0.1 Sinus kode

Kodeudsnit 1: Sinus generation for data output

```
// Write number of samples to RAM-controller
sendNumberOfSamples(full_samples);

// FIRST QUARTER
for(sample_i = 0; sample_i < quarter_samples; sample_i++)
{
  // Calculating first quarter.
  // This array will be used for the remaining quarters
  quarter_sound_samples[sample_i]
    = (1+sin((sample_i*2*PI)/(full_samples)))*HALF_MAX_CODEC_SIZE;
  sendNextSample(quarter_sound_samples[sample_i]);
}

// SECOND QUARTER
for(sample_i = quarter_samples; sample_i > 0; sample_i--)
{
  current_sample = quarter_sound_samples[sample_i-1];
  sendNextSample(current_sample);
}

// THIRD QUARTER
for(sample_i = 0; sample_i < quarter_samples; sample_i++)
{
  current_sample = MAX_CODEC_SIZE-quarter_sound_samples[sample_i];
  sendNextSample(current_sample);
}

// FOURTH QUARTER
for(sample_i = quarter_samples; sample_i > 0; sample_i--)
{
  current_sample
    = MAX_CODEC_SIZE-quarter_sound_samples[sample_i-1];
  sendNextSample(current_sample);

}
```

0.2 TransferProtocol

Kodeudsnit 2: TransferProtol valg af samples eller data

```
if address = "00000000" then
  ramSamples_to_write := writedata(7 downto 0);
  index <= 0;

  if ramSamples_to_write = X"00" then
    ramSamples_to_read <= ramSamples_to_write;
```

```
    ram_to_play <= not ram_to_play;
  end if;

--What to write on ram module
elsif address = "00000001" then   -- binary, address = 2
  ram_Addr <= index;          -- write addr to ram
  ram_Data <= writedata;        -- write data to ram
  index <= index + 1;         -- increment index
end if;
```

0.3 RamAccess-lagring

Kodeudsnit 3: Process for lagring i ram-modulerne

```
process (clk, reset_n)
begin
  if reset_n = '0' then
    ram_block <= (others => (others => '0'));
    readData <= (others => '0');

  elsif rising_edge(clk) then
    if CS = '1' then
      ram_block(writeAddr) <= writedata;
    end if;

    readData <= ram_block(readAddr);
  end if;
end process;
```