THE SYMMETRIC RUDIN-SHAPIRO TRANSFORM - AN EASY, STABLE, AND FAST CONSTRUCTION OF MULTIPLE ORTHOGONAL SPREAD SPECTRUM SIGNALS

Anders la Cour-Harbo

Aalborg University
Department of Control Engineering
Fredrik Bajers Vej 7C
9220 Aalborg East, Denmark
alc@control.auc.dk

ABSTRACT

A method for constructing spread spectrum sequences is presented. The method is based on a linear, orthogonal, symmetric transform, the Rudin-Shapiro transform (RST), which is in many respects quite similar to the Haar wavelet packet transform. The RST provides the means for generating large sets of spread spectrum signals. This presentation provides a simple definition of the symmetric RST that leads to a fast $N\log(N)$ and numerically stable implementation of the transform.

1. INTRODUCTION

The need for spread spectrum signals in various forms of digital communication has increased significantly during the past years. The purposes of spreading the spectrum are several. In some cases the purpose is to provide multiple channels in some particular domain. This is the case for CDMA, see for instance Viterbi [1], and OFDM. In other cases the purpose is encryption or disguising of signals, anti-jamming, and multi-path systems. There exists a vast amount of literature on spread spectrum methods. A good place to start is the nice tutorial by Viterbi [2], and the more comprehensive book by Dixon [3].

The presentation in this paper focuses on the mathematical aspects of a method, given as a linear transform, for generating (and post-processing) spread spectrum signals. It is based on a discovery by Shapiro in 1951 [4] and Rudin in 1959 [5] called the Rudin-Shapiro polynomials.

1.1. Rudin-Shapiro Polynomials

The Rudin-Shapiro polynomials are defined recursively as

$$P_{n+1}(\xi) = P_n(\xi) + e^{i2\pi 2^n\xi}\,Q_n(\xi), \quad P_0 = 1, \tag{1}$$

$$Q_{n+1}(\xi) = P_n(\xi) - e^{i2\pi 2^n\xi}\,Q_n(\xi), \quad Q_0 = 1, \tag{2}$$

for $\xi \in [0;1)$. It follows that each P_{n+1} has twice as many terms as P_n, and therefore that the polynomials are generated by a simple 'append' rule. We will refer to the coefficients of the RS polynomial as RS sequences. The ingenuity of these polynomials is the combination of fixed sized coefficients and the alternating signs in the recursive construction of P and Q. According to Parseval's theorem the former property gives $\|P_n\|_2^2 = 2^n$ (as each P is the Fourier transform of a ±1 sequence), while the latter property gives

$$|P_{n+1}(\xi)|^2 + |Q_{n+1}(\xi)|^2 = 2|P_n(\xi)|^2 + 2|Q_n(\xi)|^2 = 2^{n+2}, \tag{3}$$

since $|e^{i2\pi 2^n\xi}| = 1$. This leads to

$$|P_n(\xi)| \le \sqrt{2}\cdot 2^{n/2}, \quad \forall \xi \in [0;1)\,,$$

a uniform upper bound for P_n. Now, combining the two properties yields the squared crest factor

$$\frac{\|P_n\|_\infty^2}{\|P_n\|_2^2} \le 2\,. \tag{4}$$

This means that $\max|P_n(\xi)|^2$ is equal to or less than two times the energy of P. This guarantees the polynomial to be somewhat flat. Two examples of $|P_n|^2$ are shown in Fig. 1.

It is important to realize that the term 'flat' should be understood as 'not excessively far from a constant function', but not necessarily 'close to a constant function'. Usually the term flat refers to a uniform upper bound for the peak-to-power ratio, that is, the bound is independent of the length of the sequence. This upper bound is thus 2 for the Rudin-Shapiro polynomials. In most literature the coefficient sequence of a flat polynomial is called a spread spectrum sequence. To demonstrate this concept the two lower most graphs in Fig. 1 show that neither the well-known (an often used in applications) square wave nor a random ±1 sequence can be considered flat.

1.2. The Rudin-Shapiro Transform

An interesting property of the RS sequences generated according to (1) and (2) is that the P and Q coefficient sequences are orthogonal. This is immediately evident from the append rule which governs the construction of the polynomials. It is also worth noting that interchanging the $+$ and $-$ in (1) and (2) would still produce sequences with the previously presented properties. In fact, arbitrarily interchanges of the signs in each recursive step does not affect the properties of the constructed sequences.

This work is in part supported by the Danish Technical Science Foundation (STVF) Grant no. 9701481.

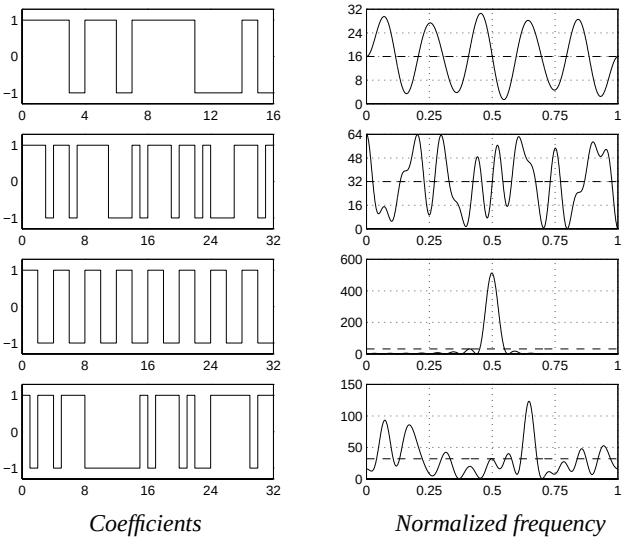

Fig. 1. The coefficients (left) and modulo squared (right) of the Rudin-Shapiro polynomials P_4 (first graph) and P_5 (second graph). Below the coefficients and modulo of the Fourier transform of a square wave (third graph) and a random sequence (fourth graph). The horizontal dashed line is the energy of the signal.

An elegant construction achieving all combinations of sign changes is found in Benke [6] (Byrnes [7, 8] gives a similar construction). In short,

$$\begin{bmatrix} P_{n+1,\epsilon}(\xi) \\ Q_{n+1,\epsilon}(\xi) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{\epsilon_n} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{i2\pi 2^n \xi} \end{bmatrix} \begin{bmatrix} P_{n,\epsilon}(\xi) \\ Q_{n,\epsilon}(\xi) \end{bmatrix}, \tag{5}$$

where $\epsilon_n \in \{0, 1\}$ is chosen in each step. A total of 2^n different P polynomials are possible after n steps. Thus, two P polynomials with two coefficients each are obtained after one step, four P polynomials with four coefficients each are obtained after two steps, and so on. The RST is then simply defined (up to normalization) as the matrix where the rows are the polynomial coefficients. This immediately gives a linear, square, orthogonal, Hadamard transform where the corresponding basis consists of spread spectrum sequences. Thus, applying the RST decomposes a signal into a basis of elements with a spread spectrum property. This is in some sense the opposite of a Fourier transform, which is a decomposition into a narrow spectrum basis. The RST is orthogonal and thus energy preserving, and the Hadamard property (equal amplitude of all the entries in the transform) makes the transform numerical stable. In general, it is an appealing transform for design and analysis of spread spectrum signals. However, in matrix form the transform is an $O(N^2)$ operation, and it is often preferable, if not crucial, to have an $O(N \log N)$ implementation, especially for real time applications. Such an implementation is demonstrated in Section 3.

While the rows of the presented matrices do have a low crest factor, this is not the case for the columns, which exhibits a Walsh-like structure rather than spread spectrum structure. Although this does not present a problem in most applications, it is indeed possible to obtain a spread spectrum property in the columns as well. In fact, it is possible with a slight alteration of the definition of the RS polynomials to obtain a symmetric transform.

2. SYMMETRIC RUDIN-SHAPIRO TRANSFORM

The idea for making the RST symmetric is communicated in Byrnes et al. [8]. There the polynomials are defined by a modification of the previously presented definition in (1) and (2). The following equations have been rewritten compared to [8], to suit the construction of a transform (most significantly, Byrnes have discarded the Q polynomials in favor of a more advanced indexing of the P polynomials). The symmetric RST is derived from the following equations.

$$\begin{aligned}
P_{j+1,4m}(\xi) &= P_{j,2m}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m+1}(\xi), \\
P_{j+1,4m+1}(\xi) &= P_{j,2m}(\xi) - e^{i2\pi 2^j \xi} Q_{j,2m}(\xi), \\
P_{j+1,4m+2}(\xi) &= P_{j,2m+1}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m+1}(\xi), \\
P_{j+1,4m+3}(\xi) &= -P_{j,2m+1}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m+1}(\xi), \\
Q_{j+1,4m}(\xi) &= P_{j,2m}(\xi) - e^{i2\pi 2^j \xi} Q_{j,2m}(\xi), \\
Q_{j+1,4m+1}(\xi) &= P_{j,2m}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m}(\xi), \\
Q_{j+1,4m+2}(\xi) &= -P_{j,2m+1}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m+1}(\xi), \\
Q_{j+1,4m+3}(\xi) &= P_{j,2m+1}(\xi) + e^{i2\pi 2^j \xi} Q_{j,2m+1}(\xi),
\end{aligned} \tag{6}$$

with

$$P_{1,0} = Q_{1,1} = 1 + e^{i2\pi\xi} \quad \text{and} \quad P_{1,1} = Q_{1,0} = 1 - e^{i2\pi\xi},$$

and for $j \geq 1$ and $m = 0, \ldots, 2^{j-1} - 1$. Note that P and Q in (6) are equal to the previous definition in (1) and (2) except for some changes of signs. The properties derived in the previous sections therefore still apply. As before it is possible to obtain two P polynomials with two terms, four with four terms, and so on. Listing the coefficient sequences in a matrix in their natural order yields the following for $j = 1, 2, 3$.

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 1 & -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}.$$

These resulting transforms can all be obtained from the non-symmetric RST of the same size simply by appropriate row permutation and change of row signs. Thus, all rows (and columns) in the symmetric matrices are mutually orthogonal, and the 'symmetric' RS sequences of length 2^J therefore also constitute an orthogonal basis of $\mathbb{R}^{2^J}$ while preserving the spread spectrum property demonstrated in the previous section.

2.1. Deriving the Symmetric Transform

Although it is possible to permute the non-symmetric RST to obtain the symmetric RST it is advantageous to construct the symmetric RST from the equations (6) since this provides the building blocks for a fast implementation.

The equations (6) can be written more compactly as

$$P_{j+1,m}(\xi) = (-1)^{m_1 m_2} P_{j,\lfloor m/2 \rfloor}(\xi) + (-1)^{m_1(m_2+1)} e^{i2\pi 2^j \xi} Q_{j,\lfloor m/2 \rfloor}(\xi), \quad (7)$$

$$Q_{j+1,m}(\xi) = (-1)^{(m_1+1)m_2} P_{j,\lfloor m/2 \rfloor}(\xi) + (-1)^{(m_1+1)(m_2+1)} e^{i2\pi 2^j \xi} Q_{j,\lfloor m/2 \rfloor}(\xi), \quad (8)$$

where m_1 and m_2 are the two least significant digits of the binary representation of m, and $\lfloor m/2 \rfloor$ means the biggest integer less than or equal to $m/2$. Rewriting to the obvious matrix form yields

$$\begin{bmatrix} P_{j+1,m}(\xi) \\ Q_{j+1,m}(\xi) \end{bmatrix} = \begin{bmatrix} (-1)^{m_1 m_2} & (-1)^{m_1(m_2+1)} \\ (-1)^{(m_1+1)m_2} & (-1)^{(m_1+1)(m_2+1)} \end{bmatrix} \times \begin{bmatrix} P_{j,\lfloor m/2 \rfloor}(\xi) \\ e^{i2\pi 2^j \xi} Q_{j,\lfloor m/2 \rfloor}(\xi) \end{bmatrix}. \quad (9)$$

This latter form of the RS equations shows the core of the transform; the 2×2 matrix. This turns out to be the 'secret' of the fast implementation.

The following definition of the symmetric RST exploits the existence of the 2×2 matrix to construct the RST as a factorization of the symmetric $2^J \times 2^J$ RST matrix. Each of the J matrix factors can then easily be applied with linear complexity.

Definition 1 *Define the mapping* $\mathbf{P}_{j,m} : \mathbb{R}^{2^j} \mapsto \mathbb{R}^{2^j}$, $j \geq 1$, *as*

$$\begin{bmatrix} y_k \\ y_{k+2^{j-1}} \end{bmatrix} = \frac{(-1)^{mk}}{\sqrt{2}} \begin{bmatrix} 1 & (-1)^k \\ (-1)^m & -(-1)^{k+m} \end{bmatrix} \begin{bmatrix} x_{2k} \\ x_{2k+1} \end{bmatrix} \quad (10)$$

for $k = 0, \ldots, 2^{j-1} - 1$ *when mapping* $\mathbf{x}$ *to* $\mathbf{y}$. *Define the* J *matrix factors*

$$\mathbf{P}_j^{(J)} \equiv \begin{bmatrix} \mathbf{P}_{j,0} & & 0 \\ & \ddots & \\ 0 & & \mathbf{P}_{j,2^{J-j}-1} \end{bmatrix}, \quad (11)$$

and finally defined the RST $\mathbf{P}^{(J)}$ *as*

$$\mathbf{P}^{(J)} \equiv \prod_{j=1}^{J} \mathbf{P}_j^{(J)}. \quad (12)$$

Note that (10) is the inverse of the transform proposed in (9). The 2×2 matrix in (10) aside, it is not immediately obvious neither how this definition is linked to (6), nor that it defines a symmetric transform. However, a fairly straightforward proof shows that the rows of $\mathbf{P}^{(J)}$ are the coefficients of the polynomials defined in (6), see la Cour-Harbo [9], and thus that the defined transform does indeed possess the properties derived above for the Rudin-Shapiro transform.

3. FAST IMPLEMENTATION

3.1. Factorization into Linear Steps

The definition of the RST given in Definition 1 is based on the recursive construction process of RS polynomials. When writing this process in matrix form the 2×2 matrix in (10) emerges along with the $2^J \times 2^J$ matrix in (11). The combination of these two matrices is the key to a fast implementation. The large matrix factors provides a factorization of the RST matrix, and the small matrix gives a simple and easy $O(N)$ implementation of these matrix factors. The principle is here demonstrated with a size 8×8 transform, but easily applies to all size 2^J RSTs.

The first factor to be applied in the 8×8 case is $\mathbf{P}_3^{(3)} = \mathbf{P}_{3,0}$. That is,

$$\mathbf{P}_3^{(3)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

This is equivalent to a Haar wavelet transform except that the filter taps are changing during filtering. The result is two parts of length 4. In the Haar case the two parts can be identified as a low and high pass part, respectively, while in the RST case the constant change of filter taps results in two parts containing a mix of frequencies. The splitting into two signal parts is also illustrated in Fig. 2 by the first (top) set of arrows. The next step in the transform is

$$\mathbf{P}_2^{(3)} = \begin{bmatrix} \mathbf{P}_{2,0} & \\ & \mathbf{P}_{2,1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & & & & \\ 0 & 0 & 1 & -1 & & & & \\ 1 & -1 & 0 & 0 & & & & \\ 0 & 0 & 1 & 1 & & & & \\ & & & & 1 & 1 & 0 & 0 \\ & & & & 0 & 0 & -1 & 1 \\ & & & & -1 & 1 & 0 & 0 \\ & & & & 0 & 0 & 1 & 1 \end{bmatrix},$$

i.e. the exact same procedure is repeat (independently) on each of the two signal parts. Notice that $m = 0$ when transforming the left part and $m = 1$ when transforming the right part of the signal. The m makes the transform symmetric in the sense that $m = 0$ throughout the transform steps would produce the non-symmetric RST. This second step is shown as the second set of eight arrows (from the top) in Fig. 2. The final step is

$$\mathbf{P}_1^{(3)} = \begin{bmatrix} \mathbf{P}_{1,0} & & & \\ & \mathbf{P}_{1,1} & & \\ & & \mathbf{P}_{1,2} & \\ & & & \mathbf{P}_{1,3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & -1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & -1 & 1 \end{bmatrix}.$$

The factorization means that the RST can be applied in J steps by multiplying a signal with all of the $\mathbf{P}_j^{(J)}$ matrices (in the right order). The mapping given in (10) shows how to reduce each multiplication to an $O(N)$ filtering process. For any choice of m and

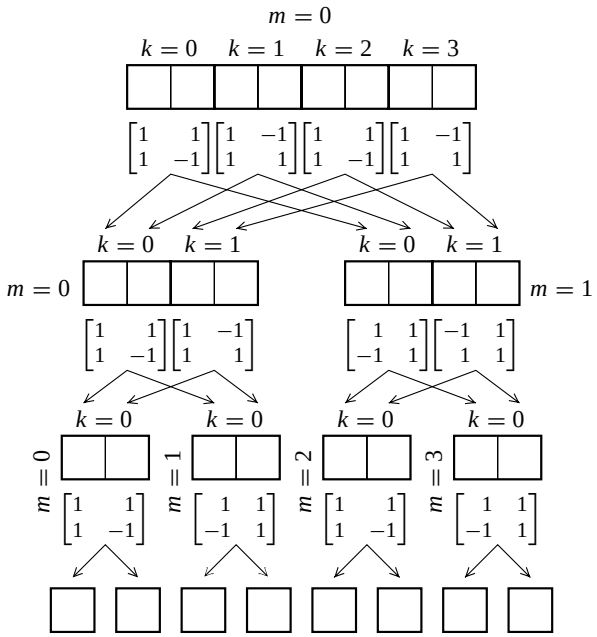

Fig. 2. This figure shows how the value of the variables change in the fast implementation of a symmetric RST. Here applied to a vector in $\mathbb{R}^8$.

k the 2×2 matrix contains 3 times $+1$ and one -1. Consequently, the output of the mapping is merely a series of sums and differences of sample pairs. A division by $\sqrt{2}$ should be applied to every sum/difference, but since the mapping is linear this scaling can be applied as division by 2 for every other step in the transform.

It is not apparent from this visualization of the fast implementation that it is its own inverse, i.e. if the resulting signal at the bottom is placed at the top, the new output is actually the original signal. But this is indeed the case since the transform is symmetric.

3.2. Stability and implementability

Applying a linear transform to a signal is basically a set of inner products with the row vectors of the transform matrix. In the case of the RST these vectors are ± 1's only, and consequently the RST is numerically very stable as all signal samples are weighted equally. This property is preserved in the fast implementation where each transform step also consists of ± 1's only. The fact that each intermediate sample depends on only two other samples makes the fast implementation even more stable than the matrix multiplication implementation. The normalization by 2 in every other transform step possess only negligible problems in the vast majority of applications.

The actual implementation of the RST can be accomplished by a regular filtering process divided into steps for even and odd k and m. In this way it is possible to avoid the computational demanding powers of (-1) in (10). Pseudo code for doing this is shown in Fig. 3. Note that the normalization is not included in this code.

The close relation to the Haar wavelet packet transform (actually, choosing $k = 0$ and $m = 0$ throughout the transform step yields exactly the full-scale Haar wavelet packet transform) provides another interesting property; instead of doing all the steps in the RST one can choose do only some of the steps and thereby

```
for j = 0 to log2(N)-1
  T = N/2^(j+1)
  for m = 0 to 2^j-1 step 2
    for k = m·T to (m+1)·T-1 step 2
      y[k]     = x[2k]   + x[2k+1]   } k even
      y[k+T]   = x[2k]   - x[2k+1]   }
      y[k+1]   = x[2k+2] - x[2k+3]   } k odd
      y[k+1+T] = x[2k+2] + x[2k+3]   }   } m even
    end for

    for k = (m+1)·T to (m+2)·T-1 step 2
      y[k]     = x[2k+1] + x[2k]     } k even
      y[k+T]   = x[2k+1] - x[2k]     }
      y[k+1]   = x[2k+3] - x[2k+2]   } k odd
      y[k+1+T] = x[2k+3] + x[2k+2]   }   } m odd
    end for
  end for
  x = y
end for
```

Fig. 3. Pseudo code for the RST. Note that to perform the first step ($j = 0$) in the transform only the 'm even' part should be used, and for the last step only 'k even' should be used.

obtain a different decomposition of the transformed signal. This is equivalent to selecting a particular basis in the Haar wavelet packet decomposition for representing the signal. Consequently, much of the theory regarding wavelet bases for $\mathbb{R}^N$ applies, and it becomes easy to construct and handle huge sets of spread spectrum signals, since it can be done within the framework of wavelet bases.

4. REFERENCES

[1] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*, Addison-Wesley, Reading, MA, 1995.

[2] A. J. Viterbi, "Spread spectrum communications: myths and realities," *IEEE Communications Magazine*, vol. 40, no. 5, pp. 34 – 41, May 2002, 50th Anniversary Commemorative Issue (first published in IEEE Communications Magazine, May 1979).

[3] R. C. Dixon, *Spread Spectrum Systems with Commercial Applications*, John Wiley & Sons, New York, 1994.

[4] H. S. Shapiro, "Extremal problems for polynomials and power series," M.S. thesis, Massachusetts Institute of Technology, may 1951.

[5] W. Rudin, "Some theorems on Fourier coefficients," *Proc. Amer. Math. Soc.*, vol. 10, pp. 855–859, 1959.

[6] G. Benke, "Generalized Rudin-Shapiro systems," *J. Fourier Anal. Appl.*, vol. 1, no. 1, pp. 87–101, 1994.

[7] J. S. Byrnes, "Quadrature Mirror Filters, Low Crest Factor Arrays, Functions Achieving Optimal Uncertainty Principle Bounds, and Complete Orthonormal Sequences – A Unified Approach," *App. and Comp. Harm. Anal.*, vol. 1, pp. 261–266, 1994.

[8] J. S. Byrnes, I. Gertner, G. Ostheimer, and M.A. Ramalho, "Discrete one dimensional signal processing apparatus and method using energy spreading coding," June 15, 1999, U.S. Patent no. 5,913,186.

[9] A. la Cour-Harbo, *Robust and Low-Cost Active Sensors by means of Signal Processing Algorithms*, Ph.D. thesis, Aalborg University, August 2002.