



Multiple Reference Clock Generator

Digital IP for Clock Synthesis

✧ August 2007

IPextreme, Inc.



This paper explains the concept behind the Multiple Reference Clock Generator (MRCG) and describes an example on-chip clock synthesis implementation using the MRCG digital IP.

HIGHLIGHTS

- ▶ Introduction
- ▶ Technical background
- ▶ Implementation example

INTRODUCTION

This document describes the direct digital clock synthesis technology developed by Motorola Labs Transceiver Research known as the Multiple Reference Clock Generator (MRCG). The technology is available to customers as IP distributed through IPextreme, Inc. The MRCG IP integrated with a DLL and a single PLL provides a low power, very flexible implementation of a clock synthesizer capable of generating multiple clocks in an SoC design.

The MRCG technology was developed as a flexible clock reference signal for integrated digital processing applications with a direct digital synthesis feature set. Some additional objectives were:

- Overcome the limited flexibility of PLL clocks such as frequency change locked loop transit
- Limited 20% voltage controlled oscillator (VCO) tuning range
- Generation of phase offset and other coherent signal pairs

Some of the MRCG signal source features are:

- Cycle-to-cycle frequency, time, or phase shifting
- 1 GHz to 2 MHz operating frequency range with one VCO reference
- Reconfigurable complex signal function generation
- Generation of multiple independent coherent signal set

TECHNICAL BACKGROUND

The MRCG uses a unique, proprietary, digital approach to frequency synthesis that is capable of direct digital synthesizer (DDS) level of performance. Instead of a digital-to-analog converter, a digital-to-time or phase converter is used to produce square wave or two-state signals. The result is edge or transition-defined direct digital signal synthesis at a much lower power drain. Two-state or binary signals are compatible with digital processing clock signals and frequency converter switching signals applied to ring mixers used in transceivers.

The MRCG synthesizer operates by concatenating a series of pulses timed to coincide with zero crossings at a desired output frequency (F_{out}). Every transition edge of the output signal is represented within a window of time defined by the implementation design. The MRCG architecture generates the timed pulses by propagating a high frequency reference signal (F_{ref}), through a series connected N-stage delay lock loop (DLL). Each stage of the DLL

provides a replica of the square wave pulse input reference signal with a delay offset equal to

$$T_d = ((1/F_{ref})/N) \text{ seconds}$$

relative to the preceding stage. The delay offset assumes the DLL is locked with stage $n = 0$ and $n = N$ with a delay offset equal to the input frequency signal period

$$T_{ref} = 1/F_{ref}$$

These $N+1$ time-delayed input reference square waves are the discrete digital-to-time or phase output signals that are the central analog function with the MRCG direct digital synthesizer. A digital algorithm and tap select network selects the proper sequence of delayed F_{ref} pulses to build the desired output signal transitions. Figure 1 shows a highly simplified functional block diagram of the MRCG synthesizer.

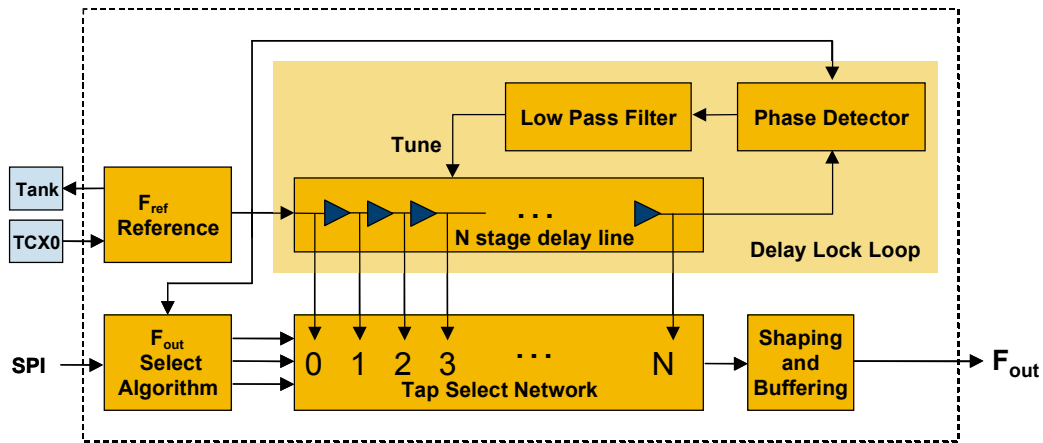


Figure 1: MRCG synthesizer architecture

The digital selection algorithm is a process, clocked at rate T_o , to determine which delay line square wave pulse to tap and when to activate the selection network. A digital accumulation function or equivalent¹ is used to process the ratio

$$C = F_{out} / F_{ref} = (1/T_{out}) / (1/T_{ref}) = T_{ref} / T_{out} \leq 1, \text{ for } F_{out} \leq F_{ref}$$

¹ One alternative accumulator function is a counter accumulator function.

The accumulator overflow signal indicates when to process a tap selection, and the overflow accumulator value (C_m) is used to determine which tap to select using the function

$$n = N - C_m * N = N (1 - C_m)$$

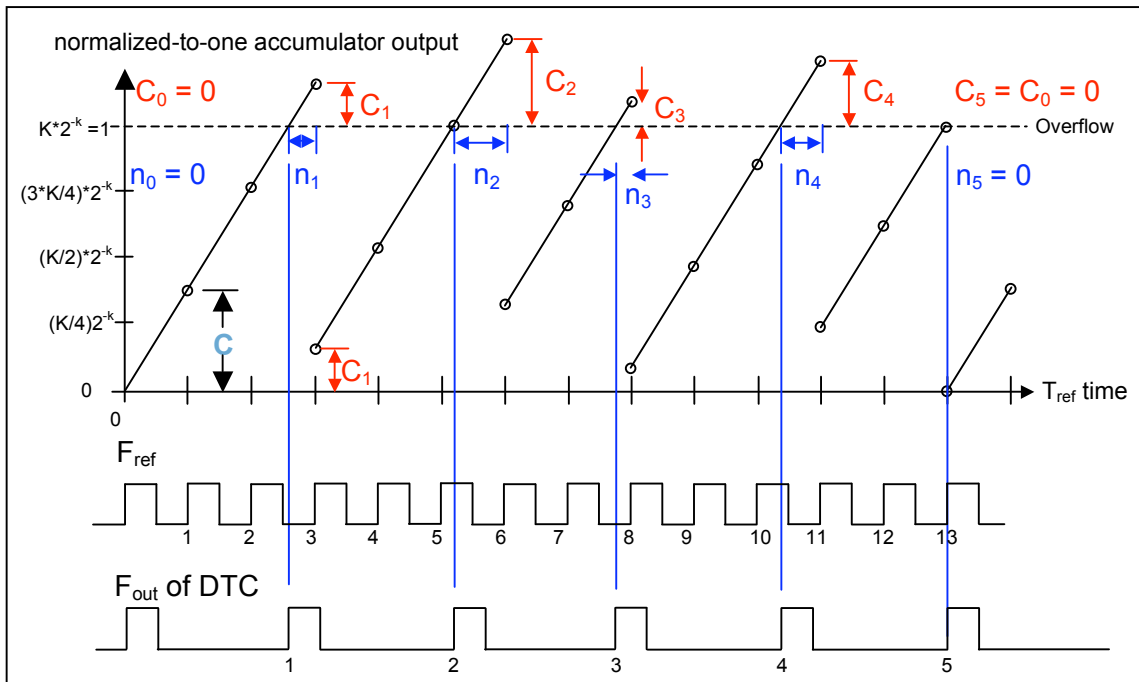


Figure 2: Accumulator tap selection algorithm plot

A simple example is shown in Figure 2, to demonstrate the tap selection algorithm for $K = 2^k$

$$C = F_{out} / F_{ref} = 5 / 13$$

$$C_1 = 3 * C - 1 = 15 / 13 - 1 = 2 / 13$$

$$n_1 = N * (1 - C_1) = N * 11 / 13 = 32 * 11 / 13 = 352 / 13 = 27 + 1/13, \text{ quantization offset of } T_d/13$$

$$C_2 = 6 * C - 2 = 30 / 13 - 2 = 4 / 13$$

$$n_2 = 32 * 9 / 13 = 288 / 13 = 22 + 2 / 13, \text{ quantization offset of } T_d 2 / 13$$

$$C_3 = 8 * C - 3 = 1 / 13$$

$$n_3 = 32 * 12 / 13 = 384 / 13 = 29 + 7 / 13, \text{ quantization offset of } - T_d 6 / 13$$

$$C_4 = 11 * C - 4 = 3 / 13$$

$$n_4 = 32 * 10 / 13 = 24 + 8 / 13, \text{ quantization offset of } - T_d 4 / 13$$

$$C_5 = 13 * C - 5 = 0 = C_0$$

$n_5 = 0 = n_0$, quantization offset of zero

The digital-to-time converter output F_{out} shown in Figure 2, has a pulse width equal to that of F_{ref} equal to $T_{ref} / 2$. This does not provide F_{out} with 50% duty cycle, but is representative of the F_{out} rising edge. A second tap selection algorithm is needed with the initial accumulator value of 0.5 instead of 0 to provide a falling edge F_{out} signal. These two tap selection networks share a common DLL to provide a sequence of signals associated with the rising edge and the falling edge of the output signal. The shaping and buffering function is designed to process rise and fall tap selection signals into the composite output signal with a common pulse shape across each pulse of the output signal F_{out} .

The number of bits (k) used to build the tap selection algorithm determines the maximum output frequency step size (F_{step}).

$$F_{stepmax} = F_{ref} (1 - ((2^k - 1) / 2^k)) = \text{maximum output frequency step size}$$

F_{out} Time Domain Parameters

A time domain representation of the output signal is helpful, with the understanding of the design implementation important to the output signal parameters. The MRCG direct digital signal synthesis process produces an output signal with every transition represented for the output signal F_{out} . A plot of F_{out} with a format similar to that of an eye diagram of a receiver decoded signal is shown in Figure 3. The pulse width and cycle-to-cycle jitter are shown relative to the rise edge of each period. A given edge resolution is within one duration of T_d , the time delay quantization value of the digital-to-time converter. If the precise real time rising or falling edge of F_{out} is approximately in the middle of a quantization value, the real time output rising or falling edge will be offset by $\pm T_d/2$ value. If the next respective rising or falling edge is also approximately in the middle of a quantization value, the real time output edge can be offset with a maximum real time of $\pm T_d/2$ or a total relative value of T_d . If this happens in two consecutive periods with opposite offset time values, the relative cycle-to-cycle jitter is $2T_d$. The pulse width is designed to have a duty cycle of 50% to represent a square wave switching signal. The digital-to-time converter quantization would provide a duty cycle tolerance value equal to:

$$F_{duty\ cycle} = T_{pulse} / T_{out} = ((1 / (2 * F_{out})) \pm T_d) / ((1 / F_{out}) \pm T_d)$$

The impact of digital-to-time converter quantization is most significant at the highest F_{out} frequency or $F_{out} = F_{ref}$, where the quantization value T_d is the largest percentage of the output signal period ($T_{out} = 1 / F_{out}$).

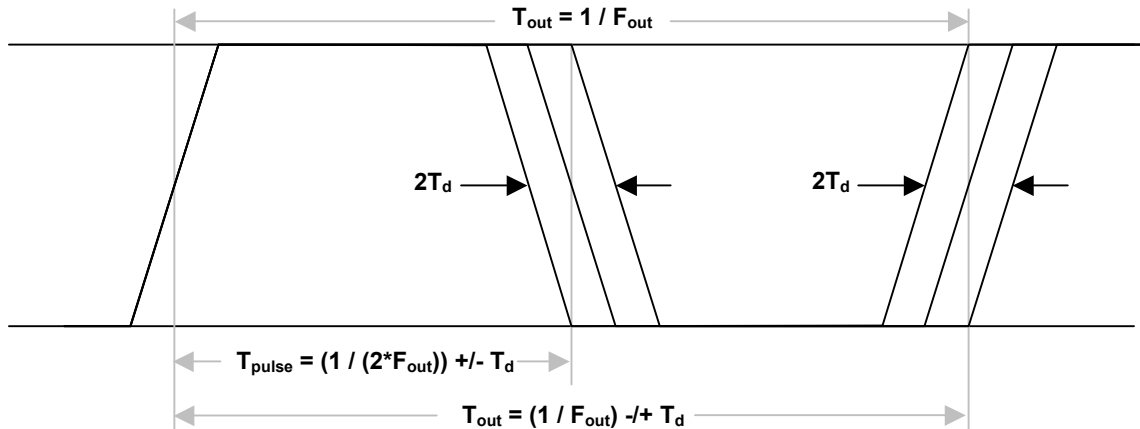


Figure 3: Output signal pulse width and cycle-to-cycle jitter as a function of edge time resolution or digital-to-time quantization

These results of duty cycle and cycle-to-cycle jitter are worst case values for a very limited set of specific output frequencies possible with the MRCG technology. For an output frequency with a period equal to an even multiple of T_d :

$$T_{out} = M \cdot (T_{ref} / N) \quad \text{for } M = N+2, N+4, N+6, N+8, \dots$$

the duty cycle is 50% and the cycle-to-cycle jitter is zero for every output pulse. For odd values of M , the output signal cycle-to-cycle jitter is still zero. However, the pulse width has a value:

$$T_{pulse} = T_{out} / 2 = M \cdot (T_{ref} / N) / 2$$

And for M/N with an odd integer value, the pulse width real time edge is half of the digital-to-time converter quantization value T_d . Worst case cycle-to-cycle jitter values occur with the following T_{out} relationship:

$$T_{out} = M \cdot (T_{ref} / (2 \cdot N)) \quad \text{for } M = 2 \cdot N+1, 2 \cdot N+3, 2 \cdot N+5, \dots$$

All other cases of M and T_{out} are between zero and the worst case duty cycle and cycle-to-cycle jitter values.

The signal quality parameter of interest for a clock signal designed for digital processing is the transition edge resolution in real or relative time. Other

secondary parameters of interest are expected to be the time or phase offset resolution of additional signals generated at the same or functionally related clock rate. Additional T_{out_x} signals would share the DLL with a separate pair of tap selection networks driven by an additional tap selection algorithm block. The tap selection output is applied into a unique pulse shaping buffer with output signal T_{out_x} . Some offset signals such as differential can be implemented as a processing function of T_{out} .

Clock or RF carrier signal parameters of interest in analog processing, such as frequency translation, are the undesired frequency component levels relative to the desired signal amplitude. Carrier signals with 50% duty cycle and zero cycle-to-cycle jitter, as described earlier, would have a harmonic frequency domain spectrum defined by an ideal square wave clock signal. All other clock signals with quantization offset from the accurate rising or falling edge real time will have non-harmonic spectrum content defined by the time domain quantization offset values. The number of non-harmonic spectrum components decreases as the periodic period of the quantization offset decreases relative to T_{ref} . The next section is an introduction to frequency domain signal quality and analysis associated with T_{out} .

F_{out} Frequency Domain Analysis

As one might expect there is a quantization impact on the non-harmonic spurious performance level associated with the digital-to-time conversion process. This is similar to the quantization results of a digital-to-analog converter used in a signal generation process. The frequency offset and level of non-harmonic spurs is a predictable function based on the number of accumulation cycles before the process repeats and the digital-to-time resolution (T_d) or effective number of bits (N) and the ratio of F_{ref} to F_{out} . Assuming the worst case quantization offset, the relation for worst case spurs relative to the desired F_{out} signal level into the same impedance is:

$$\text{dBc} = 20 \text{ Log}((N-1) * (F_{ref} / F_{out})) = 29.8 \text{ dBc}, \quad \text{for } N = 32$$

Improving the edge time resolution of the DTC improves or decreases the non-harmonic spurious level. This becomes an analog design challenge as the resolution becomes smaller and smaller. For example, a 1-GHz F_{ref} with a DTC time resolution of:

$$T_d = T_{ref} / 1024 = 0.9765625\text{ps}, \quad N = 1024 \text{ or } 10 \text{ effective DTC bits}$$

This example would have a worst case non-harmonic spurious level of 60.2dBc for F_{out} close to F_{ref} . There will be a 6-dB spurious level improvement for factor of 2 increase in the ratio of F_{ref}/F_{out} .

Dithering of the tap selection process to produce a random non-periodic quantization offset output signal will reduce the spurious level in favor of a distributed or shaped noise floor. For the worst case $N=32$ example with uniform dithering processed across a 1-GHz bandwidth, the quantization offset noise floor would be approximately -120dBc below the desired F_{out} signal level:

$$\text{dBc (noise floor)} = 10 \text{ Log (BW)} + 20 \text{ Log } ((N-1) * (F_{ref}/F_{out})) = 120 \text{ dBc}$$

Noise shaping can be applied to the tap selection process to improve the quantization noise level at selective locations in the spectrum.

IMPLEMENTATION CONSIDERATIONS

All of the analysis up to this point has assumed ideal values associated with the DTC. The two top implementation issues resulting in DTC errors are the DLL offset and mismatching in the delay elements used to build the delay line network. DLL offset is misalignment in the time lock between the delay line wavelength taps; more specifically, the time offset between tap 0 and tap 32 from a period delay or one cycle of F_{ref} . The delay offset is applied equally to each of the delay elements modifying the DTC resolution:

$$T_d = ((1/F_{ref}) + (T_{off}))/N$$

where T_{off} is the time offset distributed across each of the delay elements. DLL offset is expected to be within +/- 10% or $(N * T_d)/10$. The impact of DLL offset is an increase in the quantization offset from each tap output (n):

$$T_{off_n} = n * (T_{off} / N), \text{ where } n \text{ is the tap value between } 0 \text{ and } 32$$

In a perfect integrated circuit, all of the delay elements are identical with the same delay (T_d). There is a mismatch across the chip area that is defined by the technology process used for the design implementation. This mismatch is a random statistical value for each delay element (T_{m_n}) with a common standard deviation value (T_m) across all of the delay elements. What results is an additional quantization offset that is a statistical accumulation along the series connected delay elements:

$$T_{\text{off}_n} = n \cdot (T_{\text{off}}/N) + ((N \cdot T_d - \sum_{x=1}^N T_{m_n}) / N) + \sum_{x=1}^n T_{m_n}$$

The relationship accounts for a DLL adjustment in the DTC (T_d) resolution to account for the accumulation of the mismatch value increase or decrease on the delay lock value $N \cdot T_d$. What results is the DTC value moving further away from the ideal $n \cdot T_d$ value in the middle of the series connected delay line. As the tap selection gets close to zero, the accuracy approaches the ideal accuracy. When there is no DLL offset error, T_{off} is equal to zero and the tap selection accuracy improves going from the middle of the series connected delay elements towards tap 32.

This is shown in Figure 4, as a worse case tap delay variation versus tap position along the series connected delay network with T_m equal to 1 ps. Also shown in Figure 4 is reduction in mismatch DTC delay variation by locking the delay control over a smaller number of series connected delay elements.

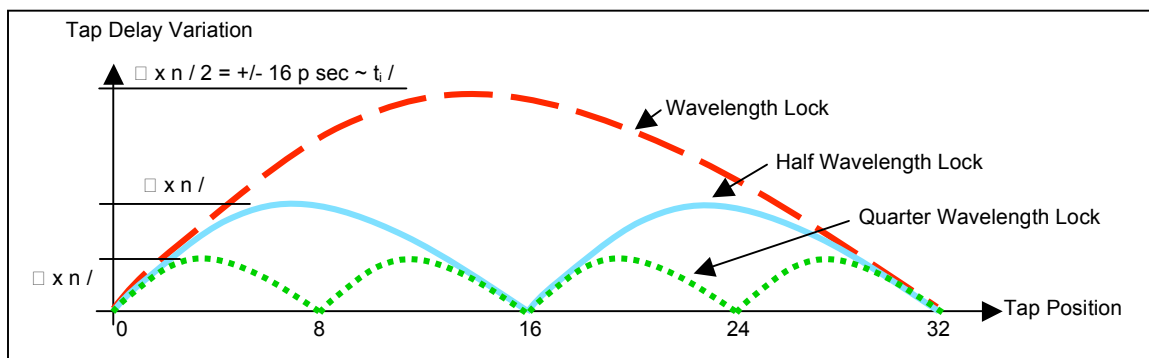


Figure 4: Worse case tap delay variation versus tap position along the series connected delay network as result of integrated device delay mismatch

Physical Implementation Example and Data

Figure 5 shows an example physical implementation of this technology in a 90-nm CMOS process. The block occupies approximately 0.054 mm². The picture shows one differential output, but multiple delay lines could be used to generate multiple differential outputs. The output is programmable, through a SPI within the digital block, from about 100 kHz to 500 MHz in 8-kHz steps. This frequency profile fit our particular application, but range and resolution can be adapted to meet other signal generation needs.

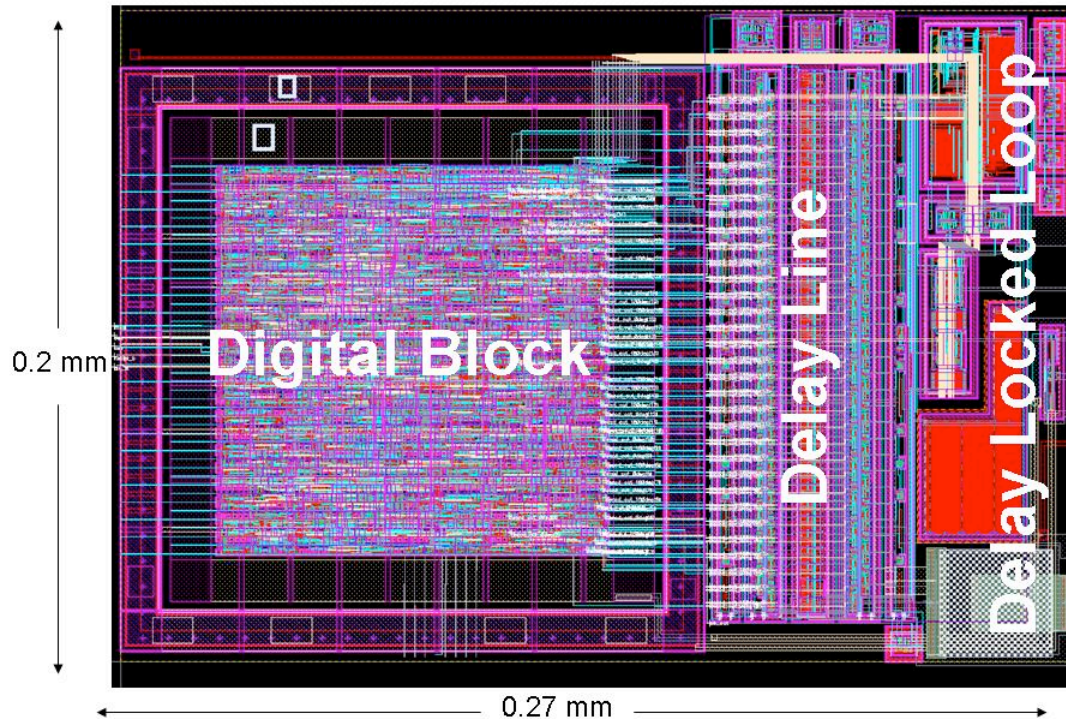


Figure 5: Recent implementation of MRCG technology in 90-nm CMOS

The power budget for this implementation is shown in the table below. The level of power consumption will depend on, among other things, the number of outputs and the maximum output frequency.

Circuit Block	Current Drain (mA)/mW	Area (mm ²)
Digital	2 / 2.4	0.029
Delay Line	1.2 / 1.44	0.013
Delay Locked Loop	1 / 1.2	0.012
Total	4.2 / 5.04	0.054

The digital block was generated with automated digital design tools such as Synopsys Physical Compiler and Cadence First Encounter. The delay line and delay locked loops were designed in Cadence Virtuoso Schematic Editor.



www.ip-extreme.com

IPextreme, Inc.

307 Orchard City Drive
M/S 202
Campbell, CA 95008
800-289-6412 (toll-free)
408-608-0421 (fax)

THIS WHITE PAPER IS FOR INFORMATIONAL PURPOSES ONLY. THE CONTENT IS PROVIDED AS IS, WITHOUT EXPRESS OR IMPLIED WARRANTIES OF ANY KIND. INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE.

© Copyright 2007, IPextreme. All rights reserved. IPextreme and the IPextreme logo are trademarks of IPextreme, Inc. All other trademarks are the property of their respective owners.
