--encoder
library IEEE;
use IEEE.std_logic_1164.all;

entity encoder is
  --this entity encodes a 8 bit pattern to a 22 bit pattern using a (22,8,6)
  --linear block code
  port(clk,reset :in std_logic;enc_in :in std_logic_vector(7 downto 0);
      enc_out :out std_logic_vector(21 downto 0)); end;

ARCHITECTURE behave of encoder is
  --description of the entity encoder
BEGIN
  Process (clk, reset) -- process to encode the 8-bit input
  function add(a,b,c: std_logic) return std_logic is
    begin return a xor b xor c;
  end;
  begin
    if (reset = '1') then
      enc_out <= "0000000000000000000000"; -- reset = 1 makes it to remain at 0 state
    elsif (rising_edge(clk)) then
      --for every rising edge of the clock it encodes the 8 bit input enc_in to enc_out
      enc_out(0) <= enc_in(0);
      enc_out(1) <= enc_in(0);
      enc_out(2) <= enc_in(1);
      enc_out(3) <= enc_in(0) xor enc_in(1);
      enc_out(4) <= enc_in(0) xor enc_in(2);
      enc_out(5) <= enc_in(1) xor enc_in(2);
      enc_out(6) <= add(enc_in(0),enc_in(1),enc_in(3));
      enc_out(7) <= add(enc_in(0),enc_in(2),enc_in(3));
      enc_out(8) <= add(enc_in(1),enc_in(2),enc_in(4));
      enc_out(9) <= add(enc_in(1),enc_in(3),enc_in(4));
      enc_out(10) <= add(enc_in(2),enc_in(3),enc_in(5));
      enc_out(11) <= add(enc_in(2),enc_in(4),enc_in(5));
      enc_out(12) <= add(enc_in(3),enc_in(4),enc_in(6));
      enc_out(13) <= add(enc_in(3),enc_in(5),enc_in(6));
      enc_out(14) <= add(enc_in(4),enc_in(5),enc_in(7));
      enc_out(15) <= add(enc_in(4),enc_in(6),enc_in(7));
      enc_out(16) <= enc_in(5) xor enc_in(6);
      enc_out(17) <= enc_in(5) xor enc_in(7);
enc_out(18) <= enc_in(6) xor enc_in(7);
enc_out(19) <= enc_in(6);
enc_out(20) <= enc_in(7);
enc_out(21) <= enc_in(7);
end if;
end process;
end behave;

DECODER

--decoder
library IEEE;
use IEEE.std_logic_1164.all;--for std_logic, std_logic_vector
use IEEE.std_logic_arith.all;
--library SYNOPSYS;
--use SYNOPSYS.bv_arithmetic.all;

package procedure_func is
procedure comp_sel(a,b: std_logic_vector(4 downto 0);
signal path_a,path_b:std_logic_vector(8 downto 1);
signal path: out std_logic_vector(8 downto 1);
signal c : out std_logic_vector(4 downto 0));
end;

package body procedure_func is
procedure comp_sel(a,b: std_logic_vector(4 downto 0);
signal path_a,path_b:std_logic_vector(8 downto 1);
signal path: out std_logic_vector(8 downto 1);
signal c : out std_logic_vector(4 downto 0)) is
begin
if (a < b) then
    c <= a ;
    path <= path_a;
else
    c <= b;
    path <= path_b;
end if;
end;
end procedure_func;

library IEEE;
use IEEE.std_logic_1164.all;--for std_logic, std_logic_vector
use IEEE.std_logic_misc.all;
use IEEE.std_logic_arith.all;
--use IEEE.std_logic_components.all;
use IEEE.std_logic_unsigned.all;
--library SYNOPSYS;
--use SYNOPSYS.bv_arithmetic.all;

entity decoder is
  port(clk,reset: in std_logic;
       dec_in:in std_logic_vector(21 downto 0);
       dec_out:out std_logic_vector(7 downto 0));
end;

ARCHITECTURE behave of decoder is
  use v^orlf.prQcedure_func.all;
  signal count: INTEGER range 0 to 11;
  signal pth0, pth1, pth2, pth3, pth4, pth5, pth6, pth7 :std_logic_vector(8 downto 1);
  signal pm0,pm1,pm2,pm3,pm4,pm5,pm6,pm7 :std_logic_vector( 4 downto 0);
BEGIN
  Process(clk,count,reset)
  begin
    if reset='1' then
      count <= 10 ;
    elsif (clk'event and clk='1') then
      if (count=11) then
        count <= 1; 
      else count <= count + 1;
      end if;
    end if;
  end process;

  Process(reset,count)
  variable r:std_logic_vector(21 downto 0);
  variable b0,b1,b2,b3 :std_logic_vector(1 downto 0);
  variable temp:std_logic_vector(1 downto 0);
  variable p0_0,p0_4,pl_0,pl_4,p2_1,p2_5,p3_1,p3_5,p4_2,p4_6,p5_2,p5_6,p6_3,p6_7,p7_3,p7_7:std_logic_vector( 4 downto 0);
  variable temp_pth0 : std_logic_vector(8 downto 1);
  variable temp_pm : std_logic_vector( 4 downto 0);
  begin
    if (reset='1') then
      dec_out <= "00000000";
      pm0<="00000";
      pm1<="00000";
      pm2<="00000";
      pm3<="00000";
      pm4<="00000";
      pm5<="00000";
    end if;
  end Process;

  Process(clk,count)
  begin
    if (count=11) then
      count <= count + 1;
    end if;
  end Process;

end;
pm6<="00000";
pm7<="00000";
temp:="00";
pth0<="00000000";
pth1<="00000000";
elif (count > 0) then
  if (count = 1) then
    r:=dec_in;
  end if;
b0 := temp + r(2*count-2) + r(2*count-1);
b1 := temp + r(2*count-2) + (not r(2*count-1));
b2 := (not b1);
b3 := (not b0);
case count is
  when 1 =>
    pm0(1 downto 0) <= b0;
    pm0(4 downto 2) <= "000";
    pm4(1 downto 0) <= b3;
    pm4(4 downto 2) <= "000";
    when 2 =>
      pm0 <= pm0 + b0;
      pm2 <= pm4 + b1;
      pm4 <= pm0 + b3;
      pm6 <= pm4 + b2;
  when 3 =>
    pm0 <= pm0 + b0;
    pm1 <= pm2 + b3;
    pm2 <= pm4 + b1;
    pm3 <= pm6 + b3;
    pm4 <= pm0 + b3;
    pm5 <= pm2 + b1;
    pm6 <= pm4 + b2;
    pm7 <= pm6 + b0;
    pth0 <= "00000000";
    pth1 <= "00000001";
    pth2 <= "00000010";
    pth3 <= "00000011";
    pth4 <= "00000100";
    pth5 <= "00000101";
    pth6 <= "00000110";
    pth7 <= "00000111";
  when 9 =>
    p0_0 := pm0 + b0;
    p1_0 := pm1 + b3;
    p2_1 := pm2 + b2;
    p3_1 := pm3 + b1;
    p4_2 := pm4 + b1;
p5_2 := pm5 + b2;
p6_3 := pm6 + b3;
p7_3 := pm7 + b0;
comp_sel(p0_0,p1_0,pth0,pth1,pth0,pm0);
comp_sel(p2_1,p3_1,pth2,pth3,pth1,pm1);
comp_sel(p4_2,p5_2,pth4,pth5,pth2,pm2);
comp_sel(p6_3,p7_3,pth6,pth7,pth3,pm3);
when 10 =>
p0_0 := pm0 + b0;
p1_0 := pm1 + b3;
p2_1 := pm2 + b2;
p3_1 := pm3 + b1;
comp_sel(p0_0,p1_0,pth0,pth1,pth0,pm0);
comp_sel(p2_1,p3_1,pth2,pth3,pth1,pm1);
when 11 =>
p0_0 := pm0 + b0;
p1_0 := pm1 + b3;
if (p0_0 < p1_0) then
  temp_pth0:=pth0;
else
  temp_pth0:=pth1;
end if;
dec_out(7 downto 0) <= temp_pth0(8 downto 1);
when others =>
p0_0 := pm0 + b0;
p1_0 := pm1 + b3;
p2_1 := pm2 + b2;
p3_1 := pm3 + b1;
p4_2 := pm4 + b1;
p5_2 := pm5 + b2;
p6_3 := pm6 + b3;
p7_3 := pm7 + b0;
p0_4 := pm0 + b3;
p1_4 := pm1 + b0;
p2_5 := pm2 + b1;
p3_5 := pm3 + b2;
p4_6 := pm4 + b2;
p5_6 := pm5 + b1;
p6_7 := pm6 + b0;
p7_7 := pm7 + b3;
comp_sel(p0_0,p1_0,pth0,pth1,pth0,pm0);
comp_sel(p2_1,p3_1,pth2,pth3,pth1,pm1);
comp_sel(p4_2,p5_2,pth4,pth5,pth2,pm2);
comp_sel(p6_3,p7_3,pth6,pth7,pth3,pm3);
library IEEE;
use IEEE.std_logic_1164.all;
-- use IEEE.std_logic_misc.all;
use IEEE.std_logic_arith.all;
-- use IEEE.std_logic_components.all;
-- use IEEE.std_logic_textio.all;
use IEEE.std_logic_signed.all;

entity test_add_com_sel is end;
architecture tb_behave of test_add_com_sel is

signal Clock,Reset: STD_LOGIC:=T;
signal path_a,path_b,path_c,path_d: std_logic_vector(7 downto 0);
signal a,b,c,d : std_logic_vector(4 downto 0);
signal bm,not_bm : std_logic_vector(1 downto 0);

component add_com_sel
port(clk,reset: in std_logic;
    a,b: in std_logic_vector(4 downto 0);
    path_a,path_b: in std_logic_vector(7 downto 0);
    bm,not_bm: in std_logic_vector(1 downto 0);
    path_c, path_d: out std_logic_vector(7 downto 0);
    c,d : out std_logic_vector(4 downto 0));
end component;

for UUT : add_com_sel
use entity work.add_com_sel(behavior);
begin
process begin
if (now < 250 us) then
a<="11111";
b<="10000";
path_a <= "10000000";
path_b <= "01000000";
bm <= "00";
not_bm <= "11";
elsif (now < 410 us) then
a<="11100";
b<="11111";
path_a <= "10000000";
path_b <= "01000000";
bm <= "00";
not_bm <= "11";
end if;
--wait for 10 us;
--wait for 10 ms;
Reset<=0';
Clock <= not Clock;
if (now> 500 us) then wait ;end if;
end process;

UUT : add_com_sel port
map(clock,reset,a,b,path_a,path_b,bm,not_bm,path_c,path_d,c,d);
end;

entity test_com_sel_add is end;
architecture tb_behave of test_com_sel_add is

signal Clock,Reset: STD_LOGIC:=1';
signal path_a,path_b,path_c,path_d: std_logic_vector(7 downto 0);
signal a,b,c,d : std_logic_vector(4 downto 0);
signal bm,not bm : std_logic_vector(1 downto 0);

component com_sel_add
port(clk,reset: in std_logic;
a,b: in std_logic_vector(4 downto 0);
path_a,path_b: in std_logic_vector(7 downto 0);
bm,not bm: in std_logic_vector(1 downto 0);
path_c, path_d: out std_logic_vector(7 downto 0);
c,d : out std_logic_vector(4 downto 0));
end component;

for UUT : com_sel_add
    use entity work.com_sel_add(behav);

begin
    process begin
        if (now < 250 us) then
            a<="11111";
            b<="10000";
            path_a <= "10000000";
            path_b <= "01000000";
            bm <= "00";
            not_bm <= "11";
        elseif (now < 410 us) then
            a<="11100";
            b<="11111";
            path_a <= "10000000";
            path_b <= "01000000";
            bm <= "00";
            not_bm <= "11";
        end if;
    wait for 10 us;
    Reset<='0';
    Clock <= not Clock;

        if (now> 500 us) then wait ;end if;
    end process;

UUT : com_sel_add port
    map(clock,reset,a,b,path_a,path_b,bm,not_bm,path_c,path_d,c,d);
end;
On-Chip Code Tracking for Multi-Carrier Multiple Access System

Abstract - Present day services and its users need new technology so as to integrate multimedia services along with existing ones within the same Bandwidth. Future communication networks need wide Bandwidth & high speed access. Ever since the World war 2, spread spectrum techniques have been widely used in a variety of military and civilian applications. A method of Spread Spectrum Multiple Access technique called Frequency Hopping/Multiple Carrier Direct Sequence-Code Division Multiple Access (FH/MC DS-CDMA), is an emerging system for future communication system. In this work an FH/MC DS-CDMA scheme is synthesized in VLSI chip of Xilinx's XC4000 architecture. The adaptability of the proposed system for future generation mobile as well as existing generation is always higher. Multiple carriers can be allotted as per user requirements with high secrecy and multiple encryptions. The performance analysis for the above scheme is shown supportive in terms of BER, power requirements, complexity, etc., compared to existing and evolving techniques. Here as a part of the research work a method for quick tracking of frequency hopped signals is performed in on-chip.

Key words: Spread spectrum, Code acquisition.

I INTRODUCTION

With the substantial increase of Internet users and with the development of new services, high-speed access in future generations of wireless systems is an important requirement. Consequently, broadband systems with bandwidths much wider than that of the 3rd-generation systems are required for meeting future requirements. Hence, compatibility with both the emerging Broadband Access Networks (BRAN), which has opted for a multi-carrier, Orthogonal Frequency Division Multiplexing (OFDM) based solution and the existing 2nd- and 3rd-generation CDMA Systems is an important consideration.

A potential multiple access scheme meeting these requirements can be proposed. The multiple-access scheme is constituted by frequency-hopping (FH) based multicarrier DS-CDMA (FH/MC DS-CDMA).

II REQUIREMENTS FOR FUTURE GENERATION NETWORKS

A) High Speed Access

Multicarrier Frequency Hopping technique in DS-CDMA gives way for very high speed.

B) Wide Bandwidth

CDMA as a technique of Spread Spectrum gives way for wide Bandwidth with multi-carrier employment. i.e., increase in spectral bandwidth with decrease in power.

C) Compatibility with 2nd and 3rd Generation Systems

Multicarriers in FH/MC DS-CDMA, provide the possibility to adjust with 2nd & 3rd generation systems for communication. i.e., required number of sub carriers can be allotted to needed 2nd or 3rd generation systems.

D) Variable Data Rate Transmission

The technique of Frequency Hopping favors way for variable data rate requirements. i.e., using the methods of Frequency Hopping (Slow, Fast, Random, Uniform and Adaptive), low to high rate of transmission can be achieved.

E) Support Multimode and Multi-Standard Communication Different Standards:

(For example)

- IS-95: Digital Cellular standards in the United States
- IS-95B: Enhanced version of IS-95
- PDC: Japanese personal digital cellular system
- IS-54: American digital advanced mobile phone system (DAMPS)
- IS-136: North American TDMA system

The multimode and multi-standard communication can be achieved in FH/MC DS-CDMA by employing compatible software. i.e., Software defined radio (SDR) implementation in FH/MC DS-CDMA technique pave way for multimode and multi-standard communication.

III FH/MC DS-CDMA TECHNIQUE

In FH/MC DS-CDMA, the entire bandwidth of future systems can be divided into a number of sub-bands and each sub-band can be assigned a sub carrier. According to the prevalent service requirements, the set of legitimate sub carriers can be distributed in line with the instantaneous information rate requirements.

FH techniques are employed for each user, in order to occupy the whole system bandwidth and to efficiently utilize the system's frequency resources. Specifically, slow FH, fast FH or adaptive FH techniques can be utilized depending on the system's design.

In FH/MC DS-CDMA systems the sub-bands are not required to be of equal bandwidth. Hence existing 2nd- and 3rd-generation CDMA systems can be supported using one or more sub carriers, consequently simplifying the frequency resource management and efficiently utilizing the entire bandwidth available. This regime can also remove the spectrum segmentation of existing 'legacy' systems, while ensuring compatibility with future BRAN and un-licensed systems. Furthermore, a number
of sub-channels with variable processing gains can be employed, in order to support various services requiring low- to very high-rate transmissions, for example for wireless Internet access.

Figure-1 illustrates transmitter model of the FH/MC DS-CDMA system. The transmitter is depicted in figure-1. Each sub-carrier of the K users in the system is assigned a PN sequence, which produces spread, wideband signals. In the figure, \( C(Q, U_k) \) represents a constant-weight code of user \( k \) with \( U_k \) number of '1's and \( (Q - U_k) \) number of '0's, hence, the weight of \( C(Q, U_k) \) is \( U_k \). This code is read from a constant-weight code book, which represents the frequency-hopping patterns.

The constant-weight code serves two different functions. Its first function is that its weight - namely \( U_k \) - determines the number of sub-carryers involved, while its second function identifies the positions of the \( U_k \). The number of binary '1's determines the selection of a set of \( U_k \) number of sub-carrier frequencies from the \( Q \) number of outputs of the frequency synthesizer. High-rate and variable rate transmissions are achieved by employing a different number of sub-carriers.

At the transmitter of the \( k \)-th user in the Figure-1, the bit stream is first serial-to-parallel (S-P) converted, yielding \( U_k \) parallel streams, which is controlled by the constant-weight code \( C(Q, U_k) \). After S-P conversion each stream is direct-sequence spread, in order to form the spread, wideband signal. These spread signals then modulate their corresponding sub-carriers, and finally are summed, in order to form the transmitted signal \( S_k(t) \).

3.2 FH/MC DS-CDMA Receiver

Figure-2 illustrates receiver model of \( k \)-th user of the FH/MC DS-CDMA system. Receiver verifies sub-carrier that is used through Band pass filter (BPF) and selects each sub-carrier. The information on hopping is received at this juncture. With this information the used sub-carrier is detected and demodulated. At the stage for detection of signal, pulse is restored using Matched Filter (MF) that discriminates the presence of pulse and original signal is restored with maximum Ratio Combiner (MRC). Restored signals are demodulated into original data stream format by parallel-serial converter using the hopping information received.

IV MULTI-CARRIER IMPLEMENTATION

The multi-carrier modulation and demodulation can be implemented using FFT techniques, provided that each of the sub-channels occupies the same bandwidth. Since not all of the sub-carriers are activated at each transmission in the proposed FH/MC DS-CDMA system, the deactivated sub-carriers is be set zero in the FFT or inverse FFT algorithm. However, if an unequal bandwidth is associated with the sub-channels, multi-carrier modulation/demodulation can only be implemented using less efficient conventional, rather than FFT-based carrier modulation/demodulation scheme.

ON-LINE CODE ACQUISITION

1) Principle

No matter, which form of spread spectrum technique is employed, it is necessary that the timing information of the transmitted signal should match with the locally generated carrier, in order to de-spread the received signal. An offset by even single chip duration may render it totally impossible to de-spread the received spread spectrum signal. This is due to the reason that the spread sequence is designed to have only a small out-of-phase autocorrelation magnitude. Conventionally the problem of timing acquisition is solved via a two-step approach: (1) Initial code acquisition that synchronizes the transmitter and receiver to within an uncertainty of \( \pm T_c \) and (2) Code tracking.

2) Problem and Solution

The drawback is that, though code tracking is a relatively easy task and is usually accomplished by a delay lock loop (DLL), the tracking loop keeps on operating during the whole communication period. If the channel characteristics change abruptly, the delay lock loop will lose track of the correct timing and initial acquisition has to be re-performed. Compared to code tracking, initial code acquisition in a spread spectrum system is very difficult. First, the timing uncertainty, which is basically determined by the transmission time of the transmitter and the propagation delay, can be much longer than the chip duration. As initial acquisition is usually achieved by a search through all possible phases (delays) of the sequence, a larger timing uncertainty results in a larger search area. Besides timing and frequency uncertainty due to Doppler shift, a mismatch between the transmitter and receiver oscillators can also
exist. This necessitates a two-dimensional search in time and frequency. This is exactly what is proposed and illustrated in this paper. Initial code acquisition is accomplished in low signal-to-noise ratio environments and in the presence of jammers. The possibility of channel fading and the existence of multiple access interference in CDMA environments are also explored in the work.

3) Illustration of Code Tracking

The process of acquiring the timing information of the transmitted spread spectrum signal is essential to the implementation of any form of spread spectrum technique. In the commonly employed DS-SS system, this is equivalent to matching the phase. In figures 1 and 2, the concept of phase synchronization between the transmitted signal (assuming no noise case) and the locally generated carrier (PN carrier) is illustrated. The de-spreader output for case (i) (figure-3) equals the data input, whereas, for case (ii) (figure-4) the de-spreader output is a random sequence. This is because, perfect phase synchronization is achieved only in case (i).

![Figure 3: Phase synchronization is perfect](image)

![Figure 4: Phase error due to one bit offset](image)

The objective of any initial code acquisition scheme is to initially achieve a coarse synchronization between the receiver and the transmitted signal. The receiver hypothesizes the phase of the spreading sequence and attempts to de-spread the received signal using this hypothesized phase. If the hypothesized phase matches the sequence in the received signal, the wide-band spread spectrum signal will be de-spread correctly to give a narrowband data signal. Then a band-pass (BPF) filter with a bandwidth similar to that of the narrowband data signal can be employed to collect the power of the de-spread signal. On the other hand, if the hypothesized phase does not match with the received signal the de-spreader will give a wideband output and the BPF will be able to collect only a small portion of the power of the de-spread signal.

Thus the power measured at the output of the BPF is a measure for the receiver to decide whether the hypothesized phase is correct or not and subsequently other phases are tried. In this paper, a two-dimensional search technique is proposed with due consideration to the channel fading effects and other CDMA environments.

4) Chip Implementation

The xilinx architecture shown in figure-5 is to be used for the proposed FH/MC DS-CDMA system.

This is a fairly complicated basic logic cell containing 2 four-input Look-up Table (LUT) that feed a three input LUT. The XC4000 CLB also has special fast carry logic hard-wired between CLBs. MUX control logic maps four control inputs (C1-C4) into the four inputs: LUT input HI, direct input (DIN), enable clock (EC), and a set/reset control (S/R) for the flip-flops. The control inputs (C1-C4) can also be used to control the use of the F' and G' LUTs as 32 bits of SRAM.

![Figure 5: Xilinx XC4000](image)

![Figure 6: Single Chip transmitter & receiver](image)
The details of the single chip transmitter and receiver are presented in figure-6. The chip supports both digital and analog inputs/outputs and is designed using mixed mode logic.

5.1) Details of Transmitter

The on-chip transmitter comprises of four parts.
A: An 8-bit PN sequence generator generates the PN sequence in random.
B: Data generator generates the Data, which has to be spreaded.
C: Multi-carrier application to the PN sequence depending on code selection logic.
D: Digital modulation is converted to analog for transmission through antenna with the help of on-chip VCO.

5.2) Effect of Phase Shift on Carrier Signal

The waveforms in figure-8 shows the effect of the phase shift introduced by the code selection at the transmitter on the carrier signal. Depending on the code selected any phase shift between 0 to 255 bit positions is introduced in the transmitted carrier signal. The 'M' denotes the number of phase shifts. This information is received in PSK form at the receiver and an identical phase shift is required at receiver to obtain coherency in the locally generated carrier.

\[
A \sin (\theta_1, K/256) \quad \text{for } K = 0, 1, 255 \\
\theta_1 = (\pi/2)/64 = (\pi/128) \text{ rad}
\]

5.3) Details of Receiver

The on-chip receiver comprises of four parts.
A: Analog modulated signal from the antenna is converted to digital with the help of on-chip VCO.
B: Based on the synchronous lock module, the multi carrier and the PN sequence are identified.
C: De-spreading is done with the help of locally generated PN sequence.
D: Data is recovered and presented as output.

5.4) Automatic Code Tracking and Lock Logic (ACTL)

This module resides on-chip and using this different codes are selected and the scanning is done for 256 combinations of the code keys. This module is triggered by the signal received from the synchronizing lock module.
5.5) Receiver Module with the ACTL

![Diagram of proposed receiver module]

Fig. 12 Proposed Receiver Module

5.6) Details of the Selection Logic

This is again intrinsic in the chip and forms the final interface section. The data output taken out from this section directly. The signal selection of this module is done using the synchronization lock module. It is to be noted that the synchronous clock signal is obtained only when the code at receiver is balanced with the code at transmitter and this balancing is done automatically.

![Diagram of selection logic]

Fig. 13 Selection logic

VI RESULTS

In the first case, the carrier code at transmitter was set equal to the binary value "00000111", i.e., 7th position. The receiver output shown in figure-14 demonstrates that the code is locked at the 7th position.

![Diagram showing results in the first case]

Fig. 14

In the second case, the carrier code at transmitter was set to the binary value "00000100", i.e., 4th position. The receiver output shown in the figure-15 demonstrates that the code is locked at the 4th position.

![Diagram showing results in the second case]

Fig. 15

VII ADVANTAGES OF PROPOSED FH/MC DS-CDMA SCHEME

The FH/MC DS-CDMA system can mitigate the effects of intersymbol interference encountered during high-speed transmissions and readily supports partial-band and multi-tone interference suppression. Moreover, multi-user detection techniques potentially approaching single user performance.

VIII CONCLUSION

The on-chip technology presented above as a part of the research work is a flexible broadband mobile wireless communication code acquisition technique. This scheme can be used for FH/MC DS-CDMA technique and is reviewed for existing as well as a range of forth coming techniques. This adoptability is required to develop broadband mobile wireless systems exhibiting high flexibility and high efficiency in future (4G).
IX REFERENCES

2. "Burst by Burst adaptive CDMA research" by Dr. Fe LIN Kuan, 2003.