



eSi-7540 Arbitrary Sample Rate Converter

1 Contents

1	Contents	2
2	Overview	3
3	Hardware Interface	4
3.1	Configuration	5
4	Software Interface	6
4.1	Register Map	6
4.2	Interrupts	7
5	Sample Rate Conversion Operation	8
5.1	Introduction	8
5.2	Interfacing	8
5.3	Operation	8
5.4	Polynomial fit to minimise the mean squared error over all delays	9
6	Performance graphs	11
6.1	N=2, M=2, MMSE	11
6.2	N=2, M=3, MMSE	11
6.3	N=3, M=2, MMSE	12
6.4	N=3, M=3, MMSE	12
6.5	N=4, M=2, MMSE	13
6.6	N=4, M=3, MMSE	13
6.7	N=2, M=3, Cubic Lagrange	14
6.8	References	14

2 Overview

The eSi-7540 core provides the control and data plane interfaces to an arbitrary sample rate converter. It supports the following features:

- Interpolates by irrational fractions
- Decimates by irrational fractions
- Can be used as part of a symbol timing feedback loop where the timing changes continuously.
- Suitable for all digital receivers using a free running clock
- Suitable for multi-standard wireless receivers
- Supplied with 7 parameterisable implementations
- ASIC or FPGA target
- Fully synchronous design
- AXI4-Streaming inputs and outputs
- APB configuration
- Verilog 2001

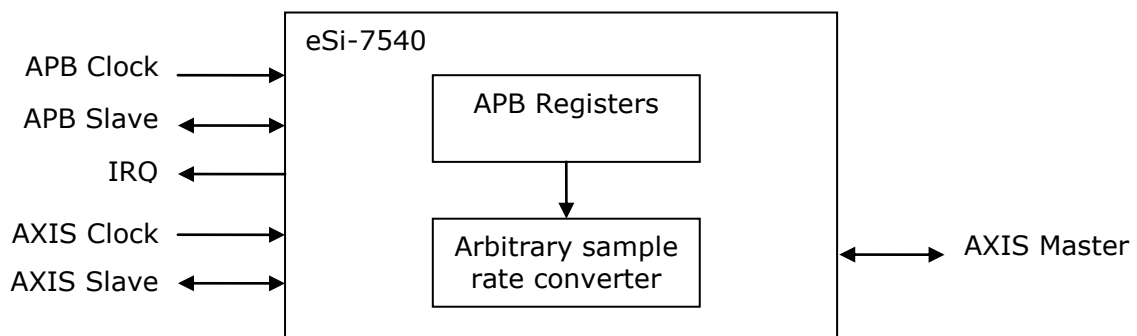


Figure 1: eSi-7540

The control logic handles the sequencing and control for handling multiple packets. This reduces the number of interrupts to the processor and significantly reduces the software overhead that would otherwise be spent on controlling the dataflow. The APB interface provides simple programmability to set resampling rate and control the IP.

3 Hardware Interface

Module Name	esi_resampler
HDL	Verilog 2001
Technology	Generic
Source Files	esi_resampler.v, resampler.v, resampler_apb.v, resampler_delay.v, resampler_filter.v, resampler_horner.v, resampler_mac.v, axis_adapter.v

Parameter	Range	Default	Description
apb_address_width	3-16	16	APB address bus width
apb_data_width	16/32	32	APB data bus width (same as BITS)
data_width	8-32	16	AXI4-Stream data width
rate_width	2-8	24	RATE register width
id_width	1-8	1	AXI4-Stream ID width

Table 1: Parameters

APB Port	Direction	Width	Description
pclk	Input	1	Clock
presetn	Input	1	Reset, active-low
paddr	Input	8	Address
psel	Input	1	Slave select
penable	Input	1	Enable
pwrite	Input	1	Write
pwdata	Input	BITS	Write data
pclk_cactive	Output	1	Clock active
pready	Output	1	Ready
prdata	Output	BITS	Read data
pslverr	Output	1	Slave error
interrupt_n	Output	1	Interrupt request, active-low

Table 2: APB I/O Ports

AXI4S Port	Direction	Width	Description
aclk	Input	1	Clock
aresetn	Input	1	Reset, active-low
s_tdata	Input	data_width	Slave data in
s_tid	Input	id_width	Slave id
s_tlast	Input	1	Slave last
s_tvalid	Input	1	Slave valid
s_tready	Output	1	Slave ready
m_tdata	Output	data_width	Master data
m_tid	Output	id_width	Master id
m_tlast	Output	1	Master last
m_tvalid	Output	1	Master valid
m_tready	Input	1	Master ready
aclk_cactive	Output	1	Clock active

Table 3: AXI4-Stream I/O Ports

For complete details of the APB and AXI4-Stream signals, please refer to the AMBA Protocol specifications available at <http://www.arm.com/products/solutions/AMBAHomePage.html>

The APB clock `pclk` and AXI clock `aclk` must be synchronous, but do not have to be the same frequency. This enables the APB interface to be run from a slower clock than the AXI interfaces.

The `aresetn` and `presetn` are used internally as active low synchronous resets. These reset signals may be asserted asynchronous to the clocks but deassertion must be synchronous after the rising edge of their respective clock.

The core may also be used without an APB interface by instantiating the file `resampler.v` as the top level and providing signals `trigger` and `rate`.

The deliverable contains a Quartus project to build a Stratix IV version of the chosen Resampler core. Simply open up the Quartus project `esi_resampler.qpf` and compile to get the top level. The following table gives some typical results using balanced optimization and 200MHz clock specified in the Synopsys Design Constraints File.

	ALUTs	Regs	M9K blocks	F_{max} (MHz)	Av SINAD	Pk SINAD
N=2, M=3, LAG	553	534	6	200	-32	-21
N=2, M=2, LS	512	403	4	200	-39	-31
N=2, M=3, LS	883	557	6	200	-41	-33
N=3, M=2, LS	621	502	4	200	-42	-32
N=4, M=2, LS	777	598	4	200	-42	-32
N=3, M=3, LS	1636	757	6	200	-57	-51
N=4, M=3, LS	2201	930	6	200	-65	-52

All F_{max} are worst case (i.e. for 85°C 1.2V slow corner). For higher speed optimization please contact EnSilica for custom effort, the current design is a trade off of speed and area.

3.1 Configuration

The available default configurations are shown below. Only one implementation should be chosen according to the system requirements. Based on that implementation all internal wordlengths are optimised to give the best output SINAD without excessive resource usage. The input/output data defaults to 16-bit. The time variable defaults to 24-bits, with 4 integer bits which gives resolution down below 1ppm, and the Input/Output rate can be between 0.0001 and 7.9999.

PARAMETER	Description	Recommended Range
SRC_CUBIC_LAGRANGE	Cubic Lagrange interpolator	Pick only one implementation
SRC_N_M_2_2	N=2, M=3, B=0.25, MMSE	
SRC_N_M_2_3	N=2, M=3, B=0.25, MMSE	
SRC_N_M_3_2	N=3, M=2, B=0.25, MMSE	
SRC_N_M_4_2	N=4, M=2, B=0.25, MMSE	
SRC_N_M_3_3	N=3, M=3, B=0.25, MMSE	
SRC_N_M_4_3	N=4, M=3, B=0.25, MMSE	
SRC_DATA_WL	Input/Output data wordlength	16
SRC_TIME_WL	Time (Rate) wordlength	24
SRC_TIME_IWL	Time (Rate) integer bits	4

4 Software Interface

4.1 Register Map

The software register map is given below

Register	Address offset	Access	Description
control	0x00	R/W	Control register
status	0x04	R/W	Status register
RATE	0x08	R/W	Rate register

Table 4: Register Map

4.1.1 Control Register

The control register contains a selection of flags that control the operation of the module.

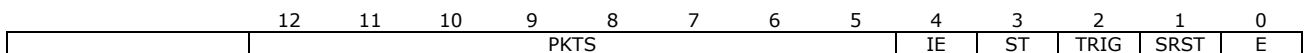


Figure 2: Format of the control register

Register	Values	Description
E	0 - Disabled 1 - Enabled	Enables the Sample Rate Converter. When disabled data will not be processed.
SRST	0 - Normal operation 1 - Synchronous reset	Synchronous reset of the Sample Rate Converter and AXI4 streaming interface
TRIG	1 - Trigger	Write 1 to trigger processing of PKTS packets. This field is self clearing. Sample Rate Conversion stops after PKTS packets are processed.
ST	0 - Packet based 1 - Continuous stream	Writing 0 to this register allow packet based control over the Sample Rate Converter using the TRIG and PKTS field. Writing 1 causes the Sample Rate converter to continuously process packets as soon as they become available.
IE	0 - Disable interrupts 1 - Enable interrupts	Writing 1 will generate an interrupt on interrupt_n once PKTS packets have been processed.
PKTS	1-255	Number of packets to process before stopping

Table 5: Fields of the control register

4.1.2 Status Register

The status register contains the interrupt bit. To clear a bit in the status register, write a 1 to it. Writing a 0 will leave it unchanged.

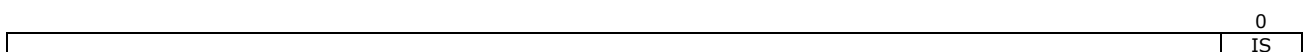


Figure 3: Format of the status register

Register	Values	Description
IS	0 - Interrupt not set 1 - Interrupt set	Interrupt status. This bit is set to 1 once PKTS packets have been processed. It is set independent of the IE field to allow software polling instead of hardware interrupt generation.

Table 6: Fields of the status register

4.1.3 RATE Register

The RATE register holds the unsigned sample rate conversion. The top 4 bits hold the integer part and the lower 20 bits hold the fractional part. This allow sample rate conversion from 0 to 7.9999. Other rates are possible through parameterisation of the IP - please contact EnSilica for your application requirements

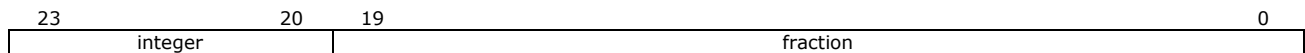


Figure 4: Format of the RATE register when rate_width=24

4.2 Interrupts

The `interrupt_n` signal will be raised when `PKTS` packets have been processed and the `IE` flag in the control register is set to 1. The interrupt status bit `IS` is set when `PKTS` packets have been processed. The `IS` bit is set independent of the `IE` flag setting. The `IS` bit and `interrupt_n` signal are both cleared by writing a 1 to the status register. The receive interrupt operation and `IS` flag are also active in streaming mode `ST=1`, but are not intended to be used in streaming mode.

5 Sample Rate Conversion Operation

5.1 Introduction

In some applications it is either inefficient or not possible to use an exact rate conversion. For example

- Two clocks may not be harmonically related by a simple rational fraction, e.g. $U/D = 2047/1023$, in which case a FIR based resampler would need to generate a large number of subfilters – in this example 2047.
- The exact conversion rate may not be known, or is continuously changing. This is encountered in wireless systems when the two clocks are remote from each other, one in the handset and one in the base-station. Even though the nominal conversion rate may be known, the actual rate depends on the frequency difference. In particular, for a base-station, the sampling instant cannot be optimized for all users, so the sampling clock is allowed to free-run and the optimum sample point is recovered by re-sampling the output of the matched filter or equalizer.
- The base station may be supporting multiple air interface standards. For instance a multi-standard base station may be based around a WCDMA sample rate of 7.68MHz, and this same receiver could be used to sample a PCN timeslot, which after adjacent channel filtering and decimation should reduce to the PCN symbol rate of 270.8333kHz. However since PCN is based on a 13MHz clock then the ratio between these clocks is 325/192.
- There may be a requirement to shift a signal in phase by a fractional amount without changing its rate as may be found in a timing error feedback system.

To implement sampling rate conversion for these applications, we need to resort to approximate rate conversion schemes. These introduce a certain amount of signal distortion, but provided the total distortion doesn't exceed the specification of the system then this can be acceptable. Note that distortion exists even in the exact rational fraction case because FIR polyphase filters cannot represent a pure time delay.

5.2 Interfacing

The slave AXI4-Stream input interface and master AXI4-Stream output interface obey a handshaking protocol using the `x_tvalid` and `x_tready` signals. The following description is taken from the AXI4-Stream specification v1.0. The `x_tvalid` and `x_tready` handshake determines when information is passed across the interface. A two-way flow control mechanism enables both the master and slave to control the rate at which the data and control information is transmitted across the interface. For a transfer to occur both the `x_tvalid` and `x_tready` signals must be asserted. Either `x_tvalid` or `x_tready` can be asserted first or both can be asserted in the same `aclk` cycle. A master is not permitted to wait until `m_tready` is asserted before asserting `m_tvalid`. Once `x_tvalid` is asserted it must remain asserted until the handshake occurs. A slave is permitted to wait for `s_tvalid` to be asserted before asserting the corresponding `s_tready`. If a slave asserts `s_tready`, it is permitted to deassert `s_tready` before `s_tvalid` is asserted.

The final sample of a packet must have `s_tlast` asserted on the slave handshake. A packet can have an ID associated with it through the AXI ID signal. This allows the system to pass on sidechannel information with each sample.

5.3 Operation

The sample rate converter has to operate on a time axis completely defined by the input sample rate, and generate output samples asynchronous to these sampling instants and

determined by the `RATE`. If the time starts at zero then the next sample that should be output is at time `RATE` on the input sample time axis. So `RATE` is added to the current time, and if the new output sample time is < 1.0 then an output sample is generated and the `RATE` is added again, otherwise an input sample is read and the output sample time is decremented by 1.0. This process continues repeatedly. When `RATE` is less than 1.0 then the core is performing interpolation, when it's greater than 1.0 then it is performing decimation, and when it is equal to 1.0 it's performing a phase shift.

Phase interpolation is performed by polynomial FIR interpolation between samples to approximate the exact interpolated value. The well-known Farrow architecture is used as the hardware implementation for maximum efficiency. This architecture is also optimal for implementing Lagrange FIR interpolation between samples, and an example of cubic Lagrange is provided. Interpolation of order 2 or 3 can be specified, where the order is related to the interpolation type as follows

- Order 1, $M=1$, provides linear interpolation
- Order 2, $M=2$, provides quadratic interpolation
- Order 3, $M=3$, provides cubic interpolation

Order 0 or nearest neighbor, and order 1 which corresponds to linear interpolation are not supported, since these are trivial. In all cases the normalized bandwidth BT s of the input signal and the order of interpolation provide bounds on the performance of this core. The normalized (one-sided) bandwidth is the fraction of the sampling frequency over which the signal integrity should be preserved. The graphs in a later section give some idea of the performance, and the SINAD for each supported design is given in the Synthesis section.

5.4 Polynomial fit to minimise the mean squared error over all delays

The theory behind the resampler can be simply stated. First a time-limited but continuous interpolation impulse response is generated according to an optimization criteria. This criteria is typically that it passes a signal in bandwidth B without amplitude or phase distortion and is unconstrained outside the passband. If the input sample rate is T_s then the family of interpolation functions that satisfy this criteria are called Nyquist Type 1 pulses and their impulse response crosses zero at lags of $\pm nT_s$.

In a communications system where the oscillator is uncorrected the probability density of the delay is likely to be uniformly distributed between 0 and 1 so it makes sense to optimise for minimum mean squared error (MMSE) from an ideal delay element jointly over the bandwidth B and over the continuum of fractional delays from 0 to 1 sample period. Outside of this bandwidth the optimisation is unconstrained. In mathematical terms this can be expressed as

$$\bar{\sigma}_e^2 = \frac{1}{2B} \int_0^1 \int_{-B}^B |e^{j2\pi f T_s \mu} - H_{opt}(e^{j2\pi f T_s}, \mu)|^2 df d\mu \rightarrow \min$$

In practice the optimum frequency response is calculated for a large number of small fractional delays between 0 and 1.

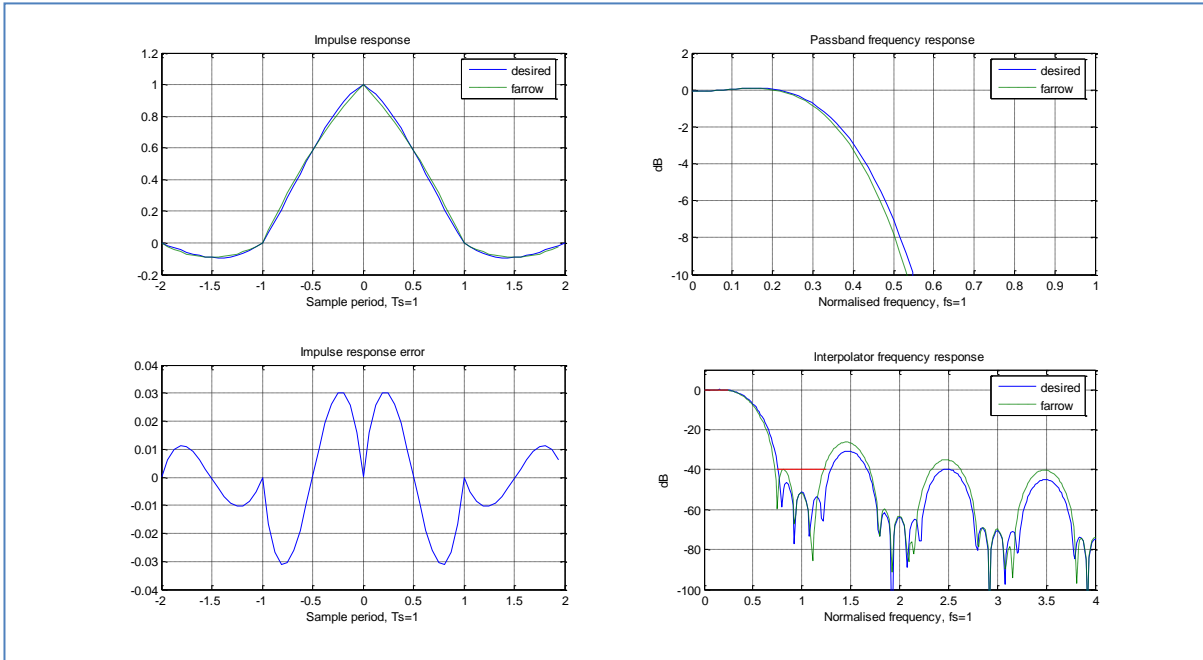
Next we approximate the impulse response in each T_s period by a low order polynomial. Although in principle each T_s period could use a different order polynomial the simple case is where a common order is applied. Again a criteria needs to be chosen for the polynomial fit, for example least squares or equiripple. In the case of a least squares fit the polynomial must pass through zero at the lags $\pm nT_s$ forming a continuous error function.

In the supported configurations we have optimised for $BT_s=0.25$, which means that the one-sided bandwidth should be less than half the Nyquist frequency. For instance in a single carrier system with symbol period T and excess bandwidth α , then $B \geq (1+\alpha)/T$, which reads that the passband bandwidth should be more than the pulse bandwidth including any excess.

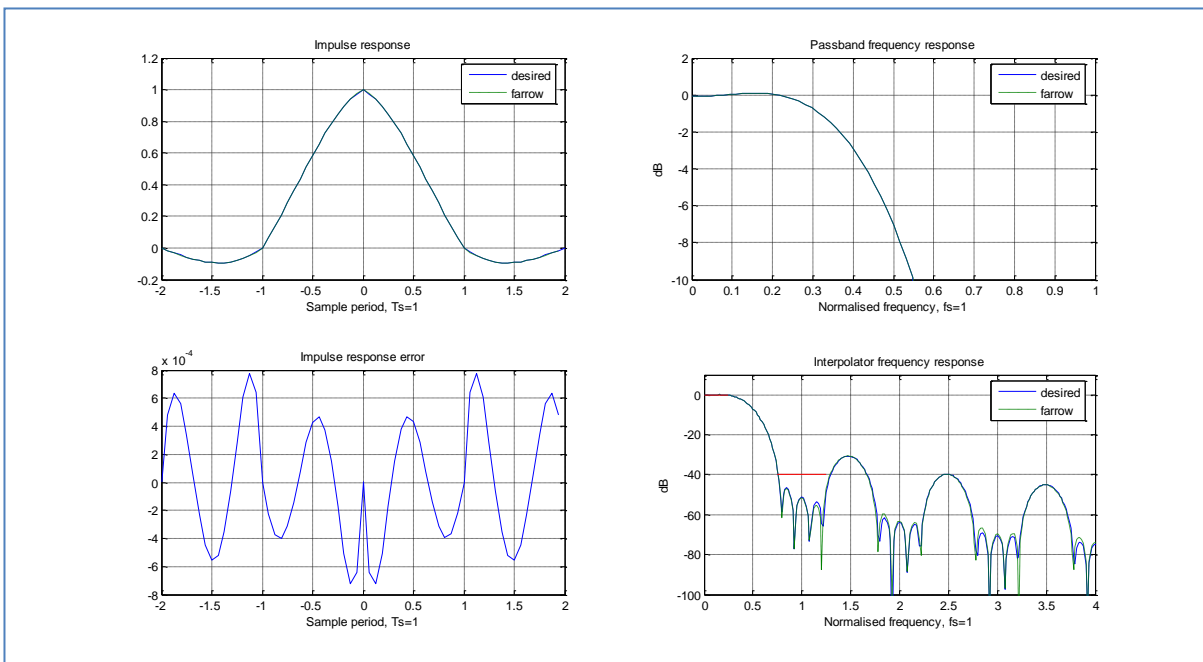
6 Performance graphs

In the following graphs we show the ideal continuous time impulse response, and the polynomial fit to it in each sample period. The error in the polynomial fit is also shown. In the frequency domain we show the passband and stopband of the ideal and approximated continuous time filter. A common reference in red helps to compare the different configurations - it corresponds to a 0.1dB passband droop between 0 and B, and in the stopband it is -40dB between 1-B and 1+B.

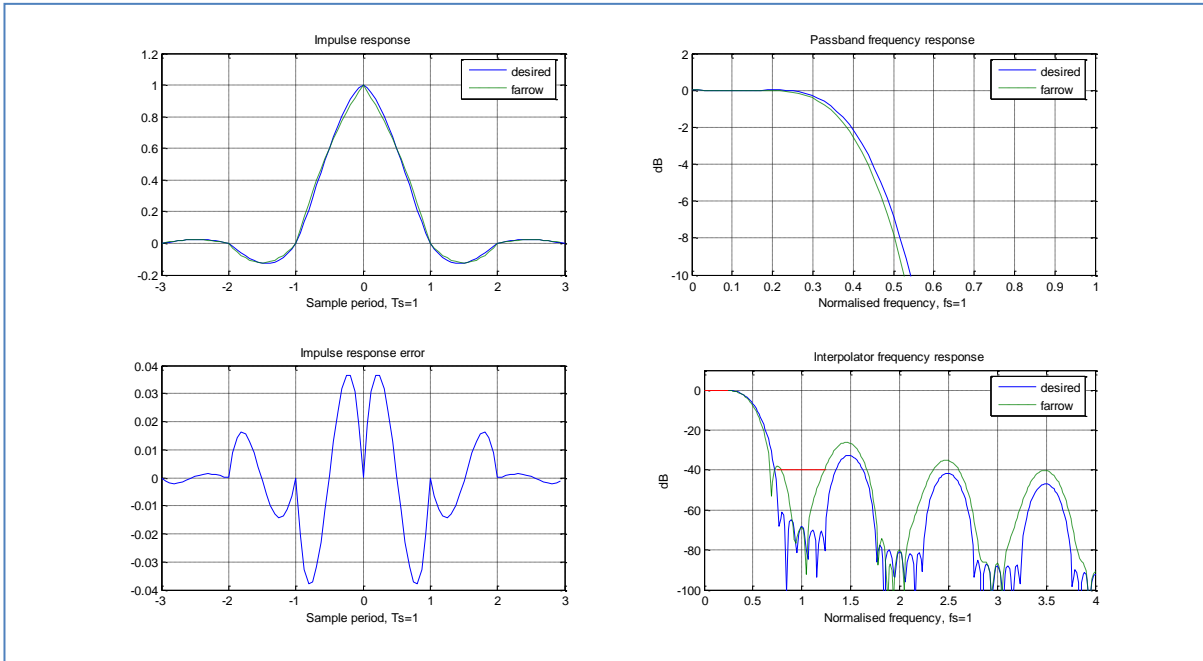
6.1 N=2, M=2, MMSE



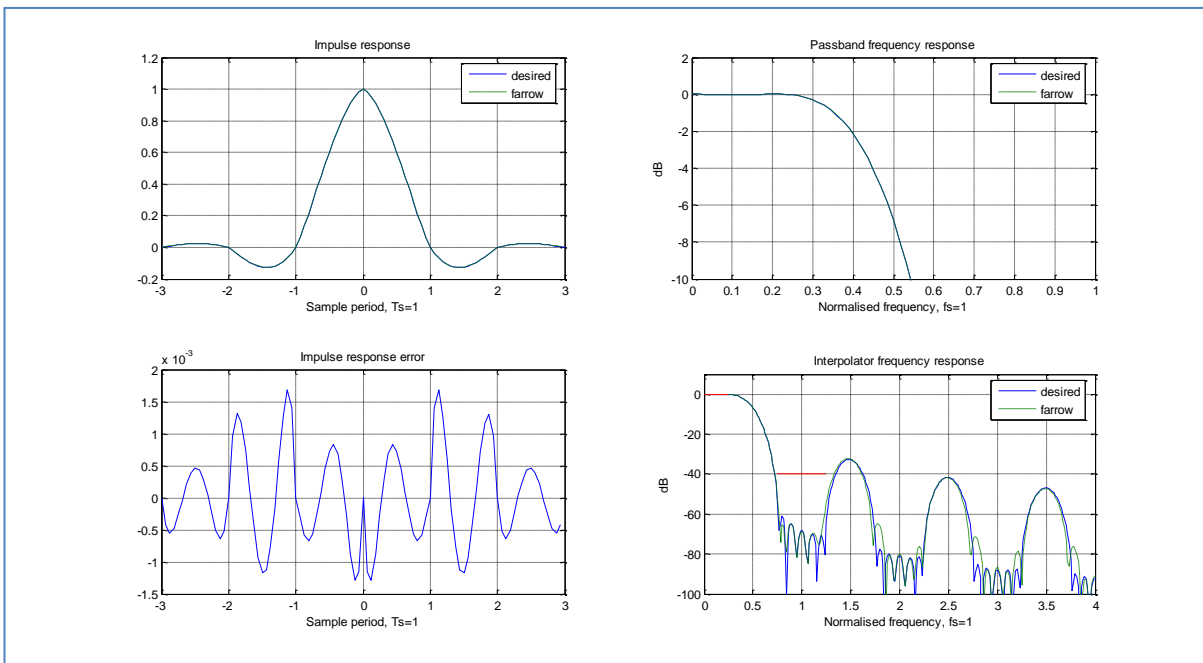
6.2 N=2, M=3, MMSE



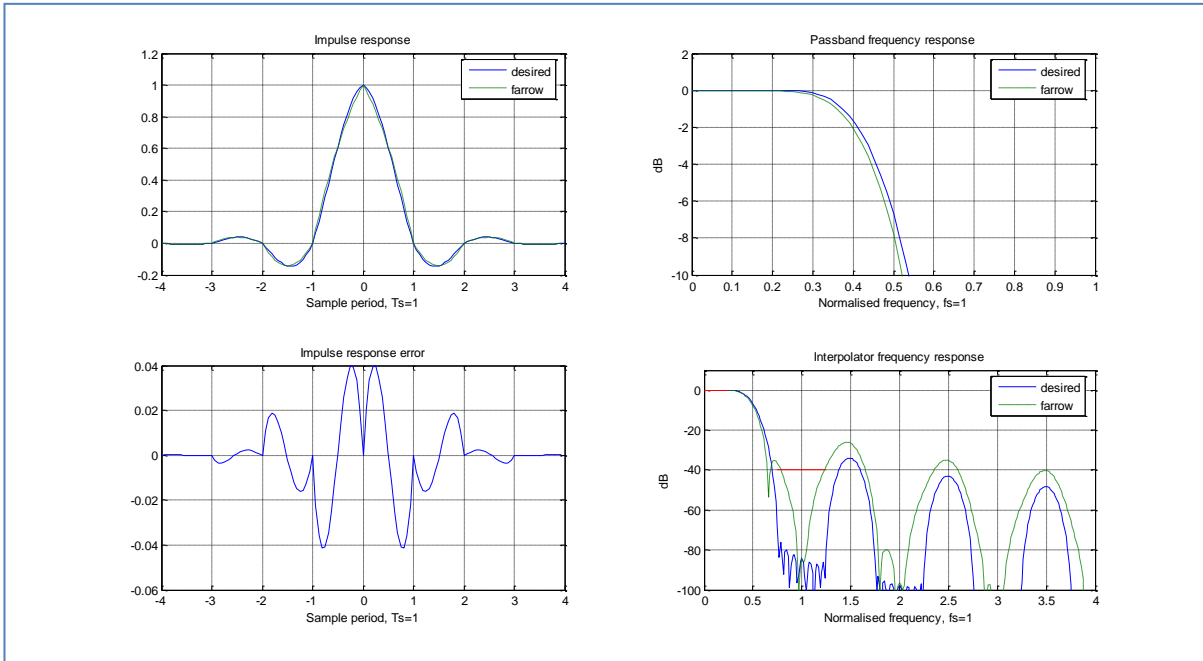
6.3 N=3, M=2, MMSE



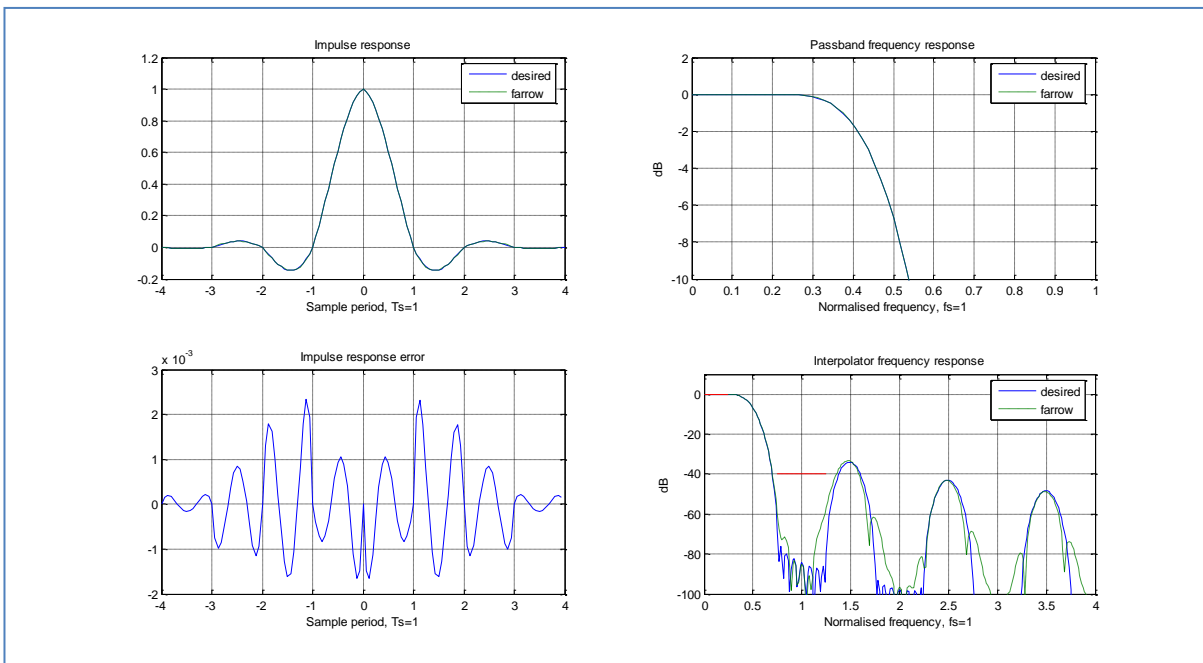
6.4 N=3, M=3, MMSE



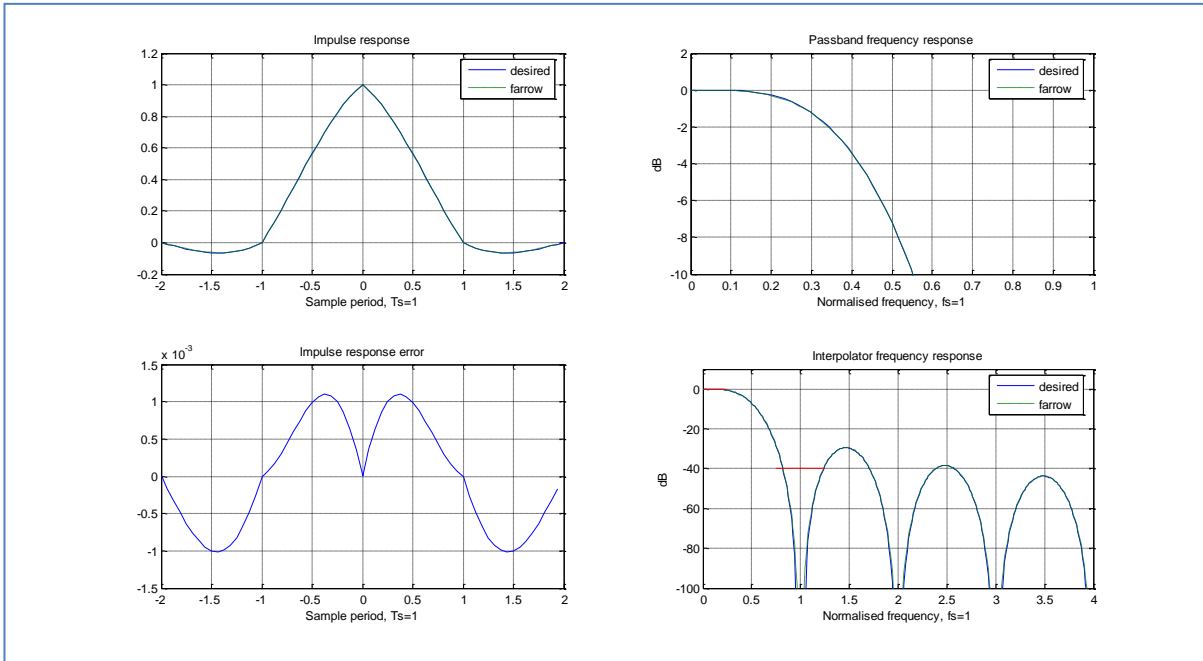
6.5 N=4, M=2, MMSE



6.6 N=4, M=3, MMSE



6.7 N=2, M=3, Cubic Lagrange



6.8 References

[1] Digital Communication Receivers. Heinrich Meyr, Marc Moeneclaey, Stefan Fechtel. John Wiley and Sons Publishing 1998.