Documentation for JTAG Switcher

18 October 2019

License

The MIT License

Table of Contents

Chapter 1

Introduction

Lauterbach has published the VHDL source code of the JTAG Switcher component at:
`https://www.lauterbach.com/jtag_switcher.html`

1.1 IEEE 1149.1 aka JTAG

The IEEE standard 1149.1 defines a physical interface to an integrated circuit (IC).

This interface allows a suitable tool to exchange data with circuitry inside the IC. This interface was originally intended for testing assembled printed circuit boards (PCBs).

Nowadays this interface is used as a general purpose testing and debugging interface, allowing a suitable tool to communicate with various debug logic included in an IC.

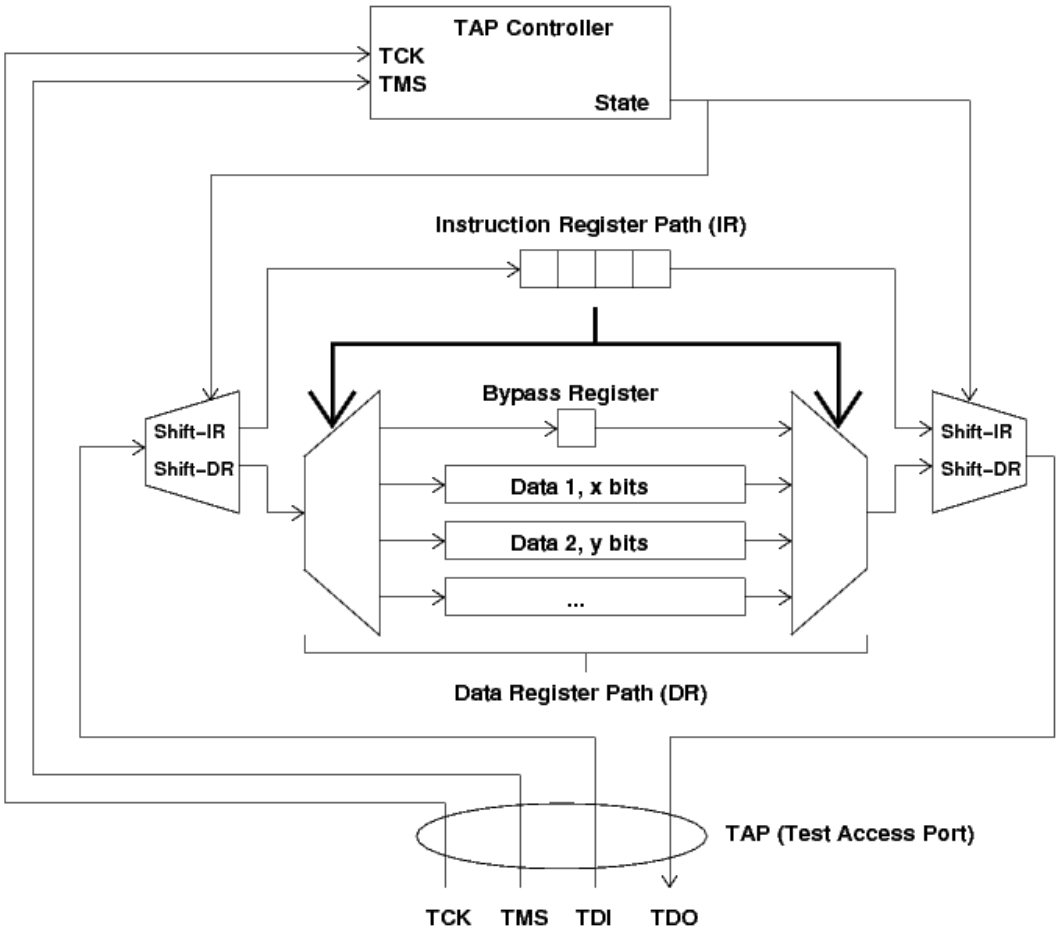
1.2 JTAG overview

A JTAG interface consists of four mandatory and one optional signal. These are:

- TCK : Test Clock, clocks all D-flip-flops, which are part of the JTAG architecture.
- TMS : Test Machine State, controls the JTAG TAP Controller state machine
- TDI : Test Data In, serial data send into the JTAG architecture.
- TDO : Test Data Out, serial data received from the JTAG architecture.
- TRST* : Test Reset, optional reset signal, which will *asynchronously* reset the JTAG TAP Controller state machine, putting it into the "Test-Logic-Reset" state.

These four (or five) signals grouped together are called Test Access Port (TAP).

These four signals connect to the internal JTAG architecture. The overall architecture looks like this (TRST* is excluded here).



Details:

The `Instruction Register` (IR) shift path is accessed, when the TAP Controller is in the `Shift-IR` state.

The `Data Register` (DR) shift path is accessed, when the TAP Controller is in the `Shift-DR` state.

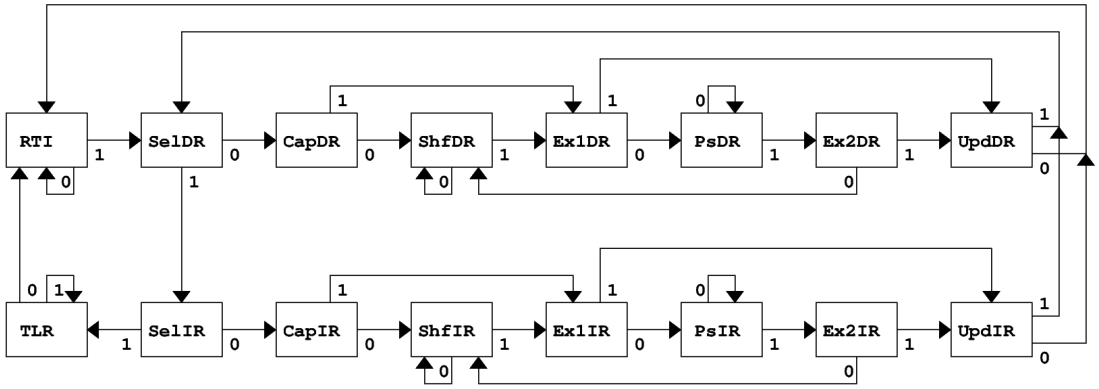
In all other states (all except `Shift-DR` and `Shift-IR`) the data coming in from TDI should be ignored and TDO should be tristated.

The IR should have a fixed length. The content of the IR defines which DR shift path is accessed in the Shift-DR state.

The 1149.1 IEEE standard defines that a device *must* support the BYPASS instruction; this instruction has to have a code/value of all ones (so 1111...). If the IR contains the BYPASS instruction the shift path in the Shift-DR state *must* access the bypass register.

The TAP Controller is a state machine, which changes state at each rising edge of TCK. The state progression is controlled via the value of the TMS signal which is sampled on a rising edge of TCK.

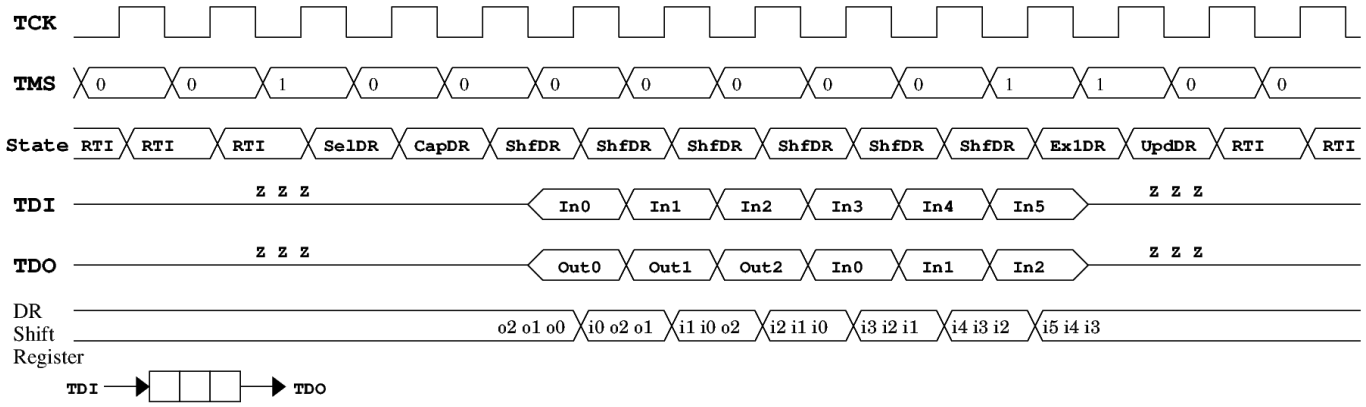
The state machine has sixteen states and behaves as described in the following diagram.



The sixteen states are abbreviated in the above picture. They are called:

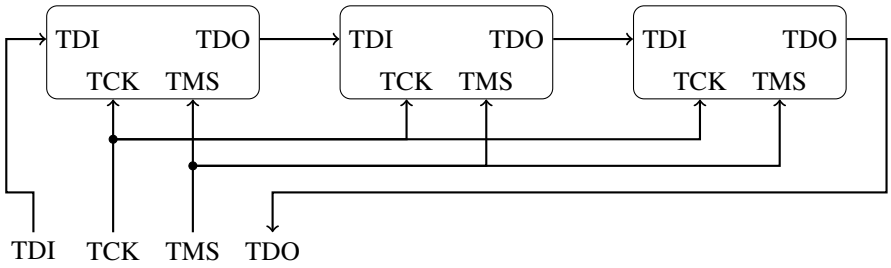
Test-Logic-Reset	(TLR)	Select-DR-Scan	(SelDR)	Select-IR-Scan	(SelIR)
Run-Test/Idle	(RTI)	Capture-DR	(CapDR)	Capture-IR	(CapIR)
		Shift-DR	(ShfDR)	Shift-IR	(ShfIR)
		Exit1-DR	(Ex1DR)	Exit1-IR	(Ex1IR)
		Pause-DR	(PsDR)	Pause-IR	(PsIR)
		Exit2-DR	(Ex2DR)	Exit2-IR	(Ex2IR)
		Update-DR	(UpdDR)	Update-IR	(UpdIR)

As an example, here is a waveform diagram, which shows an access to a three-bit DR. The TAP controller state sequence starts and ends in the Run-Test/Idle state.



1.3 JTAG chaining

If you have several ICs on a PCB and each of these ICs has a JTAG TAP (JTAG Test Access Port), JTAG allows you to daisy chain the TAPs in the following manner (the following diagram shows three ICs connected to one PCB wide TAP):



TCK and TMS are routed in a star fashion from the PCB TAP to each of the IC TAPs.

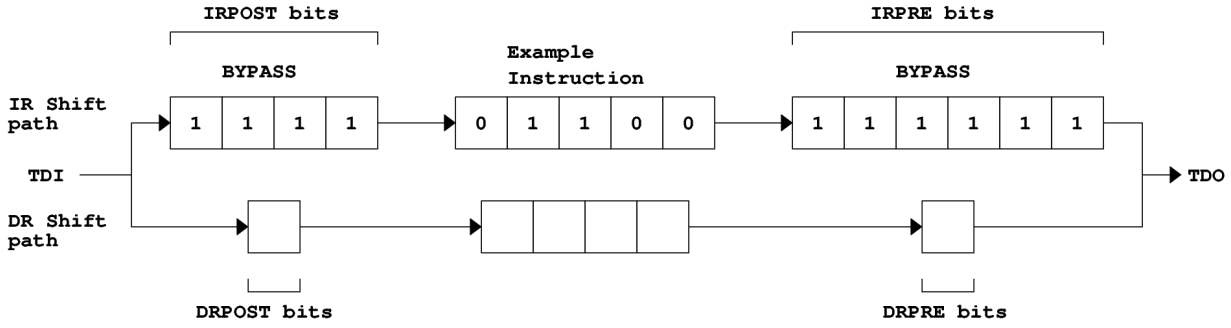
TDI and TDO are chained so that the TDO of one IC TAP is routed to the TDI of the next IC TAP.

With this approach, the IR and DR shift paths are chained together. This means that you access all IRs and all DRs at once when executing a shift TAP controller state sequence.

The usual approach to communicate with one TAP is to load BYPASS instructions into the IRs of all other TAPs. When you then execute a DR shift TAP controller state sequence, you will have to receive and send some additional bits to account for the BYPASS registers of the additional TAPs, but this is not a big problem. In such a scenario you only have to know how many bits precede and follow the real instruction bits of the TAP you want to access and how many extra bits you have to shift before and after the DR data. So basically you only need four values which define how to access the TAP you want. These four values are:

- `IRPRE` : Number of '1' bits to shift before the IR bits.
- `IRPOST` : Number of '1' bits to shift after the IR bits.
- `DRPRE` : Number of BYPASS registers before the DR.
- `DRPOST` : Number of BYPASS registers after the DR.

The following picture tries to illustrate how that works. In this example the middle TAP is accessed.



Chapter 2

JTAG Switcher overview

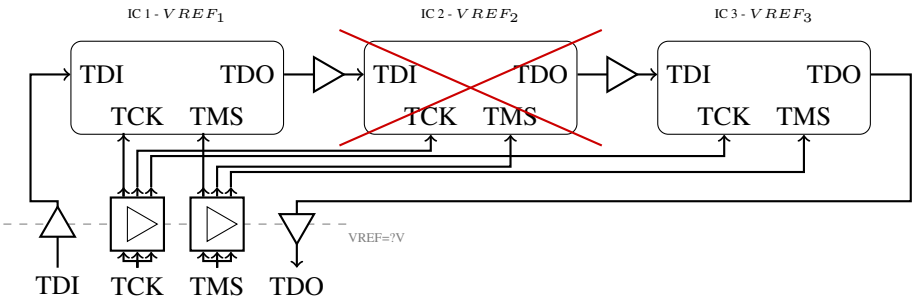
2.1 Motivation

When trying to chain several ICs together into a JTAG chain there are various potential problems.

Two of the most common ones are:

- Different signaling levels
- TDI to TDO chain might become non-functional in a single IC, which will break the *whole* JTAG chain.

Here is a picture illustrating these problems:

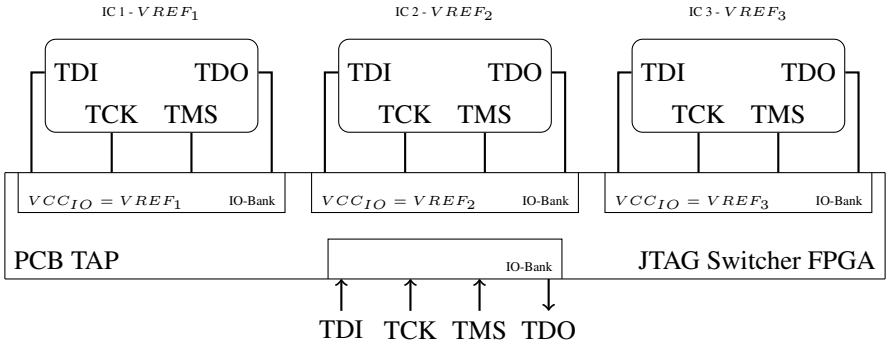


In the situation depicted above, you might circumvent the different signaling voltage levels by including appropriate dual-voltage buffers.

But if additionally `IC 2` is sometimes powered, sometimes not, the chain will be broken once `IC 2` is unpowered, making it impossible to access `IC 1` and `IC 3`.

The "JTAG Switcher" implements a solution for these problems.

The idea is to use an FPGA as buffer between the PCB TAP and the different TAPs of the ICs, like this:

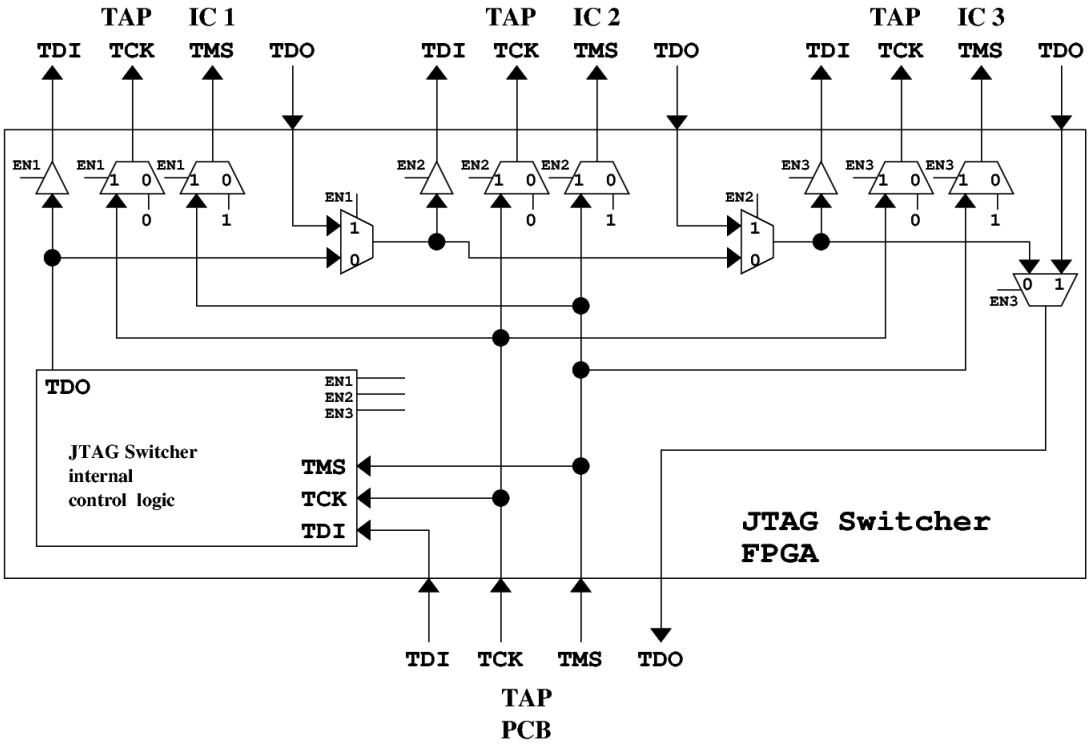


The JTAG Switcher has two main functions:

- The JTAG Switcher translates to the appropriate voltage levels.
- The JTAG Switcher can modify which TAPs are visible on the PCB TAP. So it provides the ability to choose which TAPs are included in the JTAG chain.

2.2 Basic circuitry

Here is an overview of the internal circuitry of the JTAG Switcher FPGA. In this example three JTAG slave TAPs are connected to the FPGA. In general the number of JTAG slave TAPs is configurable.



Explanations:

- The "JTAG Switcher internal control logic" is responsible for controlling three control bits: `EN1`, `EN2`, `EN3`.
- A JTAG slave TAP is completely excluded from the JTAG chain if the corresponding `ENx` bit is '0'. Exclusion from the JTAG chain means:
  - TDI to this JTAG slave TAP is tri-stated.
  - TCK to this JTAG slave TAP is gated and kept at value '0'.
  - TMS to this JTAG slave TAP is kept at value '1'.
  - TDO from this JTAG slave TAP is ignored.
- When all `ENx` bits are '0', no JTAG slave TAP is visible on the PCB TAP. In this case *only* the "JTAG Switcher internal control logic" is visible on the PCB TAP.

2.3 Design decisions

2.3.1 JTAG Switcher should conform to IEEE 1149.1

The JTAG Switcher internal logic should behave like a IEEE 1149.1 TAP.

Rationale:

- Configuration of the JTAG Switcher does not require any extra pins.
- Regular JTAG tools are easily able to just ignore the JTAG Switcher, so these tools do not need to have any knowledge of the internal logic of the JTAG Switcher.

2.3.2 Transparency

You might configure the JTAG Switcher once. The JTAG Switcher *will not* modify the selected JTAG slave TAPs because of entering the Test-Logic-Reset state or because of assertion of the TRST* signal.

Rationale:

- Once the JTAG Switcher is configured, a JTAG tool does not need to know how the JTAG Switcher behaves internally. All JTAG TAP Controller state sequences are allowed.
- This makes the JTAG Switcher transparent to a JTAG tool; the configuration stays stable and the JTAG Switcher will act like an inert extra TAP on the JTAG chain.

2.3.3 First in JTAG chain

The JTAG Switcher internal logic is directly connected to the TDI signal of the PCB TAP. So the JTAG Switcher internal logic always appears to be the "first" device in the JTAG chain.

Rationale:

- This construction will always allow recovery if the JTAG chain is broken by an inaccessible JTAG slave TAP: Since the JTAG Switcher will still be directly connected to TDI, a series of *write* accesses to the JTAG Switcher internal logic IR and DR shift paths is possible. This way arbitrary commands can still be sent to the JTAG Switcher internal logic and thus allow for the exclusion of all JTAG slave TAPs from the JTAG Chain, putting the JTAG Chain back into a usable state.

2.3.4 JTAG slave TCK clock gating

JTAG slave TAPs which are not selected, will see no toggling on the corresponding JTAG slave TCK signal.

Rationale:

- Some ICs exhibit JTAG TAPs, which produce side effects, if TCK toggles. By keeping the TCK signal of an excluded JTAG slave TAP stable, all these side effects are avoided.

2.3.5 Change selection *only* in the Run-Test/Idle JTAG TAP Controller state

The JTAG Switcher will change the value of the `ENx` bits, which control the in- and exclusion of JTAG slave TAPs *only* in the Run-Test/Idle JTAG TAP Controller state.

Rationale:

- A JTAG slave TAP is always ex- and included onto the JTAG TAP chain while the TAP Controller state is in Run-Test/Idle. With this construction all excluded JTAG slave TAP Controllers are kept in the Run-Test/Idle state. This way there is one consistent state for all excluded TAP Controllers.
- This allows a series of DR and IR shift path accesses to the JTAG Switcher control logic, while then still modifying the selection of *all* JTAG slave TAPs at exactly the same TCK clock edge: As long as the Run-Test/Idle state is not reached the selection will not change. Only by entering the Run-Test/Idle state the configured selection will get activated.

Chapter 3

JTAG Switcher implementation

This section tries to describe the architecture and behavior of the "JTAG Switcher internal control logic". This logic is responsible for implementing access to all control registers.

3.1 JTAG to bus access translation

Internally the JTAG Switcher uses a very simple 16-bit wide data bus. This bus behaves similarly to any other known peripheral buses: So a bus master drives:

- An Address to all bus slaves.
- A 16-bit wide data word to all bus slaves.
- A write strobe (asserted for one TCK clock cycle)
- A read strobe (asserted for one TCK clock cycle)

The bits in the 16-bit wide data word are numbered from 0 to 15.

Bit 0 corresponds to the least significant bit (2^0).

Bit 15 corresponds to the most significant bit (2^{15}).

The function of the bus master is to translate JTAG accesses to bus accesses on the internal bus. The basic operations supported are:

- Set address for bus access
- Write to bus
- Read from bus
- Read from and write to bus
- Set address and write to bus

All these operations are executed via a DR Shift Sequence.

Read operations are triggered by the Capture-DR state.

Write operations and setting the address are triggered by the Update-DR state.

The content of the IR defines which operation is currently selected.

Example Sequences:

Set bus address and then write a 16 bit value to this address

- Shift "Set Address" instruction into IR of JTAG Switcher.
- Shift address for bus access into DR of JTAG Switcher.
- Shift "Write to bus" instruction into IR of JTAG Switcher.
- Shift 16-bit value (via TDI) into DR of JTAG Switcher.

Set bus address and then read a 16 bit value from this address

- Shift "Set Address" instruction into IR of JTAG Switcher.
- Shift address for bus access into DR of JTAG Switcher.
- Shift "Read from bus" instruction into IR of JTAG Switcher.
- Shift out 16-bit value (via TDO) from DR of JTAG Switcher.

Combined: Set bus address and write a 16 bit value to this address

This operation is implemented for more efficient access to modify control bits.

- Shift "Set address and write to bus" instruction into IR of JTAG Switcher.
- Shift address + 16-bit value for bus access into DR of JTAG Switcher.

3.2 Set and Clear control bit accesses

Control bits inside the JTAG Switcher use the concept of "Set and Clear" bits for modification.

To modify control bits a 16-bit write access to the bus will be used to modify up to 8 control bits.

Each control bit reacts to two bits of the 16-bit data word. The four possible combinations for the two bits of the 16-bit data word are interpreted in the following manner:

- `00`: "No Operation"; the corresponding control bit is *not* modified
- `01`: "Set Bit"; the corresponding control bit is set to '1'.
- `10`: "Clear Bit"; the corresponding control bit is set to '0'.
- `11`: "Clear Bit"; the corresponding control bit is set to '0'.

For consistency the JTAG Switcher uses a fixed mapping from bits of a 16-bit data word to modify up to 8 control bits:

Data Bits	15..14	13..12	11..10	9..8	7..6	5..4	3..2	1..0
Slave TAP Nr	8	7	6	5	4	3	2	1

All even numbered bits of a 16-bit data word (0, 2, 4, 6, 8, 10, 12, 14) are used to indicate "Set Bit" operations. All odd numbered bits of a 16-bit data word (1, 3, 5, 7, 9, 11, 13, 15) are used to indicate "Clear Bit" operations. "Clear Bit" operations have a higher priority, so if both "Set Bit" and "Clear Bit" are set, a "Clear Bit" operation is performed.

A read access to a 16-bit data word will output the value of the corresponding control bits on the even numbered bits of the 16-bit data word.

Rationale:

This scheme allows to modify single control bits without knowledge of any other control bits. Without the "Set/Clear" scheme, you usually would have to read out all control bits accessed via one specific data word, then mask or set the control bits you want to modify and then to write back the resulting value.

With the "Set/Clear" scheme you do not need to know any values of control bits to modify a single control bit: You write the operation you need (`01` or `10`) for the control bits you want to modify; for all other control bits you write (`00`) leaving the corresponding control bits unmodified.

Disadvantage:

You can only control up to 8 control bits with one 16-bit data word address.

3.3 Internal bus, banked access

As mentioned in the previous section, a 16-bit value is used to modify up to 8 control bits. Without any extension this naturally corresponds to controlling 8 JTAG slave TAPs.

To support more JTAG slave TAPs, all control registers corresponding to one JTAG slave TAP are "banked". This means that the bus address for these control registers might be extended by up to 4 bits. These 4 extra bits are used as "bank number". The full address for one control register is the bus address plus (additionally) the bank number.

Per default only one bank is implemented (bank number zero) for up to 8 JTAG slave TAPs.

If more JTAG slave TAPs are needed, up to 15 banks might be implemented (up to 4 bank bits, bank numbers 0 to 14), giving a total of up to 120 JTAG slave TAPs.

Bank 15 is reserved to access registers which are global for the whole JTAG Switcher logic (*not* per JTAG slave TAP).

3.4 JTAG Switcher internal logic TAP Details

The TAP implemented inside the JTAG Switcher internal logic has a 4-bit wide JTAG Instruction Register.

In addition to the Instructions described in the "JTAG to bus access translation" section it has:

- A "Reset all control registers" instruction. When loaded, this instruction will assert an internal synchronous reset to all control registers, when *additionally* the TAP Controller State is Run-Test/Idle.
- A "Reset all Select registers" instruction. When loaded, this instruction will deselect all slave TAPs, when *additionally* the TAP Controller State is Run-Test/Idle.
- A "BYPASS" instruction. This instruction has a code of "1111" as specified by IEEE 1149.1. When loaded a 1-bit bypass register will be put into the DR Shift path.
- An "IDCODE" instruction. As specified by IEEE 1149.1, this instruction is automatically loaded into the IR, when the TAP Controller State is Test-Logic-Reset. When loaded this instruction allows to read out the IDCODE of the JTAG Switcher internal logic TAP via the DR Shift path.

The following DR Shift path lengths are used:

- When the "BYPASS" instruction is loaded, 1 bit.
- When the "IDCODE" or "Set address and write to bus" instruction is loaded, 32 bit.
- For all other instructions, 16 bit

3.5 Selection and Deselection of JTAG slaves

Selection and Deselection of JTAG slaves only changes when the JTAG TAP Controller state is "Run-Test/Idle".

So the configuration set via internal bus writes to the corresponding control bits takes effect once the JTAG Switcher internal logic TAP Controller reaches the "Run-Test/Idle" state.

When the configuration is changed, it is *mandatory* to spend at least **8** consecutive TCK clock cycles in the "Run-Test/Idle" state to activate the configuration.

These clock cycles allow the internal logic to cleanly turn the TCK signals for the JTAG slaves on and off.

Chapter 4

JTAG Switcher programming interface guide

It is assumed a reader has a working knowledge of how JTAG accesses work in principle.

You might read the section *1.2 JTAG overview* for a short introduction.

The JTAG Switcher FPGA is connected to a configurable number of JTAG slave TAPs.
The JTAG slave TAPs are numbered from 1 to n.

4.1 Terminology, bit numbering

Some terminology:
DR-Shift Sequence
A JTAG TAP Controller state sequence which starts with a transition from the *Select-DR-Scan* state to the *Capture-DR* state and ends with a transition from the *Update-DR* state to either the *Run-Test/Idle* state or the *Select-DR-Scan* state.

IR-Shift Sequence
A JTAG TAP Controller state sequence which starts with a transition from the *Select-IR-Scan* state to the *Capture-IR* state and ends with a transition from the *Update-IR* state to either the *Run-Test/Idle* state or the *Select-DR-Scan* state.

An *IR-Shift Sequence* will define the content of all JTAG Instruction Registers of all TAPs on the JTAG chain.

Bit numbering:
In this document little-endian bit order is used. This means: If you have an n-bit value, the bits are numbered from bit 0 to bit $n-1$. Bit 0 corresponds to the value 2^0, Bit $n-1$ corresponds to the value 2^{n-1}.

Example:
`0x9A = 1001 1010`

Bit Nr	7	6	5	4	3	2	1	0
Value	1	0	0	1	1	0	1	0

4.2 JTAG to bus translation

The JTAG Switcher internally uses a simple bus system to access control bits.

The content of the 4-bit JTAG Instruction Register of the JTAG Switcher TAP defines which bus operation is executed by a *DR-Shift Sequence*.

The following 4-bit Instruction Register codes are defined:

- **0101** IEEE 1149.1 IDCODE Instruction Register code.
  Per default the JTAG Switcher implementation outputs an IDCODE of **0x11111FFF**.
- **0110** Reset all control registers. Will assert a synchronous reset to all bus slaves,
  when additionally the JTAG Switcher TAP Controller is in the *Run-Test/Idle* state.
  This reset *includes* to reset all Select registers (see **0111** Instruction Register code).
- **0111** Reset all Select registers. Will deselect all slave TAPs,
  when additionally the JTAG Switcher TAP Controller is in the *Run-Test/Idle* state.
- **1001** Write to bus
- **1010** Read from bus
- **1011** Read from and Write to bus
- **1100** Set address for bus access
- **1101** Set address and write to bus
- **1111** IEEE 1149.1 BYPASS Instruction Register code.

Bus read accesses are triggered by reaching the *Capture-DR* JTAG TAP controller state.
Bus write accesses are triggered by reaching the *Update-DR* JTAG TAP controller state.

The address is set via a DR-Shift Sequence with a 16-bit value. Internally by default 3 address bits are used, which are set by bit `2..0` of the 16-bit value. Additionally bit `15..12` of the 16-bit value sets the bank to be accessed. Unused bits should be always set to 0; this enables the number of address bits to be easily increased in a JTAG Switcher implementation.

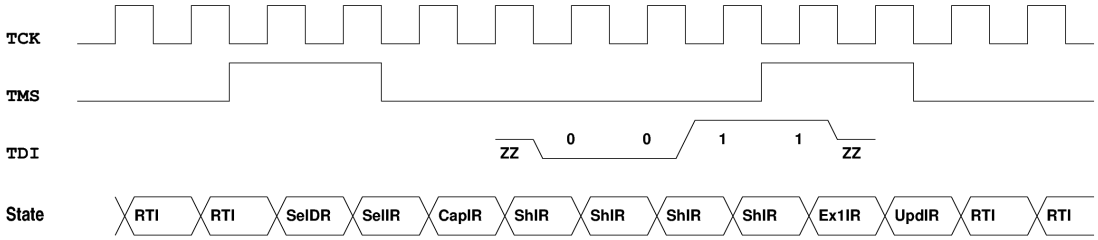
Up to 15 banks (bank 0..14) are supported (this is configurable, when the JTAG Switcher is implemented).
Bank 15 is reserved for global registers (which are not-banked and are global for the whole JTAG Switcher module).

If the IR contains the **1101** code (Set address and write to bus), then a 32-bit DR-Shift Sequence is expected. For this DR-Shift Sequence, bit `31..16` of the 32-bit word define the bus address to be accessed, bit `15..0` define the 16-bit value to be written to this address.

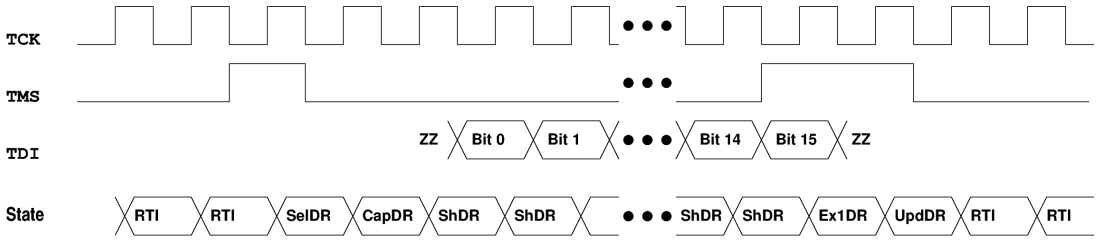
All DR-Shift Sequences will output the 16-bit value read from the current bus address via the TDO signal. If the IR contains the **1101**, the 16-bit value will be repeated twice (so the value will appear on bits `31..16` and bits `15..0` of the sampled TDO value).
If the IR contains a "Read from bus" code (**1010** or **1011**) then a read strobe will also be triggered on the internal bus by the *Capture-DR* JTAG TAP controller state. (This mechanism might be used for destructive reads).

4.2.1 Example JTAG sequences

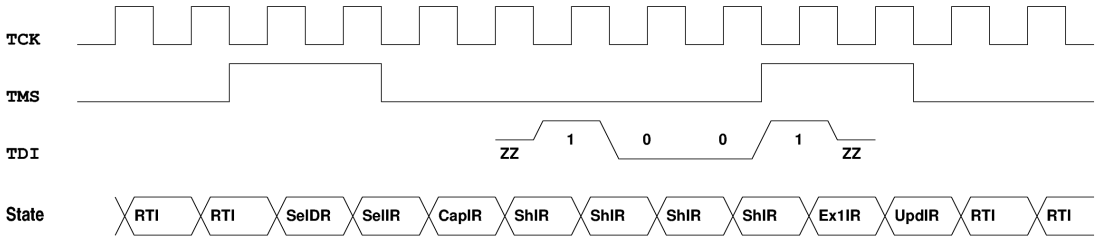
Prepare to set the address for a bus access via an IR-Shift Sequence.



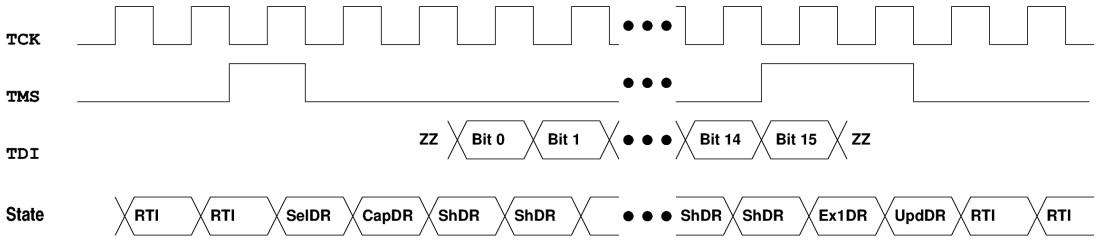
The JTAG TAP Controller state sequence starts in the *Run-Test/Idle* state. The IR code **1100** is shifted into the IR (least-significant bit first). After setting the IR code, it is now possible to set the bus address via a DR-Shift Sequence:



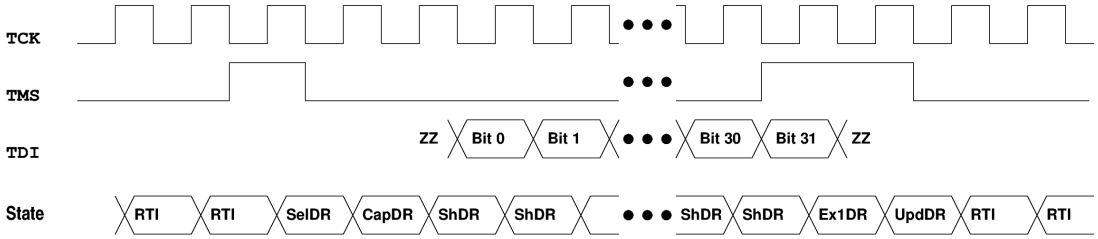
Bit `2..0` of the 16-bit value are used to set the address, bit `15..12` are used to set the bank. To execute a write access, you next have to set the IR to another value:



The IR code **1001** is shifted into the IR. After setting this IR code, it is now possible to write a 16-bit value to the specified bus address and bank:



To execute a series of writes to different bus addresses, you first set the IR code to **1101** and then execute several DR-Shift Sequences with 32-bits:



Here bits `15..0` define the 16-bit value to be written to the address specified by bits `31..16`.

4.3 Global registers

Global registers are accessible via bank 15 (`0xF`). Global registers are *not* implemented per JTAG slave TAP; they exist only once in the whole JTAG Switcher.

Bus Address	Name	
`0x0`	ConfigA	mandatory (read only)
`0x1`	ConfigB	reserved for future use
`0x2`	ControlA	reserved for future use

Example
How to read the *ConfigA* register:

- Set the bus bank/address via the value `0xF000` (bank 15 for global register access, bus address `0x0`).
- Read the value of the ConfigA register from this address:
  - Set the JTAG Instruction Register to the code **1010** (*Read from bus*)
  - Read the 16 bit value via a DR-Shift Sequence.

4.3.1 ConfigA Register

This global read-only register might be used to read out the configuration of the JTAG Switcher. The 16-bit word has the following meaning:

- Bit `6..0` number of JTAG slave TAPs.
- Bit `7` If 1, then the JTAG Switcher implements reading of control bits (only for debugging).
- Bit `8` If 1, then the JTAG Switcher implements "TDO-Sync" control bits.
- Bit `9` If 1, then the JTAG Switcher implements "Ungate" control bits.
- Bit `10` If 1, then the JTAG Switcher implements "TMS-Low" control bits.
- Bit `11` If 1, then the JTAG Switcher implements "TRST-Ena" control bits.

4.3.2 ControlA Register

To allow to control more bits via a single global register address, the following scheme is used:

- Bit `15..12` bit **group** which is accessed.
- Bit `11..0` six Set/Clear bit pairs to control up to six different global register bits.

With this scheme up to $16 \cdot 6 = 96$ bits can be set and cleared.

4.4 JTAG slave TAP control bits

The JTAG Switcher FPGA connects to a configurable number of JTAG slave TAPs.
The JTAG slave TAPs are numbered from 1 to n.
Several control bits control the operation of one JTAG slave TAP.
Each control bit for one JTAG slave TAP controls one particular function.
The control bits for one function, for a set of eight JTAG slave TAPs are accessed via one specific bus bank/address.

To modify a single control bit for a single JTAG slave TAP a set/clear scheme is used. This means to modify a single control bit you have to write *two* bits via a bus write access. A two bit pair of the written value, defines the operation which is performed on a single control bit. The following operations are defined:

- `00`: "No Operation"; the corresponding control bit is *not* modified
- `01`: "Set Bit"; the corresponding control bit is set to '1'.
- `10`: "Clear Bit"; the corresponding control bit is set to '0'.
- `11`: "Clear Bit"; the corresponding control bit is set to '0'.

A fixed mapping is used between the 16-bit data value written and the JTAG slave TAP number for which a control bit is modified:

Data Bits	15..14	13..12	11..10	9..8	7..6	5..4	3..2	1..0
JTAG slave TAP	8	7	6	5	4	3	2	1

Writing a 16-bit value of `0x0000` will *not* modify any control bits.

When you read control bit values (which is only supported when you enable it in the JTAG Switcher implementation), the current control bit values will appear on the even numbered bits of the read 16-bit data word.

If more than eight JTAG slave TAPs are implemented, the bus bank will additionally define which JTAG slave TAP control bits are accessed.
Bank 0 addresses JTAG slave TAPs 1-8,
Bank 1 addresses JTAG slave TAPs 9-16,
Bank 2 addresses JTAG slave TAPs 17-24 and so on.

The following control bit functions are supported

Bus address	Function	
`0x1`	Select	mandatory
`0x2`	TDO-Sync	optional
`0x3`	Ungate	optional
`0x4`	TMS-Low	optional
`0x5`	TRST-Ena	optional

Modifying control bits does not directly change the behavior of the JTAG slave TAPs.

To take effect, you *additionally* have to move to the *Run-Test/Idle* TAP controller state and stay at least for 8 TCK clock cycles in the *Run-Test/Idle* TAP controller state.

This scheme allows the modification of a bunch of control bits via a series of IR/DR-Shift Sequences without taking effect (by avoiding to enter the *Run-Test/Idle* TAP controller state). After these modifications, when then entering the *Run-Test/Idle* TAP controller state all modifications will take effect at the same time.

Example
How to set the Select Control Bit of JTAG slave TAP number 20:

- Set the bus bank/address via the value `0x2001` (bank 2, bus address `0x1`).
- Write the value `0x0040` to this address.

Bits `7..6` of the written value are `01`. So this is a "Set Bit" operation for the fourth JTAG slave TAP in bank 2, which means JTAG slave TAP number 20.

To make the selection effective, you also have to stay in the *Run-Test/Idle* TAP controller state for at least 8 TCK clock cycles.

4.4.1 Select Control Bit

This bit controls if a JTAG slave TAP is included in the chain or not.

To take effect you also have to enter the *Run-Test/Idle* state and stay there for at least 8 TCK clock cycles.

The visible JTAG chain will be constructed in the following manner:

- TDI will be directly connected to the JTAG Switcher TAP
- All JTAG slave TAPs will appear in increasing order; so JTAG slave TAP 1 (if included) will appear directly after the JTAG Switcher TAP, JTAG slave TAP 2 (if included) will appear next and so on.

4.4.2 TDO-Sync Control Bit (optional)

If the JTAG Switcher includes this control bit, then this bit allows the insertion of a Flip-Flop (clocked on the falling edge of TCK) after the corresponding JTAG slave TAP.

This means the input of the Flip-Flop will be connected to the TDO of the corresponding JTAG slave TAP. When the Flip-Flop is inserted into the chain, the output of the Flip-Flop will be connected to the TDI of the next included JTAG slave TAP, or to the TDO of the JTAG switcher if no further JTAG slave TAP is included.

The insertion of such a Flip-Flop relaxes timing a lot and allows higher frequencies for TCK.

Caveat: Such a Flip-Flop might *only* be inserted if the corresponding JTAG slave TAP has its TDO signal synchronized to the falling edge of TCK (as the 1149.1 standard mandates). Unfortunately there exist devices, for which TDO is synchronized to the rising edge of TCK (under certain circumstances). If this is the case, you *must not* insert a TDO-Sync Flip-Flop for this particular JTAG slave TAP.

4.4.3 Ungate Control Bit (optional)

By default the TCK signal of a deselected JTAG slave TAP is kept low.

If the JTAG switcher includes this control bit, then this bit allows to un-gate the TCK signal of a JTAG slave TAP, without including it in the JTAG chain (so even if the corresponding JTAG slave TAP is deselected).

This control bit might be useful if a JTAG slave TAP looses state (for example because it is powered down).

By default the TMS signal of a deselected JTAG slave TAP will be kept high. By un-gating the TCK signal to a deselected JTAG slave TAP you are able to ensure that the TAP Controller of this JTAG slave TAP will reach the *Test-Logic-Reset* state. So this control bit moves a deselected JTAG slave TAP to the *Test-Logic-Reset* state, *without* moving the rest of the JTAG chain through *Test-Logic-Reset*.

There is an alternative way to achieve the same:

- De-Select all JTAG slave TAPs (so that no JTAG slave TAPs are included).
- Select the nominated JTAG slave TAP; go to *Run-Test/Idle* to make sure the selection takes effect.
- Send a sequence of at least 5 TCK clock cycles with TMS high. This will ensure that the nominated JTAG slave TAP is now in the *Test-Logic-Reset* state.
- Re-Select all previously selected JTAG slave TAPs; go to *Run-Test/Idle* to make sure the selection takes effect.

This sequence of operations makes sure that only the nominated JTAG slave TAP "sees" the *Test-Logic-Reset* state. This is usually important because many devices exhibit side effects, when the TAP Controller enters the *Test-Logic-Reset* state.

4.4.4 TMS-Low Control Bit (optional)

This Control bit usually only makes sense if you also include the Ungate Control bit.

By default the TMS signal of a deselected JTAG slave TAP is kept high. The TMS-Low Control Bit configures the TMS signal of such a deselected JTAG slave TAP to be pulled low.

In conjunction with the Ungate Control Bit this drives the TAP Controller of a deselected JTAG slave TAP to the *Test-Logic-Reset* state and then drives it to the *Run-Test/Idle* state.

This might be useful if keeping the TAP Controller of a JTAG slave TAP in *Test-Logic-Reset* has unwanted side-effects.

As with the Ungate Control Bit, there is an alternative to achieve the same:

- De-Select all JTAG slave TAPs (so that no JTAG slave TAPs are included).
- Select the nominated JTAG slave TAP; go to *Run-Test/Idle* to make sure the selection takes effect.
- Send a sequence of at least 5 TCK clock cycles with TMS high. This will ensure that the nominated JTAG slave TAP is now in the *Test-Logic-Reset* state.
- De-Select all JTAG slave TAPs; this will make sure that the nominated JTAG slave TAP is parked in the *Run-Test/Idle* state.
- Re-Select all previously selected JTAG slave TAPs; go to *Run-Test/Idle* to make sure the selection takes effect.

4.4.5 TRST-Enable Control Bit (optional)

If a JTAG slave TAP is included in the JTAG chain, its TRST* signal will have the same value as the TRST* input of the JTAG Switcher.

If a JTAG slave TAP is not included in the JTAG chain, by default its TRST* signal will be kept high (which means it is de-asserted).

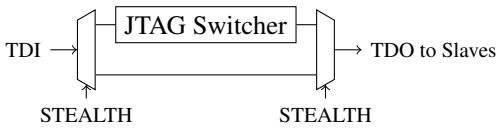
Including this control bit allows to assert the TRST* signal of a deselected JTAG slave TAP.

Chapter 5

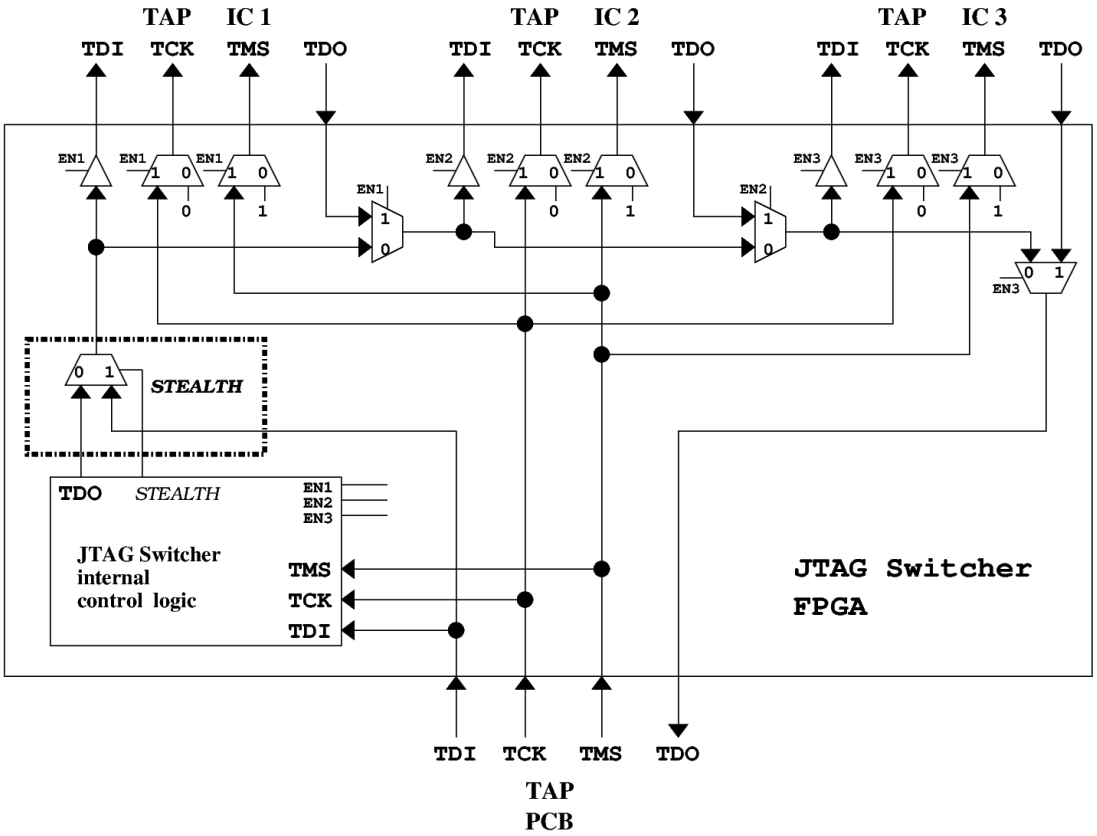
JTAG Switcher Stealth Mode

Stealth Mode is an advanced feature, for which suppport has to be enabled when the JTAG Switcher is implemented.

Stealth Mode allows the JTAG Switcher to completely hide itself from the JTAG chain.



To support this feature, the circuitry of the JTAG Switcher is slightly modified like this:



As depicted above: Once Stealth Mode is enabled, the TDI-TDO path of the JTAG Switcher TAP is bypassed. The effect is, that from the outside the JTAG Switcher TAP is not visible anymore.

Note: The JTAG Switcher TAP is still connected to the TCK/TMS/TDI signals of the PCB TAP. This means the JTAG Switcher TAP Controller will move through the states defined by JTAG and the JTAG Switcher TAP will still see all bits transmitted via TDI.

While Stealth Mode is active the Instruction Register of the JTAG Switcher TAP is forced to always hold the Instruction Register code for BYPASS (all ones). This means the JTAG Switcher will ignore all regular JTAG accesses done via DR or IR-Shift Sequences.

5.1 Stealth Mode activation

To activate stealth mode the global register **ControlA** is used.

Stealth Mode is *scheduled* for activation by writing `0x0001` to the global register **ControlA**. This corresponds to a "Set Bit" operation for bit 0 in bit group 0 (bits 15..12) (see **ControlA** register description).

To finally activate Stealth Mode you (as always) have to spend at least **8** consecutive TCK clock cycles in the "Run-Test/Idle" state.

Example Sequence:

- You start in a state where none of the JTAG slave TAPs is selected.
- You select one JTAG slave TAP, by using a "Set Bit" operation to the corresponding "Select Control Bit".
  You **do not** enter the Run-Test/Idle state yet.
- You write `0x0001` to the global register **ControlA**, to schedule activation of Stealth Mode.
- You drive the TAP Controller to the Run-Test/Idle state and spend at least **8** consecutive TCK clock cycles in Run-Test/Idle.

When the JTAG Switcher TAP Controller reaches the Run-Test/Idle state, the selected configuration will be activated, which means the JTAG Switcher will activate Stealth Mode and include the selected JTAG slave TAP into the JTAG Chain.

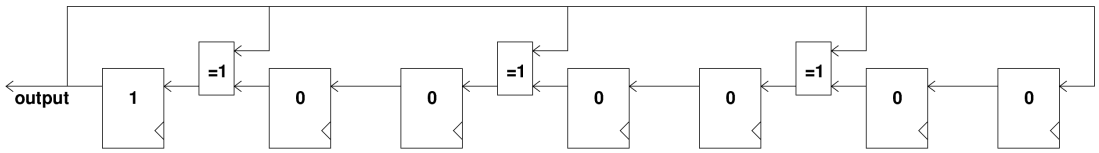
After this operation you should only detect the selected JTAG slave TAP on the JTAG chain. The JTAG Switcher TAP should be hidden.

5.2 Stealth Mode deactivation

Once Stealth Mode is activated, you cannot access the JTAG Switcher TAP via regular means, because the JTAG Switcher is hidden.

To get the JTAG Switcher back into the chain, the JTAG Switcher TAP listens to the TCK/TMS/TDI signals. To *schedule* the deactivation of Stealth Mode, you have to send a specific sequence of TDI bits, while the TAP Controller is in the Shift-IR state.

The sequence of TDI bits is defined via a 7 bit linear feedback shift register (LFSR), constructed in the following manner:



Here is a small program in C to output the TDI bit sequence as a sequence of bytes.

```
#include <stdint.h>
#include <stdio.h>
int main()
{
   uint32_t lfsr,byte;
   lfsr=0x40;byte=0;
   for(int i=0;i<128;i++) {
      byte=(byte>>1)|((lfsr&0x40)<<1);
      if ((i&0x7)==0x7)
         printf("0x%02x\n",byte);
      lfsr<<=1;
      if (lfsr&0x80)
         lfsr^=0xD5;
   }
   return 0;
}
```

The produced sequence of bytes is:

first							last
0xb7	0xcf	0x5f	0x52	0x57	0x39	0xa6	0x19
0xbd	0x13	0x61	0x2d	0x1e	0x64	0x43	0x81

Each byte has to be send in via TDI with least-significant-bit first. So the resulting bit sequence is:

```
1 1 1 0 1 1 0 1   1 1 1 1 0 0 1 1   1 1 1 1 1 0 1 0   0 1 0 0 1 0 1 0   ...
0xb7              0xcf              0x5f              0x52
```

The precise sequence of TDI bits needed is more flexible: To *schedule* deactivation of Stealth Mode you have to send in via TDI (while you are in the Shift-IR TAP Controller state):

- An arbitrary sequence of bits. (This might allow you to pacify whatever JTAG slave TAPs are on the chain.)
- One byte (8 bit) of 0 bits. (This ensures a mismatch between TDI and the most significant bit of the LFSR and in turn loads and keeps the LFSR at an initial value of `0x40`.)
- The LFSR sequence of bits described above. (Once the sequence has been sent the deactivation of Stealth Mode is unrevocably scheduled.)
- An arbitrary sequence of bits. (This allows you to load all Instruction Registers included in the JTAG chain with for example the BYPASS instruction code.)

To finally deactivate Stealth Mode you (as always) have to spend at least **8** consecutive TCK clock cycles in the "Run-Test/Idle" state.

When the JTAG Switcher TAP Controller reaches the "Run-Test/Idle" state, Stealth Mode will be de-activated and at the same time *all* JTAG slave TAPs are excluded from the JTAG chain.

After this operation only the JTAG Switcher itself should appear on the JTAG chain (all JTAG slave TAPs should be excluded). Additionally the JTAG Switcher Instruction Register still contains the BYPASS instruction register code.