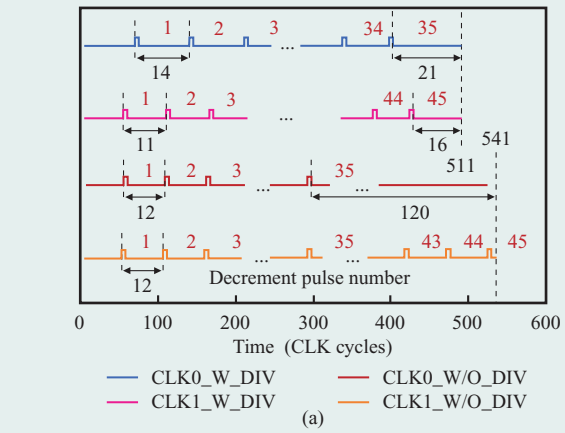
levels, or 3) the other card has been marked defective in the past.

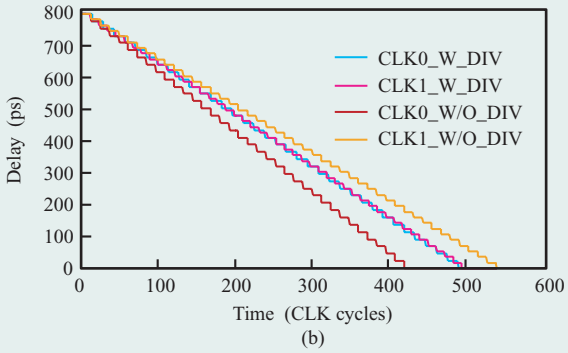
Because of hardware dependencies among chips on one card and between the two oscillator cards in the system, the cards are set up in stages such that all signals are available at the right time. The power to card 0 is switched on, then the power to card 1. All PLLs are set up on card 0, then on card 1. The IDCS chips are set up and can use the signals of the previously set-up PLLs. At the end of the power-on sequence, the IDCS chip on the secondary card is reset to allow it to use an output of the IDCS on the primary card. The firmware now enables the monitoring of all status signals.

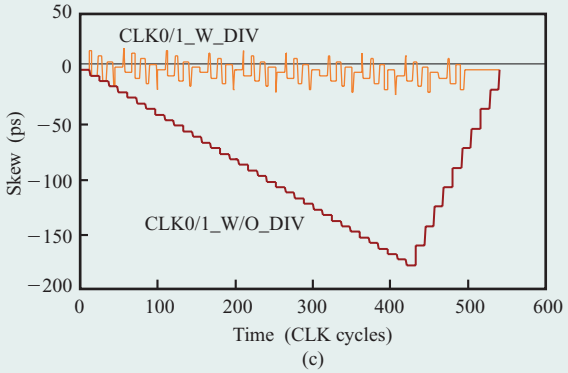
The logic that guarantees that all CLKs get their signal from the same oscillator card is disabled during this phase in order to ensure that, despite errors, all CLKs receive a signal from at least one oscillator card. The OSC_SELECT signals are given a fixed setting according to the oscillator card mode. This allows a CLK to switch to the secondary oscillator card if it does not receive a signal from the primary card, and to do so without causing all other CLKs to switch to the secondary card. The fact that some CLKs possibly do not use the same signal does not matter at this point in time because the interfaces between the processing nodes are not yet in use. The final setting up of the oscillator states is done during the POST.

POST

The oscillator cards are tested just before the full-chip self-test step in the power-on sequence. This is done to ensure that all chips have valid clock signals. The chips on the oscillator card have no logical built-in self-test to verify them, but with the status information reported back by the oscillator card and feedback information from the clock switch logic of each book, it is possible to ensure that both oscillator cards work correctly and that the CEC logic receives valid clock signals. Every card is cycled through each mode to verify the clock switch logic. This also ensures that all feedback wires are functional. The correct cycle time is set up on the oscillator cards. All







T_R = 511 CLK cycles (Dividend = 0×0000FF)

CLK0:

D_0 = 800 ps	Value of the delay line on CLK0
$d_{cir,0}$ = 22,86 ps	Delay of delay circuit on CLK0
x_0 = 35	Number of delay circuits on CLK0
R_0 = 14	Removal step rate on CLK0
r_0 = 21	Remainder on CLK0

CLK1:

D_1 = 800 ps	Value of the delay line on CLK1
$d_{cir,1}$ = 17,78 ps	Delay of delay circuit on CLK1
x_1 = 45	Number of delay circuits on CLK1
R_1 = 11	Removal step rate on CLK1
r_1 = 16	Remainder on CLK1

Figure 6

Quotient calculation method comparisons.

chips on the multichip module are given a PLL reset to ensure that the system has valid clock signals and that all PLLs are locked. If a problem is detected during the oscillator card test, the FRU that was identified to be defective is disabled. The system posts an error log and calls the card for replacement.

The system is set up to ensure that the maximum amount of hardware is available for the customer. When an oscillator card is disabled, the clock switch propagation remains disabled. The system continues in a redundant mode. If both oscillator cards and the clock switch logic of all configured books have passed their tests, the clock switch propagation is enabled. This allows all clock state to be propagated to the other clocks, and the clock signals stay synchronized. The system runs with a redundant oscillator setup, and the support element is registered for oscillator events. Chip self-test is executed on all CEC and I/O chips with the dynamic oscillator switch feature enabled.

Errors during system operation

The firmware controlling the clock distribution runs on a pair of redundant controllers. This ensures that reliable control is available for the oscillator function. The firmware is always ready to read and evaluate the current status of the clock distribution. Interrupts from the IDCS chip direct the firmware to read, to evaluate, and, if necessary, to recover the clock distribution hardware.

Oscillator problems that occur on a running system are divided into two categories:

1. Problems on the signal source and every component on the input side of the IDCS.
2. Problems in the clock signal distribution from the IDCS to the CLKs.

Both described recoveries of oscillator signal failures are implemented entirely in hardware and do not require firmware interaction. Firmware actions are started after all CLKs have switched to the alternate oscillator card. The firmware is triggered when the oscillator card indicates a state change on a monitored signal. The firmware identifies and disables the defective oscillator card. It posts an error log, calls for an FRU replacement, and prepares the defective card for concurrent replacement. All oscillator card repairs are concurrent; the system continues to run on one oscillator card.

The replacement card is installed and powered on. The configuration is updated, and an oscillator card mode change is triggered. The new card up is now a valid backup card. The card is verified, and a new defective card would be disabled and removed during the repair procedure.

There is one special error scenario in which the oscillator hardware does not report any problem, but the card is defective. This is when the firmware detects a frequency drift. Frequency drift detection and recovery are discussed in the section on oscillator drift detection and recovery.

Table 3 summarizes the major oscillator failure and switchover scenarios, the hardware action taken to maintain the integrity and coherence of the clock signals, and the firmware actions to deconfigure the failing card and to prepare the concurrent repair of the isolated failed card.

Changing the configuration while running

A valid configuration is selected for the oscillator cards during initialization. A fault on one of the cards can dynamically change the modes of the cards, and the firmware ensures that the modes are kept consistent between the two oscillator cards. Every mode change of an oscillator card is performed in a sequence that does not affect system operation. A mode can only change to certain other modes; the firmware ensures that changes to the mode do not violate the hardware constraints.

Executing a mode change on a running system is difficult. Dozens of signals are distributed over the two oscillator cards that must be switched in the correct sequence. This sequence varies depending on the target modes and on the initial mode. Since any incorrect sequence step can cause a system outage, every sequence step must be implemented such that a sudden interruption after any operation is possible without any impact to the system. After becoming active, the redundant controller must be able to continue the interrupted sequence.

A target sequence is executed by one controller. It is saved as persistent data on both controllers even though one controller is not active. Every sequence is divided in as many sub-sequences as necessary. The execution of any sub-sequences is confirmed at a synch point. The redundant controller can take over if the active controller should fail. After the redundant controller takes control, it checks first to determine whether any sequence is incomplete. The controller restarts the incomplete sequence, beginning at the last confirmed synch point. The sub-sequence between two synch points is defined such that every operation between two synch points can be redone without any impact.

Oscillator drift detection and recovery

One type of oscillator problem is a sudden failure. Another type is described as a creeping failure—for example, deterioration or aging of the oscillator circuitry. Hardware detects only sudden failures, while firmware detects long-term problems using special hardware.

Table 3 Oscillator failures and switchover scenarios (the numbers in parentheses refer to signal numbers in Figure 2).

Failure description (OSC0: Primary, OSC1: Secondary)			*Oscillator card mode*	
Component failure	*Hardware action*	*Firmware action*	*Before replacement*	*After replacement*
OSC_M0 or PLL0	IDCS0 switches from input signal (1) to (5). Sends interrupt to CTL0.	Changes OSC_SEL signal (7), forcing CLKs to switch from input signal (2) to (4) and IDC1 to switch from input signal (3) to (6). Sets the modes for the OSC cards for part replacement.	OSC0: Disabled OSC1: Single card	OSC0: Secondary OSC1: Primary
PS0: Under voltage	Changes OSC_SEL signal (7), forcing CLKs to switch from input signal (2) to (4) and IDC1 to switch from input signal (3) to (6). Generates interrupt for firmware.	Sets the modes for the OSC cards for part replacement.		
IDCS0: Loss of lock	Deactivates output_ enable signal. Generates interrupt for firmware.			
IDCS0: Output to a single CLK	CLK_OSF switches from input (2) to (4). CLK_OSF informs CTL0 that switchover occurred via OSC_STATUS signal. CTL0 changes OSC_SEL signal (7). Generates interrupt for firmware.			
IDCS0/CLK: Wire break	Same as above.			

Hardware resources used for oscillator drift detection

Long-term firmware error detection is supported by hardware on the CLK. This hardware, a precision oscillator comparator, compares the oscillator signal from each card. The time-of-day (TOD) function is also used as an input. It is stepped by an external time reference when available, providing a third independent time source. Pulses from each card are counted in separate counters by the precision oscillator comparator. When either count is exhausted, both counters and the TOD value are captured. The firmware is interrupted, and it determines whether the long-term drift is tolerable. The precision oscillator comparator can be configured to interrupt for different conditions.

Firmware used for oscillator drift detection

The firmware uses different interrupt conditions in the precision oscillator comparator to detect drift under varying conditions. An interrupt can be posted only when a specified frequency deviation is exceeded, or measurement data can be collected and used to build a history for a card. The z9 is configured to actively collect measurement data which is used to monitor oscillator drift over time. The deviation of the oscillator card pair is determined from the measurement data. However, determining which card is correct requires a third time reference, which is provided by the TOD function. The TOD function may be stepped from a Sysplex Timer [3] as an external time reference. Comparing the drift of the oscillators to the externally stepped TOD

function makes it possible to determine which oscillator card is drifting.

Firmware escalation

The firmware monitoring routine uses three thresholds for three different situations. The first threshold indicates that the oscillator card frequencies are diverging. It is configured when the TOD function is stepped to an external time reference. The threshold value is selected such that the defect oscillator card can be detected and identified. When a drifting oscillator card reaches this threshold, a system reference code is posted, and the system calls for repair of the defective card. The second threshold is used when a third time reference is not available. This threshold indicates that the oscillator card frequencies are diverging, but the system cannot determine which card is drifting. When this threshold is reached, a system reference code is posted, and the system calls for repair. The repair action is to change the inactive oscillator card. A third and smaller threshold is applied after the repair action to determine whether the repair action has fixed the problem. It is needed to prevent simple card tolerances from exceeding the threshold following the repair. Should this third threshold value be reached, the repair is repeated, with the active card being changed. This trial-and-error scenario cannot be avoided without an external time reference.

Test approach, methodology, results

Because of the complex interactions of numerous involved firmware components and the many possible parameter variations, we determined that the only feasible way to provide good test coverage was to perform a black box test of the whole system for a limited, well-chosen number of error scenarios in a representative system configuration. The error injection was done with an instrumented oscillator card that allowed the injection of signal errors (tie signal to 0/1 or break signal) on 20 signals covering the areas of the various oscillator signals, control, and feedback lines among the nodes and the oscillator card, I^2C (Inter-Integrated Circuit serial bus interface) control lines between the node controller and the oscillator card chips, and power surveillance.

The signal error injection was realized by miniature relays, which enabled remote control by means of special hardware connected to the system controller printer port and controlled by software running on the system controller. Each signal inject was performed with the card being the primary or secondary oscillator card during system activation and on a fully operational system. A few additional test cases with only the instrumented oscillator card installed (no backup oscillator available) were also run to verify the firmware capability to handle nonrecoverable oscillator hardware errors.

The total number of test cases was about 100. A test suite was developed to run the test cases almost fully automated. The plug position of the instrumented cards had to be exchanged manually. The test suite was executed by the system controller, automating the use of the error-injection component, controlling and supervising system status to ensure that the correct hardware was identified as being defective, and monitoring the system error log to verify that the correct hardware error message and call home to the IBM Remote Technical Assistance Information Network system was performed. For each test case, a full error-data collection, including gathering of traces, dump, and log data, was performed.

The test cases were executed in two shifts: the first shift with the instrumented card used as the primary oscillator, the second shift with the instrumented card used as the secondary. Each shift took nine hours to complete, resulting in approximately 3,000 tests executed over a six-week test period.

Development of the test suite and the test-case driver and result collection, together with debugging the test cases, was a cumbersome and time-consuming process. The effort was well spent and created a highly automated regression test package, which is used and enhanced to verify that this critical firmware component works flawlessly after fixes are introduced.

Conclusion

The increasing demand for continuous availability necessitates that each function be designed such that virtually any failure within its domain can be tolerated. Furthermore, the repair of the failing unit must be concurrent with system operation. The dynamic oscillator switchover function introduced on the z9 enriches the availability functions suite over previous generations.

The IBM System z9 fully redundant clock generation and distribution design with a fully dynamic switchover provides a mechanism that allows instantaneous switching from one input source to a backup without introducing any damaging disturbance to the running system. The oscillator signal generation, control, switch logic, and control firmware are redundant. The design is optimized to handle a variety of failure scenarios by dynamically switching to a backup oscillator signal. The replacement of the failing unit is performed concurrently with system operation. The design tolerance and concurrent repair capability supports a high level of availability.

Acknowledgments

We gratefully acknowledge the contributions to this design by our colleagues at Freescale, Inc., and IBM, especially Andreas Günther for good cooperation with

the IDCS design; Scott Swaney, Andreas Wagner, and Tim McNamara for design discussion and reviews; Hans Kittl for building the error-injection hardware; and the Global RAS team for their good cooperation on requirements specification.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

References

1. M. Mueller, L. C. Alves, W. Fischer, M. L. Fair, and I. Modi, "RAS Strategy for IBM S/390* G5 and G6," *IBM J. Res. & Dev.* **43**, No. 5/6, 875–888 (1999).
2. M. L. Fair, C. R. Conklin, S. B. Swaney, P. J. Meaney, W. J. Clarke, L. C. Alves, I. N. Modi, F. Freier, W. Fischer, and N. E. Weber, "Reliability, Availability, and Serviceability (RAS) of the IBM eServer* z990," *IBM J. Res. & Dev.* **48**, No. 3/4, 519–534 (2004).
3. N. R. Dhondy, R. J. Schmalz, R. M. Smith, Sr., J. Thomas, and P. Yeh, "Coordination of Time-of-Day Clocks Among Multiple Systems," *IBM J. Res. & Dev.* **36**, No. 4, 655–665 (1992).
4. F. Baitinger, H. Elfering, G. Kreissig, D. Metz, J. Saalmueller, and F. Scholz, "System Control Structure of the IBM eServer z900," *IBM J. Res. & Dev.* **46**, No. 4/5, 523–535 (2002).
5. Freescale Semiconductor, *Quad Input Redundant IDCS Clock Generator,* Datasheet MPC9894, July 2005; see *http://pdf1.alldatasheet.com/datasheet-pdf/view/103208/ETC/MPC9894.html.*
6. M. D. Ercegovac and T. Lang, *Digital Systems and Hardware/Firmware Algorithms,* John Wiley & Sons, Inc., New York, 1985; ISBN 0-471-88393-X.

Received March 21, 2006; accepted for publication June 16, 2006; Internet publication January 19, 2007

Michael J. Mueller *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (mulm@de.ibm.com).* Mr. Mueller is a Senior Technical Staff Member responsible for the platform architecture and design of the RAS functions for IBM iSeries*, pSeries*, and zSeries eServers. He received his Dipl. Ing. degree in electrical engineering from the University of Stuttgart, Germany. Mr. Mueller has held various positions in S/390 firmware development and system design.

Ulrich Weiss *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (uweiss@de.ibm.com).* Mr. Weiss is responsible for the oscillator generation and distribution function on IBM zSeries and pSeries eServers. He has worked on the clocking of high-performance chips, high-performance system oscillator generation, and distribution and chip-pervasive functions.

Tobias Webel *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (webel@de.ibm.com).* Mr. Webel is a Senior Development Engineer responsible for processor hardware development of RAS functions on IBM zSeries and pSeries eServers. He received his Dipl. Ing. degree in electrical engineering from the University of Stuttgart, Germany. Mr. Tobias is one of the leading hardware design engineers responsible for the transition of the system run control structure on zSeries 900 to a multibook system, including RAS aspects.

Luiz C. Alves *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (alves@us.ibm.com).* Mr. Alves is a Senior Technical Staff Member working in z9 system assurance. He received his B.S. degree from New York University and his M.S. degree from Polytechnic Institute of New York, both in electrical engineering. He was previously the RAS manager for the 9021 processor families and is currently responsible for defining RAS requirements for future products.

William J. Clarke *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (wjclarke@us.ibm.com).* Mr. Clarke is an Advisory Engineer working in z9 system assurance. He received his B.S. degree in electrical engineering from Rutgers University. He previously worked on the storage subsystem for the IBM 3090* processor, logic design, recovery systems test, and zSeries product engineering. Mr. Clarke currently defines RAS requirements for future products.

Markus Strasser *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (markus.strasser@de.ibm.com).* Dr. Strasser works in firmware development. Prior to joining IBM, he was with the distributed systems group of Stuttgart University, Germany. He received an M.A. degree and a Ph.D. degree in computer science, both from the University of Stuttgart. Dr. Strasser's current primary responsibility is the development of zSeries firmware.

Eberhard Engler *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (eengler@de.ibm.com).* Mr. Engler received an M.S. degree in physics (Dipl. Phys.) from the University of

Tuebingen, Germany. He worked initially on a handheld data acquisition system, later joining S/390 firmware development. Mr. Engler works primarily in the areas of IBM Sysplex Timer and TOD clock synchronization, service word communication, and on demand functionality.

Giovanni Cautillo *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (cautillo@de.ibm.com)*. Mr. Cautillo received his Dipl. Ing. (FH) degree in information technology from the Esslingen University of Applied Sciences, Germany. He joined the IBM Development Laboratories at Boeblingen in 2003 after working for several years as an application developer. Mr. Cautillo's current main responsibility is the development of firmware for zSeries servers.

Harm Osterndorf *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (osterndo@de.ibm.com)*. Mr. Osterndorf received his Dipl. Ing. degree in computer science from the Berufsakademie Stuttgart, Germany. He joined IBM zSeries firmware development in 1993.

Joerg Schulze *IBM Systems and Technology Group, IBM Deutschland Entwicklung GmBH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (jschulze@de.ibm.com)*. Mr. Schulze studied physics and computer science at the Technical University of Braunschweig, Germany, and received an M.A. degree in physics. In 1999 he joined IBM, where he works primarily in firmware development.