Test and debug features of the RTO7 chip

DOI:
10.1109/TEST.2005.1583985

Document status and date:
Published: 01/01/2005

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Test and Debug Features of the RTO7 Chip

Kees van Kaam Bart Vermeulen Henk Jan Bergveld

Philips Research Laboratories
High Tech Campus 5, 5656 AE Eindhoven
The Netherlands

{Kees.van.Kaam, Bart.Vermeulen, HenkJan.Bergveld}@philips.com

Abstract

The Philips RTO7 chip consists of a complete receive chain from RF up to and including digital demodulation for Bluetooth-like radio communication. This paper describes both the implementation and verification of the test and debug hardware for the digital core of the RTO7. The core-based DfT and DfD flow of the RTO7 is presented. The experimental results show that the RTO7 is both a fully testable and debuggable chip. State dump analysis results are also presented, showing that the state dumps obtained in the application are 100% stable, and match the state dumps made in simulation, and on the digital test system.

1 Introduction

In recent years, several standards for low-cost short-distance wireless communication, such as Bluetooth, IEEE802.11x and Zigbee have emerged. Systems that implement these standards typically contain a transceiver component to enable wireless communication. The cost of these systems can be reduced by either reducing the number of external components and/or using a standard baseline IC process technology. The latter option takes advantage of the decreasing feature sizes available in each successive CMOS generation.

The main goal of the RTO7 design team is to integrate a Bluetooth-like receiver in a baseline CMOS process, thereby integrating analog, Radio Frequency (RF), and digital modules on a single die. The chosen architecture for the RTO7 chip is highly digitized, offering flexibility at low power and low cost. The transceiver is designed for 2.4 GHz-band, Gaussian Frequency-Shift Keying (GFSK) applications. The specifications are derived from the Bluetooth standard. The digital block in the design performs filtering and demodulation. To allow manufacturing test and prototype silicon debug, this block had to be made fully testable and debuggable. Testing the mixed-signal and RF parts of the design is not covered in this paper.

This paper is organized as follows. The architecture of the RTO7 chip, and particularly the digital filtering and demodulation, is described in Section 2. In addition, this section describes the test and debug requirements for the RTO7 chip. Section 3 gives an overview of prior work in the field of Design-for-Testability (DfT) and Design-for-Debug (DfD). The test and debug architecture used for the RTO7 chip is described in detail in Section 4. The DfT and DfD insertion flow is the topic of Section 5. Our experimental results and conclusions are presented in Sections 6 and 7 respectively.

2 The RTO7 chip architecture

A simplified block diagram of the RTO7 design is shown in Figure 1.

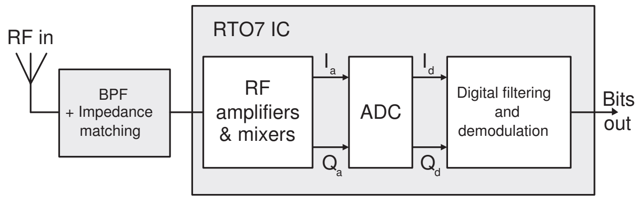

Figure 1: Simplified diagram of the RTO7 design.

The RTO7 design contains three main modules: the RF front-end, a high-resolution complex $\Sigma\Delta$ ADC and a digital module performing filtering and demodulation.

An external Band-Pass Filter (BPF) selects the 2.4 GHz-band and performs impedance matching. The main components in the RF front-end are two matched low noise amplifiers and mixers. The signal received from the RF front-end contains a complex data stream. The amplifiers and mixers split this signal into two separate channels: I_a, and Q_a. The I_a channel contains the real part of the complex data stream and the Q_a channel the imaginary part. Furthermore the front-end converts the RF input signal to a lower frequency of 500 kHz. The high-resolution, complex $\Sigma\Delta$ ADC converts the quadrature low frequency signals from the front-end into noise-shaped bit streams, called I_d and Q_d. The digital module performs channel filtering, decimation (to reduce the sample rate), and demodulation. A complete and detailed description of the receiver can be found in [1].

Figure 2 shows a block diagram of the digital filter and demodulator.

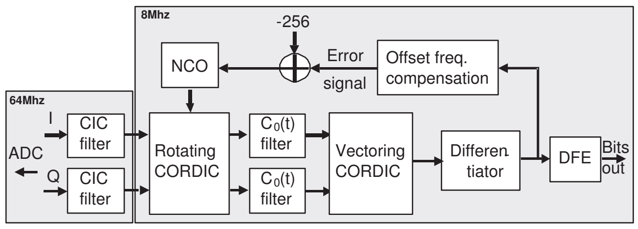

Figure 2: Diagram of the digital filter and demodulator.

The I_d and Q_d bit streams from the $\Sigma\Delta$ ADC are sampled at a clock frequency of 64 MHz. After filtering and decimation in the Cascaded Integrator-Comb (CIC) filters, the clock frequency in the remainder of the digital module is 8 Mhz. Both down-sampled signals are received by the rotating COordinate Rotation DIgital Computing (CORDIC) block. This block shifts the frequency of its input signals from 500 kHz to 0 Hz. Furthermore, the rotating CORDIC compensates for phase shifts in the input signal. The compensation angle ϕ is provided by the Numeric Controlled Oscillator (NCO). The output of the rotating CORDIC can be approximated with good accuracy using only the main impulse response $C_0(t)$. Therefore the output signal of the rotating CORDIC is filtered with filters matched to $C_0(t)$ to suppress white noise. The vectoring CORDIC calculates the phase ϕ of the complex signal. This phase signal is differentiated $(d\phi/dt)$ at the output of the vectoring CORDIC to obtain the instantaneous frequency. The combination of the vectoring CORDIC and phase differentiation forms a Frequency Modulation (FM) demodulator. The last block is a linear Decision-Feedback Equalizer (DFE). This yields a final output bit stream that is 8 times over-sampled compared to a 1 Mb/s symbol rate.

Due to differences in crystal frequencies of transmitters and receivers in for example Bluetooth systems, a frequency offset typically will be present in the signal that needs to be demodulated. This offset leads to a DC shift in the differentiator's output. Therefore, a frequency-compensation loop including first-order filtering is used. The loop error signal is added to a default frequency shift of -256, which is the digital representation of the low frequency of 500 kHz, and subsequently input to the NCO. This leads to a frequency shift of the incoming signal to 0 Hz by the rotating CORDIC as described above.

The RTO7 design has been realized in a 6-metal-layer 0.18-μm standard CMOS process on a 10-Ωcm substrate. The core area of the chip is 3.5 mm^2. The digital filter and demodulator covers 1.25 mm^2 of the total area. The digital core consists of approximately 32k standard library cells of which 9% is used for test and debug purposes. Due to the decimation step, the digital core has two different clock domains: 64 Mhz, and 8 Mhz. The die has been packaged in a 48-pin LQFP plastic package. Apart from supply decoupling capacitances the only external components needed are an antenna filter and impedance matching network, and a 64 MHz crystal.

To enable structural test for manufacturing defects, and to be able to debug prototype silicon on the application board, the design team decided to implement both Design-for-Test and Design-for-Debug. In addition, to allow easy reuse of the design, steps were taken to conform to existing core-based design, test, and debug rules and guidelines.

3 Prior Work

The core-based test architecture used for the RTO7 chip follows company-internal guidelines, which have previously been described in [2], [3], and [4]. This architecture closely resembles the architecture currently under standardization as IEEE 1500 [5]. The architecture combines the advantages of isolating a core from its environment during test, allowing high-coverage test, with the ease of integrating the testable core in a larger SoC design. This approach requires that the individual cores are wrapped for structural test, and a test access mechanism is used to transport the test data to and from the wrapped cores. To further enable easy re-use of the RTO7 cores, we also followed (internal) design standards on IP reuse.

The core-based debug architecture used for the RTO7 chip also follows company-internal guidelines that have previously been presented in [6]. This architecture allows creating state dumps via the IEEE 1149.1 Standard Test Access Port (TAP) [7], while the chip is located on the application board. The DfD hardware required to enable this usage consists of three on-chip components; (1) for access to the internal scan chains, (2) for breakpoint control to freeze the state of the chip at important points in time, and (3) for clock and reset control to functionally reset the chip, stop the clocks, and single-step individual clock domains. All three components are controlled from the IEEE1149.1 TAP by adding private instructions. Although already implemented on several SoCs, we wanted to further develop and verify the full automation of the DfD insertion using the RTO7 design.

To validate the chip design in silicon, we use the well-known state dumping technique [6, 8, 9]. Several issues however were anticipated with respect to the usage of state dumping on chips with multiple clocks. In contrast to most published chip designs that use state dumping techniques, the RTO7 chip uses multiple clock domains, and also features analog components. Both have previously been shown to cause non-deterministic state dumps [10, 11]. For the proper validation of the RTO7 silicon on its application board, we have to eliminate all sources of state dump noise. In [10] the removal of this 'latch divergence' as it was called, is done by placing the microprocessor in a deterministic environment, i.e. a digital tester. Statistical techniques are subsequently used to filter out the state noise. We take a novel and complementary approach, based on the work described in [11]. Our approach involves analyzing the design and its intended environment at design time, eliminate as much state noise as possible by design, and at run time utilize statistical methods.

4 DfT and DfD Hardware

4.1 Architecture Overview

The different RTO7 components for test and debug and their hierarchy in the top-level are shown in Figure 3.

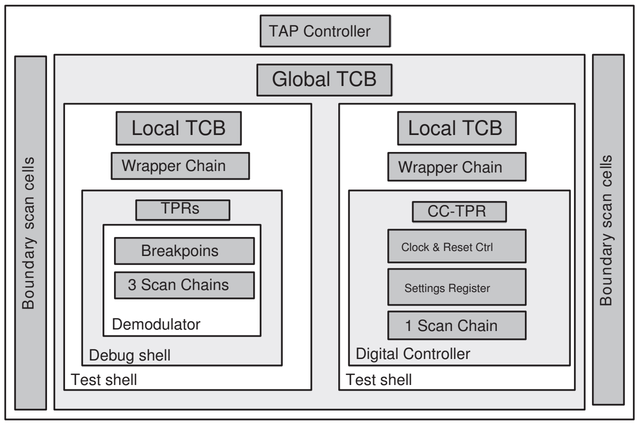

Figure 3: Top-level DfT and DfD hierarchy and components.

The top-level of the chip contains the boundary scan cells, a TAP with controller, and two separate cores; the clock and reset control core and the demodulator core. A global Test Control Block (TCB) is used to provide the global test control, and each core contains a local TCB to provide dedicated, local test control. Each TCB consists of a shift register and an update register, which allows new control values to first be shifted in without changing the test control outputs, and subsequently applied. The shift register is a TAP data register, and is accessible from the TAP controller through the PROGRAM_TCB instruction. This enables full test control via the TAP.

The digital controller is wrapped with a test shell, containing a local TCB and a wrapper scan chain to isolate all core inputs and outputs. This test shell is used for testing the interconnections of the controller with the rest of the chip. The controller itself contains a Clock Control Test Point Register (CC-TPR), a scan chain, a functional clock and reset control module and a serial application settings register. The CC-TPR [6] is used for debug clock control. It has a similar design as a TCB and can be accessed via the TAP by using the PROGRAM_DBG_CC instruction. The application settings register is used to program the digital, analog and RF parts of the RTO7 chip, and is accessible, both directly from chip pins, or via the TAP using the PROGRAM_STATUS instruction.

The demodulator is wrapped with test and debug shells. The test shell contains a local TCB, and a wrapper scan chain. This TCB and the wrapper scan chain have the same purpose as for the controller. The debug shell for the demodulator allows the concatenation of the different scan chains into one single debug scan chain per clock domain. It contains two TPRs; an Access Control TPR (AC-TPR) for controlling the debug chain selection, and an Breakpoint Control TPR (BC-TPR) to program and monitor the breakpoint modules. These two TPRs can be accessed from the TAP using the PROGRAM_DBG_AC, and PROGRAM_DBG_BC instruction respectively. There are five debug breakpoint modules and three scan chains inside the demodulator core.

Due to the limited number of digital pins on the RTO7 chip, we can only have four scan chains at the top-level. Two of the scan chains in the demodulator core are directly connected to the chip pins. The two remaining scan chains are concatenated with the two scan chains in the digital controller.

4.2 The Digital Controller Core

The digital controller core contains the blocks required for on-chip clock and reset generation, including the DfT and DfD for clock and reset control. An special application register has been added to control the settings of the receiver. A block diagram of this digital controller core is shown in Figure 4.

The digital controller core receives a functional, 64 MHz input clock from the analog front-end and a test clock from the TAP. The clock generator and divider generate a 8 Mhz clock signal, which is derived from the 64 Mhz clock, and which are both synchronized to the rising edge of the active low, functional reset signal. The functional 64 and 8 Mhz output clock signals start with their rising edge when the functional reset signal rises. A functional reset can be issued from a chip pin, or from the TAP controller. The latter is achieved through the DBG_RESET TAP instruction, and is very useful during debugging. Both clock signals can be gated by either setting the functional power down bit in the application control register or by asserting the debug stop clocks signal. The debug stop clocks signal is received from the breakpoint modules in the demodulator core, and discussed in Section 4.3.1.

4.2.1 Clock control

The clock controller uses a CC-TPR, which was previously described in [6]. This TPR is a fully testable and TAP-programmable control and observe register. The CC-TPR allows a user to program which of the two on-chip clock domains is stopped on an internal breakpoints, and which clock is activated on a subsequent debug scan. Clock control slices perform the actual gating, and allow switching between the functional clock, the test clock, and the debug clock. These slices are controlled from the CC-TPR and the TCB. The test clock can also be gated, which provides the option to suppress clock pulses in normal mode during test. A possible use for this is to ensure proper data communication between multiple clock domains during test. For this RTO7 design, this is however not strictly required, as all transitions between the two on-chip clock domains in the demodulator have been made clock skew tolerant by inserting anti-skew flipflops, also often referred to as lock-up latches, on the clock domain crossings.

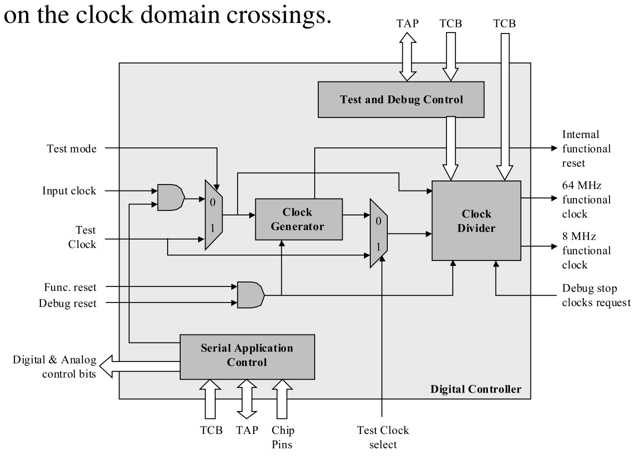

Figure 4: Digital controller block.

4.2.2 Reset control

The chip can be reset either by pulling the reset pin of the chip low, or via the TAP. This latter option makes it much easier to control the execution of the chip from the debugger software, especially on an application board. We will come back to this functionality in Section 6. Next to the two clock signals, the clock generator also generates a new internal reset signal that is synchronized to the rising edge of the 8 MHz clock signal. This internal reset signal is routed to the demodulator core, where it causes a synchronous reset. Both the 8 MHz clock signal and the internal reset signal are output on dedicated chip pins. These two outputs help determine whether the clock and reset generation is correct. The flip-flops in the clock and reset controller core reset asynchronously. A diagram of the clock and reset controller core is shown in Figure 5.

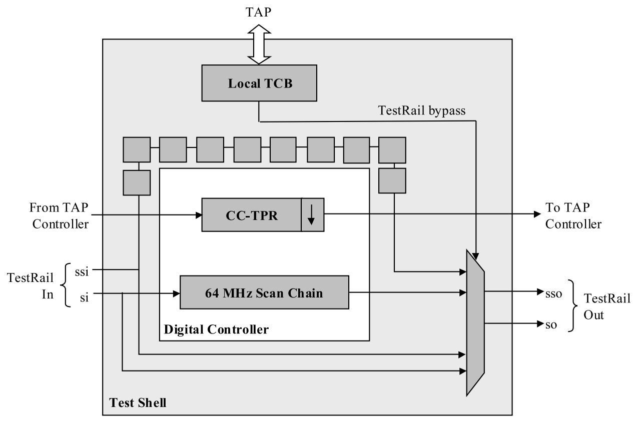

Figure 5: Digital controller core with test shell.

The core has one scan chain and the CC-TPR. In test mode, the CC-TPR is connected in series with the wrapper scan chain. Several trade-off were made between the testability of this core and the required number of wrapper cells. First, the control signals for the application setting register, which are connected to the TAP controller, are not isolated. Although this will result in some coverage loss for the structural test of this interface, this can be easily regained through the use of a dedicated test. Control over the application settings register remains available directly from the chip pins. Second, the outputs of the application settings register that are connected to the analog core are also not isolated. The analog core does not have a test shell. Therefore wrapping the signals used to configure the analog core is of little use, as the analog core, and therefore the interconnections between the analog core and the digital core are checked through a dedicated, functional test any way.

4.3 The Digital Demodulator Core

The digital demodulator has been extended for debug with five breakpoint modules, to detect internal events that mark important stages in the data processing of the core (see points A,B,C,D, and E in Figure 6).

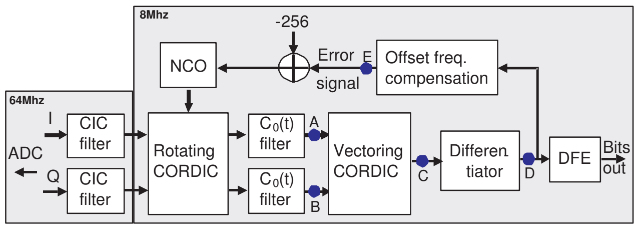

Figure 6: Observation points for the breakpoint modules.

The breakpoint flags from these five modules are connected to a single OR gate, the output of which is directly connected to the debug stop clocks input of the digital controller. This causes a request from any of the breakpoint modules to immediately stop both on-chip clocks.

4.3.1 Breakpoint modules

Together with the designers of the RTO7, a set of breakpoint modules were defined. The block diagram of the demodulator in Figure 6 shows the observation points for the breakpoint modules in the demodulator. For every observation point a separate breakpoint module is used. Figure 7 shows the implementation for the breakpoint modules for points A and B.

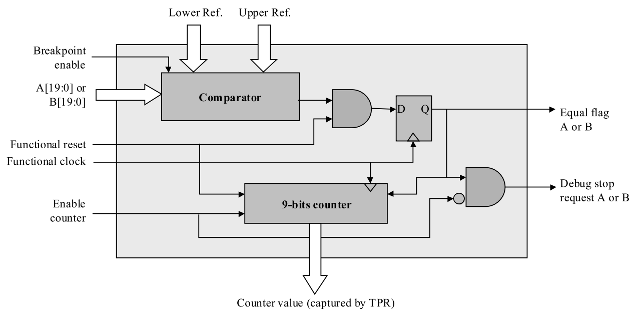

Figure 7: Breakpoint modules for points A and B.

This breakpoint module has two modes of operation: (1) to trigger on an out of range value on the input, or (2) count the number of clock cycles in which the input value is out-of-range. The mode of operation is determined by the enable counter signal.

The comparator checks if the input signal from the demodulator lies between the lower reference and upper reference values provided by the BC-TPR. All signals use two-complement notation. If the input signal is out of range, the output of the comparator will be asserted, resulting in an asserted output flag of the breakpoint module. This flag can be captured by the BC-TPR and observed by debugger software connected to the TAP. Together with the enable counter signal, these two signals enable the 9-bits counter. The counter will increment when the counter is enabled and the value on the input (A[19:0] or B[19:0]) is out of range. In this mode, no debug clock stop request can be issued, and therefore the functional clock cannot be stopped. The complete module is synchronously reset using the functional reset signal and is clocked by the 8 Mhz clock signal in the demodulator. The reference values are controlled, and the output flag captured using the BC-TPR in the demodulator core.

The breakpoint modules for points C, D, and E had similar requirements and hence a similar implementation, see Figure 8.

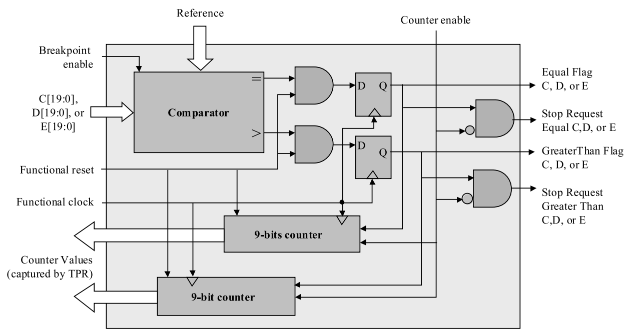
Reference
Counter enable
Breakpoint enable
C[19:0], D[19:0], or E[19:0]
Comparator
=
>
D Q
Equal Flag C, D, or E
Stop Request Equal C,D, or E
GreaterThan Flag C, D, or E
Stop Request Greater Than C,D, or E
Functional reset
Functional clock
9-bits counter
Counter Values (captured by TPR)
9-bit counter

Figure 8: Breakpoint modules for points C, D, and E.

The main difference is the comparator, which has 'greater than' and 'equal' output flags. Both signals can be used to generate a debug stop clock signal when the counters are disabled. Two counters are used to separately count the number of clock cycles the input value is larger, respectively equal to the reference value. The counter values and the output flags can be observed by external debugger software connected to the TAP using the BC-TPR.

The demodulator core is instantiated in a so-called debug shell. This debug shell contains the structures needed to perform scan based debug on the demodulator core and is shown in Figure 9.

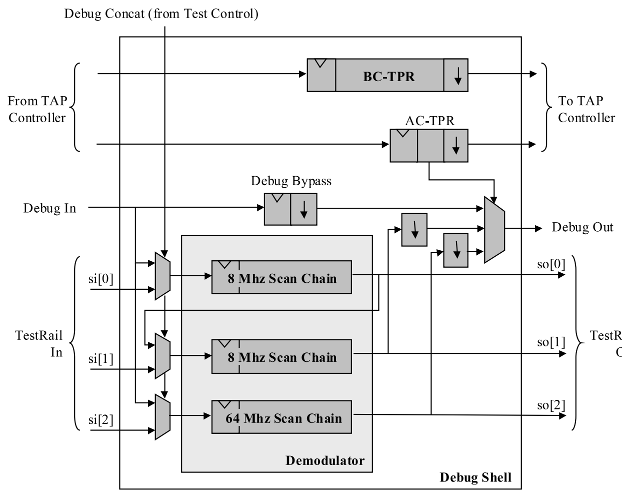

Figure 9: Implementation of the debug shell.

This debug shell contains the following five elements; (1) an AC-TPR, (2) a BC-TPR, (3) a debug bypass, (4) several multiplexers, and (5) several anti-skew elements. There are three different debug chains; the concatenated chain for the 8 Mhz clock domain, the chain for the 64 Mhz domain, and the bypass register. The AC-TPR is a 2-bit register and selects one of the three debug scan chains. The BC-TPR is used to provide all breakpoint reference values, and capture all breakpoint flags and counter values. In total the complete BC-TPR uses 204 bits.

The test shell, TCB and bypass for the demodulator are implemented in the same way as for the digital controller. The control signals for debug, which are connected to the TAP controller, do not have a wrapper cell. Also for the Debug In and Debug Out no wrapper cells are inserted. Figure 10 gives the schematic overview of the test shell of the demodulator core, which contains the debug shell module and surrounding test shell, the local TCB and the bypass.

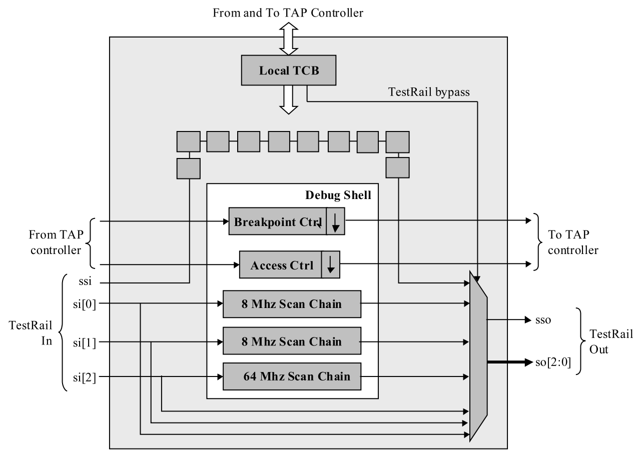

Figure 10: Demodulator test shell.

The local TCB for the demodulator has standard test control signals, but also includes the control signals for the TPRs in the debug shell. There is a special scan enable signal for debug. This signal is necessary, because the default scan enable signal reuses the TDI pin of the TAP interface to indicate the normal cycles during a scan test. This is certainly not allowed while the chip is debugged in the application. The scan enable of the scan chains in the demodulator must then be controllable without the influence of the pins of the TAP, because these pins are then allocated for debug.

5 DfT and DfD Insertion Flow

This section covers the complete DfT- and DfD-insertion flow. Due to the core-based approach used, the flow is split into a core-level part and a top-level part. The steps we took were the following:

- DfT and DfD component generation
- Synthesis
- Test and debug data collection
- Scannable flip-flop insertion
- Layout-optimized flip-flop reordering
- Automated test pattern generation
- Protocol expansion
- Test program assembly

Each of these steps is discussed in more detail below.

5.1 DfT and DfD Component Generation

Several RTL components were generated using internal tools to implement the required DfT and DfD. These components include the debug and test shells, the Test Control Blocks, and the Test Point Registers.

The RTL descriptions of the cores were extended to include *all* test and debug control and data signals. The advantage of including all these signals already at the RTL is that it allows us to develop test benches at RTL to verify the implementation of the DfT and DfD components, and to re-use these exact same test benches without modification at gate-level. This is possible as the interface of our cores will have stayed the same through synthesis and scan chain insertion. The drawback of this approach is that at the RTL design stage, decisions have to be made regarding the number of scan chains that will be inserted in the final, synthesized design. We considered this a small price to pay for the ease of re-use of functional test benches.

5.2 Synthesis

All RTL descriptions of the RTO7 design are synthesized using a commercial synthesis tool. For test and debug there are several synthesis constraints.

- Clock constraints are put on the functional clocks and the test clock;
- No buffers are allowed in the scan enable net;
- Constants and assigns are not allowed;
- No scannable flip-flops or latches are allowed during the logic mapping.

We do not allow latches during synthesis, because we want a synchronous test with well-defined logic paths between register stages. Scannable flipflops are not allowed, because the test and debug structures that we have implemented at RTL themselves are already testable. We only want scannable flipflops to appear in the functional part of the chip, which is what we take care of in the scannable flip-flop insertion step, described in Subsection 5.4.

The scan chain input and output ports of each core receive specific synthesis constraints to leave them dangling in the resulting netlist. An in-house tool takes care of connecting them to scan chains in a later step. The demodulator is flattened to make optimization during synthesis easier and to enable the use of layout-optimized scan chain routing later in the flow.

5.3 Test and Debug Data Collection

A test and debug data collection phase is used before the scan insertion step. First, the inter-clock domain data transfers are analyzed. Because the demodulator has two different functional clock domains, unsafe data transfers between these clock domains can exist in test mode. This is checked and subsequently corrected by inserting anti-skew elements in the paths between clock domains. In our case, negative-edge triggered flip-flops are used. In total, 79 anti-skew elements have been inserted to make the design clock skew safe in test mode. Timing verification was subsequently performed to verify that the timing constraints on the affected paths are still met. Then the flip-flop names that need to become part of the wrapper scan chains of the two cores were extracted. Finally information on existing TPR scan chains is extracted to allow merging of these scan chains with the wrapper scan chains.

At this stage we examined the inter-clock domain communication and determined that for debug, we would not suffer from the data invalidation problem we previously described in detail in [11]. In the RTO7 design we only have data communication from the 64 MHz clock domain to the 8 MHz clock domain (refer to Figure 2. As was proven in [11], data invalidation does not occur under these conditions.

5.4 Scannable Flip-flop Insertion

As the specification of the number of scan chains for both cores was already done at RTL, this now becomes a constraint for the scan insertion tool. This means that the scan insertion tool no longer has the freedom to select the optimal number of scan chains, and number of flipflops per chain. This is the consequence and a disadvantage of introducing as much as possible test and debug hardware in the RTL description. To insert for example the debug multiplexers and bypass at RTL, the number of scan chains of the cores has to be known. The only alternative for a better automation is by introducing the test and debug hardware automatically during synthesis in the Verilog netlist. Unfortunately this is not supported (yet) by any of the available synthesis and/or DfT tools. The scannable flip-flop insertion before placement does not route the scan chains, because we want to take the layout into account when we route the scan chains.

5.5 Layout-optimized Flip-flop Reordering

After the scannable flipflop insertion step, re-synthesis and timing driven placement is performed. The new positions of the scannable flip-flops are used to generate a new ordering of the scannable flip-flops in the scan chains to minimize the total scan chain wire length. The results before and after layout-optimized ordering are shown in Figure 11 and Figure 12 respectively. There is a total reduction in a total scan chain wire length of 64.85%, from 222 mm to 78 mm.

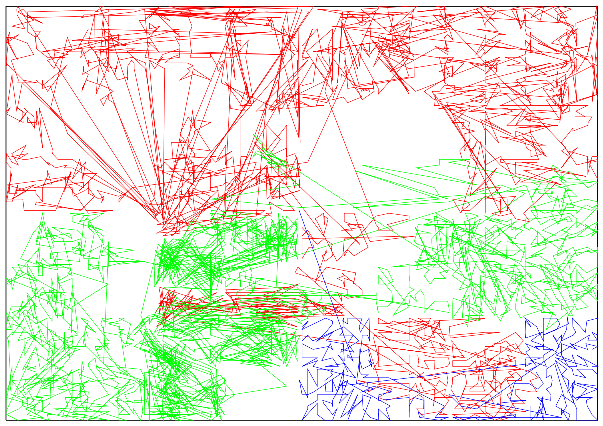

Figure 11: Chain routing before layout-based reordering.

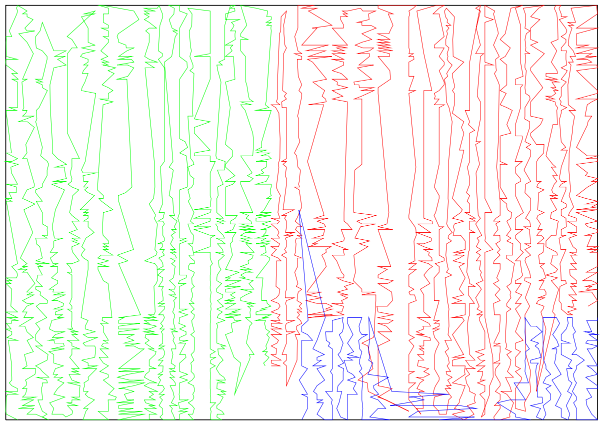

Figure 12: Chain routing after layout-based reordering.

5.6 Automated Test Pattern Generation

Our own state-of-the-art and proprietary ATPG tool was used for the generation of the test patterns. First stuck-at patterns were generated for the two cores, secondly patterns were generated to test the top-level interconnect. As the RTO7 is a test chip, designed to evaluate the options for integrating digital, analog, and RF circuitry on a single die, it was decided to initially only generate stuck-at patterns, and not a extensive, production-worthy set which would also require Iddq and delay fault patterns. Given the flexibility in the clock controller we do not expect any problems with the generation and application of either Iddq or delay fault patterns should they become required in the future. The total stuck-at fault coverage reported for the wrapped demodulator is 98.75% with 286,457 faults and a total of 215 patterns. The same approach was used to generate the stuck-at patterns for the wrapped digital controller. In the ATPG control file additional control signals had to be set for this core during shift and normal mode. The ATPG tool also used additional constraints to allow the testing of the asynchronous reset signals in the digital controller. The total fault coverage report for the digital controller is almost 70% with 3,325 faults and a total of 6 patterns. The combined stuck-at fault coverage for the two cores is 98.42%. The missing coverage is caused by stuck-at faults that are implicitly tested for, for example in the clock control logic. If there is no clock during test due to a defect in the clock control logic, this is also detected. In total 11 additional test patterns were needed to test the chip-level interconnect.

Our internal tool that generates the boundary scan and TAP hardware also generates the test patterns to test this hardware. The initial file that is needed to generate these patterns is the BSDL description. With the BSDL, the tool generates a control file for the test program assembler with an accompanying test protocol and pattern file. These files can then be used during the final step of test assembly.

5.7 Protocol expansion

We subsequently expanded the core-level test patterns to top-level through a procedure called test protocol expansion, which has been described in detail in [12]. This procedure basically involves translating the method of apply test patterns from the core-level to the top-level. Subsequently filling in the specific bits results in the test patterns that need to be applied to the chip pins. The test patterns of the two cores are expanded separately to the top-level, so the cores are not tested in parallel. Of course parallel testing is possible, but the gain in test time is so small that it was not done.

5.8 Test program assembly

We generated test patterns for our automated test equipment. The complete test program for the RTO7 chip contains the following tests (in order of execution):

- Boundary scan and TAP structural test;
- Scan chain continuity for the demodulator;
- Stuck-at test for the demodulator;
- Scan chain continuity for digital controller;
- stuck-at test for for the digital controller;
- Interconnect test;
- Functional test for the demodulator.

6 Experimental Results

We conducted various experiments on the RTO7 design in three different environments: (1) in simulation, (2) on the digital tester, and (3) on an application board. In this section we report on our findings in each of these three environments.

6.1 RTO7 in Simulation

Our test assembly program can generate the necessary files to simulate and verify the generated structural test patterns using the final netlist. For the demodulator core the first 25 patterns were chosen and successfully simulated. Simulating the first 25 patterns verifies all essential test control hardware, from scan chain (access), to clock and reset control. Next to these simulations, additional timing verification was done for the design in test mode to ensure no timing violations were present.

6.2 RTO7 on the Digital Tester

We ran the generated test patterns on an initial batch of 51 samples on our automated test equipment. Two out of these 51 samples did not pass this structural test, all others ran without problems using the test patterns out-of-the-box. We did not do a yield estimation based on these numbers, but given the amount of logic and the maturity of the process technology used, this result is as expected.

To verify the functional design on our automated test equipment, the functional test-bench of the design was also converted to test system input files. This functional test is very helpful, because it can demonstrate the functional correctness of the design. For example if there are timing violations in the real silicon that were missed during timing analysis then the functional test would fail. The simulator was used to write out a VCD file containing the functional input stimuli and design responses. This VCD file was subsequently converted to the tester and applied to the chip. On our ATE, the converted functional test patterns ran without problem.

We also verified the state dumping functionality on our test system. A breakpoint was programmed via the TAP controller, and the functional test stimuli applied. After all functional stimuli were applied, the TAP controller was used to scan out the complete state of the chip, at the moment of the breakpoint, via the TDO pin. We used a simulation to capture all input and output data from the chip during the execution of this debug scenario. This data was converted in the same way to our test system as the conversion of the functional test bench. As the output data of the chip in simulation is used as the reference value for the chip on the tester, we could repeatedly and automatically verify that the state dump obtained from the chip on the digital test system was correct. Repeating this check 10,000 times resulted in identical state dumps each run. This was an important result before we moved to the application board.

6.3 RTO7 in its Application Board

We verified the debug hardware on the application board by connecting our in-house silicon debugger software to the TAP of the RTO7 chip.

We verified a typical debug scenario, which starts off with programming an internal breakpoint. In our case, this was a breakpoint on node A (refer to Figure 6). The chip was subsequently reset using the TAP controller DBG_RESET instruction. The application then starts and the internal clocks are gated when the breakpoint is hit. We had intended to use the debugger software to poll the state of the breakpoint module to know when the internal breakpoint was hit. Unfortunately we discovered that an error had been made matching the implementation of the Breakpoint Control TPR and the TAP instruction to query the state of the breakpoint module. In Figure 13 the implementation of the capture and shift functionality inside the Breakpoint Control TPR is shown.

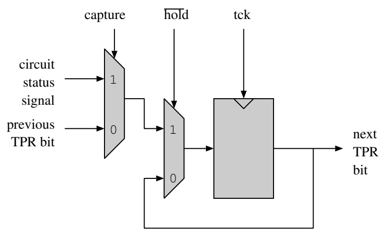

Figure 13: Multiplexer order in breakpoint TPR.

In order to capture the status of the breakpoint module, one can see that both the '$\overline{\text{hold}}$' as well as the 'capture' signal need to be activated at the same time. We should have designed the TAP controller to do this in the Capture Data Register state. However in the implementation of our TAP instruction, this was not the case. Identical to how the instruction was implemented for the Access Control TPR (which doesn't have a capture or update function), the '$\overline{\text{hold}}$' signal is only deactivated in the Shift Data Register stage. This encoding prevents the breakpoint TPR to correctly capture the breakpoint module's state, and hence prevents the debugger software from precisely detecting when a breakpoint was hit. However, given that the test bench for the application board has a fixed length, we programmed the debugger software to wait until the entire test bench was finished to work around this problem.

Once we resolved our breakpoint problem, we were able to capture state dumps from the chip on the application board. However when we examined these state dumps we discovered that, out of a total set of 10,000 state dumps, these state dumps were not all identical. A substantial part was stable (approximately 75-80%), but the remaining 20-25% varied in content. Given that the state dumps on the digital tester were completely deterministic, we conjectured that the variability in the state dumps was caused by the application environment not being completely in sync with the chip and hence causing different results over multiple debug runs.

On the application board, we issue a functional reset via the TAP to start each debug run. This functional reset is also output on a chip pin, and we could connect this signal to the input of the arbitrary waveform generator (AWG) connected to the digital inputs of the chip. We initially thought that some non-deterministic delay was introduced at the beginning of each run by the AWG, causing it to apply the input data slightly out of sync with the chip, causing the state dumps to vary. When we checked the inputs, we discovered that, at the beginning of each run, the AWG in fact was sending 5 samples with an analog value of $\frac{1}{2}V_{DD}$ to the chip inputs. This gets interpreted by the digital circuit as either binary-0 or binary-1, but we cannot guarantee that every run the same interpretation is made. When we replaced the $\frac{1}{2}V_{DD}$ samples with samples identical to the binary stimuli also used in simulation, we saw a significant improvement in state dump stability. Now 98% of all state dumps (again out of a total set of 10,000 state dumps) were identical to the state dumps obtained both in simulation and on the digital tester. However, 2% of all state dumps were not, but were among themselves identical. Another reason had to be found why this was the case.

Using the debugger software and a logical analyzer we observed the reset behavior of the chip over multiple debug runs, each time checking the state dump to see if it matched the 2% class. If so, we would store the logical analyzer results for closer examination. In doing so, we saw the (unexpected) difference that was causing these two distinct state dump class (see Figure 14 and 15).

These two figures show that after a chip reset and one initial clock pulse, either a gated or a non-gated clock pulse occurs. The gated clock pulse causes the class in which 98% of all state dumps are located, the non-gated clock pulse caused the other class. The reason why this pulse either gets gated or not, lies in the reset implementation of both the digital core, the breakpoint module, and the clock control TPR.

A closer examination revealed that the time between programming a breakpoint via the TAP and issuing a reset via the TAP determined whether the breakpoint was hit before the reset. If the breakpoint was hit, the result of Figure 15 would be produced, otherwise the result of Figure 14. As the amount of time between programming the breakpoint and resetting the chip is dependent on the TAP hardware, the software driver, and the PC, we had to change to TCK frequency to solve this problem. By changing the frequency, we changed the time between programming the breakpoint and issuing the reset. For 10,000 runs, we verified that at low TCK frequencies (< 2 MHz) state dumps corresponding to Figure 15 were consistently produced, indicating that each time a functional breakpoint was hit before the chip was reset. For high TCK frequencies (> 10 MHz) state dumps corresponding to Figure 14 were observed, indicating that each time the chip was reset before a functional breakpoint was hit.

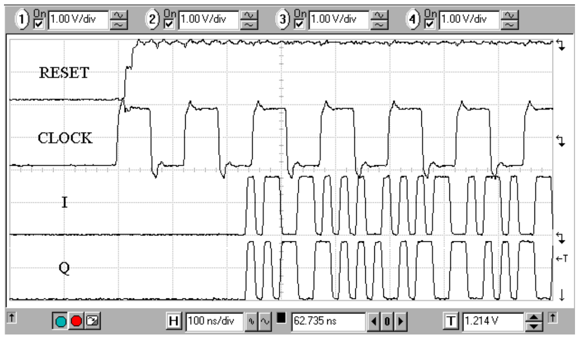

Figure 14: Non-gated 8 MHz clock pulse after reset.

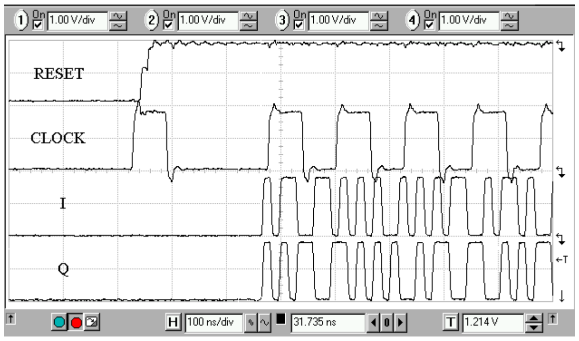

Figure 15: Gated 8 Mhz clock pulse after reset.

7 Conclusion

In this paper we presented the DfT and DfD implementation for the digital filter and demodulator of the RTO7 chip. Both a core-based test and debug approach have been implemented and verified. The chip design was verified in simulation, on a digital tester, and in an application board. We have been able to obtain matching and reproducible results between all three environments, despite initial problems.

Our work has resulted in three new recommendations for the core-based DfD implementation published in [6]. These are holdable debug bypass registers, an updated capture bit implementation, and proper synchronous deassertion of asynchronous reset signals inside the cores, breakpoint modules, and clock control modules.

Acknowledgments

The authors thank the RTO7 design team for their help in making the RTO7 chip testable and debuggable, especially Ad Vaassen, and Gunner Wetzker. The authors also thank Harald Vranken and Tom Waayers for their valuable feedback on the draft version of this paper.

References

[1] H.J. Bergveld, K.M.M. van Kaam, D.M.W. Leenaerts, K.J.P. Philips, A.W.P. Vaassen, and G. Wetzker. A Low-Power Highly Digitized Receiver for 2.4-GHz-Band GFSK Applications. *IEEE Transactions on Microwave Theory and Techniques*, 53(2):453–461, February 2005.

[2] Erik Jan Marinissen, Robert G.J. Arendsen, Gerard Bos, Hans Dingemanse, Maurice Lousberg, and Clemens Wouters. A structured and scalable mechanism for test access to embedded reusable cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, 1998.

[3] B. Vermeulen, S. Oostdijk, and F. Bouwman. Test Infrastructure Design for the PNX8525 Nexperia^{TM} Digital Video Platform System Chip. In *Proceedings IEEE International Test Conference (ITC)*, pages 121–130, 2001.

[4] S.K. Goel, K. Chiu, E.J. Marinissen, T. Nguyen, and S. Oostdijk. Test Infrastructure Design for the Nexperia^{TM} Home Platform PNX8550 System Chip. In *Proceedings Design, Automation, and Test in Europe (DATE) Designers Forum*, pages 108–113, Paris, France, February 2004.

[5] F. DaSilva, Y. Zorian, L. Whetsel, K. Arabi, and R. Kapur. Overview of the IEEE P1500 Standard. In *Proceedings IEEE International Test Conference (ITC)*, pages 988–997, Charlotte, NC, September 2003.

[6] B. Vermeulen, T. Waayers, and S.K. Goel. Core-based Scan Architecture for Silicon Debug. In *Proceedings IEEE International Test Conference (ITC)*, pages 638–647, Baltimore, MD, USA, October 2002.

[7] IEEE Computer Society. *IEEE Standard Test Access Port and Boundary-Scan Architecture - IEEE Std. 1149.1-2001*. IEEE, New York, July 2001.

[8] K. Holdbrook, S. Joshi, S. Mitra, J. Petolino, R. Raman, and M. Wong. microSPARC: A Case-Study of Scan Based Debug. In *Proceedings IEEE International Test Conference (ITC)*, pages 347–350, 1995.

[9] D. Josephson, S. Poehhnan, and V. Govan. Debug methodology for the McKinley processor. In *Proceedings IEEE International Test Conference (ITC)*, pages 451–460, 2001.

[10] P. Dahlgren, P. Dickinson, and I. Parulkar. Latch divergency in microprocessor failure analysis. In *Proceedings IEEE International Test Conference (ITC)*, pages 755–763, 2003.

[11] S.K. Goel and B. Vermeulen. Hierarchical Data Invalidation Analysis for Scan-based Debug on Multiple-Clock System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 1103–1110, Baltimore, MD, USA, October 2002.

[12] Erik Jan Marinissen. The Role of Test Protocols in Automated Test Generation for Embedded-Core-Based System ICs. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):435–454, August 2002.